Master Thesis

Optimization of the Low-Level Torque Controller of the Quadruped Robot HyQ

Author(s):
Hubacher, Sven

Publication Date:
2014

Permanent Link:

https://doi.org/10.3929/ethz-a-010341012 →

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ADRL

Sven Hubacher

Optimization of the Low-Level Torque Controller of the Quadruped Robot HyQ

Master's Thesis

Institute of Robotics and Intelligent Systems
Agile and Dexterous Robotics Lab
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Dr. Thiago Boaventura
Farbod Farshidian
Prof. Dr. Jonas Buchli

April 2014

Abstract

The hydraulic actuated quadruped robot HyQ at the ETH Zurich is a four legged robot capable of versatile, robust and dynamic maneuvers. The robot is fully position and torque controlled, the latter enabling the use of many control laws that produce torque commands as outputs. To allow best performance of these control laws, a *perfect* torque source is desired. Currently a nonlinear partial feedback linearization controller with fixed gains is used for torque tracking in the hydraulic actuation system. Changing load conditions, e.g. when the robot is walking, pose difficulties for this fixed gain low-level controller not being able to follow the torque command satisfactorily. Therefore, this work presents an adaptive control framework to cope with changing system dynamics by recursively estimating the current system dynamics and updating the low-level controller when needed. The hydraulic cylinder dynamics are first studied and analysed. Then an adaptive control framework is proposed and its effectiveness is shown in simulation. Controller and estimator were implemented on a real-time hydraulic cylinder test set-up. Experimental results assess the performance of the estimator. The controller is verified for different load conditions and the possible gain in performance is pointed out when using the proposed adaptive control framework.

Acknowledgements

I would like to sincerely thank all the people who have contributed to this work in various ways. First, I would like to thank Prof. Jonas Buchli for letting me do this master's thesis. Thank you for your open and friendly way of communicating. There is an inspiring spirit within your robotics lab. I also want to sincerely thank my two supervisors Dr. Thiago Boaventura and Farbod Farshidian for their helpful and friendly and sometimes critical advices. I very much enjoyed working with you.
I had the opportunity to spend two months at the IIT in Genova for performing the experiments. Many thanks to Dr. Michele Focchi, who helped me a lot in doing the experiments. Without his support, I would not have been able to get the experiments done within a reasonable time frame. He was always there for me, even after I returned to Zurich to finish the thesis.
Many thanks to all the members of the HyQ team at the IIT, who made these two months a very enjoyable experience.
I would also like to thank Michael Neunert for his advice on construction details related to the new test set-up design and on how to write proper template classes in C++. Many thanks also to Pascal Wespe, who gave many very useful tips on how to design parts that are easier and quicker to manufacture.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

$\alpha = \frac{A_r}{A_p}$	Factor of the rod side area vs. area of the other side	[−]
β	Weight to reduce the influence of the old measurements	[−]
β_e	Effective bulk modulus of the oil	[Pa]
ϵ	Prediction error	[−]
θ	Joint position	[rad]
θ_{ref}	Joint angle reference	[rad]
$\dot{\theta}$	Joint velocity	[rad/s]
$\ddot{\theta}_{ref}$	Joint angle acceleration reference	[rad/s²]
λ	Forgetting factor	[−]
ρ	Number of additional integrator states of the LQR controller	[−]
τ	Torque of the hydraulic system	[Nm]
τ_{fb}	Torque command from feedback controller	[Nm]
τ_{ff}	Torque command from feedforward controller	[Nm]
τ_{ref}	Torque reference command for low-level controller	[Nm]
Γ	Feedforward part of the LQR controller	[−]
Θ	Unknown parameter vector	[−]
ω	Measurement noise	[−]
ω_v	Valve spool angular frequency	[rad/s]
$\mathcal{A}$	State transition matrix (general)	[−]
A_e	Equivalent area of the piston	[m^2]
A_p	Area of the piston	[m^2]
A_r	Area of the piston side with the rod	[m^2]
$\mathcal{B}$	Input transition matrix	[−]
B	Combined friction $B = d_p + d_l$ of load mass and cylinder	[Ns/m]
$\mathcal{C}$	Output transition matrix	[−]
C	Combined stiffness of the springs $C = k_1 + k_2$	[N/m]
$\frac{d}{dt}$	Symbol for calculating the time derivate of an expression	[−]
$\frac{\partial}{\partial ...}$	Symbol for calculating the partial derivate of an expression	[−]
d_l	Velocity-dependent friction of the load mass and bearings	[Ns/m]
d_p	Velocity-dependent friction of the cylinder	[Ns/m]

$\mathcal{D}$	Input transition matrix for output channel	[−]
D_{pl}	Diameter of the pipe line	[m]
D_v	Valve spool damping	[−]
e	Controller error $e[k] = r[k] - y[k]$	[−]
f	General nonlinear function for the state dynamics	[−]
	Frequencies of a signal, for example, a multisine with $f = [2, 10]Hz$	[Hz]
$f(n)$	Part of the RLS derivation	[−]
F	General force symbol	[N]
$\dot{F}_{...}$	General force dynamics symbol (for any force)	[N/s]
F_1	Spring force on load mass side	[N]
F_2	Spring force on cylinder side	[N]
F^a	General external force symbol	[N]
F_{cL}	Constraint force of load mass side	[N]
F_{cp}	Constraint force of cylinder side	[N]
F_h	Hydraulic force of the cylinder	[N]
F_{Lf}	Friction of the load mass	[N]
F_{pf}	Friction of the cylinder piston	[N]
F_s	Load cell sensor force	[N]
$F_{s\oslash}$	Operating point of the load cell sensor force	[N]
F_t	Total force	[N]
g	General nonlinear function for the output measurements	[−]
G_{F_s}	Load cell force gain for simplified Fs model	
G_{u_v}	Controller output gain for simplified Fs model	
G_{x_p}	Position gain for simplified Fs model	
H	Regression matrix relating measurement to parameter	[−]
$H_{...}$	Part of the regression matrix H containing the different contributions H_A, H_B, H_C, H_D of state, input etc.	[−]
I	Diagonal unity matrix	[−]
J	Cost function (matrix) for the Riccari equation	[−]
k_1	Spring stiffness of spring on load mass side	[N/m]
k_2	Spring stiffness of spring on cylinder side	[N/m]
$\mathcal{K}$	Optimal gain calculated using the Riccati solution $\mathcal{K} = [K_I K]$	[−]
K	State feedback gain of the LQR controller	[−]
$K(n)$	Correction gain of the RLS algorithm	[−]
K_I	Integrator feedback gain of the LQR controller	[−]
K_{pl}	Load pressure gain for simplified model	
K_{sp}	Steady-state valve input-to-spool position gain	[m/A]
K_{th}	Hydraulic transmission stiffness $K_{th} = K_{tha} + K_{thb}$	[Pa/m]
K_{tha}	Hydraulic transmission stiffness for chamber A	[Pa/m]
K_{thb}	Hydraulic transmission stiffness for chamber B	[Pa/m]
K_{u_v}	Valve input gain for simplified model	

K_v	Valve gain	$[\mathrm{m^3/s}/(\mathrm{A}\sqrt{\mathrm{N/m^2}})]$
K_{x_pA}	Position pressure gain for simplified model	
K_{x_pB}	Velocity gain for simplified model	
$K_{\dot{x}_p}$	Position velocity gain for simplified model	
L_0	Unstressed length of the springs $L_0 = L_{01} = L_{02}$	[m]
L_{01}	Unstressed length of spring on load mass side	[m]
L_{02}	Unstressed length of spring on cylinder side	[m]
L_c	Length of the cylinder stroke	[m]
L_{pl}	Length of the pipe line	[m]
L_{vsp}	Pre-compressed length of the springs	[m]
m	General mass symbol or number of measurements	[kg]
m_l	Load mass	[kg]
m_p	Cylinder piston mass	[kg]
M	Combined mass $M = m_l + m_p$ of load mass and cylinder or general mass symbol	[kg]
n	Discrete-time notation, or number of states	[−]
n_u	Number of inputs	[−]
N_0	Number of samples a system remains constant	[−]
N	Weighting matrix for the cross-coupling of state and input of the Riccari equation	[−]
$\oslash$	General operating point (symbol)	[−]
$\mathcal{O}$	Zero matrix of proper size	[−]
p_a	Pressure inside chamber A	[Pa]
$p_{a\oslash}$	Operating point of the pressure inside chamber A	[Pa]
p_b	Pressure inside chamber B	[Pa]
$p_{b\oslash}$	Operating point of the pressure inside chamber B	[Pa]
p_l	Load pressure across the chambers A and B	[Pa]
$p_{l\oslash}$	Load pressure operating point	[Pa]
p_s	Supply pressure	[Pa]
p_t	Reservoir pressure	[Pa]
$\dot{p}$	Pressure dynamics (general)	$[\mathrm{Pa/s}]$
$\dot{p}_a$	Pressure dynamics inside chamber A	$[\mathrm{Pa/s}]$
$\dot{p}_b$	Pressure dynamics inside chamber B	$[\mathrm{Pa/s}]$
$\dot{p}_l$	Load pressure dynamics	$[\mathrm{Pa/s}]$
Δp_n	Nominal pressure drop across to ports of the valve	[Pa]
$P(n)$	Variance matrix of the RLS algorithm	[−]
q_a	Fluid flow of chamber A	$[\mathrm{m^3/s}]$
q_b	Fluid flow of chamber B	$[\mathrm{m^3/s}]$
q_e	Equivalent fluid flow	$[\mathrm{m^3/s}]$
q_{in}	Ingoing flow (general)	$[\mathrm{m^3/s}]$
q_n	Nominal flow through the valve	$[\mathrm{m^3/s}]$

q_{out}	Outgoing flow (general)	[m^3/s]
Q	Weighting matrix for the states of the Riccari equation	[−]
$r(t), r[k]$	Reference signal to the controller	[−]
$R(n)$	Part of the RLS derivation	[−]
R	Weighting matrix for the inputs of the Riccari equation	[−]
s	Laplace variable (complex number)	[−]
S	Solution of the Riccari equation	[−]
S_0	Initial solution of the Riccari equation	[−]
S_{k+1}, S_k	Solution of the Riccari equation of the next and current time step	[−]
t	Time, and (t) for a variable at Time t	[s]
T	Normalization matrix for the states	[−]
T_s	Sampling time of signals of the FC1D hardware	[s]
u	Controller output command	[−]
$u_\oslash$	Operating point of the controller output command	[−]
$\vec{u}$	Controller output vector command	[−]
$\hat{u}$	Normalized controller output	[−]
Δu	Controller output command in the frequency domain	[−]
u_v	Equivalent valve spool position command	[A]
Δu_v	Equivalent valve spool command in the frequency domain	[A]
$u_{v\oslash}$	Operating point of the equivalent valve spool command	[A]
u_{vn}	Nominal equivalent valve spool position command	[A]
v	General velocity symbol	[m/s]
V	Volume (general)	[m^3]
V	Normalization matrix for the inputs	[−]
V_a	Volume of the chamber A	[m^3]
$V_{a\oslash}$	Operating point of the volume of the chamber A	[m^3]
V_b	Volume of the chamber B	[m^3]
$V_{b\oslash}$	Operating point of the volume of the chamber B	[m^3]
V_{pl}	Pile line volume	[m^3]
$\dot{V}$	Change of volume (general)	[m^3/s]
W	Normalization matrix for the outputs Weighting matrix of the RLS algorithm	[−]
$\vec{x}$	State space vector	[−]
$\hat{x}$	Normalized states	[−]
$x_1, \ldots$	General states (numbered)	[−]
xi	Additional gain for the integrator action of the LQR controller	[−]
x_l	Position of the load mass	[m]
x_{l0}	Equilibrium position of the load mass	[m]
x_p	Position of the cylinder piston	[m]
x_{p0}	Equilibrium position of the cylinder	[m]
$x_{p\oslash}$	Operating point of the position of the cylinder	[m]

x_v	Position of the valve spool	[m]
Δx_v	Position of the valve spool in the frequency domain	[m]
x_{vn}	Nominal valve spool position input	[m]
$\dot{x}_l$	Velocity of the load mass	[m/s]
$\dot{x}_p$	Velocity of the cylinder piston	[m/s]
$\dot{x}_{p\oslash}$	Operating point of the velocity of the cylinder piston	[m/s]
$\ddot{x}_l$	Acceleration of the load mass	[m/s^2]
$\ddot{x}_p$	Acceleration of the cylinder piston	[m/s^2]
$\vec{y}$	Output measurement vector	[−]
$\vec{y}_\oslash$	Operating point of the output measurement vector	[−]
$\hat{y}$	Normalized measurements	[−]
y	Position of the load mass $y = x_l - x_{l0}$ measured from the equilibrium position	[m]
$\dot{y}$	Velocity of the load mass measured from the equilibrium position	[m/s]
$\ddot{y}$	Acceleration of the load mass measured from equilibrium position	[m/s^2]
y_0	New equilibrium position of the load mass measured from the equilibrium position x_{l0}	[m]
$y_\oslash$	Operating point of the position of the load mass measured from the equilibrium position. Also, general operating point of output measurement	[m] [−]
Y	Output measurement vector of the RLS algorithm	[−]
$\dddot{\square}$	General third time-derivative of an expression	[−]
$[\dots], [k]$	Symbol for a variable in discrete time	[−]
$\tilde{\square}$	Symbol for augmented variables or matrices	[−]

Acronyms and Abbreviations

ADRL	Agile and Dexterous Robotics Lab
AFF	Adaptive Forgetting Factor
CAD	Computer-Aided Design
C/C++	A general purpose programming language
DARE	Discrete-time Algebraic Riccati Equation
dB	deciBel
ETH	Eidgenössische Technische Hochschule
FC1D	Force Control One-Dimensional test set-up
FFT	Method based on the frequency response
FL	Feedback Linearization controller
GmbH	Gesellschaft mit beschränkter Haftung (company with limited liability)
HyL	Hydraulically actuated Leg test set-up
HyQ	Hydraulically actuated Quadruped robot
IIT	Istituto Italiano di Tecnologia
LQG	Linear Quadratic Gaussian regulator

LQR	Linear Quadratic Regulator
LQR-I	Linear Quadratic Regulator with Integrator
MATLAB	MAtrix LABoratory, a simulation and numerical computing environment
MIMO	Multiple-Input and Multiple-Output system
PD	Proportional Derivative controller
PE	Persistence of Excitation
RLS	Recursive Least Squares algorithm
RTOL	Relative TOLerance
SISO	Single-Input and Single-Output system
SL	Simulation Laboratory software package
VC	Velocity Compensation

Chapter 1

Introduction

The robot HyQ [36, 37, 35] is a hydraulic actuated quadruped robot developed at the Istituto Italiano di Tecnologia (IIT) in Genoa, Italy. The ETH Zurich owns a copy, the HyQ-blue, of the original HyQ. Both robots, shown in figure 1.1, are four legged robots capable of versatile, robust and dynamic maneuvers. The robot's versatility, such as reflex generation [17], reactive controller framework [5], and trotting assisted by stereo-vision [6], has been shown in many publications. The robot is both position and torque controlled [12], the latter enabling the use of many control concepts that produce torque outputs, such as impedance control [16]. Force or torque[1] commands are also used in rigid body inverse dynamics feedforward controllers and for the robot's balance where the leg contact points are not moved, but forces are introduced to keep the robot in balance. The interaction with the environment occurs via forces, thus force controlled robots allow for compliant behavior [10].

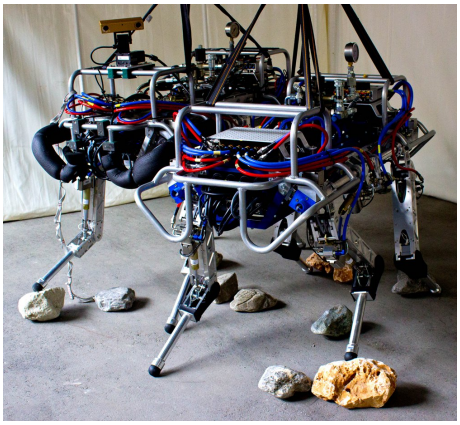

Figure 1.1: The two robots HyQ (IIT) and HyQ-blue (ETH) [24]. These are hydraulically quadruped robots capable of highly versatile, robust and dynamic maneuvers.

All these mentioned concepts, among others, depend on the performance of the low-level torque controller. A better-performing low-level controller maximizes the effectiveness of these higher-level control concepts. Therefore, a *perfect* torque source is desired, that is, a low-level controller able to track a torque reference in any circumstance.

[1] In this thesis the terms *force* and *torque* are used interchangeably. On HyQ the actuator is rigidly attached to the joint, hence the torque is equal to the force multiplied by the lever arm.

However, providing such a *perfect* torque source is a difficult task and cannot always[2] be fulfilled, especially when the system dynamics are changing during operation. This is the case when the robot HyQ performs, for example, a walking task, its torque controlled legs need to support the full weight of the robot in stance phase and are only subjected to their own weight during flight phase while tracking a torque reference. Thus, the load conditions for a leg change quite heavily and rapidly, that is, the system dynamics change accordingly. Experiments on the real-time force control one-dimensional (FC1D) test set-up were carried out in order to assess the impact of such changes in load conditions. More details about this test set-up can be found in appendix D.

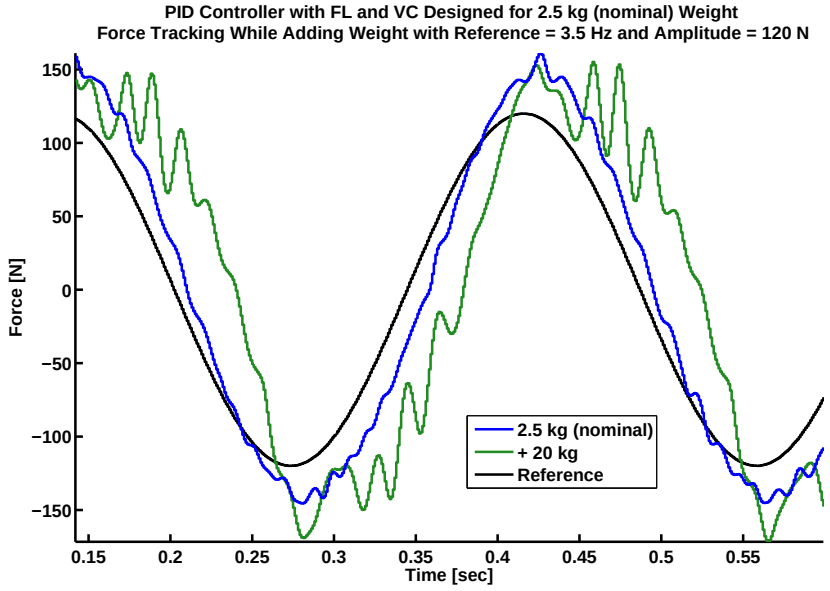

Figure 1.2: Experimental results exemplifying the impact of changing system dynamics on the force tracking performance of a *state-of-the-art* feedback linearization controller [12]. The blue line shows the response of the controller designed for a system having load mass of 2.5 kg. Whereas the green line shows the decrease in performance using this very same controller, but subjected to a system having an added mass of 20 kg. It is evident that this fixed gain controller is not able to give consistent force tracking performance, thus a controller that adapts itself to the environment is seeked for.

Figure 1.2 shows the force tracking of a *state-of-the-art* feedback linearization controller [12] for a sinusoidal force reference signal (black line). The blue line shows the force tracking[3] for the controller designed for a system having load mass of 2.5 kg. The system dynamics are then altered by adding 20 kg of additional weight to the test set-up while keeping the controller gains the same. This added weight imitates the conditions of a leg in stance phase. The green line shows the force tracking of this fixed gain controller. As can be seen, the force tracking performance using the same controller decreases when adding weight to the test set-up, that is, when the system dynamics change. In general, fixed gain controllers

[2] Actually, it can never be fulfilled. However, an engineer's interpretation of the word *perfect* is anticipated.

[3] Note that this experiment does not try to demonstrate the force tracking capabilities, it only exemplifies the impact of changing system dynamics. High-performance force tracking of this *state-of-the-art* feedback linearization (FL) controller is shown in [12].

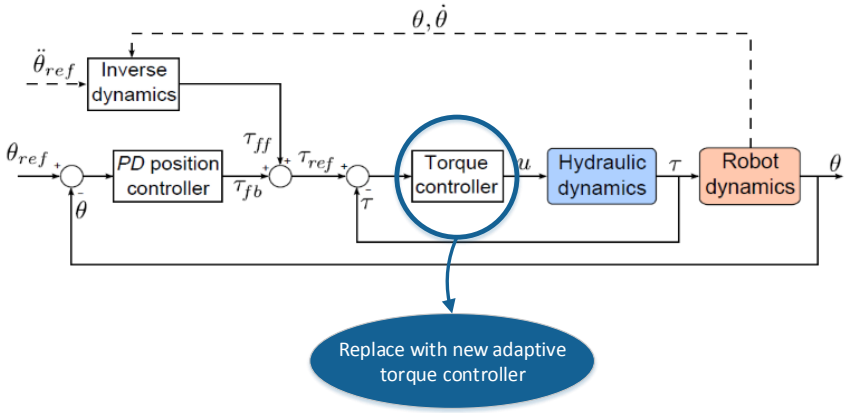

Figure 1.3: The HyQ control architecture with outer position controller implemented as a proportional derivative (PD) controller and inner torque controller along with a feedforward term using rigid body inverse dynamics. The inner torque controller receives a torque reference $\tau_{ref} = \tau_{ff} + \tau_{fb}$ to be tracked using its controller output u acting on the system (robot). The measured joint position θ and velocity $\dot{\theta}$ are fed back and used along with the joint acceleration reference $\ddot{\theta}_{ref}$ to calculate the feedforward torque needed. The inner fixed gain torque controller shall now be replaced by a new adaptive torque controller developed in this thesis. The diagram was copied from [10].

are not able to give consistent performance when the system dynamics change, thus a low-level controller is seeked for that adapts itself to the environment.

Therefore, this work presents an adaptive control framework to cope with changing system dynamics by recursively estimating the current system dynamics and updating the low-level controller when needed.

1.1 The HyQ Control Architecture

The control architecture of the robot HyQ, as shown in figure 1.3, consists of two main loops; an outer position loop and an inner torque loop [10, 12]. The outer position proportional derivative (PD) controller tracks the joint angle references θ_{ref} and commands a torque τ_{fb} to be applied to the joint. The reference input $\tau_{ref} = \tau_{ff} + \tau_{fb}$ to the inner torque controller is the sum of the outputs of the position controller plus the feedforward torque computed by the inverse dynamics. This inverse dynamics block uses a rigid body dynamics model of the robot along with joint acceleration references $\ddot{\theta}_{ref}$ and measurements of joint position θ and velocity $\dot{\theta}$. The torque controller tracks the desired torque reference by acting on the hydraulic system (blue box) with its output u. The hydraulic system consists of the servo-valve and hydraulic cylinder which is moved according to the command u, which results in movements of the rigid body parts of the robot represented by the red block. Measurements of the joint position θ and velocity $\dot{\theta}$ are fed back and the control cycle starts again, running at 1 kHz.

The new adaptive control framework developed in this thesis leaves the HyQ control architecture unchanged, it simply replaces the inner torque controller by a new adaptive torque controller with the goal of improving the overall control performance.

1.2 Literature Review

This thesis is based on the findings of the PhD thesis [10]. This thesis explains the hydraulic system in detail, introduces the concept of load velocity feedback [11], and shows how this concept can be used to improve the FL torque controller performance [12]. Related work regarding the robot HyQ and its capabilities was already mentioned at the beginning of this chapter. A similar capable robot is, for example, the StarlETH [23], which is, however, electrically actuated.

A good overview about robot force control can be found in [41]. This study characterizes two categories of adaptive control concepts, that is, the direct adaptive control and the indirect adaptive control. The former method applies an adaptive scheme to the controller gains such that these are self-adjusting. The latter explicitly estimates unknown system parameters and uses these to update the controller gains.

A control concept belonging to the former category is, for example, the model reference adaptive control concept. Its idea is to continuously update the controller gains such that the closed loop performance matches that of the reference model. An early implementation of this classical approach can be found in [40], where the controller parameters are directly identified by using a least squares algorithm.

An example of an indirect adaptive method can, for example, be found in [26], where they use a nonlinear controller derived via backstepping. The unknown system parameters are linear and estimated using a recursive least squares method. These identified parameters are then used to update the nonlinear controller equations. However, only simulation results are shown and the framework is very much tailored to the system.

Our approach discussed in detail in chapters 5, 6, and 7, belongs to the indirect adaptive control category.

Additional references to related works are given within the respective sections.

1.3 Goals and Work Packages

The goals of this thesis are to design, implement and test a new adaptive control framework for the low-level torque controller used on the robot HyQ. Figure 1.4 shows these and additional goals. Based on these goals, the following work packages were defined, shown in figure 1.5. The system used in this thesis is a one-dimensional force test set-up (FC1D), which is first analysed and simulated. The adaptive control framework is outlined and designed, then tested in simulation and afterwards implemented on the real-time hardware. This first part is done at the ETH Zurich, the second part, which involves experiments on the real-time hardware, is done at the IIT in Genoa. The initial plan was to also use the rotary actuator test set-up if the FC1D experiments deliver satisfactory results. The plan was then to generalize the idea to be implemented on a leg test set-up (HyL) in order to prepare implementation and experiments on the full HyQ robot, which was planned to be done again at ETH Zurich. Unfortunately, this plan proved to be too ambitious, thus only preliminary results using the FC1D test set-up at IIT were performed. The generalization of the code to the leg test set-up and the HyQ robot are part of future work.

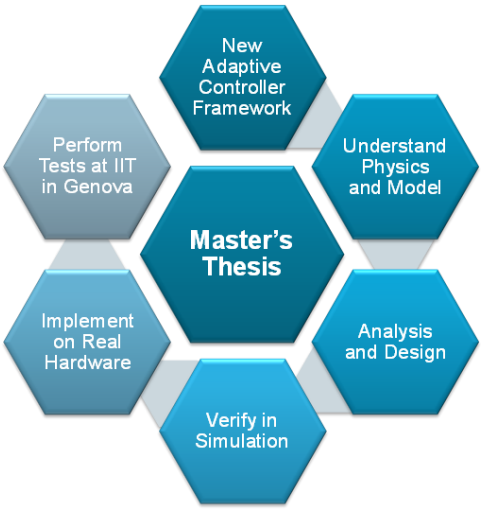

Figure 1.4: Master's thesis goals

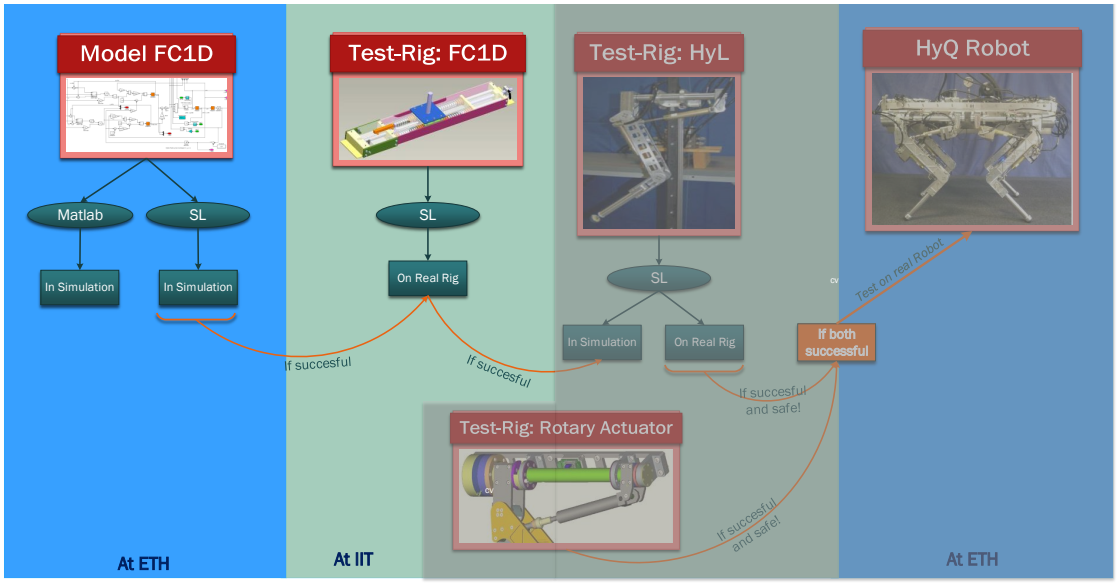

Figure 1.5: Master's thesis work packages and overview of project plan. The gray parts of the figure correspond to the work packages that were planned but have not yet been realized.

1.4 Outline

The hydraulic cylinder system and the derivation of the nonlinear model are shown in the next chapter. Chapter 3 discusses the simplified linear models. Chapter 4 then analyses these three presented models, followed by chapter 5 illustrating the adaptive control framework. The estimation part is discussed in more detail in chapter 6 followed by the discussion of the controller within chapter 7. Implementation details are shown in chapter 8, whereas experimental data is shown in chapter 9, followed by the conclusion (chapter 10).

Chapter 2

Nonlinear Model of the Hydraulic System

The hydraulic system set-up is sketched and all the relevant variables are defined. Then the load mass dynamics, hydraulic cylinder dynamics, valve and load cell dynamics are explained and a nonlinear model is derived.

2.1 System Layout and Specification

The two systems, the hydraulic system and the load mass system, are first discussed as independent systems. In subsection 2.1.1 it is then shown how to interconnect these systems. The hydraulic system consists of an asymmetric double acting cylinder with a valve which controls the flow of fluid between the two chambers. Such an asymmetric cylinder with valve is shown in figure 2.1. A piston inside the cylinder creates these two chambers. The piston moves due to the applied pressure difference across the two chambers. A single rod is rigidly attached at the right side of the piston, which can be used to connect a load. Since there is only one rod, the volume of chamber A (left side) is larger than the volume of chamber B (right side) because the volume of the rod reduces the volume the fluid can displace; therefore, the cylinder is called asymmetric. The pressure inside the two chambers is controlled by a valve. The valve consists of a spool which opens or closes the ports connecting the pressure source (or reservoir) and the two chambers.
The load mass system, shown in figure 2.2, consists of a mass which is guided along the direction of the cylinder stroke (in the above figure 2.1 the mass would be restricted to only move in horizontal direction). This guidance, for example, could be achieved by a cart mounted via an axial bearing on a cylindrical shaft. The bearings introduce friction acting in opposite direction of motion. In addition, springs are considered which are placed between cart and end-stop on both sides. The springs and friction terms can be understood as modeling stiffness and damping of the environment the cylinder load mass system is operated in. Figure 2.2 depicts the described linear load mass system set-up. The springs are intentionally added to the modeling to be able to capture the dynamics of the linear FC1D test set-up, while knowing that the HyQ robot does not have any springs. The springs can easily be deactivated later on by setting the corresponding terms to zero. One might ask why there are springs on the test set-up in the first place, whereas the real robot does not have any. This is due to the fact that in case of the real robot or leg, gravity is the

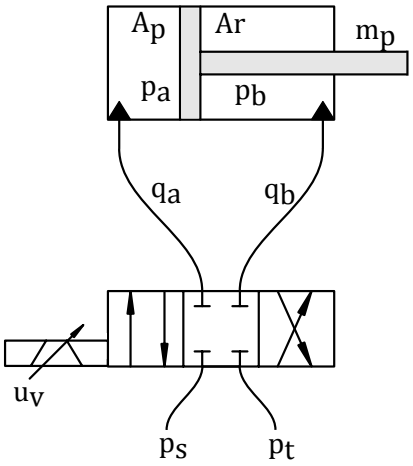

Figure 2.1: Schematic of an asymmetric double acting hydraulic cylinder (top) with valve and solenoid (bottom). The variables are defined in chapter 2.

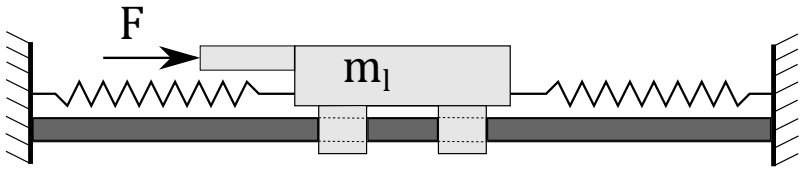

Figure 2.2: Schematic of a load mass m_l system mounted on a shaft by axial bearings. Springs are placed between cart and end-stop on both sides. This drawing represents the basic mechanic design of the linear FC1D test set-up. The cart can be pushed via the external force F.

potential force. Since the linear FC1D test set-up only moves horizontally, the influence of gravity is replaced by springs[1].

2.1.1 Combined System: Hydraulic Cylinder and Load Mass

The system considered in this chapter consists of the combination of the two systems introduced above, that is, the complete linear FC1D test set-up. The end of the cylinder rod is connected to the load mass and a load cell (force sensor) is mounted in between. In most cases, the behavior of the combined system is of interest; however, for modeling purposes, the combined system is cut in two parts in order to derive the equations of motion and to reveal the internal forces. Figure 2.3 shows the combined system already cut open and all relevant dimensions and forces are drawn. It is assumed that only the load m_l and the piston m_p have mass, the rod and load cell connecting load to cylinder are considered to be an *ideal* connection, thus infinitely stiff and without mass. Each mass is associated with an unique coordinate; however, since the connection between the two masses is *ideal*, the coordinate of the load mass x_l is determined by the coordinate of the piston mass x_p, thus a single variable is sufficient to fully describe the motion of this combined system[2].

[1] Of course, the potential of the springs and gravity is not the same. However, both allow steady-state force reference signals to be commanded

[2] It is assumed that the piston and load can only move in the direction of the stroke L_c (horizontally), which is a legitimate assumption since the real system is guided by two rails effectively locking the other two degrees of freedom.

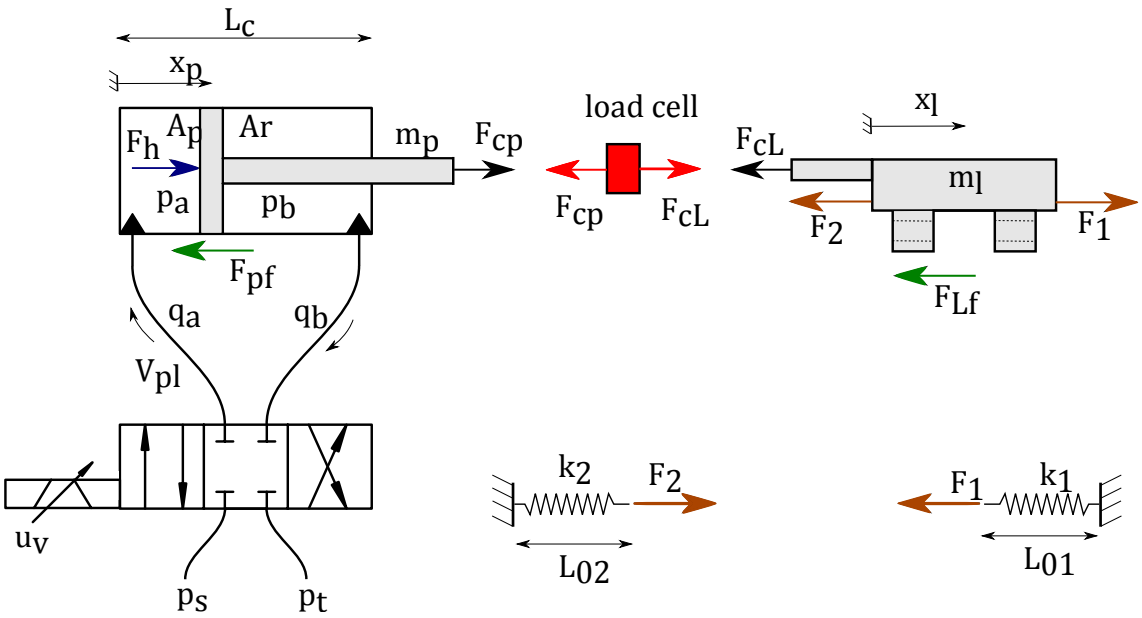

Figure 2.3: Free-body diagram of the combined system consisting of an asymmetric double acting hydraulic cylinder (left), valve and solenoid (bottom left), load mass (right) with springs, and a load cell (force sensor) in between. The piston coordinate is x_p, the load coordinate is x_l. Load cell and connecting rod are assumed to be massless. The internal forces F_{cp} and F_{cL} appear due to cutting open the connecting rod just before and after the load cell. The variables are defined in chapter 2.

The friction is drawn as a force acting in opposite direction of motion, determined by any general nonlinear friction model, noted as F_{pf} for the piston friction and F_{Lf} for the load mass friction. The springs are attached to the load mass and environment and can consist of any nonlinear spring model, with F_1 and F_2 being the two spring forces and L_{01}, L_{02} denoting their unstressed length, respectively. The load cell is assumed to be massless (and thus does not need a coordinate) and its resonance frequency is at around 12 kHz[3], which means that its relevant dynamics are at frequencies one decade higher than those of the dynamics of the cylinder load mass system. Therefore, the sensor dynamics can be neglected. The internal forces F_{cp} and F_{cL} appear due to cutting open the connecting rod just before and after the load cell. The hydraulic force acts on the piston and is positive in positive x_p direction. The fluid flowing into chamber A is denoted by q_a and is positive denoted by the arrow beneath, the flow out of chamber B is denoted by q_b and positive when flowing out, denoted by the corresponding arrow beneath. All other dimensions are listed in table A.1 of appendix A.

2.2 Load Mass Dynamics

In order to model the load mass system sketched on the right of figure 2.3, we first define the system boundary to enclose the cart which is represented by the mass m_l. Thus all forces acting on the cart (or mass m_l) are external forces. Then we write down the forces due to the springs and damping. The springs are modeled as an *ideal* spring, having stiffness,

[3] This is the resonance frequency of the load cell in isolation. Value according to e-mail (3.12.2013) from Mr. Hans Joachim Legat, product engineer at Burster Präzisionsmesstechnik GmbH & co kg.

but no mass. The friction model considered in this work is assumed to consist only of velocity-dependent friction d_l. Thus we can write:

$$F_1 = k_1 \cdot (L_{vsp} - x_l - L_0) \tag{2.1}$$

$$F_2 = k_2 \cdot (x_l - L_0) \tag{2.2}$$

$$F_{Lf} = d_l \cdot \dot{x}_l \tag{2.3}$$

where k_1 and k_2 are the constant spring stiffnesses and d_l is the friction coefficient of the bearings. L_0 is the free length of the springs (assumed to be equal for all springs $L_0 = L_{01} = L_{02}$) and L_{vsp} denotes the pre-compressed length (distance between left and right end-block) of the setup via a movable end-block on the right. The equation of motion is then given by the principle of linear momentum[4], where a constant mass is assumed. The complete equation of motion is then given by:

$$m_l \cdot \ddot{x}_l = -F_{cL} + k_1 \cdot (L_{vsp} - x_l - L_0) - k_2 \cdot (x_l - L_0) - d_l \cdot \dot{x}_l \tag{2.4}$$

where m_l is the load mass and x_l defines the position of the load with respect to the very left mounting point of the spring. F_{cL} is the force applied to the load mass through the load cell. The equilibrium (or steady-state) position x_{l0} of the cart is determined by the equation:

$$x_{l0} = \frac{k_2 \cdot L_0 + k_1 \cdot (L_{vsp} - L_0)}{k_1 + k_2} \tag{2.5}$$

By defining a new coordinate $y = x_l - x_{l0}$ which describes the position measured from steady-state equilibrium, we can simplify the above equation (2.4), such that:

$$m_l \cdot \ddot{y} + d_l \cdot \dot{y} + (k_1 + k_2) \cdot y = -F_{cL} \tag{2.6}$$

with the new equilibrium position of $y_0 = 0$.

2.3 Hydraulic Cylinder Dynamics

The piston of the hydraulic cylinder has a mass m_p and a coordinate x_p associated with it. The system boundary encloses the piston and rod. Using the same concept as in section 2.2, the equation of motion for the piston is given by:

$$m_p \cdot \ddot{x}_p = -F_{pf} + F_h + F_{cp} \tag{2.7}$$

where m_p is the mass of the piston. The friction force is denoted by F_{pf}. The force F_{cp} is the force applied to the load cell. The hydraulic force F_h results from the pressure difference inside the cylinder, thus:

$$F_h = A_p \cdot p_a - A_r \cdot p_b = A_p \cdot (p_a - \alpha \cdot p_b) \tag{2.8}$$

where A_p is the area of the piston without the rod and A_r is the area of the piston with the rod, which is smaller than the other side by a factor of $\alpha = A_r/A_p$.

[4]Which states that the change of momentum is the result of the sum of all external forces acting on the mass: $(m \cdot v)' = \sum F^a$

The pressure dynamics in chamber A and B depend on the bulk modulus of the fluid (in our case the fluid is oil), the volume of the chamber and the total fluid trapped inside. According to fluid dynamics the pressure equation is given by [25]:

$$\dot{p} = \frac{\beta_e}{V} \cdot (q_{in} - q_{out} - \dot{V}) \tag{2.9}$$

where q_{in}, q_{out} are the in- and outgoing flows and $\dot{V}$ is the flow due to changing volume. In our case the general equations for the pressures in the two chambers are as follows [25]:

$$\dot{p}_a = \frac{\beta_e}{V_a} \cdot (q_a - A_p \cdot \dot{x}_p) \tag{2.10}$$

$$\dot{p}_b = \frac{\beta_e}{V_b} \cdot (-q_b + \alpha \cdot A_p \cdot \dot{x}_p) \tag{2.11}$$

where β_e is the effective bulk modulus of the oil. V_a and V_b are the volumes of the chamber A and B, respectively. The fluid flow of the chambers are denoted by q_a (positive inflow) and q_b (positive outflow). The term $A_p \cdot \dot{x}_p$ is the amount of volume that changes due to the motion of the piston. The volumes V_a and V_b thus depend on the position of the piston. These chamber volumes are given by [25]:

$$V_a = V_{pl} + x_p \cdot A_p \tag{2.12}$$

$$V_b = V_{pl} + (L_c - x_p) \cdot \alpha \cdot A_p \tag{2.13}$$

where V_{pl} is the pipe line volume, given by its length and diameter: $V_{pl} = L_{pl} \cdot \pi \cdot (D_{pl})^2/4$. The chamber flows q_a and q_b depend on the valve properties and on the pressure difference between input and output port. These are determined by the valve spool position x_v or equivalent position u_v (see section 2.4). The valve switches for each chamber between supply pressure p_s and reservoir pressure p_t. For chamber A, the flow equation is [25]:

$$q_a = \begin{cases} K_v \cdot |x_v| \cdot \sqrt{p_s - p_a} & x_v > 0 \\ -K_v \cdot |x_v| \cdot \sqrt{p_a - p_t} & x_v < 0 \end{cases} \tag{2.14}$$

and for chamber B, the flow equation is:

$$q_b = \begin{cases} K_v \cdot |x_v| \cdot \sqrt{p_b - p_t} & x_v > 0 \\ -K_v \cdot |x_v| \cdot \sqrt{p_s - p_b} & x_v < 0 \end{cases} \tag{2.15}$$

where the valve gain is denoted by K_v and is calculated from data sheet information. It is the nominal flow divided by the nominal input multiplied by the nominal pressure drop Δp_n across two ports, thus:

$$K_v = \frac{q_n}{x_{vn} \cdot \sqrt{\frac{\Delta p_n}{2}}} \tag{2.16}$$

where the nominal valve input x_{vn} is used. The friction force F_{pf} can be modeled by any nonlinear fiction model. In this work, only a viscous friction model is used, where the friction is proportional to the piston velocity. The viscous friction coefficient is denoted by d_p.
The full equation of motion for the hydraulic cylinder is thus given by:

$$m_p \cdot \ddot{x}_p + d_p \cdot \dot{x}_p = A_p \cdot (p_a - \alpha \cdot p_b) + F_{cp} \tag{2.17}$$

2.4 Valve Spool Dynamics

The servo-valve consists of a solenoid that moves a spool back and forth, thus blocking or opening the valve channels. The currently used servo-valve accepts current as input and results in position of the spool, which determines how much oil can flow through the valve and orifice. The dynamics of the spool can be modeled as a second-order system which has the following transfer function (for more details on the valves used on HyQ see [10]) from the input current u to the spool position output x_v:

$$\Delta x_v(s) = \frac{K_{sp}}{\frac{1}{\omega_v^2}s^2 + \frac{2 \cdot D_v}{\omega_v} \cdot s + 1} \Delta u(s) \tag{2.18}$$

where,

K_{sp} : steady-state valve input-to-spool position gain [m/A]

ω_v : valve spool angular frequency [rad/s]

D_v : valve spool damping [−]

The position of the spool of the servo-valve used on the HyQ robot cannot be measured, as it is the case for most servo-valves. Therefore, we seek a way to express the valve spool position in terms of the input signal which is measurable and has the unit ampere. Dividing the above equation (2.18) by K_{sp} results in:

$$\Delta u_v(s) :\equiv \frac{\Delta x_v(s)}{K_{sp}} = \frac{1}{\frac{1}{\omega_v^2}s^2 + \frac{2 \cdot D_v}{\omega_v} \cdot s + 1} \Delta u(s) \tag{2.19}$$

where a new output $\Delta u_v(s)$ of the valve dynamics is defined which has the same unit [A] as the input $\Delta u(s)$. This new valve output u_v can be seen as a filtered version of the actual valve input u, capturing in essence the lag introduced by the valve spool dynamics.
The advantage of this transfer function approach is that the complexity of the model can be varied. A more complete model would then simply consist of multiplying the above valve transfer function (2.18) with the model transfer function. Thus the combined system can be analysed in terms of u or u_v as an input, depending on the complexity needed. Of course, the valve gain K_v (2.16) needs to be expressed in the proper units (as given in appendix A.3). The valve gain using the equivalent valve output u_v is now given by:

$$K_v = \frac{q_n}{u_{vn} \cdot \sqrt{\frac{\Delta p_n}{2}}} \tag{2.20}$$

where the equivalent valve output u_v along with the equivalent nominal valve input u_{vn} is used. For the remainder of this work, the valve spool dynamics are not included in the analysis since its dynamics are fast enough to be neglected. In the following, the valve spool position is thus represented by the equivalent valve output: $u_v = x_v$.

2.5 Modeling the Load Cell

The load cell is a force sensor which typically consists of a membrane perpendicular to the direction of the force. On the inner side of this membrane, several strain gauges are applied.

External forces are induced via the casing into the membrane whose deformation the strain gauges measure and an appropriate voltage signal is generated. The load cell used on HyQ is made by Burster GmbH[5] and is sold under the description *type* $8417 - 6005$. This is a sub-miniature force sensor rugged in construction and designed to measure forces up to 5000 N according to [14].
The load cells resonance frequency is at around 12 kHz[6], which means that its relevant dynamics are at frequencies one decade higher than those of the hydraulic system we consider. We therefore neglect the dynamics of the load cell, as already mentioned in section 2.1.1, and assume it to be massless and infinitely stiff. From the first assumption, it immediately follows from figure 2.3 that a simple force balance must hold:

$$F_{cp} = -F_{cL} \tag{2.21}$$

Not only the casing, but also the membrane introduces some dynamics, which we again neglect since the load cell is assumed to be infinitely stiff. We can thus simplify the force measured by assuming that we can exactly measure the applied force, thus we define the sensor-force as $F_s = -F_{cp}$ where a unit measurement gain is assumed. The equation describing the force introduced to the membrane is either equation (2.6) or (2.17), depending on the point of view. With these two equations, the load cell force measurement is then given by:

$$\begin{aligned} F_s &= F_h - d_p \cdot \dot{x}_p - m_p \cdot \ddot{x}_p && \text{(left side)} \\ &= (k_1 + k_2) \cdot y + d_l \cdot \dot{y} + m_l \cdot \ddot{y} && \text{(right side)} \end{aligned} \tag{2.22}$$

As can be seen, the load cell measures not only the hydraulic force, but also the forces due to friction and inertia. This equation (2.22) usually confuses people since one would expect only one equation. The fact that we chose to cut free the complete system made it necessary to define two system boundaries for the left system and for the right system shown in figure 2.3. Due to Newton equations (the momentum balance), the previous internal forces acting on the load cell are now external forces for these two systems and thus they appear in both equations. When connecting the two systems back together, these forces become internal forces again and disappear (as will be seen in the section below). Hence, we know that the load cell force measurement permits us to treat either side as an independent system. This means that if we know the load cell force, we can completely ignore one system, for example the left hydraulic cylinder system, and still be able to fully describe the dynamics of the other system, here the right load mass system, using the load cell force measurement.

2.6 Hydraulic Cylinder and Load Mass

Equation (2.17) is the equation of motion for the piston and equation (2.6) is the equation of motion for the load mass. As mentioned in section 2.1.1, these two equations can be merged into one single equation describing the motion of the combined system[7]. Without

[5] Burster Präzisionsmesstechnik GmbH & co kg, `www.burster.de`

[6] This is the resonance frequency of the load cell in isolation. Value according to the e-mail (3.12.2013) from Mr. Hans Joachim Legat, product engineer at Burster Präzisionsmesstechnik GmbH & co kg.

[7] More precisely, we define the system boundary to include the piston and the mass, thus the whole assembly is the system and all previously internal forces disappear since the Newton equations are defined in terms of external forces.

loss of generality, the load mass in equilibrium is attached to the connecting rod such that the piston is in the middle of its stroke $x_{p0} = L_c/2$, thus it holds that:

$$x_p = y + L_c/2 \tag{2.23}$$

Therefore, the combined system equation is derived by solving equation (2.17) for the constraint force F_{cp} and inserting this force in equation (2.6) by using equation (2.21). After rearranging and combining the common terms, we can write the equation of motion for the combined system as:

$$(m_l + m_p) \cdot \ddot{y} + (d_l + d_p) \cdot \dot{y} + (k_1 + k_2) \cdot y = A_p \cdot (p_a - \alpha \cdot p_b) \tag{2.24}$$

Clearly, the two friction terms and the masses are lumped together and they both influence the dynamics of the overall system. The cut forces, which are internal forces, have disappeared and are not visible when considering the combined system. It is important to realize that the forces acting on the load cell are internal forces and cannot be determined from the above equation (2.24). With the definition of some lumped parameters, the equation can be written in a shorter form; these coefficients are:

$$M = (m_l + m_p) \tag{2.25}$$

$$B = (d_l + d_p) \tag{2.26}$$

$$C = (k_1 + k_2) \tag{2.27}$$

The above equation of motion (2.24) takes a simpler form given by:

$$M\ddot{y} + B\dot{y} + Cy = A_p(p_a - \alpha p_b) \tag{2.28}$$

All parameters used for the modeling are listed in table (A.1) in appendix A.3 along with their nominal values and units.

2.7 Model in State Space Form

When considering again figure 2.3, it is clear that there are at least three relevant reservoirs storing energy. These reservoirs and their associated level variables are: the combined mass with level variable velocity $\dot{y}$ (kinetic energy), pressure in chamber A with level variable p_a (potential energy), and pressure in chamber B with level variable p_b (potential energy). Additional reservoirs are, for example, the kinetic energies of the valve and of the load cell; however, these dynamics are fast compared to the dynamics of the combined system, therefore these can be neglected[8]. The dynamics of the level variable velocity $\dot{y} \equiv \dot{x}_p$ are determined by the equation of motion, which depends on the load position y, or using equation (2.23), on the piston position x_p. The position x_p is therefore additionally included as a level variable. Thus, we define the state vector as:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_p = \text{position of piston} \\ \dot{x}_p = \text{velocity of piston} \\ p_a = \text{pressure inside chamber A} \\ p_b = \text{pressure inside chamber B} \end{pmatrix} \tag{2.29}$$

[8] As already mentioned within section 2.5.

where the position variable x_p is used instead of y. This is done because the pressure dynamics, chamber volumes and flow equations were derived above in terms of x_p.

Considering the simplification mentioned in section 2.4, which relates x_v to u_v, we can now write the full nonlinear state space equation using (2.28), (2.10), (2.11) along with (2.12), (2.13), (2.14) and (2.15) as follows:
for $u_v > 0$ we have:

$$\ddot{\vec{x}} = f^+_{u_v}(\vec{x}, u_v) \rightarrow \begin{pmatrix} \dot{x}_p \\ \ddot{x}_p \\ \dot{p}_a \\ \dot{p}_b \end{pmatrix} = \begin{bmatrix} \dot{x}_p \\ \frac{1}{M}\left(-B\dot{x}_p - C(x_p - \frac{L_c}{2})\right) + \frac{A_p}{M}(p_a - \alpha p_b) \\ \frac{\beta_e}{V_{pl}+x_p A_p}\left(K_v|u_v|\sqrt{p_s - p_a} - A_p\dot{x}_p\right) \\ \frac{\beta_e}{V_{pl}+(L_c-x_p)\alpha A_p}\left(-K_v|u_v|\sqrt{p_b - p_t} + \alpha A_p\dot{x}_p\right) \end{bmatrix} \tag{2.30a}$$

and for $u_v < 0$:

$$\ddot{\vec{x}} = f^-_{u_v}(\vec{x}, u_v) \rightarrow \begin{pmatrix} \dot{x}_p \\ \ddot{x}_p \\ \dot{p}_a \\ \dot{p}_b \end{pmatrix} = \begin{bmatrix} \dot{x}_p \\ \frac{1}{M}\left(-B\dot{x}_p - C(x_p - \frac{L_c}{2})\right) + \frac{A_p}{M}(p_a - \alpha p_b) \\ \frac{\beta_e}{V_{pl}+x_p A_p}\left(-K_v|u_v|\sqrt{p_a - p_t} - A_p\dot{x}_p\right) \\ \frac{\beta_e}{V_{pl}+(L_c-x_p)\alpha A_p}\left(K_v|u_v|\sqrt{p_s - p_b} + \alpha A_p\dot{x}_p\right) \end{bmatrix} \tag{2.30b}$$

On the real test set-up, we can measure the position and velocity of the load mass by a linear magnetic encoder system. The sensors on HyQ provide measurements of the same information. The load cell force measurements are also available, but no pressure measurement. Acceleration could also be provided through differentiation. The measurement equation, using (2.22), is given by:

$$\vec{y} = \begin{pmatrix} x_p \\ \dot{x}_p \\ F_s \end{pmatrix} = \begin{pmatrix} x_p \\ \dot{x}_p \\ \begin{cases} F_h - d_p \cdot \dot{x}_p - m_p \cdot \ddot{x}_p \\ (k_1 + k_2) \cdot y + d_l \cdot \dot{y} + m_l \cdot \ddot{y} \end{cases} \end{pmatrix} \tag{2.31}$$

2.8 Hydraulic Force Dynamics

The hydraulic force given by equation (2.8) is a function of the pressure difference between the two chambers. We are now interested in its dynamic behavior, thus enabling us to see what factors influence the hydraulic force. This section presents the most important formulas of the study already done in [10]. The equation describing the dynamics is obtained by taking the derivative with respect to time:

$$\dot{F}_h = A_p \cdot (\dot{p}_a - \alpha \cdot \dot{p}_b) = \frac{A_p \beta_e}{V_a}(q_a - A_p \cdot \dot{x}_p) - \frac{\alpha A_p \beta_e}{V_b}(-q_b + \alpha \cdot A_p \cdot \dot{x}_p) \tag{2.32}$$

We can bring this formula into a simpler version by defining transmission stiffness coefficients for the two chambers:

$$\dot{F}_h = K_{tha} \cdot q_a + K_{thb} \cdot q_b + A_p \dot{x}_p (-K_{tha} - \alpha K_{thb}) \tag{2.33}$$

with

$$K_{tha} = \frac{A_p \cdot \beta_e}{V_a} \quad [\text{Pa/m}] \quad \text{hydraulic transmission stiffness for chamber A} \tag{2.34}$$

$$K_{thb} = \frac{\alpha \cdot A_p \cdot \beta_e}{V_b} \quad [\mathrm{Pa/m}] \quad \text{hydraulic transmission stiffness for chamber B} \tag{2.35}$$

these two coefficients are a function of time since the chamber volumes V_a and V_b change with the piston position according to equations (2.12) and (2.13).
By defining yet another set of coefficients, the equation can be written even simpler. The coefficients are:

$$K_{th} = K_{tha} + K_{thb} = A_p \cdot \beta_e \cdot \left(\frac{1}{V_a} + \frac{\alpha}{V_b}\right) \tag{2.36}$$

$$q_e = \frac{K_{tha} \cdot q_q + K_{thb} \cdot q_b}{K_{tha} + K_{thb}} = \frac{V_b \cdot q_a + \alpha \cdot V_a \cdot q_b}{V_b + \alpha \cdot V_a} \tag{2.37}$$

$$A_e = \frac{A_p}{K_{th}} \left(K_{tha} + \alpha \cdot K_{thb}\right) = A_p \cdot \frac{V_b + \alpha^2 \cdot V_a}{V_b + \alpha \cdot V_a} \tag{2.38}$$

The above equation (2.33) can be simplified to:

$$\dot{F}_h = K_{th} \cdot (q_e - A_e \cdot \dot{x}_p) \tag{2.39}$$

which describes the dynamics of the hydraulic force driven by an equivalent flow q_e and a change in volume through an equivalent area A_e and the piston velocity $\dot{x}_p$. The difference of equivalent flow minus change in volume can be considered as the effective flow. This effective flow multiplied with the transmission stiffness determines the propagation of the hydraulic force.
From this short notation (2.39), the influence of the load velocity[9] through the term $A_e \cdot \dot{x}_p$ is evident. By controlling the pressure via the input u_v and therefore the flow q_e, we create a hydraulic force acting on the mass which starts to move. Its movement is immediately fed back by the term $A_e \cdot \dot{x}_p$. Therefore, if one wants to control the hydraulic force, this effect has to be compensated for. In the remainder of the thesis, this observation is called *velocity feedback* [11] and the compensation strategy is called *velocity compensation*.

2.9 Total Force Dynamics

The total force is the effective force that is responsible for the acceleration of the load mass. According to the Newton principle of linear momentum, the right side of the equation represents the sum of all external forces. Despite the common misconception, this total force is not the force the load cell measures. There has to be a clear distinction between these two forces. The total force can be calculated from the equation of motion (2.28), where the change of the impulse is equal to the sum of all the forces, thus we have:

$$M \cdot \ddot{y} = \sum_i F_i :\equiv F_t \tag{2.40}$$

$$\rightarrow F_t = F_h - B \cdot \dot{y} - C \cdot y \tag{2.41}$$

The dynamics of the total force is again derived by taking the derivative with respect to time, thus $\dot{F}_t = \dot{F}_h - B \cdot \ddot{y} - C \cdot \dot{y}$. Inserting the hydraulic force equation (2.39) and changing the variables using equation (2.23), we can write the total force dynamics in terms of acceleration and velocity of the piston and the equivalent flow q_e as:

$$\begin{aligned} \dot{F}_t &= K_{th} \cdot (q_e - A_e \cdot \dot{x}_p) - B \cdot \ddot{x}_p - C \cdot \dot{x}_p \\ &= K_{th} \cdot q_e - (K_{th} \cdot A_e + C) \cdot \dot{x}_p - B \cdot \ddot{x}_p \end{aligned} \tag{2.42}$$

[9] Because of equation (2.23), the load velocity is the same as the piston velocity.

This equation clearly shows that the total force applied to the combined system is not only influenced by the friction and spring forces of the load mass itself, but also by the friction components of the cylinder system due to equation (2.26).
Comparing this equation (2.42) to the hydraulic force dynamic equation (2.39), it can be observed that the velocity feedback is extended by a term stemming from the springs and an additional acceleration feedback term is introduced via the friction of load and piston. Therefore, the velocity compensation strategy has to be extended.
However, the hydraulic force cannot be measured, only the load cell force F_s, thus using the equation (2.22), the total force dynamics becomes:

$$\begin{aligned}\dot{F}_t &= \dot{F}_s + d_p \cdot \ddot{x}_p + m_p \cdot \dddot{x}_p - B \cdot \ddot{x}_p - C \cdot \dot{x}_p \\ &= \dot{F}_s - C \cdot \dot{x}_p - d_l \cdot \ddot{x}_p + m_p \cdot \dddot{x}_p\end{aligned} \tag{2.43}$$

As can be seen, the velocity feedback from the hydraulic force dynamics has disappeared and only the feedback due to springs (C), load friction (d_l) and piston inertia (m_p) has remained. The velocity feedback, as seen in equation (2.39), is now internal to the force F_s. Thus, a controller using the load cell force measurement F_s already knows the influence of the velocity feedback seen in (2.39). Only the contributions of the load mass system are unknown and introduce feedback. However, the feedback term of equation (2.39) can be used within the controller as a feedforward compensator to achieve faster force tracking. Usually, the terms $\ddot{x}_p$ and $\dddot{x}_p$ are only available through differentiation of the velocity measurement and are therefore noisy. Since their magnitude is high only for a short duration and otherwise low or zero for steady state, they are being neglected. Thus only springs, if present, contribute to the feedback. The HyQ robot does not have springs, thus, as an approximation, one can see when controlling the load cell force, it is possible to fully control the total force dynamics, i.e:

$$\dot{F}_t = \dot{F}_s \tag{2.44}$$

Chapter 3

Linear Models of the Hydraulic System

Three different linear models are derived from the nonlinear model, which was explained in chapter 2. First, a full linear model with four states is obtained, then two simplified models requiring only three states are derived. The respective state space matrices are given for each model.

3.1 Full Linear Model

The set of nonlinear equations (2.30a) and (2.30b) can be linearized around an operating point for the state vector (defined by equation (2.29)) to enable the use of linear system analysis techniques such as transfer functions. A linear system description also allows the use of linear estimation techniques for identifying the parameters. The first step in linearization is to define the states, inputs and outputs. Secondly, an appropriate operating point must be chosen which should be a point where the system operates normally or most of the time. Finally, the linearized equations can be derived.
We use the same states as used by the nonlinear system given by (2.29), repeated here for convenience:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_p \\ \dot{x}_p \\ p_a \\ p_b \end{pmatrix} \tag{3.1}$$

We define the input as $\vec{u} = u_v$, where u_v is the input signal to the valve and, as before, the valve spool dynamics is neglected (for details see section 2.4).
The output vector is defined as:

$$\vec{y} = \begin{pmatrix} x_p \\ \dot{x}_p \\ F_s \end{pmatrix} \tag{3.2}$$

where F_s represents the force measured by the load cell. The equation (2.22) cannot be used directly due to the fact that acceleration terms appear which are not part of the state vector (3.1) used in this work. Since the mass of the piston m_p is usually much smaller than

the mass of the load m_l, its influence on the measured load cell force can be neglected[1], resulting in the load cell measurement equation given by:

$$F_s = F_h - d_p \cdot \dot{x}_p \tag{3.3}$$

A similar simplification for the other part of equation (2.22) is not possible because the mass m_l is typically high and thus the acceleration term cannot be neglected. By inserting the simplified load cell measurement equation (3.3) along with equation (2.8), the output vector can be written in terms of state variables only. This load cell force measurement should not be confused with the total force F_t or the hydraulic force F_h. The output vector is given by:

$$\vec{y} = \begin{pmatrix} x_p \\ \dot{x}_p \\ F_s \end{pmatrix} = \begin{pmatrix} x_p \\ \dot{x}_p \\ A_p\left(p_a - \alpha p_b\right) - d_p \cdot \dot{x}_p \end{pmatrix} \tag{3.4}$$

The nonlinear system (2.30a) and (2.30b) can now be written in the form:

$$\begin{aligned} \frac{d}{dt}\vec{x}(t) &= f(\vec{x}(t), u_v(t)) \\ \vec{y}(t) &= g(\vec{x}(t), u_v(t)) \end{aligned} \tag{3.5}$$

In the second step, the operating points for all states and inputs are defined so that they represent a typical system state. A typical state for our hydraulic cylinder is an equilibrium such that the piston is in the middle of its stroke. The corresponding operating points are thus:

- The position and velocity operating point is: $x_{p\oslash} = L_c/2$ and $\dot{x}_{p\oslash} = 0$.
- The pressure operating point is: $p_{a\oslash} = \frac{\alpha \cdot p_s}{1+\alpha}$ and $p_{b\oslash} = \frac{p_s}{1+\alpha}$. These expressions account for the different areas of the piston. With this choice, the piston remains stationary, thus the load pressure is zero: $p_l = p_a - \alpha p_b = 0$).
- The input operating point is chosen to be in the middle of its range, thus $u_{v\oslash} = 0$.
- The output operating point is derived by inserting the operating points of the states into the measurement equation, thus: $\vec{y}_\oslash = \begin{pmatrix} L_c/2 \\ 0 \\ 0 \end{pmatrix}$

3.1.1 Full Linear Model in State Space Form

The third and final step of the linearization consists of calculating the Taylor series expansion of the set of nonlinear equations (2.30a), (2.30b) around an operating point $x_{i\oslash}, u_\oslash, y_{i\oslash}$ (with i denoting the ith variable) and neglecting all terms of higher order than one. The above nonlinear system (3.5) is thus approximated by the linear system given in the form below as:

$$\begin{aligned} \frac{d}{dt}\delta\vec{x}(t) &= \mathcal{A} \cdot \delta\vec{x}(t) + \mathcal{B} \cdot \delta u(t) \\ \delta\vec{y}(t) &= \mathcal{C} \cdot \delta\vec{x}(t) + \mathcal{D} \cdot \delta u(t) \end{aligned} \tag{3.6}$$

[1] Assuming a maximum acceleration of the piston of about $10\,\mathrm{m/s^2}$ and a piston mass of at most 0.4 kg, the contribution of the piston inertia on the force measurement is only about 4 N. Compared to the maximum force of about 3200 N that our hydraulic cylinder can provide, this simplification is admissible.

where the matrices relating state and input are defined by:

$$\begin{aligned} \mathcal{A} &= \frac{\partial f}{\partial \vec{x}}\Big|_{\vec{x}=\vec{x}_\oslash, u=u_\oslash} & \mathcal{B} &= \frac{\partial f}{\partial u}\Big|_{\vec{x}=\vec{x}_\oslash, u=u_\oslash} \\ \mathcal{C} &= \frac{\partial g}{\partial \vec{x}}\Big|_{\vec{x}=\vec{x}_\oslash, u=u_\oslash} & \mathcal{D} &= \frac{\partial g}{\partial u}\Big|_{\vec{x}=\vec{x}_\oslash, u=u_\oslash} \end{aligned} \tag{3.7}$$

Considering the operating point defined above, the individual matrices are given by:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} & \frac{-\alpha A_p}{M} \\ 0 & \frac{-\beta_e A_p}{V_{pl}+\frac{L_c}{2}A_p} & 0 & 0 \\ 0 & \frac{\alpha\beta_e A_p}{V_{pl}+\frac{\alpha L_c}{2}A_p} & 0 & 0 \end{bmatrix} \tag{3.8a}$$

$$\mathcal{B} = \begin{bmatrix} 0 \\ 0 \\ \frac{\beta_e K_v \sqrt{p_s/(1+\alpha)}}{V_{pl}+\frac{L_c}{2}A_p} \\ \frac{-\beta_e K_v \sqrt{p_s/(1+\alpha)}}{V_{pl}+\frac{\alpha L_c}{2}A_p} \end{bmatrix} \tag{3.8b}$$

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -d_p & A_p & -\alpha A_p \end{bmatrix} \tag{3.8c}$$

$$\mathcal{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.8d}$$

where the matrices $\mathcal{B}, \mathcal{C}$ should not be confused with the total friction coefficient B and the total spring stiffness C of the combined system. Note that these linearized matrices are the same regardless of the sign of the input, meaning that the distinction present in the equations (2.30a) (for $u_v > 0$) and (2.30b) (for $u_v < 0$) is not needed anymore. However, one has to be careful, since this is only the case for this particular operating point.
Furthermore, it makes sense to normalize the system such that all variables remain in the same range, which reduces the potential of occurring numerical issues. The states are transformed by: $x = T \cdot \hat{x}$, the inputs with $u = V \cdot \hat{u}$ and the outputs with $y = W \cdot \hat{y}$. Where the variables with a "hat" denote the normalized variables.
The transformation matrix is chosen as:

$$T = \begin{bmatrix} \frac{L_c}{2} & 0 & 0 & 0 \\ 0 & 1.2\,\mathrm{m/s} & 0 & 0 \\ 0 & 0 & p_s & 0 \\ 0 & 0 & 0 & p_s \end{bmatrix} \tag{3.9}$$

$$W = \begin{bmatrix} \frac{L_c}{2} & 0 & 0 \\ 0 & 1.2\,\mathrm{m/s} & 0 \\ 0 & 0 & 5000\,\mathrm{N} \end{bmatrix} \tag{3.10}$$

$$V = 10\,\mathrm{mA} \tag{3.11}$$

where for each state the maximum expected magnitude is used such that the normalized variables remain in the range $[-1, 1]$. The piston can move half a stroke distance to either

side of its middle position, thus the normalization variable is $L_c/2$. The velocity normalization is set to an empirical value of 1.2 m/s. The pressure states are normalized using the supply pressure p_s. The output normalization matrix W uses the same values for position and velocity and for the load cell force measurement a value of $5000N$ is chosen since this value is the maximum force the sensor can measure. The input normalization matrix V consists of only one term since there is only one input to the system. The value is set to the maximum current of 10 mA the valve accepts.
The matrices for general linearizations of the nonlinear system for an arbitrary operating point defined by $(x_{p\oslash}, \dot{x}_{p\oslash}, p_{a\oslash}, p_{b\oslash}, u_\oslash, y_\oslash)$ are given in appendix A.1.

3.2 Simplified Linear Model

The linear model derived above (subsection 3.1.1) has one significant drawback, that is, the state vector (3.1) is not fully measurable since we do not measure the pressure inside the cylinder chambers. The linear model was derived by defining the relevant reservoir variables, resulting in the matrices (3.8a) through (3.8d). However, upon closer inspection, it can be seen that what drives the piston is just the pressure difference. We control the hydraulic cylinder only by applying pressure to both sides at the same time, governed by a single valve with a single spool. More clearly this means that if we choose the pressure for chamber A, it immediately fixes the pressure that has to be applied to chamber B. This is because if the valve spools opens port A by 10 %, then port B opens exactly the same amount (this relationship is only valid for symmetric valves). The two pressure reservoirs of the chambers A and B can thus be expressed as a new reservoir of the pressure difference. This simplification allows us to reduce the order of the state space system. In the following, the necessary assumptions are elaborated and the new equations are derived.

3.2.1 Derivation of Load Pressure Dynamics

Based on equation (2.8), we can define a new pressure $p_l = p_a - \alpha p_b$, called *load pressure*. This is the pressure that effectively results in the hydraulic force: $F_h = A_p \cdot p_l$. For the linearization we need the load pressure dynamics purely as a function of itself and possibly the other states, thus:

$$\dot{p}_l = f(p_l, x_p, \dot{x}_p, u_v) \tag{3.12}$$

where $f(\dots)$ is some function of the arguments. The following is derived by using [10] and [25].

Steady-State Pressure Equation

We need to relate the pressures p_a, p_b inside the two chambers to the load pressure p_l, supply pressure p_s and reservoir pressure p_t. This can be achieved by assuming a steady-state pressure condition which has the property that the pressure dynamics are zero:

$$\dot{p}_a = 0 \qquad \dot{p}_b = 0 \tag{3.13}$$

Inserting equation (2.10) and (2.11) results in:

$$q_a = A_p \dot{x}_p \qquad q_b = \alpha A_p \dot{x}_p \tag{3.14}$$

Inserting the flow governed by the equations (2.14) and (2.15) while considering the simplification mentioned in section 2.4, gives:

$$A_p\dot{x}_p = \begin{cases} K_v \cdot |u_v| \cdot \sqrt{p_s - p_a} & u_v > 0 \\ -K_v \cdot |u_v| \cdot \sqrt{p_a - p_t} & u_v < 0 \end{cases} \tag{3.15}$$

$$\alpha A_p\dot{x}_p = \begin{cases} K_v \cdot |u_v| \cdot \sqrt{p_b - p_t} & u_v > 0 \\ -K_v \cdot |u_v| \cdot \sqrt{p_s - p_b} & u_v < 0 \end{cases} \tag{3.16}$$

Where again the valve spool dynamics have been neglected. Setting these equations equal and solving for the pressure differences yields then:

$$p_s - p_a = \frac{p_b - p_t}{\alpha^2} \qquad u_v > 0 \tag{3.17}$$

$$p_a - p_t = \frac{p_s - p_b}{\alpha^2} \qquad u_v < 0 \tag{3.18}$$

Now substituting p_a for the load pressure as: $p_a = p_l + \alpha p_b$ yields the equation for the pressure in chamber B as:

$$p_b = \begin{cases} \frac{\alpha^2(p_s - p_l) + p_t}{1+\alpha^3} & u_v > 0 \\ \frac{\alpha^2(p_t - p_l) + p_s}{1+\alpha^3} & u_v < 0 \end{cases} \tag{3.19}$$

and substituting p_b using the same load pressure relationship $p_b = (p_a - p_l)/\alpha$ for p_a in chamber A yields:

$$p_a = \begin{cases} \frac{p_l + \alpha p_t + \alpha^3 p_s}{1+\alpha^3} & u_v > 0 \\ \frac{p_l + \alpha p_s + \alpha^3 p_t}{1+\alpha^3} & u_v < 0 \end{cases} \tag{3.20}$$

Linearized Load Pressure Dynamics

Again starting from the load pressure equation $p_l = p_a - \alpha p_b$ and taking the derivative with respect to time, we can write the load pressure dynamics using equations (2.10), (2.11), (2.14) and (2.15) resulting in:

$$\dot{p}_l^+ = \begin{cases} \frac{\beta_e}{V_{pl} + x_p A_p}\left(\frac{K_v|u_v|}{\sqrt{1+\alpha^3}}\sqrt{p_s - p_l - \alpha p_t} - A_p\dot{x}_p\right) \\ \quad - \frac{\alpha\beta_e}{V_{pl} + (L_c - x_p)\alpha A_p}\left(\frac{-K_v|u_v|}{\sqrt{1+\alpha^3}}\sqrt{\alpha^2 p_s - \alpha^2 p_l - \alpha^3 p_t} + \alpha A_p\dot{x}_p\right) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad u_v > 0 \end{cases} \tag{3.21a}$$

$$\dot{p}_l^- = \begin{cases} \frac{\beta_e}{V_{pl} + x_p A_p}\left(\frac{-K_v|u_v|}{\sqrt{1+\alpha^3}}\sqrt{\alpha p_s + p_l - p_t} - A_p\dot{x}_p\right) \\ \quad - \frac{\alpha\beta_e}{V_{pl} + (L_c - x_p)\alpha A_p}\left(\frac{K_v|u_v|}{\sqrt{1+\alpha^3}}\sqrt{\alpha^3 p_s + \alpha^2 p_l - \alpha^2 p_t} + \alpha A_p\dot{x}_p\right) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad u_v < 0 \end{cases} \tag{3.21b}$$

This nonlinear load pressure equation, which is of the form given in equation (3.12), can now be linearized using the same method as in section (3.1.1). The linear load pressure takes on the form:

$$\frac{d}{dt}\delta p_l = \frac{\partial \dot{p}_l}{\partial p_l}\Big|_{\oslash} \cdot \delta p_l + \frac{\partial \dot{p}_l}{\partial x_p}\Big|_{\oslash} \cdot \delta x_p + \frac{\partial \dot{p}_l}{\partial \dot{x}_p}\Big|_{\oslash} \cdot \delta \dot{x}_p + \frac{\partial \dot{p}_l}{\partial u_v}\Big|_{\oslash} \cdot \delta u_v \tag{3.22}$$

where $\oslash$ denotes an arbitrary operating point for the four variables: $\oslash = (p_{l\oslash}, x_{p\oslash}, \dot{x}_{p\oslash}, u_{v\oslash})$. The four terms relating state and input to output are given by:

$$\frac{\partial \dot{p}_l}{\partial p_l}\Big|_{\oslash} = \begin{cases} K_{pl}/\sqrt{p_s - p_{l\oslash} - \alpha p_t}\,, & u_v > 0 \\ K_{pl}/\sqrt{\alpha p_s + p_{l\oslash} - p_t}\,, & u_v < 0 \end{cases} \tag{3.23}$$

$$\frac{\partial \dot{p}_l}{\partial x_p}\Big|_{\oslash} = \begin{cases} -K_{x_pA} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash}\,, & u_v > 0 \\ K_{x_pA} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash}\,, & u_v < 0 \end{cases} \tag{3.24}$$

$$\frac{\partial \dot{p}_l}{\partial \dot{x}_p}\Big|_{\oslash} = \begin{cases} K_{\dot{x}_p}\,, & u_v > 0 \\ K_{\dot{x}_p}\,, & u_v < 0 \end{cases} \tag{3.25}$$

$$\frac{\partial \dot{p}_l}{\partial u_v}\Big|_{\oslash} = \begin{cases} K_{uv} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t}\,, & u_v > 0 \\ K_{uv} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t}\,, & u_v < 0 \end{cases} \tag{3.26}$$

The chamber volumes at the operating point are given by:

$$V_{a\oslash} = V_{pl} + x_{p\oslash} Ap \tag{3.27}$$

$$V_{b\oslash} = V_{pl} + (L_c - x_{p\oslash})\alpha Ap \tag{3.28}$$

and the gains are defined as:

load pressure gain:

$$K_{pl} = \frac{-K_v |u_{v\oslash}|}{2\sqrt{1+\alpha^3}} \left(\frac{\beta_e}{V_{a\oslash}} + \frac{\alpha^2 \beta_e}{V_{b\oslash}} \right) \tag{3.29}$$

position pressure gain:

$$K_{x_pA} = \frac{K_v |u_{v\oslash}|}{\sqrt{1+\alpha^3}} \left(\frac{\beta_e A_p}{V_{a\oslash}^2} - \frac{\alpha^3 \beta_e A_p}{V_{b\oslash}^2} \right) \tag{3.30}$$

velocity gain:

$$K_{x_pB} = \left(\frac{\beta_e A_p^2}{V_{a\oslash}^2} - \frac{\alpha^3 \beta_e A_p^2}{V_{b\oslash}^2} \right) \tag{3.31}$$

position velocity gain:

$$K_{\dot{x}_p} = A_p \beta_e \left(-\frac{1}{V_{a\oslash}} - \frac{\alpha^2}{V_{b\oslash}} \right) \tag{3.32}$$

valve input gain:

$$K_{uv} = \frac{K_v}{\sqrt{1+\alpha^3}} \left(\frac{\beta_e}{V_{a\oslash}} + \frac{\alpha^2 \beta_e}{V_{b\oslash}} \right) \tag{3.33}$$

This completes the derivation and linearization of the load pressure dynamics equation, which may look complicated with these five gains; however, they allow a simple and intuitive analysis.

3.2.2 Simplified Linear Model in State Space Form

The linearized load pressure dynamics equation (3.22) allows us now to simplify the system and write it as a similar state space system with only three states.
The states are defined as:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_p = \text{position of piston} \\ \dot{x}_p = \text{velocity of piston} \\ p_l = \text{load pressure} \end{pmatrix} \tag{3.34}$$

The measurement vector is defined as:

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_p = \text{position of piston} \\ \dot{x}_p = \text{velocity of piston} \\ F_s = \text{load cell force} \end{pmatrix} \tag{3.35}$$

The state space form, given by the equations (3.6) and (3.7), is derived for the same operating point $\oslash = (p_{l\oslash}, x_{p\oslash}, \dot{x}_{p\oslash}, u_{v\oslash}) = (0, L_c/2, 0, 0)$ as defined above in chapter 3. The linear system matrices are given by:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} \\ 0 & K_{\dot{x}_p} & 0 \end{bmatrix} \tag{3.36a}$$

$$\mathcal{B} = \begin{bmatrix} 0 & \\ 0 & \\ \begin{cases} K_{uv}\sqrt{p_s - \alpha p_t} & u_v > 0 \\ K_{uv}\sqrt{\alpha p_s - p_t} & u_v < 0 \end{cases} \end{bmatrix} \tag{3.36b}$$

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -d_p & A_p \end{bmatrix} \tag{3.36c}$$

$$\mathcal{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.36d}$$

where the matrices $\mathcal{B}, \mathcal{C}$ should again not be confused with the total friction coefficient B and the total spring stiffness C of the system. For ease of notation, the matrices $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ have no distinguished name with respect to the complex state space model; however, they are not the same and it is always clearly stated which system model is used. The matrices related to the measurement equation are basically identical, but adjusted to fit three states. Care must be taken when using these system descriptions since one easily forgets to subtract the chosen operating point from the inputs and add it again to the outputs. This will be important when using system identification because the results can differ vastly when, for example, x_p instead of $x_p - x_{p\oslash}$ is used as a state measurement.

3.3 Simplified Linear Model in Terms of Load Cell Force

One drawback of the simplified system is that the load pressure is a state which cannot be measured directly. We measure it indirectly through the load cell force measurement given by equation (2.22) along with velocity and acceleration measurements of the piston.

We therefore seek to write the system dynamics purely in terms of measurable states only, which permits the use of pure state feedback control techniques without the need of additional state estimation algorithms. Therefore, the complexity of the required control system can be significantly reduced. The new state vector is thus defined as:

$$\vec{x} = \begin{pmatrix} x_p \\ \dot{x}_p \\ F_s \end{pmatrix} \tag{3.37}$$

The hydraulic force term $F_h = A_p \cdot p_l$ appearing in the load mass dynamic equation (2.28) needs to be replaced by a term of the form $F_h = f(x_p, \dot{x}_p, F_s)$. And the load pressure dynamics is replaced by the load cell force dynamics, having a form like: $\dot{F}_s = f(x_p, \dot{x}_p, F_s, u_v)$. We can rewrite the load mass dynamics (2.28) using the equation (2.22) for expressing the hydraulic force in terms of the load cell force measurements as:

$$F_h = A_p \cdot p_l = F_s + d_p \dot{x}_p + m_p \ddot{x}_p \tag{3.38}$$

where the measurement equation for the left side is used. The next step is to find an equation describing the dynamics of the load cell force, i.e.:

$$\dot{F}_s = f(x_p, \dot{x}_p, F_s, u_v) \tag{3.39}$$

a function that only depends on the states $(x_p, \dot{x}_p, F_s)$ and the input u_v. Using the left side measurement equation of equation (2.22), we have:

$$\dot{F}_s = f(x_p, \dot{x}_p, F_s, u_v) = A_p \dot{p}_l - d_p \dddot{x}_p - m_p \ddddot{x}_p \tag{3.40}$$

Linearizing this equation, using again the same method as in section (3.1.1), around a general operating point along with the known linear load pressure dynamics given by the equation (3.22), we have for $u_v > 0$:

$$\begin{aligned} \frac{d}{dt}\delta F_s^+ =& \frac{K_{pl}}{\sqrt{p_s - p_{l\oslash} - \alpha p_t}} \cdot \delta F_h + A_p \left(-K_{x_pA} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash}\right) \cdot \delta x_p \\ &+ A_p K_{\dot{x}_p} \cdot \delta \dot{x}_p + A_p K_{uv} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t} \cdot \delta u_v - d_p \delta \ddot{x}_p - m_p \delta \dddot{x}_p \end{aligned} \tag{3.41}$$

and for $u_v < 0$:

$$\begin{aligned} \frac{d}{dt}\delta F_s^- =& \frac{K_{pl}}{\sqrt{\alpha p_s + p_{l\oslash} - p_t}} \cdot \delta F_h + A_p \left(K_{x_pA} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash}\right) \cdot \delta x_p \\ &+ A_p K_{\dot{x}_p} \cdot \delta \dot{x}_p + A_p K_{uv} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t} \cdot \delta u_v - d_p \delta \ddot{x}_p - m_p \delta \dddot{x}_p \end{aligned} \tag{3.42}$$

The linearized hydraulic force δF_h is now substituted by equation (3.38), where the term $m_p \delta \ddot{x}_p$ is assumed to be negligible since the piston mass is small compared to the load mass. Moreover, the underlying load pressure model assumes steady-state pressure conditions, thus the piston acceleration is assumed to be small in the first place. Hence, the term $m_p \delta \dddot{x}_p$ is also negligible. Defining the following new abbreviations to simply notation:

$$G_{F_s} = \begin{cases} \frac{K_{pl}}{\sqrt{p_s - p_{l\oslash} - \alpha p_t}}, & u_v > 0 \\ \frac{K_{pl}}{\sqrt{\alpha p_s + p_{l\oslash} - p_t}}, & u_v < 0 \end{cases} \tag{3.43}$$

$$G_{x_p} = \begin{cases} A_p \left(-K_{x_pA} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash} \right) , & u_v > 0 \\ A_p \left(K_{x_pA} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t} + K_{x_pB} \cdot \dot{x}_{p\oslash} \right) , & u_v < 0 \end{cases} \tag{3.44}$$

$$G_{u_v} = \begin{cases} A_p K_{uv} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t} , & u_v > 0 \\ A_p K_{uv} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t} , & u_v < 0 \end{cases} \tag{3.45}$$

Using these abbreviations, we can now write for:

$$\begin{aligned} \frac{d}{dt}\delta F_s = {} & G_{F_s} \cdot (\delta F_s + d_p \cdot \delta \dot{x}_p) + G_{x_p} \cdot \delta x_p \\ & + A_p K_{\dot{x}_p} \cdot \delta \dot{x}_p + G_{u_v} \cdot \delta u_v - d_p \cdot \delta \ddot{x}_p \end{aligned} \tag{3.46}$$

Solving the right side measurement equation of equation (2.22) for $\ddot{y} \equiv \ddot{x}_p$ gives:

$$\delta \ddot{x}_p = \frac{1}{m_l} \cdot \delta F_s - \frac{C}{m_l} \cdot \delta x_p - \frac{d_l}{m_l} \cdot \dot{x}_p \tag{3.47}$$

where the linearized variables were used. This equation will now be used for substituting the term $-d_p \cdot \delta \ddot{x}_p$, resulting in the final equation:

$$\begin{aligned} \frac{d}{dt}\delta F_s = {} & \left(G_{F_s} - \frac{d_p}{m_l} \right) \cdot \delta F_s + \left(G_{F_s} d_p + \frac{d_p \cdot d_l}{m_l} + A_p K_{\dot{x}_p} \right) \cdot \delta \dot{x}_p \\ & + \left(G_{x_p} + \frac{d_p \cdot C}{m_l} \right) \cdot \delta x_p + G_{u_v} \cdot \delta u_v \end{aligned} \tag{3.48}$$

3.3.1 State Matrices of the Load Cell Force Linear Model

The simplified system can now be written in state space form replacing the load pressure state with the load cell force state:

$$\tilde{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_p = \text{position of piston} \\ \dot{x}_p = \text{velocity of piston} \\ F_s = \text{load cell force} \end{pmatrix} \tag{3.49}$$

The measurement vector is now just the same as the state vector since all states are measured:

$$\vec{y} = \vec{x} \tag{3.50}$$

The state space form, given by the equations (3.6) and (3.7), is derived for the same operating point $\oslash = (F_{s\oslash}, x_{p\oslash}, \dot{x}_{p\oslash}, u_{v\oslash}) = (0, L_c/2, 0, 0)$ as defined above in chapter 3.

The linear system matrices are given by:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} \\ G_{x_p} + \frac{d_p \cdot C}{m_l} & G_{F_s} d_p + \frac{d_p \cdot d_l}{m_l} + A_p K_{\dot{x}_p} & G_{F_s} - \frac{d_p}{m_l} \end{bmatrix} \tag{3.51a}$$

$$\mathcal{B} = \begin{bmatrix} 0 \\ 0 \\ \begin{cases} A_p K_{uv} \cdot \sqrt{p_s - p_{l\oslash} - \alpha p_t}\,, & u_v > 0 \\ A_p K_{uv} \cdot \sqrt{\alpha p_s + p_{l\oslash} - p_t}\,, & u_v < 0 \end{cases} \end{bmatrix} \tag{3.51b}$$

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.51c}$$

$$\mathcal{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.51d}$$

Note that the abbreviations G_{x_p}, G_{F_s} and G_{u_v} all depend on the sign of the valve opening u_v. The normalization matrices are the same as given by (3.9)-(3.11), but adjusted for three states, that is, the state normalization matrix is the same as the output normalization matrix:

$$T = W = \begin{bmatrix} \frac{L_c}{2} & 0 & 0 \\ 0 & 1.2\,\mathrm{m/s} & 0 \\ 0 & 0 & 5000\,\mathrm{N} \end{bmatrix} \tag{3.52}$$

$$V = 10\,\mathrm{mA} \tag{3.53}$$

Chapter 4

Model Analysis

Chapter 2 has developed a nonlinear model for the hydraulic cylinder combined with the load mass. Chapter 3 derived a linear model and sections 3.2 and 3.3 presented two even simpler linear models based on the load pressure and the load cell force measurement. The nonlinear model is usually used for simulations since the physical effects are well described. The linear models are used to study the system and to design controllers. This chapter analyses the four presented models in terms of their accuracy, model assumptions, different transfer functions and studies both the "Degree of Nonlinearity" and the parameter sensitivity.

4.1 Linear vs. Nonlinear System Responses

The accuracy of the linearized system dynamics vanishes as the operating point moves away from the linearization point. Therefore, it is important to check the response of the linear system against the response of the nonlinear system. Figure 4.1 shows the comparison of the nonlinear model (2.30) versus the three linear models. These are the full model (3.8), the simplified model (3.36) (based on the load pressure), and the simplified model (3.51) (based on the load cell force measurement F_s). The simulation input signal to the valve is a step at time 0.001 s with amplitude 1 % of the nominal valve opening followed by an equal step at time 0.05 s with the same but negative amplitude.
When examining the load cell force F_s shown on top of figure 4.1, it can be clearly seen that all linear models are able to accurately reproduce the force compared to the response of the nonlinear model. The force response of the linear models seems to be less damped than the nonlinear response. The second subplot shows the position of the hydraulic cylinder starting from the middle of its stroke. The full model almost exactly reproduces the position signal of the nonlinear model, while the two simplified models do not fully reproduce the position. However, these two models are almost indistinguishable. When looking at the pressure shown in subplot three, it is evident that the full linear model is not able to reproduce the whole pressure dynamics existing in the chambers and a pressure offset results. This is probably due to the highly nonlinear structure of the pressure equations (2.10) and (2.11) since they are influenced by position, velocity, pressure and input current at the same time where the pressure appears within the square root. However, when looking at the fourth subplot, where the load pressure $p_l = p_a - \alpha p_b$ is shown, the response of all three systems is again nearly identical. This suggests that the linearization does not well at reproducing the pressure dynamics; however, the relative pressure dynamics p_l, which

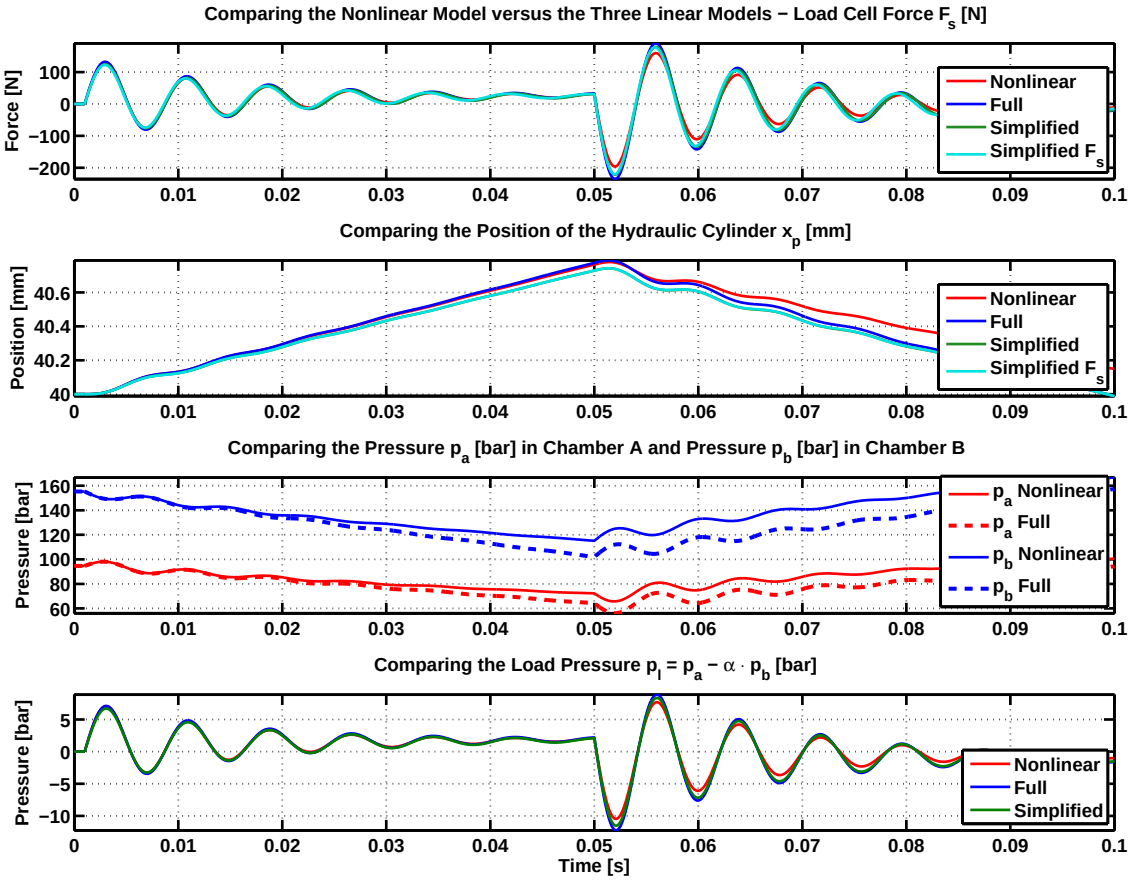

Figure 4.1: Comparison of the nonlinear model (2.30) versus the three linear models. These are the full model (3.8), the simplified model (3.36) (based on the load pressure), and the simplified model (3.51) (based on the load cell force measurement F_s). The simulation input signal to the valve is a step at time 0.001 s with amplitude 1 % of the nominal valve opening followed by an equal step at time 0.05 s with the same but negative amplitude. On top the load cell force of the nonlinear system (red) is compared to the responses of the linear models in blue, green and cyan, respectively. The second subplot compares the position of the hydraulic cylinder. Pressure inside chambers A and B and the load pressure $p_l = p_a - \alpha p_b$ are shown in the third and fourth subplot. Notice the good match of both force and load pressure, but only marginal accuracy of the full linear model in reflecting the pressure dynamics. Note also that the two simplified models are able to accurately reproduce the load cell force.

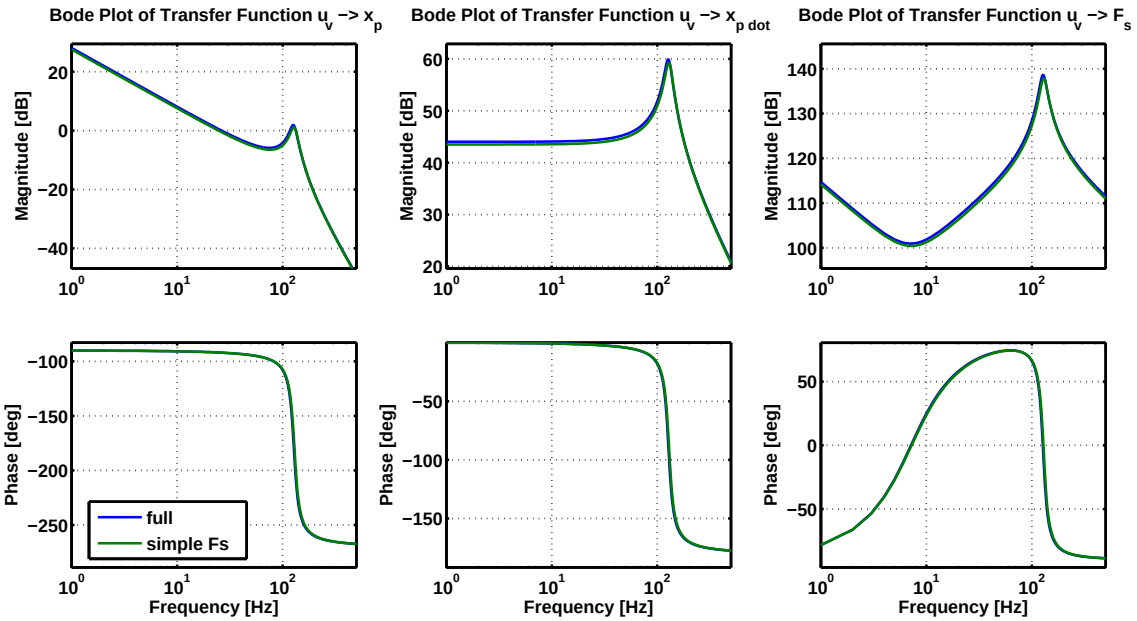

Figure 4.2: Bode plot of all three output transfer functions of the linear models. From left to right these are: the transfer function from input u_v to the position of the piston x_p, to the velocity of the piston $\dot{x}_p$ and to the force measured by the load cell F_s. The full system is drawn in blue and the simple Fs in green.

are only slightly nonlinear, are well approximated. This is also the reason why the forces, shown in subplot one match better than the pressures shown in subplot three since the force is essentially the load pressure but scaled with the piston area A_p.

Overall we can say that the three linear models are able to accurately reproduce both the load cell force F_s as well as the load pressure p_l. However, the actual pressure inside the chambers is not reliably reproduced. Moreover, the additional simplification of the simplified Fs model (given by equations (3.51)) does not result in less accurate results. Hence, this model (simplified Fs) will be used for any further analysis and also for designing the estimator and controller since little accuracy is lost due to the simplifications.

4.2 Transfer Functions

Transfer functions describe the relationship of the output to the input and are most often visualized by Bode plots, where on top the magnitude in dB and at the bottom the phase in degree (deg) is shown. The magnitude describes the amplification of the output with respect to the input and the phase describes the lag of the output with respect to the input. The linear systems have three outputs measuring position, velocity and load cell force. Figure 4.2 shows those three transfer functions for both the full (given by equations (3.8)) and the simple Fs model (given by equations (3.51)) for the operating point defined in chapter 3.

The position transfer function has an integrator at the origin (negative slope at low frequencies) and only a small resonance peak followed by a quick roll-off. The third column shows the valve input u_v to the load cell force F_s transfer function; note the high amplification of the input. This is inherent to the hydraulic system dynamics, where a very small

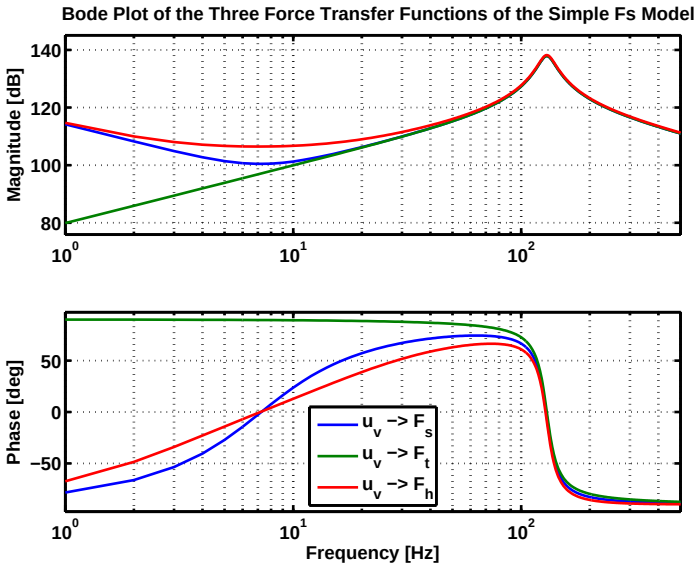

Figure 4.3: Bode plot of the three force transfer functions of the simple Fs model. These are: the load cell force transfer function (blue) from input u_v to F_s, the hydraulic force transfer function (red) from input u_v to F_h, and the total force transfer function (green) from input u_v to F_t. As can be seen, these three transfer functions differ quite a lot at low to middle frequencies, but coincide at higher frequencies.

valve input $\leq 10\,\text{mA}$ can produce a very high force output $\approx 3200\,\text{N}$. Note the steep slope at low frequencies, the dent around middle frequencies (determined by the spring stiffness and friction coefficient), and the sharp resonance peak at high frequencies (determined by the load mass dynamics and pressure dynamics of the hydraulic system).

The force transfer function studied so far is not the only force that acts on the system. There are two other forces, namely the hydraulic force F_h (discussed in 2.8) and the total force F_t (discussed in 2.9). In many papers about force control of a hydraulic system there is usually no clear statement which force is considered or which transfer function is looked at. The measurement device (the load cell) is rarely discussed. However, there are papers, for example [28], that discuss the force measurement sensor. The three force transfer functions are not the same, therefore figure 4.3 shows all these force transfer functions for the simple Fs model linearized around the operating point given in chapter 3.

Note that all three transfer functions coincide for frequencies above $10^2\,\text{Hz}$; hence, they capture the same high frequency dynamics of the system. Their shape differs at frequencies below $10^2\,\text{Hz}$. Unfortunately, this is the region where the system is operated in most of the time, that is, walking with frequencies around $\approx 4\,\text{Hz}$. Hence, choosing the right transfer function for the controller design will be important. As can be seen, the total force transfer function $u_v \to F_t$ has two zeros at the origin (and an integrator), hence the steep slope at low frequencies. The other two transfer functions only differ slightly due to the fact that the load cell transfer function $u_v \to F_s$ incorporates additional velocity and acceleration terms (see equation (2.22)). In this work, we use the load cell force transfer function since the load cell force is the only force that can currently be measured directly. An advantage of the simplification of using the load cell force transfer function is that the velocity and acceleration terms are inherently included.

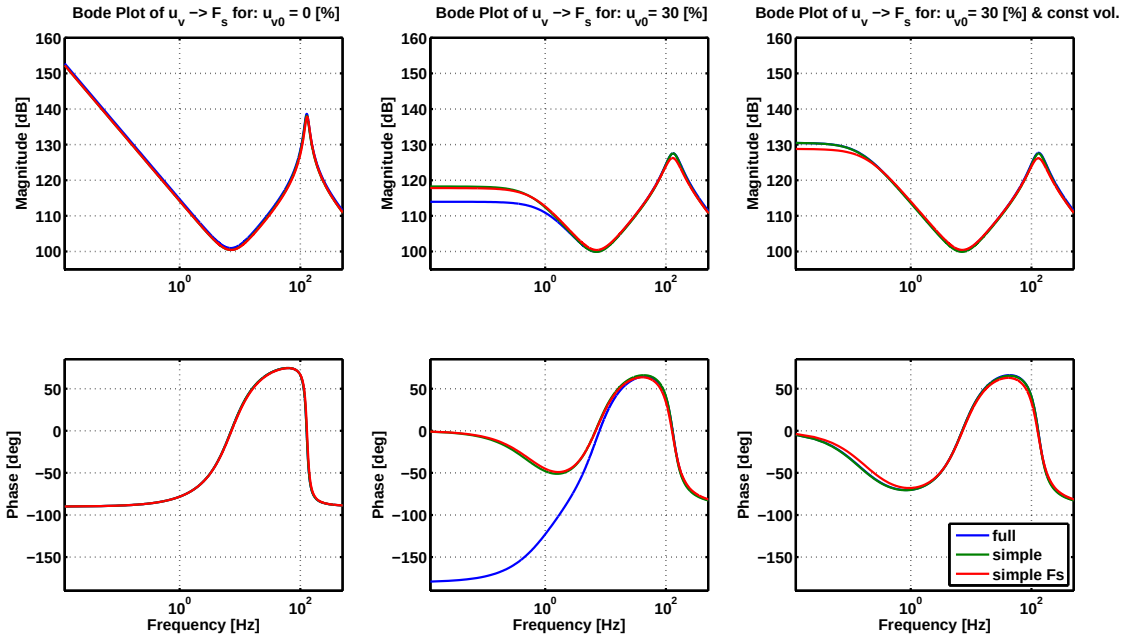

Figure 4.4: Bode plot of the load cell force transfer function of the full model (3.8), the simplified model (3.36) (based on the load pressure), and the simplified model (3.51) (based on the load cell force measurement F_s). The first column shows the impact of the assumptions needed to derive these three models for the nominal operating point (chapter 3). The second column shows those for the operating point: $\oslash = (p_{l\oslash}, x_{p\oslash}, \dot{x}_{p\oslash}, u_{v\oslash}) = (0, L_c/2, 0, 30\%)$. The third column shows the transfer function of the second column, but for the additional "constant volume" assumption. As can be seen, the "constant volume" assumption results in transfer functions not able to capture the low frequency dynamics that are present in column two.

4.2.1 Influence of the Modeling Assumptions on the Transfer Functions

The three models discussed above are derived using different modeling assumptions. The full linear model (3.8) assumes the linearization concept. The simplified model (3.36) additionally assumes steady-state pressure dynamics, and the simplified Fs model (3.51) further adds the assumption of negligible acceleration terms of the load cell force measurement. Hence, we compare the impact that these assumptions have on the load cell force transfer function of each system, respectively, shown in figure 4.4. The first column shows the Bode plot without additional assumptions for an operating point as defined in chapter 3. As can be seen, all models give the same transfer function, thus the assumptions appear to have no influence. However, when the operating point is changed, for example the valve operating point to $u_{v0} = 30\,\%$ of its nominal range, then the Bode plot shown in the second column results. As is evident, the simple (green) and simple Fs (red) model are not able to capture the impact of the changed operating point, hence they differ from the transfer function of the full model (blue). Similar observation also holds when the velocity operating point is changed, that is, $\dot{x}_{p0} = 1\,\mathrm{m/s}$, see figure 4.5 column one and two. The two simplified models are again not able to fully cope with the changed operating point, that is, they differ from the blue response. However, in both cases they only differ in the low frequency range.

A common additional assumption is the so called "constant volume" assumption which assumes that the chamber and pipeline volumes remain constant since only small movements

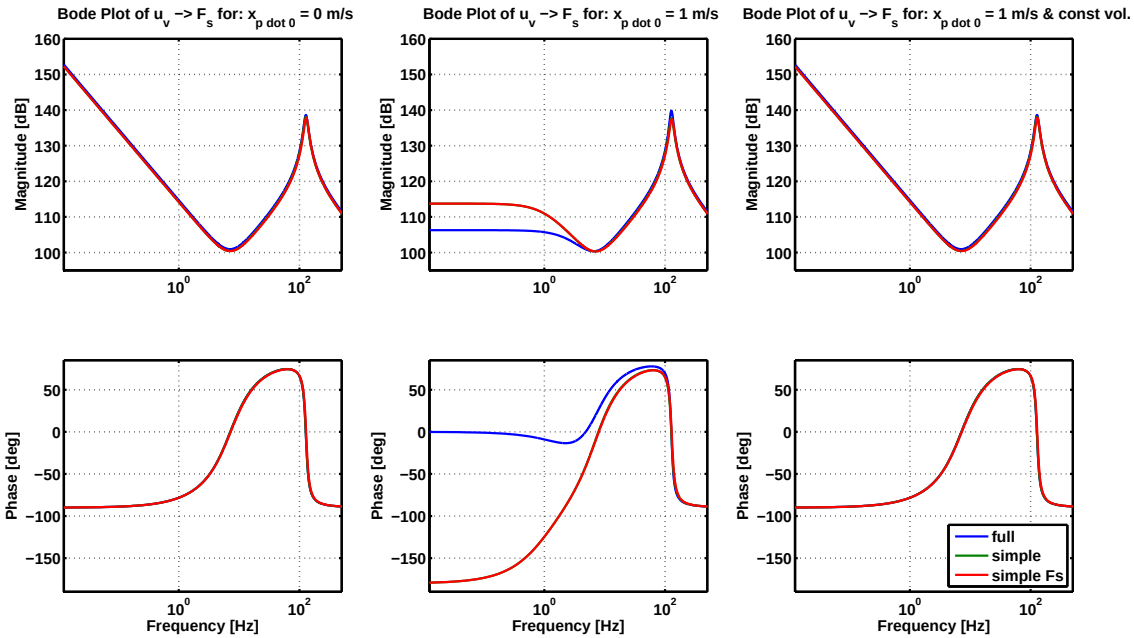

Figure 4.5: Bode plot of the load cell force transfer function of the full model (3.8), the simplified model (3.36) (based on the load pressure), and the simplified model (3.51) (based on the load cell force measurement F_s). The first column shows the impact of the assumptions needed to derive those three models for the nominal operating point (chapter 3). The second column shows these for the operating point: $\oslash = (p_{l\oslash}, x_{p\oslash}, \dot{x}_{p\oslash}, u_{v\oslash}) = (0, L_c/2, 1, 0)$. The third column shows the transfer function of the second column, but for the additional "constant volume" assumption. Again this assumption renders the three models unable to accurately describe the low level frequency dynamics.

of the piston are considered and the pipelines are typically long and hold large amounts of volume (fluid). Therefore, the equations for the chamber volumes (2.12) and (2.13) are set to their respective nominal value, that is, $V_a = V_{a\oslash}$ and $V_b = V_{b\oslash}$. Hence, the position x_p is no longer an argument of the load pressure equation (3.12), thus the corresponding element of the state space matrix A is linearized to zero. The impact of this "constant volume" assumption is shown in column three of the two figures 4.4 and 4.5. Considering the third column of figure 4.4, it can be seen that this assumption influences all three models not being able to completely capture the low frequency dynamics that are present in the blue line of the second column of figure 4.4 without the "constant volume" assumption. The influence of this "constant volume" assumption is even more severe when changing the velocity operating point as shown by column three of figure 4.5. As can be seen, all three models are no longer able to capture the impact of changing the velocity operating point, that is, the transfer functions of column three look exactly like the ones from column one where the velocity operating point is zero. This is no surprise since the "constant volume" assumption effectively sets the coefficients K_{x_pA} (3.30) and K_{x_pB} (3.31) to zero, hence the velocity operating point $\dot{x}_{p\oslash}$ can no longer influence the linearized pressure dynamics (3.22) through the term (3.24). We can conclude the following:

1. The two simplified models are not able to fully capture the changes of the transfer function due to changing operating point (compare columns one and two, respectively, of figures 4.4 and 4.5). However, the deficit by choosing the simplified Fs model is still reasonable.

2. The reason why the "constant volume" assumption has such a high impact lies in the fact that the hydraulic pressure lines used on the robot HyQ are very short and thin, thus very low amounts of fluid are contained inside. Therefore, the total volume V_a (or V_b) is largely determined by the volume of chambers A (or B).

3. Due to the previous argument and the discussion above, the "constant volume" assumption is not recommendable in our case since it severely deteriorates the accuracy and capability of all three models. Hence this assumption is not used any further.

4. The "constant volume" assumption may be a valid assumption for hydraulic systems that use long pipelines holding large amounts of volume (fluid), where the volume of the chambers is negligible.

4.2.2 Symbolic Equations of the Three Force Transfer Functions

As mentioned above in section 4.2, there are three force transfer functions shown in figure 4.3. This work uses the simplified Fs model since it is a good compromise between accuracy and simplicity. Due to these modeling assumptions, the force transfer function can be studied more easily, giving important insights into the behavior of the system. Below the three force transfer functions for the simplified system (3.36) are given in symbolic notation. We only show the equations for this simplified model since it allows additional insights into the explicit pressure dependencies. The equations are valid for positive valve opening, that is: $u_v \geq 0$.
The load cell force transfer function is:

$$\frac{F_s(s)}{u_v(s)} = \frac{A_p K_{uv}\left(Ms^2 + d_l s + C\right)}{Ms^3 + (d_l + d_p + MK_{pl})\,s^2 + \left(C + (d_l + d_p)K_{pl} - A_p K_{\dot{x}_p}\right)s \\ -A_p\left(K_{x_pA} - K_{x_pB}\dot{x}_{p\oslash}\right) + CK_{pl}} \tag{4.1}$$

The hydraulic force transfer function is:

$$\frac{F_h(s)}{u_v(s)} = \frac{A_p K_{uv}\left(Ms^2 + (d_l + d_p)s + C\right)}{\begin{array}{c} Ms^3 + (d_l + d_p + MK_{pl})\,s^2 + \left(C + (d_l + d_p)K_{pl} - A_p K_{\dot{x}_p}\right) s \\ -A_p\left(K_{x_pA} - K_{x_pB}\dot{x}_{p\oslash}\right) + CK_{pl} \end{array}} \tag{4.2}$$

The total force transfer function is:

$$\frac{F_t(s)}{u_v(s)} = \frac{A_p K_{uv} M s^2}{\begin{array}{c} Ms^3 + (d_l + d_p + MK_{pl})\,s^2 + \left(C + (d_l + d_p)K_{pl} - A_p K_{\dot{x}_p}\right) s \\ -A_p\left(K_{x_pA} - K_{x_pB}\dot{x}_{p\oslash}\right) + CK_{pl} \end{array}} \tag{4.3}$$

Comparing equations (4.1), (4.2), and (4.3), it is immediately clear that only the nominator, that is, the zeros of the transfer function, changes. As can be seen, the zeros of the load cell force transfer function (4.1) are determined by the load mass as well as load and piston friction coefficients, and stiffness of the load.

4.3 Sensitivity Analysis

This section analyses how the system behavior changes when the operating point or the values of the model parameters are varied. First, the influence of the operating point is analyzed since the linearized system is only valid around its operating point chosen for the linearization. In the remainder of this thesis this analysis is called "Degree of Nonlinearity". The range of validity is determined by the nonlinearity of the underlying nonlinear system. If, for example, the nonlinear system behaves almost linearly, then the validity of the linearized system will span a large set of operating points. If, however, the nonlinear system behaves strongly nonlinear, then the validity of the linear system will be limited to a small range. Second, the sensitivity of the model to variations in the model parameters around their nominal value is studied.
Since a controller is usually designed for a particular linearized system, its validity[1] will depend on the validity of the linearized system. Therefore, valuable insights can be gained from the "Degree of Nonlinearity" analysis.
The difference in the "Degree of Nonlinearity" analysis (subsection 4.3.1) and the "Parameter Sensitivity" analysis (subsection 4.3.2) is that the dynamics of changing operating points are inherently included in the nonlinear system, whereas variations in the modeling parameters, such as mass and friction, affect both the nonlinear and the linear models.

4.3.1 Degree of Nonlinearity

The "Degree of Nonlinearity" of the nonlinear model (2.30) is studied by linearizing it at different operating points. The nominal or reference operating point is defined in chapter 3. Only the analysis of the transfer function from input u_v to the load cell force F_s of the simplified Fs system (3.51) is shown here since this model was selected in section 4.1. The analysis of the other systems (3.2.2 and 3.1.1) results in roughly the same findings, just extended by their additional operating point dependencies.

[1] With the term "validity" I mean the range of systems for which the controller will still provide reasonable performance. This is of course the basic idea of robust control theory. In this thesis, however, these theories are not further considered.

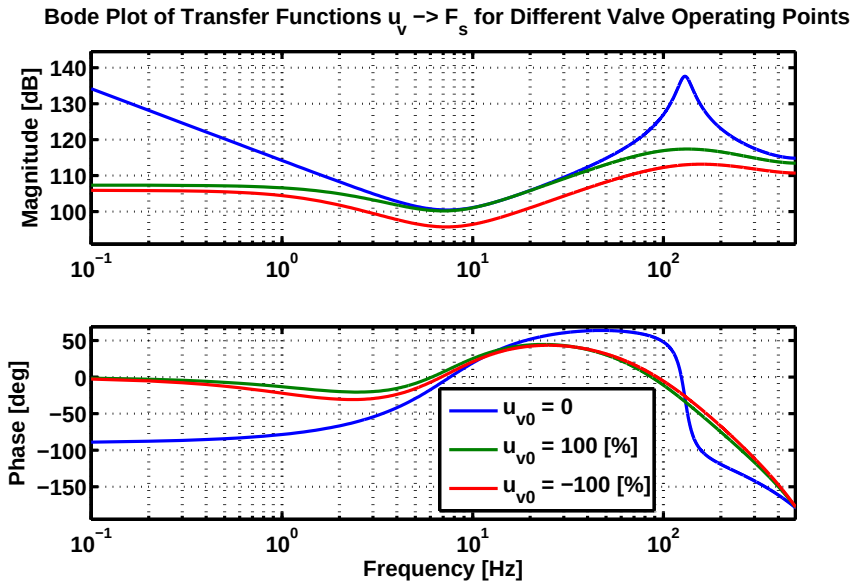

Figure 4.6: Bode plot of the transfer function from valve input to load cell force measurement for the simplified Fs system in terms of load cell force. Shown are the transfer functions for the linearization around the nominal operating point $u_v = 0\,\%$ (blue) and fully open $u_v = \pm 100\,\%$ (green and red, respectively). The valve operating point and hence the pressure dynamics dominate the resonance peak significantly. The response magnitude of the negative open valve (red) is lower than that of the positive open valve (green) due to the asymmetric cylinder.

The operating point of the simplified Fs system has a complexity of four dimensions because there are four variables defining the operating points. The influence of each variable is looked at first while the others remain fixed. It turns out that varying the operating point of the valve $u_{v\oslash}$ results in the most dramatic change. The individual influence of the other three operating points is smaller. Hence, figure 4.6 shows the changes of the load cell force transfer function for the nominal operating point $u_v = 0\,\%$ and fully open valve $u_v = \pm 100\,\%$. It can clearly be seen how the force $F_h = A_p \cdot p_l$ created by the load pressure dynamics (3.21) dominates the resonance peak of the hydraulic system. The response magnitude of the fully negative open valve (red) is less than that of the positive open valve (green) because the hydraulic cylinder is asymmetric, hence less force is created in negative direction. The equivalent valve position u_v, amplified by the valve gain K_v, is a direct input of this load pressure equation hence its strong influence. This dominant behavior of the load pressure dynamics is actually desirable since it means that the controller does not need to additionally suppress the resonance peak; the load pressure does it already. However, this introduces of course some unwanted changes in the dynamics of the system. As one can imagine, when applying certain reference trajectories, the range of the valve inputs might span its full range. Hence, the system response is different at every instance. As will be seen later in subsection 6.3.2, this causes some problems during system identification.

During operation, when the piston is moving, the state vector travels through a wide range of its state space. Therefore, an arbitrary combination of operating points must be chosen to characterize the behavior of the system. Figure 4.7 shows the extent the transfer functions change when letting the operating point vary through the complete state space. Shown are the responses of 2400 different systems calculated by linearizing the nonlinear model around operating points varying between 10 % and 190 % of their nominal value (defined in

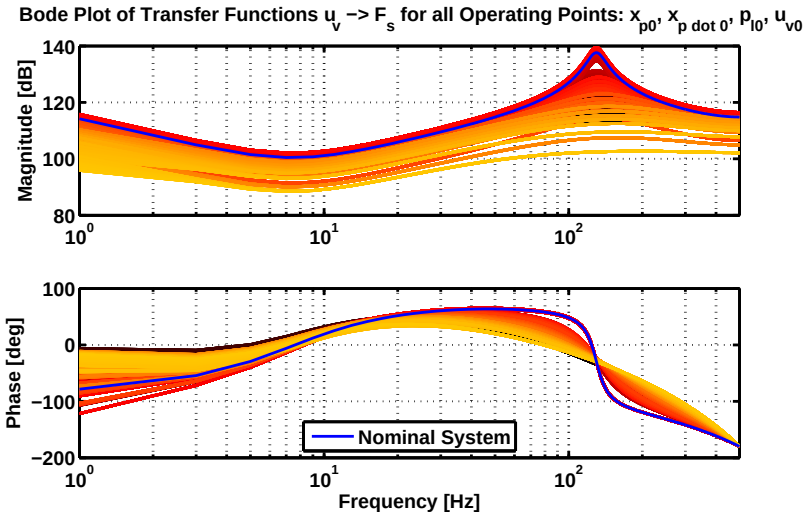

Figure 4.7: Bode plot of load cell force transfer functions for linearizations of the nonlinear model around 2400 different operating points in the state space. All the operating points are varied between 10 % and 190 % of their nominal value. The fundamental characteristic of the transfer functions remains approximately the same; however, the resonance peak is damped more or less, depending on the operating point. Note that the whole range of transfer functions can be captured by the three (or even two) transfer functions shown above in figure 4.6.

chapter 3). As can be seen, the fundamental characteristic of the transfer functions remains approximately the same; however, the resonance peak is damped more or less, depending on the operating point. Note that the whole range of transfer functions can be captured by the three (or even two) transfer functions shown above in figure 4.6. One system for the "undamped" response and a second one for the "damped" response are essentially sufficient to describe the region the nonlinear system occupies.

4.3.2 Parameter Sensitivity

The dynamics of the hydraulic system depend to a large degree on the values of the mass, friction coefficient and spring stiffness of the load. These three parameters are varied while the others are held at their nominal values[2]. The mass of the load is varied within the set: $m_l \in [1\,\mathrm{kg}, 127\,\mathrm{kg}]$, the friction coefficient within: $d_l \in [0\,\mathrm{Ns/m}, 2000\,\mathrm{Ns/m}]$, and the stiffness of the springs within: $C = k_1 + k_2 \in [10\,\mathrm{N/m}, 35000\,\mathrm{N/m}]$. Again, only the load cell force transfer function is shown since it is of main interest. Figure 4.8 depicts the sensitivity of the transfer function due to mass variations, figure 4.9 shows the sensitivity due to friction coefficient variations, and figure A.2, shown in appendix A, depicts the sensitivity of the transfer function due to spring stiffness variations. The parameter m_l, as can be seen from figure 4.8, has quite a significant impact on the shape of the force transfer function, thus the sensitivity of the model to this parameter is high. The resonance peak frequency is shifted from $\approx 38\,\mathrm{Hz}$ for heavy mass up to 500 Hz for light mass. Moreover the magnitude of the response is altered from 160 dB for heavy mass down to 115 dB for very light mass. Hence, these changes in mass have a drastic influence on the dynamics of the system. The sensitivity of the force transfer function to changes in the friction coefficient d_l, as shown in figure 4.9, is less severe, though still altering the low frequency behavior of the system

[2]Given in appendix A.3.

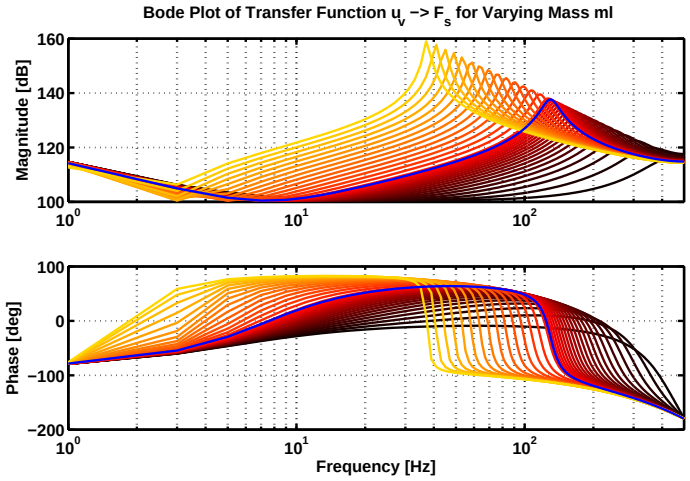

Figure 4.8: This figure shows the effect on the force transfer function due to varying load mass m_l within the set: $m_l \in [1\,\mathrm{kg}, 127\,\mathrm{kg}]$. The color changes from black to yellow with increasing mass. The region around the nominal system (in blue) is drawn in bright red. As can be clearly seen, the changes in mass shift the location of the resonance peak and also increase the magnitude for increasing weight.

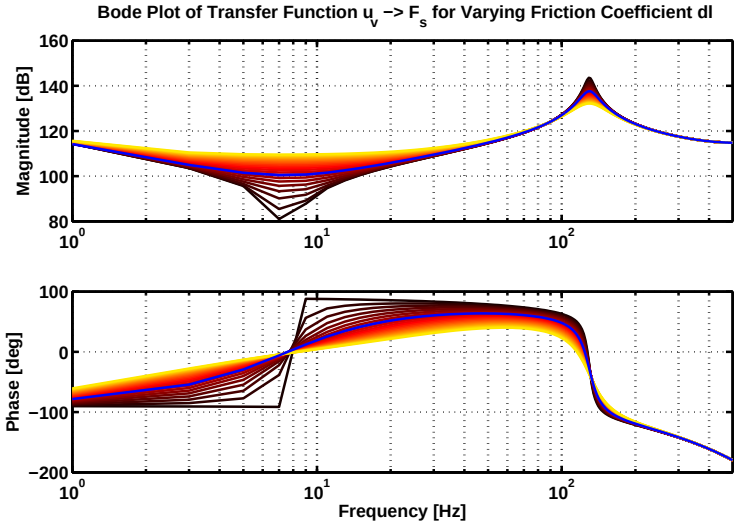

Figure 4.9: This figure shows the effect on the shape of the force transfer function due to varying the friction coefficient of the load cart d_l within the set: $d_l \in [0\,\mathrm{Ns/m}, 2000\,\mathrm{Ns/m}]$. The color changes from black to yellow with increasing friction coefficient. The region around the nominal system (in blue) is drawn in bright red. As can be clearly seen, the changes in friction coefficient mainly dampen the low frequency content of the system and only dampen the resonance peak slightly.

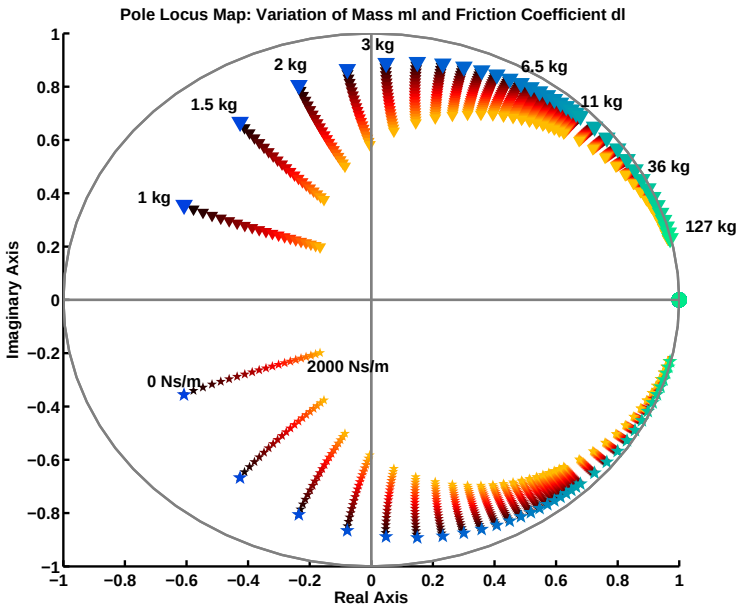

Figure 4.10: This figure shows the movement of the discrete-time pole locations due to varying mass m_l and friction coefficient d_l (for the simple Fs model), using again the variation sets as defined in subsection 4.3.2. The color changes from black to yellow with increasing friction coefficient and, additionally, the color changes from blue to light green for increasing mass. As can be clearly seen, the changes in friction coefficient effectively pull the poles closer to the origin, thus making them more robust. The changes in mass have a profound effect on the location of the poles, which basically cover the complete unit circle.

and some slight damping of the resonance peak. Variations in the spring stiffness, shown in figure A.2, are even less severe.

Besides the changes of the transfer function, the movement of the location of the discrete-time poles and zeros is also important. The sampling time is $Ts = 0.001\,\mathrm{s}$ using standard zero-order hold for the discretization the dynamics. The largest effect on the location of poles shows again the variation of the mass m_l and friction coefficient d_l. The variation of both parameters at the same time is shown in figure 4.10. Only the three poles of the system, marked by the symbols $\star$, Δ and o, are shown here since the location of zeros is only slightly affected. The color changes from black to yellow with increasing friction coefficient and, additionally, the color changes from blue to light green for increasing mass. This pole locus clearly shows that increasing the friction coefficient effectively pulls the poles closer to the origin, therefore making the system more robust. The changes in mass have a profound effect on the location of the poles, too, which are getting shifted almost all across the unit disc. The location of these poles represents the damping and resonance frequency of the system's force response, which can be better seen in figure 4.8. Increasing mass shifts the poles closer to the edge of the unit circle, thus decreasing the robustness. The poles of the system are closer to a discrete-time integrator, hence the control task might become easier since the controller has to force the system less to behave as anticipated.

All in all we can summarize that the hydraulic system has a high degree of nonlinearity due to the valve and pressure dynamics creating the hydraulic force. Identifying the underlying system for input signals exciting these nonlinearities will pose a difficult task

for any estimator. The force transfer function (and system) is quite sensitive to changes in both mass m_l and friction coefficient d_l, hence completely diverse systems for different parameters may result. Therefore, an adaptive controller makes sense to cope with these severe changes in the system dynamics.

Chapter 5

Adaptive Control Framework

The introduction (chapter 1) discussed and motivated the need for an adaptive low-level force controller. This chapter explains the chosen approach more thoroughly, leaving the details of the algorithms and implementation to the following two chapters. There are many ways on how to design an adaptive control framework, the two most commonly used methods are:

a) Directly learning the gains of the controller via some learning algorithm, like PI^2 [13], where the controller gains are learned for a specific task.

b) First (recursively) estimate the time-varying system and then use this improved system knowledge to calculate new controller gains.

In general, one advantage of method **a)** is that once the controller gains for a specific task or system are learned, they can be reused every time this specific task is executed. One advantage of method **b)** is, usually, that it is more generalizable since there are no system or task specific assumptions needed. Its disadvantage is, however, that the system and controller gains need to be recalculated even when the same task is executed repetitively, unless some logic is incorporated. A disadvantage of method **a)** is that a changing system needs to be somehow detected and the controller gains need to be switched to the gains learned for the changed system. This set of controller gains needs to be learned in advance, thus depends on a design choice.
We have chosen method **b)** since it is additionally rather simple and computationally inexpensive, which is important since the adaptive framework will be implemented on real-time hardware. Moreover, the chosen approach is also modular, meaning that different estimators and controllers can be used without disturbing the adaptive control framework[1]. With this structure, controller requirements can be changed without changing the estimation algorithm at all, which is in general not possible when using learning algorithms since these are typically more integrated. Our approach is therefore more general and modular. Moreover, since there are no explicit model assumptions needed, our adaptive framework can be used for any robot, will it be electrically or hydraulically actuated. Different degrees of freedom are also possible. The same framework can, for example, be used for the legs of the robot HyQ as well as for its arms (currently under development).
Figure 5.1 shows the flow chart of the chosen adaptive control framework. Inputs (e.g. measurements and reference signals) are fed to the controller which then acts on the system

[1] Of course, the inputs and outputs of these modules need to be tailored to match, such that they fulfill the respective requirements of each block.

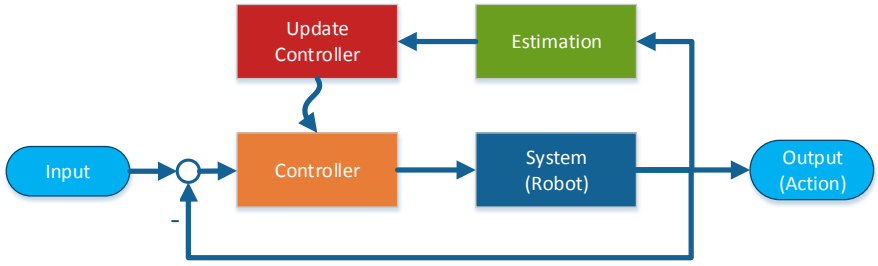

Figure 5.1: Flow chart of the adaptive control framework. Inputs (e.g. measurements and reference signals) are fed to the controller which then acts up on the system (robot). The estimator uses the state measurements to give an estimate of the system. This improved model knowledge is then used by the update algorithm to calculate the corresponding (new) controller gains, which are then sent to the controller, where the cycle starts again.

(robot) such that it performs the desired actions. The estimator uses the latest state measurements to improve estimate of the system (robot) at each time step. This improved model knowledge is then used by an update algorithm which calculates new controller gains. The controller is then updated using these new gains and the adaption cycle starts again.

5.1 Adaptive Control Framework - Controller

The previous section 3.3 derived a linear system model in terms of a fully measurable state vector, thus enabling full state feedback control without the need for an observer (e.g. a Linear Quadratic Gaussian (LQG) controller). Therefore, the first and most obvious choice is to use a Linear Quadratic Regulator (LQR) not only taking advantage of the fully measurable state but also (and very important) providing a guaranteed (minimum) phase margin[2] of $\pm 60°$ for all controlled inputs of the system [38, page 381]. This guaranteed phase margin has the advantage that the controller is more robust against estimation errors of the system matrices $\mathcal{A}$ and $\mathcal{B}$. Additionally, because of full state feedback, the LQR controller has the velocity compensation concept [11] already "built-in" because the velocity of the load $\dot{x}_l$ is part of the state.

There are only a few parameters that need to be tuned, namely the weighting matrices for the states Q and inputs R (sometimes the cross-weighting of states and inputs N is also considered). Of course, the number of parameters depends on the size of the state vector. Usually, it is sufficient to limit the weighting matrices to be diagonal matrices, therefore, greatly reducing the number of parameters. Once tuned, these parameters remain the same unaffected by system changes.

The final controller used in simulation and on the real-time hardware is a LQR controller augmented by a feedforward term and an integrator for the force channel.

More details about the structure of the controller and the calculation of the LQR gains are given in chapter 7 and section 8.2.2.

[2]Note that there is no such guaranteed phase margin for LQG controllers [15].

5.2 Adaptive Control Framework - Estimator

The estimation block (green) shown in figure 5.1 can consist of virtually any estimator as long as the inputs and outputs remain consistent with the framework. Actually, only the outputs need to be consistent with the requirements of the selected controller (red and orange blocks).
Again, due to the linear system model derived in section 3.3, which suggests a linear relationship between unknowns and measurements, a very simple estimator, that is, a (linear) recursive least squares (RLS) estimator, was selected. Details can be found in chapter 6. This simple estimator was chosen due to several reasons. These are:

- The unknown parameters, which are the elements of the system matrices, depend linearly on the measurements. Therefore, no additional model simplifications are needed to be able to write the system in terms of unknown parameters.
- The estimator is able to estimate a discrete-time single-input and single-output (SISO) system as well as a multiple-input and multiple-output (MIMO) system, hence the estimator is a rather general algorithm. The estimator can thus be used for any robot, will it be electric or hydraulically actuated, with any number of degrees of freedom.
- There are no model assumptions necessary, fixing the measurement vector is sufficient. There is only one single parameter, that is, the forgetting factor λ.
- The LQR controller requires a linear system, which the RLS estimator can provide directly.

5.3 Adaptive Control Framework - Update Logic

Figure 5.1 shows an additional block in red, which is the update logic responsible for calculating the new controller gains and, more importantly, deciding when to update the controller gains. As the time-varying system changes, the recursive estimator results in system descriptions mixing the response of the old system (before the system changes) with the response of the actual system (while the system changes), thus resulting in unreliable system estimates. The update logic is therefore used to allow the update of the controller gains only, when the system estimate has converged and the actual system is identified accurately enough. Moreover, when the system varies only little, the controller designed for the previously estimated system will still be able to give satisfactory performance due to the phase margin (robustness) of the LQR controller. Therefore, the controller gains do not need to be updated. An additional advantage of the update logic is that the computational demand can be kept low by only updating the controller gains when really needed. The update logic shall be kept as simple as possible, though some complexity is needed to give a robust adaptive control framework. The following subsection discusses the details about the chosen update logic, whereas the implementation details are discussed in subsection 8.2.1.

5.3.1 Adaptive Controller Update Logic in Detail

Diagram 5.2 lists four possible approaches on how to implement the update logic, these are:

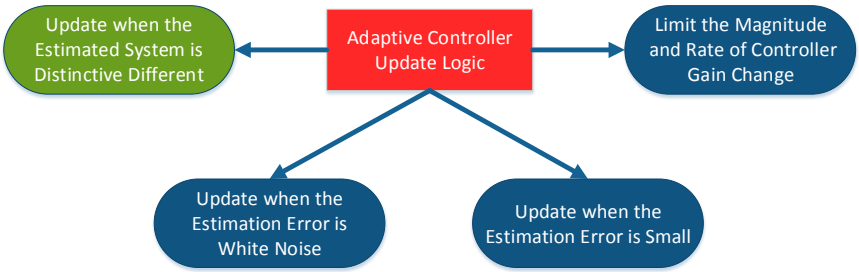

Figure 5.2: Four possible methods for implementing the adaptive controller update logic. The green shape is the method that is implemented on the real-time hardware and in simulation. Details can be found in subsection 8.2.1.

(a) Limit the magnitude and rate of controller gain changes

(b) Update when the estimation error is small

(c) Update when the estimation error is white noise

(d) Update when the estimated system is distinctively different

Each of these four methods has its advantages and disadvantages. We give a short reasoning on why method (d) seems to be the most appropriate for our scenario. The implementation of this method will be discussed in subsection 8.2.1. Method (a) is the most general approach, with the advantage that no additional information about the system or the estimator is needed. The controller gains are re-calculated every time step and the new controller is only applied to the system when the gains are bounded and change slowly, indicating that the system estimate has converged. The disadvantage is that there are many parameters[3] to tune and it is not straightforward to select appropriate bounds that will work in any circumstance. Method (b) has the advantage that it only depends on the estimator and not the actual controller since only the *error* of the estimation is considered. With *error* we mean the difference between measurement and predicted response of the estimated system. When this *error* is small, then the estimated system captures the essence of the system well and a new controller can be calculated. Method (c) extends method (b) by requiring that the error has to be white noise. This means that the controller is only updated when the estimator has extracted all useful information within the measurement and only noise remains (uncorrelated error). The advantage over method (b) is that not only the size of the *error* but also its quality is considered. A common disadvantage of all these three methods is, however, that many parameters need to be tuned and that it is difficult to select appropriate values that do not limit the generality of the algorithm. Therefore, method (d) is suggested having only a few parameters (three for our case) and additionally minimizing computational requirement. The idea of method (d) is to calculate and update the controller gains only when the estimated system is distinctively different enough, as will be discussed in subsection 8.2.1, from the current system the controller was calculated on. In essence, the controller is updated only when it is not able to deliver satisfactory performance and when the system estimate has converged to a new system description. The robustness capability of the controller is thus fully taken into account. The

[3]For our case there are two parameters for each controller component (LQR, LQR-I, Feedforward) and an additional convergence parameters, resulting in a total of seven parameters to tune.

advantages of this method (d) are that it minimizes the computational burden, considers the robustness of the controller, and is able to incorporate knowledge about the system and its estimator. Because of these reasons, method (d) is used in the implementation on the real-time hardware and in simulation.

Chapter 6

Recursive System Identification Algorithm

This chapter explains the RLS estimator (outlined in section 5.2) in great detail, including the derivation of the algorithm. This is followed by the discussion of the necessary steps needed to rewrite the state space system into a form applicable for the estimator. Simulation results assess the performance of the estimator for different input signals and the impact of the forgetting factor λ is studied.

6.1 Least Squares Algorithm

The general problem formulation is as follows: Given a linear relationship $y(t) = A(t) \cdot x(t)$, where $y(t)$ is the general measurement at time t, $x(t)$ is the input and $A(t)$ is the *linear* matrix, find the unknown elements of the matrix $A(t)$, which is assumed to be time-varying. Reformulating the problem into a linear equation relating the nth measurement to the unknown parameter vector Θ yields:

$$y[n] = H[n]\Theta + \omega[n] \tag{6.1}$$

where $y[n]$ is the nth measurement, $H[n]$ is the regression matrix relating the measurement to the parameter and $\omega[n]$ is the measurement noise. The noise is assumed to be zero-mean white noise. The measurement $y[n]$ is assumed to be a general multivariable vector. The parameter vector Θ is as long as the number of elements of the matrix $A(t)$. The prediction error, that is, the discrepancy between the latest measurement and latest "model" (the linear relationship between regression matrix and latest estimated parameters) is defined as:

$$\epsilon[n] = y[n] - H[n]\Theta \tag{6.2}$$

The error shall be minimized for some norm, where we choose the most simple euclidean distance norm $\frac{1}{2}\epsilon^T[n] \cdot \epsilon[n]$, since the minimization can then be calculated analytically by taking the derivative with respect to Θ. The minimum is where the derivative is equal to zero since the norm is convex. Hence, the "best" estimate of the parameter vector, for the

multivariable case, using all past measurements, is [27, page 366]:

$$\begin{aligned}\Theta[n] &= \arg\min_{\Theta} \frac{1}{2}\sum_{k=1}^{n} \epsilon^T[k]\cdot\epsilon[k] \\ &= \arg\min_{\Theta} \frac{1}{2}\sum_{k=1}^{n} \beta(n,k)\Big[y[k]-H[k]\Theta\Big]^T\Big[y[k]-H[k]\Theta\Big]\end{aligned} \tag{6.3}$$

where a weight $\beta(n,k)$ is introduced to reduce the influence of the old measurements since the system has changed in the meantime. Taking the derivative of equation (6.3) with respect to the unknown parameter Θ, setting it to zero and solving for the unknown gives:

$$\Theta_n = \left[\sum_{k=1}^{n}\beta(n,k)H^T[k]H[k]\right]^{-1}\cdot\sum_{k=1}^{n}\beta(n,k)H^T[k]y[k] \tag{6.4}$$

where Θ_n is the "best" estimate of the unknown parameters Θ using all past measurements up to time n. Using matrix notation the equation (6.4) can be shortened, that is:

$$\Theta = (H^TWH)^{-1}H^TWy \tag{6.5}$$

where the weight $\beta(n,k)$ has been transformed into a weighting matrix W. Vector y is now a vector of all past measurements and H is the full regression matrix connecting the unknown parameters to the measurement.

6.1.1 Weighted Recursive Least Squares Algorithm

To be able to run the estimator in real-time, we seek a recursive algorithm of equation (6.4). In [27, page 363], the recursive least squares equations for the single-variable case are derived. In here we follow the outlined derivation but derive the equations directly for the multivariable case.
The weight $\beta(n,k)$ is assumed to have the following property:

$$\begin{aligned}&\beta(n,k) = \lambda(n)\beta(n-1,k) \qquad 0\le k\le n-1\\ &\beta(n,n) = 1\end{aligned} \tag{6.6}$$

Because the system to be estimated is time-varying, we want to assign less weights to the old measurements, hence we require that the parameter, which is called *forgetting factor*, is smaller one: $\lambda(n)<1$. For the sake of simplicity, we additionally assume that $\lambda(j)\equiv\lambda$. Explicitly writing down the weight $\beta(n,k)$ for the the first values of n and k gives:

$$\begin{aligned}&\beta(1,0) = \lambda\beta(0,0) = \lambda\\ &\beta(2,0) = \lambda\beta(1,0) = \lambda^2 \qquad \beta(2,1) = \lambda\beta(1,1) = \lambda\\ &\beta(3,0) = \lambda\beta(2,0) = \lambda^3 \qquad \beta(3,1) = \lambda\beta(2,1) = \lambda^2 \qquad \beta(3,2) = \lambda\\ &\quad\vdots\\ &\beta(n,k) = \prod_{k=1}^{n}\lambda = \lambda^{n-k}\end{aligned} \tag{6.7}$$

Hence, the weight is exponentially decreasing for older measurements.

The first step of the derivation is to rewrite equation (6.4) by introducing two new variables defined as:

$$\Theta_n = R(n)^{-1} \cdot f(n) \tag{6.8}$$

$$R(n) = \sum_{k=1}^{n} \beta(n,k) H^T[k] H[k] \tag{6.9}$$

$$f(n) = \sum_{k=1}^{n} \beta(n,k) H^T[k] y[k] \tag{6.10}$$

Inserting equation (6.6) into (6.9) gives:

$$R(n) = \sum_{k=1}^{n} \lambda\beta(n-1,k) H^T[k] H[k] = \lambda\beta(n-1,n) H^T[n] H[n] + \underbrace{\sum_{k=1}^{n-1} \lambda\beta(n-1,k) H^T[k] H[k]}_{\lambda R(n-1)} \tag{6.11}$$

with $\beta(n-1,n) = \lambda^{n-1-n} = \lambda^{-1}$ we have:

$$R(n) = \lambda R(n-1) + H^T[n] H[n] \tag{6.12}$$

which is a recursive equation for calculating $R(n)$ based on the previous $R(n-1)$. Inserting (6.6) into (6.10), using similar calculations, yields:

$$f(n) = \lambda f(n-1) + H^T[n] y[n] \tag{6.13}$$

which is again a recursive equation for calculating $f(n)$ based on the previous $f(n-1)$. Inserting (6.13) into (6.8) gives:

$$\Theta_n = R(n)^{-1} \cdot \Big[\lambda f(n-1) + H^T[n] y[n]\Big] \tag{6.14}$$

using equation (6.8) to substitute $f(n-1)$ with $R(n-1) \cdot \Theta_{n-1}$ results in:

$$\Theta_n = R(n)^{-1} \cdot \Big[\lambda R(n-1) \cdot \Theta_{n-1} + H^T[n] y[n]\Big] \tag{6.15}$$

using equation (6.12) we have:

$$\begin{aligned} \Theta_n &= R(n)^{-1} \cdot \Big[\big(R(n) - H^T[n] H[n]\big) \cdot \Theta_{n-1} + H^T[n] y[n]\Big] \\ &= I \cdot \Theta_{n-1} - R(n)^{-1} H^T[n] H[n] \cdot \Theta_{n-1} + R(n)^{-1} H^T[n] y[n] \end{aligned} \tag{6.16}$$

Finally, the *recursive least squares equations* emerge as:

$$\Theta_n = \Theta_{n-1} + \underbrace{R(n)^{-1} H^T[n]}_{K(n)} \Big(y[n] - H[n] \cdot \Theta_{n-1}\Big) \tag{6.17}$$

which is a recursive equation for the latest estimate of the unknown parameter vector Θ_n based on the old estimate Θ_{n-1} corrected by the prediction error $y[n] - H[n] \cdot \Theta_{n-1}$ scaled by some gain $R(n)^{-1} H^T[n]$, which will be called the correction gain $K(n)$. In order to avoid the inversion of the matrix $R(n)$ at each time step, the following substitution is introduced [27, page 364]:

$$P(n) = R(n)^{-1} \tag{6.18}$$

The inversion can be calculated using the matrix inversion lemma (B.15). Hence,

$$\begin{aligned} P(n) &= \frac{R(n-1)^{-1}}{\lambda} - \frac{R(n-1)^{-1}}{\lambda} H^T[n] \Big[H[n] \frac{R(n-1)^{-1}}{\lambda} H^T[n] + I \Big]^{-1} H[n] \frac{R(n-1)^{-1}}{\lambda} \\ &= \frac{1}{\lambda} \Big[P(n-1) - P(n-1) H^T[n] \Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} H[n] P(n-1) \Big] \end{aligned} \tag{6.19}$$

which is again a recursive equation for calculating $P(n)$ based on the previous $P(n-1)$. The correction gain $K(n)$ can be written explicitly by inserting equation (6.19) giving:

$$\begin{aligned} K(n) &= R(n)^{-1} H^T[n] = P(n) H^T[n] \\ &= \frac{1}{\lambda} P(n-1) H^T[n] \Big[I - \Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} H[n] P(n-1) H^T[n] \Big] \\ &= \frac{1}{\lambda} P(n-1) H^T[n] \Big[\Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} \\ &\qquad \Big(\big[H[n] P(n-1) H^T[n] + \lambda \cdot I \big] - H[n] P(n-1) H^T[n] \Big) \Big] \\ &= P(n-1) H^T[n] \Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} \end{aligned} \tag{6.20}$$

where the correction gain $K(n)$ depends on the past $P(n-1)$ as well as the current regression matrix $H[n]$. This completes the derivation of the recursive least squares (RLS) algorithm which is given by the three equations (6.17), (6.19) and (6.20) with the single parameter, the forgetting factor λ. Additional interpretations of the RLS algorithm, that is, a Kalman algorithm and an instrumental variable method, are briefly given in appendix B.3, but not used any further in this thesis.

The Complete RLS Algorithm

The RLS estimation algorithm consists of the following equations, repeated here for convenience.

$$\boxed{\begin{aligned} K(n) &= P(n-1) H^T[n] \cdot \Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} \\ \Theta_n &= \Theta_{n-1} + K(n) \cdot \Big(y[n] - H[n] \cdot \Theta_{n-1} \Big) \\ P(n) &= \frac{P(n-1)}{\lambda} \cdot \Big(I - H^T[n] \Big[H[n] P(n-1) H^T[n] + \lambda \cdot I \Big]^{-1} H[n] P(n-1) \Big) \end{aligned}} \tag{6.21}$$

6.1.2 Calculating the Forgetting Factor λ

The choice of the forgetting factor depends on the system characteristics. In [27, page 378], an equation for calculating the forgetting factor is given for a system remaining approximately constant over N_0 samples:

$$N_0 = \frac{1}{1-\lambda} \tag{6.22}$$

where a suitable λ can be calculated. The influence of the forgetting factor on the conversion speed of the estimator is studied below in subsection 6.3.3.

6.1.3 Estimator Wind-Up or Persistence of Excitation

There is a phenomena called "estimator wind-up" [4, page 473], which occurs when the regression matrix $H[n]$ is all zero or close to zero. The reason is that the input signals do not persistently excite the system [19], hence the measurements contain too little information about the system. The consequence is that the gain matrix $K(n)$ tends to zero and thus the estimates are not being updated:

$$\Theta_n \approx \Theta_{n-1} \tag{6.23}$$

The matrix $P(n)$, however, increases steadily, since:

$$P(n) = \frac{1}{\lambda} \cdot P(n-1) \tag{6.24}$$

As soon as the regression matrix $H[n]$ becomes information rich again, the gain matrix $K(n)$ will be large, due to the large variance matrix $P(n)$, resulting in abrupt changes of the estimate Θ_n. This behavior can be suppressed by not updating the variance $P(n)$ in the case of near zero regression matrix $H[n]$.

6.2 Direct Discrete-Time State Space Identification

The basis for this identification idea is the discrete-time system given by equation (B.6) in appendix B, but shifted by one time step, since only measurements up to the recent time step k can be available in real-time due to causality.

$$\begin{aligned} x[k] &= \mathcal{A}[k-1] \cdot x[k-1] + \mathcal{B}[k-1] \cdot u[k-1] \\ y[k-1] &= \mathcal{C}[k-1] \cdot x[k-1] + \mathcal{D}[k-1] \cdot u[k-1] \end{aligned} \tag{6.25}$$

The fundamental idea is to realize that we want to estimate the four transition matrices describing the linear system. It is therefore possible to rewrite the system description (6.25) as a linear equation in terms of the unknown parameters which are all the elements of the transition matrices. This idea has been mentioned in the past, for example, in [29] where a discussion of a subspace method is presented for directly identifying linear models. However, we do not need to deploy subspace methods since we can measure the full state vector $\vec{x}$ along with output measurements $\vec{y}$.
The first line for the state $x_1[k]$ is written down explicitly to exemplify the idea of the rewrite process. The first line reads:

$$\begin{aligned} x_1[k] = \mathcal{A}_{11}[k-1] \cdot x_1[k-1] + \mathcal{A}_{12}[k-1] \cdot x_2[k-1] + \cdots + \mathcal{A}_{1n}[k-1] \cdot x_n[k-1] \\ + \mathcal{B}_{11}[k-1] \cdot u_1[k-1] + \cdots + \mathcal{B}_{1n_u}[k-1] \cdot u_{n_u}[k-1] \end{aligned} \tag{6.26}$$

where n is the number of states and n_u is the number of inputs. The unknowns are the coefficients of the transition matrices $\mathcal{A}$ and $\mathcal{B}$. The above equation can be rewritten into a matrix times vector notation since the relationship is linear in the unknown parameters,

thus:

$$\begin{pmatrix} x_1[k] \\ \vdots \end{pmatrix} = \begin{bmatrix} x_1[k-1] & \dots & x_n[k-1] & u_1[k-1] & \dots & u_{n_u}[k-1] \\ & \vdots & & & \vdots & \end{bmatrix} \cdot \begin{pmatrix} \mathcal{A}_{11}[k-1] \\ \vdots \\ \mathcal{A}_{1n}[k-1] \\ \mathcal{B}_{11}[k-1] \\ \vdots \\ \mathcal{B}_{1n_u}[k-1] \end{pmatrix} \tag{6.27}$$

The equations for additional states are similar, thus we can rewrite the system (6.25) as a set of equations in terms of the unknown coefficients of the transition matrices along with measurements of the states (current and past), inputs and outputs. This rewritten equation has a linear structure, given by:

$$Y = H \cdot \Theta \tag{6.28}$$

where H is the regression matrix (a large matrix containing all past measurements) relating the unknown parameter vector Θ to the latest measurement vector Y. The regression matrix H can be partitioned into four parts:

$$H = \begin{bmatrix} H_A & H_B & \mathcal{O} & \mathcal{O} \\ \mathcal{O} & \mathcal{O} & H_C & H_D \end{bmatrix} \tag{6.29}$$

where each part contains measurements of states or inputs, respectively. H_A contains the states, H_B the inputs, H_C the states visible at the output and H_D contains the inputs influencing directly the outputs. The matrix $\mathcal{O}$ stands for the zero-matrix of appropriate dimension. In more detail, the four individual matrices are:

$$H_A = \begin{bmatrix} x_1 & \dots & x_n & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & x_1 & \dots & x_n & \dots & 0 & \dots & 0 \\ & & & & \vdots & & & & & \\ 0 & \dots & 0 & & \dots & 0 & \dots & x_1 & \dots & x_n \end{bmatrix} \qquad (n \times n^2) \tag{6.30}$$

$$H_B = \begin{bmatrix} u_1 & \dots & u_{n_u} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & u_1 & \dots & u_{n_u} & 0 & \dots & 0 \\ & & & & \vdots & & & & \\ 0 & \dots & 0 & & \dots & & u_1 & \dots & u_{n_u} \end{bmatrix} \qquad (n \times n_u \cdot n) \tag{6.31}$$

$$H_C = \begin{bmatrix} x_1 & \dots & x_n & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ & & & & \vdots & & & & & \\ 0 & \dots & 0 & & \dots & 0 & \dots & x_1 & \dots & x_n \end{bmatrix} \qquad (m \times m \cdot n) \tag{6.32}$$

$$H_D = \begin{bmatrix} u_1 & \dots & u_{n_u} & 0 & \dots & 0 & 0 & \dots & 0 \\ & & & & \vdots & & & & \\ 0 & \dots & 0 & & \dots & & u_1 & \dots & u_{n_u} \end{bmatrix} \qquad (m \times n_u \cdot n) \tag{6.33}$$

where n is number of states, m is the number of measurements and n_u is the number of inputs. For short notation reasons, the time index $[k-1]$ of the states x_i and inputs u_i

within the matrices H_A, H_B, H_C and H_D has been omitted. The regression matrix H contains state and input measurements only from the past time step $[k-1]$.
The measurement vector Y contains the state and output measurements, that is, the left side of the system (6.25):

$$Y = \begin{pmatrix} x_1[k] \\ x_2[k] \\ \vdots \\ x_n[k] \\ y_1[k-1] \\ \vdots \\ y_m[k-1] \end{pmatrix} \tag{6.34}$$

where the state measurements from the current time step $[k]$ and the measured outputs from the previous time step $[k-1]$ are used. The parameter vector contains all unknown elements of the transition matrices $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$, thus

$$\Theta = \begin{pmatrix} \mathcal{A}_{11} \\ \mathcal{A}_{12} \\ \vdots \\ \mathcal{B}_{11} \\ \mathcal{B}_{12} \\ \vdots \\ \mathcal{C}_{11} \\ \mathcal{C}_{12} \\ \vdots \\ \mathcal{D}_{m,n_u \cdot n} \end{pmatrix} \tag{6.35}$$

which is a vector with a total number of unknowns of: $\left(n^2 + n \cdot n_u + m \cdot n + m \cdot n_u\right)$.

6.2.1 Specifying the Estimator for the Hydraulic System

The previous section derived the general RLS estimator algorithm, which can be used for any single-input and single-output (SISO) or multiple-input and multiple-output (MIMO) system. Now, we tailor the estimator algorithm to the specific needs of our hydraulic system, which is then used both in simulation and on the real-time hardware.
Since our simplified linear model, given by equations (3.51), shows that the state vector is fully measurable, only the first line containing the states, of the state space system (6.25), needs to be identified. Therefore, the measurement vector Y (6.34) of the RLS estimator only contains the state measurements, thus:

$$Y = \begin{pmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \end{pmatrix} = \begin{pmatrix} x_p[k] \\ \dot{x}_p[k] \\ F_s[k] \end{pmatrix} \tag{6.36}$$

Consequently, the regression matrix H (6.29) is only made up using the matrices (6.30) and (6.31). The regression matrix H uses state and input measurements from the previous time step $[k-1]$. The input used in simulation is the equivalent input u_v after the valve, as discussed in section 2.4, whereas on the real-time hardware, the input is the controller

output, that is, the input u to the valve. The unknown parameter vector Θ (6.35) contains only 12 elements which are the elements of the system matrices $\mathcal{A}$ and $\mathcal{B}$.

6.3 Estimator Performance in Simulation

The performance of the above discussed RLS estimator is studied by simulating the nonlinear hydraulic cylinder model given by equations (2.30). An overview of the simulation is given in appendix E. At each simulation time step, the latest input and output measurements are fed into the estimation block of the simulation. The linearization point is first subtracted from the signals and these are then normalized (pre-processed). The estimation performs the update, as given by equations (6.21), and outputs the latest estimates of the state space matrix elements. While the simulation is running, the system parameters, e.g. mass or friction of the load, are changed, mimicking the rapid changes in the system dynamics when the robot HyQ is, for example, performing a walking trot. Mass and friction of the load are changed according to figure 6.1, where the left axis shows the variation of the load mass m_l and the right axis shows the variation of the friction coefficient d_l plotted over the simulation time.

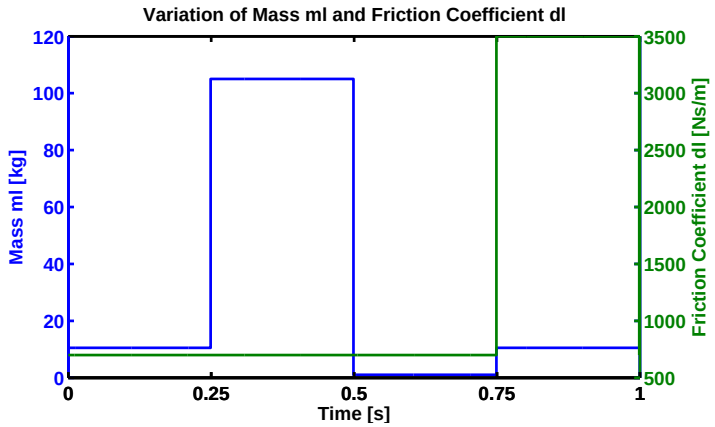

Figure 6.1: Variation of the model parameters for assessing the estimator performance in simulation. The left axis shows the variation of the load mass m_l and the right axis shows the variation of the friction coefficient d_l. The time (horizontal axis) corresponds to the simulation time. The chosen sequence of variation (changes in weight from low to high) tries to mimic a walking cycle of the HyQ robot.

The chosen sequence of variation tries to mimic a walking cycle of the HyQ robot, that is, during the first section the weight is low (mimicking a leg in the air) followed by a heavy weight section (mimicking a sudden touch down where the leg is in stance phase). The weight is then quickly reduced again (mimicking again a leg in the flight phase), followed by the last section where the weight is only increased little, but the friction coefficient greatly to mimic a sticky environment, such as mud or snow. The idea is to mimic a situation where the leg does not yet support the full weight of the robot, but tries to find good footholds within a high friction environment. The duration of the sections corresponds to a walking cylce of 2 Hz [36], which results in a stance phase duration of 0.25 s.
The following sections analyse the performance of the estimator for various input signals, while the system is changing according to figure 6.1. The simulation, using the nonlinear model, represents an idealized environment where there are no disturbances, the operating

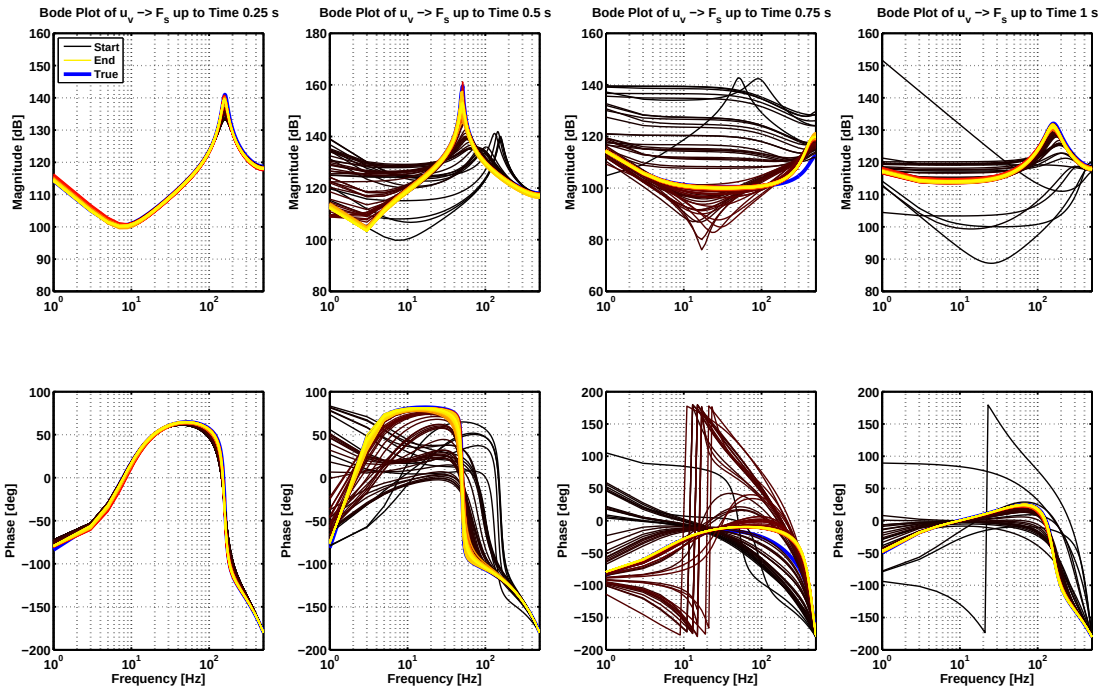

Figure 6.2: Bode plot of the load cell force transfer function for each section of the figure 6.1. The true transfer function within the sections is drawn in blue, while the estimates are drawn in colors ranging from black to yellow for increasing time. The input signal is zero-mean white noise with an amplitude of 5 % of the nominal valve opening.

point of the states as given in chapter 3 and the measurements are perfect (no noise, no phase delay). The following simulation results all use a forgetting factor of $\lambda = 0.85$.

6.3.1 Estimator Performance for Ideal Input Signals

Figure 6.2 shows the estimation performance of the RLS estimator when subjected to ideal valve input signals having rich frequency content and not exciting the nonlinearity of the system (see subsection 4.3.1), that is, the inputs have low amplitude. The input signal used in this section is zero-mean white noise with an amplitude of $\pm 5\,\%$ of the nominal valve opening. The Bode plot is divided into sections (columns) corresponding to the sections of figure 6.1. The time stated in the column titles refers to the end time of the respective section. The columns display the transfer functions, for each time step, starting at the end time of the previous section till the end time of its respective section. The color of the transfer function varies accordingly from black (initial system within the section), via red to yellow (final transfer function of the section), whereas the true transfer function of the system, according to figure 6.1, is shown in blue. These transfer functions are calculated during post-processing by first creating state space matrices $\mathcal{A}$ and $\mathcal{B}$ from the parameter vector Θ and then calculating the transfer function from the input u_v to the load cell force F_s measurement.

When examining figure 6.2, several things are worthy to note: When the system is not changing (column one), the estimates do not diverge from the truth and their variance is very low. As soon as the system changes (column two), the estimates start to converge to the new true transfer function and are converged within 50 ms. The convergence speed

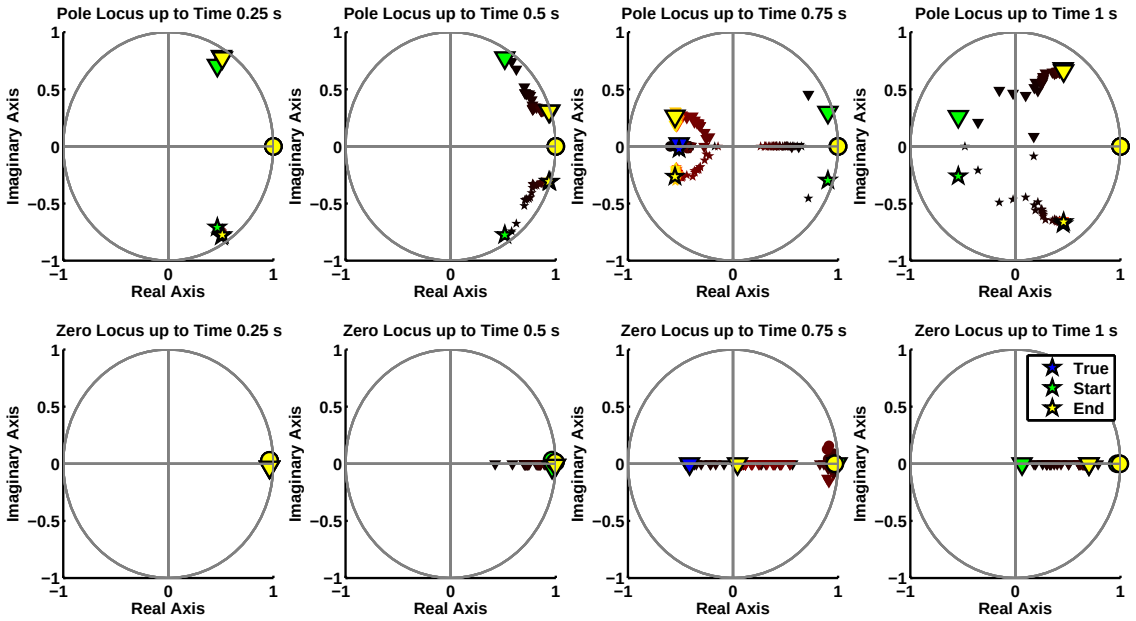

Figure 6.3: Pole and zero locus, where the top row shows the poles (or eigenvalues) of the estimated system matrix $\mathcal{A}$, whereas the second row shows the zeros associated with the transfer function from input to the measured force, as shown in the Bode plot 6.2. The color scheme is also kept similar, with the addition of green markers highlighting the poles and zeros at the beginning of the sections.

depends on the forgetting factor λ, which is discussed below in subsection 6.3.3. Similar observations hold for the other two columns. The estimates of column three are converged after 60 ms and those of column four after just 25 ms. The third column shows that there might be an issue with aliasing effects since the magnitude of the transfer function at the Nyquist frequency has not yet decayed. Which means that when these high frequencies of the system are excited, the measurements will also contain some high frequency[1] content close to the Nyquist frequency, which then results in aliasing effects diminishing the estimator performance. Hence, doubling the current sampling frequency of $T_s = 1000\,\mathrm{Hz}$ might improve the estimation performance in general.

The pole locus plot 4.10 shown within the parameter analysis of subsection 4.3.2 revealed that the poles are pushed closer to the stability boundary for increased mass, a fact that is not readily visible when examining Bode plots. Therefore, pole zero locus plots are also studied in order to judge the estimator performance. Such a plot is shown in figure 6.3, where the same column layout is kept. The top row shows the poles (or eigenvalues) of the estimated system matrix $\mathcal{A}$, whereas the second row shows the zeros associated with the transfer function from input to the measured force, as shown in the Bode plot 6.2. The color scheme is also kept, with the addition of green markers highlighting the poles and zeros at the beginning of the section. There are three different markers for each pole and zero.

Additional insights can be gathered from this pole zero locus plot. The second column reveals how close the complex conjugate poles come to the stability boundary. Thus, it can happen that the estimator returns a system with poles just slightly outside the stability

[1] A continuous-time filter (anti-aliasing filter) can be used before sampling the signals to prevent such aliasing effects on the measurements.

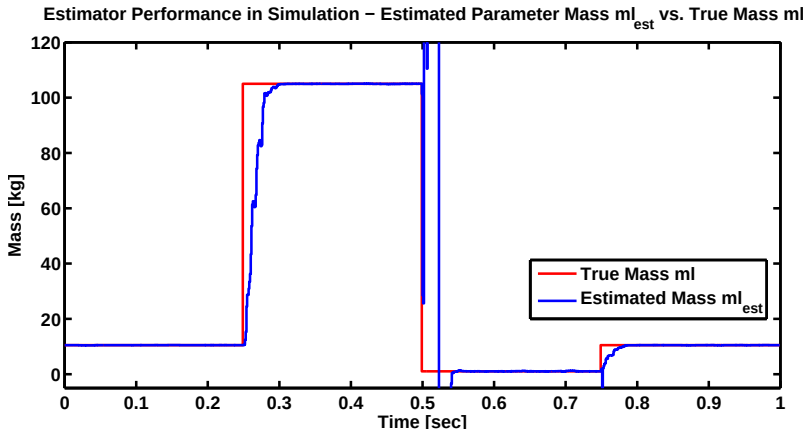

Figure 6.4: True load mass m_l (red) according to figure 6.1 compared to the extracted load mass $m_{l\,est}$ (blue) from the estimated state space matrices whose transfer function is shown in the Bode plot of figure 6.2.

boundary, caused by noise or numerical issues. Hence, either a constrained estimator or a mapping of the unstable poles into the stable region might be incorporated. This work implemented the latter approach. Column three reveals more clearly that the estimator was not able to exactly estimate the poles, a fact that is only barely visible in the Bode plot of figure 6.2.

So far the (overall) accuracy of the estimated systems (or transfer functions) against the true system has been shown. Let us now consider the performance of the estimator by comparing a single parameter, that is, we compare the true load mass m_l to the estimated $m_{l\,est}$. Figure 6.4 shows this comparison. The estimated mass needs to be extracted from the estimated state space matrices during post-processing. Several things are worthy to note: Firstly, once the estimated mass has converged, it does not diverge, as can clearly be seen, for example, within the intervals $0\,\mathrm{s} - 0.25\,\mathrm{s}$ or $0.3\,\mathrm{s} - 0.5\,\mathrm{s}$. Secondly, the convergence of the estimated mass $m_{l\,est}$, starting at time $0.25\,\mathrm{s}$, is rather smooth and fast, in contrast to the almost chaotic response after the time $0.5\,\mathrm{s}$. The estimation appears to be superior in case the system mass is increased, rather than decreased. Thirdly, the single black transfer function starting at $150\,\mathrm{dB}$ and diagonally crossing the Bode plot in column four of figure 6.2 might be related to the sharp negative peak visible in the estimated load mass just after the system dynamics have changed at $0.75\,\mathrm{s}$. These sharp peaks might be due to the fact that the estimator needs to explain sudden changes in the system dynamics. Fourthly, the convergence speed of the estimated mass compared to the convergence speed of the transfer function seems to be about the same.

6.3.2 Estimator Performance for Non-Ideal Input Signals

So far, valve input signals having a rich frequency spectrum and low amplitude have been considered. In this section, we consider non-ideal signals having a low frequency spectrum and/or high amplitude that do not excite all the frequencies and/or excite the nonlinearities of the system. Estimating a system using such non-ideal signals is more difficult. In reality, these are the signals that the estimator has to deal with. Figure 6.5 shows the estimated transfer functions of a simulation using the exact same parameters as used in the previous subsection, but the input amplitude of the white noise signal is increased to the full range

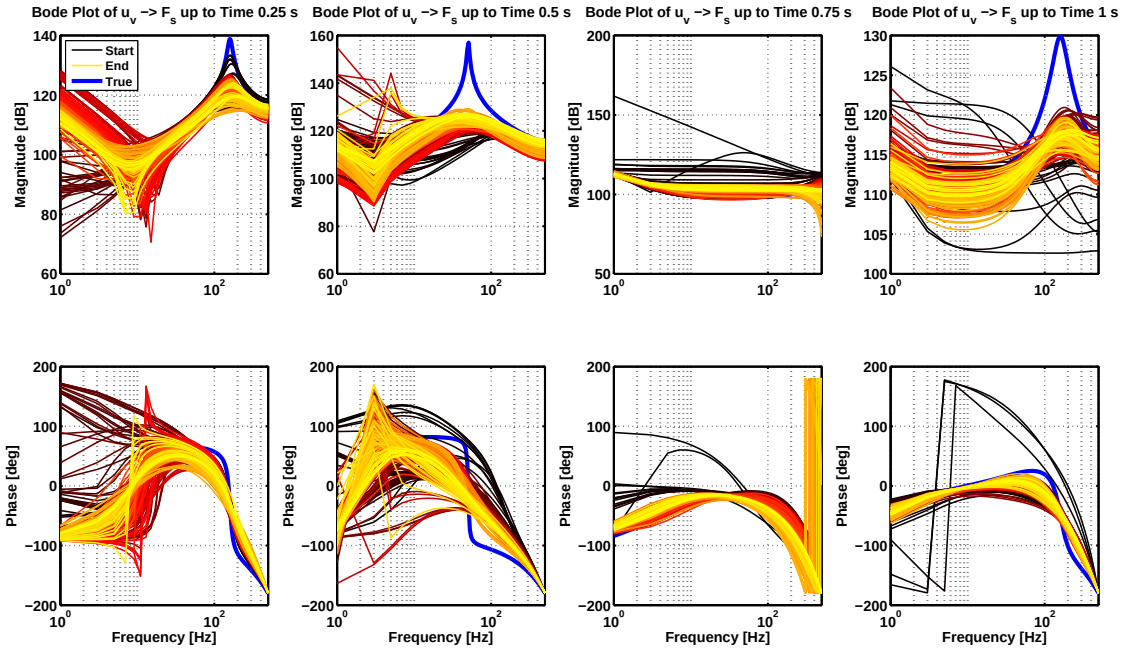

Figure 6.5: Bode plot of the load cell force transfer function for each section of the figure 6.1. The true transfer function within the sections is drawn in blue, while the estimates are drawn in colors ranging from black to yellow for increasing time. The input signal is zero-mean white noise with an amplitude of 100 % of the nominal valve opening, thus exciting the nonlinearity of the system.

of the valve, that is: $u_v = \pm 100\%$. When comparing figure 6.5 to 6.2, it is immediately evident that the estimator is no longer able to correctly identify the system dynamics. The hydraulic system is now operated over its full operation range. As has been discussed in subsection 4.3.1, the system response changes from time step to time step (shown in figure 4.7), hence the estimator "sees" this family of responses. Since the currently used estimator is a linear estimator, the best linear approximation is a system lying within this family of responses, hence the estimates shown in figure 6.5. Therefore, the estimator is not able to identify the resonance peaks since there is too little data within the measurements. The nonlinearity and transient responses due to the high amplitude also increase the variance of the estimates. A similar Bode plot for a multisine input signal with frequencies $f = [2, 10]Hz$ is shown in appendix C.1 by figure C.2.

To summarize the following conclusions about the performance of the linear estimator can be drawn. Table 6.1 lists the main findings for all combinations of frequency and amplitude of the input signal. The combination of signals with low frequency content and high amplitude is not shown, but was verified by simulation. The linear estimator performs very well for input signals having rich frequency content and low amplitude, exciting only the system dynamics behaving linearly. The performance decreases when high amplitude signals are used since these excite the nonlinearities of the system resulting in blurred estimates. Low frequency content signals do not excite the system enough to give accurate estimates. Signals with low frequency content and high amplitude excite the nonlinearities while not exciting enough frequencies of the system, hence the worst performance can be expected.

Due to this extensive analysis, we can conclude that the chosen estimator may not be the right choice for our system, hence other estimators may be used in the future, such as

Performance of the Estimator	Low Frequency Content	Rich Frequency Content
Low Amplitude	*limited*	*good*
High Amplitude	*limited*	*acceptable*

Table 6.1: Performance of the linear estimator for each combination of frequency and amplitude of the input signal. Low frequency signals do not excite the system enough to give accurate estimates and high amplitude input signals excite the nonlinearities in the system resulting in blurred estimates.

nonlinear estimators that are able to cope with the high nonlinearities in the system as has been analysed in section 4.3.

6.3.3 Influence of the Forgetting Factor λ

The only parameter of the recursive estimation algorithm, given by the equations (6.21), is the forgetting factor λ, which essentially determines how much of the past data should be forgotten. The forgetting factor λ plays an important role, since, firstly it determines how fast the system estimates converge when an abrupt change in the system dynamics occurs. Secondly, it determines the sensitivity of the estimates to noise or disturbance. Generally speaking, when the forgetting factor is low < 0.8, then the estimates converge quickly to the new system, but the variance of the estimates is high when the system is not changing. In contrary, when the forgetting factor is high > 0.95, the convergence is slow, however, noise or disturbance have little impact on the estimates. Moreover, the estimates have a low variance when the system is not changing. This results in a trade-off between fast convergence and robustness (against noise and disturbances). The influence of the forgetting factor was studied by simulating the nonlinear system given by equations (2.30) using the recursive least squares algorithm given by equations (6.21). Four different forgetting factors $\lambda = \{0.8, 0.85, 0.9, 0.98\}$ were considered. For each parameter, the simulation was initialized with a load mass of $m_l = 105\,\text{kg}$, while the estimator was initialized with the nominal load mass of $m_l = 10.5\,\text{kg}$. The duration of the simulation was set to $T_{sim} = 0.15\,\text{s}$ and the valve input signal was white noise with an amplitude of $u_v = 5\%$. Figure 6.6 shows the Bode plot of the load cell force transfer functions for the four different forgetting factors. The convergence speed using a low forgetting factor (top left) is higher compared to using a high forgetting factor (bottom right). As soon as the estimates have converged, the variance of the estimates starts to increase more for a low forgetting factor (top left) than for a high forgetting factor (bottom left). The advantage of using a very high forgetting factor, e.g. $\lambda = 0.98$ (bottom right), is that there is a smooth transition from one estimate to the next, which can be useful when calculating controller gains directly based on the latest estimates without using some update logic[2] in between.
The convergence speed of the estimator can be more clearly seen by extracting the estimated mass $m_{l\,est}$ and comparing it to the true mass m_l. Figure 6.7 shows the estimated and true mass for the four forgetting factors. The same observations about the convergence speed also hold for this figure. The estimated mass using a forgetting factor of $\lambda = 0.8$ con-

[2]Details about the update logic used in this work can be found in section 8.2.1.

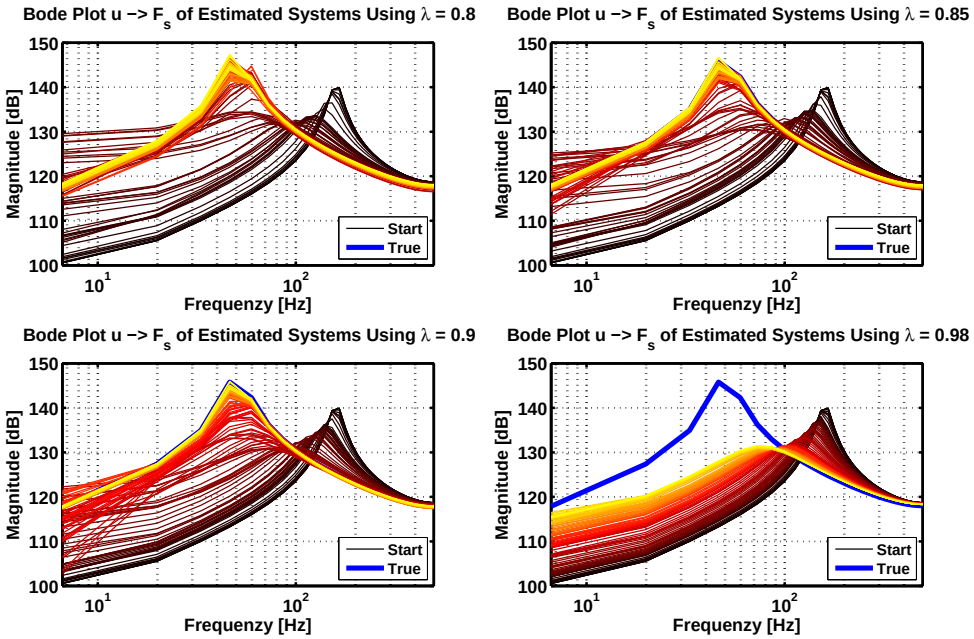

Figure 6.6: Bode plot of the load cell force transfer functions for four different forgetting factors $\lambda = \{0.8, 0.85, 0.9, 0.98\}$ using simulation that was initialized with a load mass of $m_l = 105\,\text{kg}$, while the estimator was initialized with the nominal load mass of $m_l = 10.5\,\text{kg}$. The convergence speed using a low forgetting factor (top left) is higher compared to using a high forgetting factor (bottom right). As soon as the estimates have converged, the variance of the estimates starts to increase more for a low forgetting factor (top left) than for a high forgetting factor (bottom left).

verged after just 50 ms, whereas for a forgetting factor of $\lambda = 0.9$ almost 90 ms are needed. Note the very slow convergence speed in case of a very high forgetting factor $\lambda = 0.98$.
In practice the forgetting factor must be adopted to the situation and the system. There are some guidelines that help to find an appropriate forgetting factor, e.g. the equation (6.22) introduced above. The usual trotting frequency [36] of HyQ is at around 2 Hz. Assuming a "worst-case" trotting frequency for the HyQ robot of about 4 Hz, the leg will be on the ground during 0.125 s. Since we aim to use the improved knowledge of the estimator while the leg is still on the ground, we require that the estimates converge within (at least) the first quarter of the stance phase. Using equation (6.22), this requirement results in a forgetting factor of $\lambda = 0.968$. When considering again figure 6.7, one can see that the estimates, using this calculated forgetting factor, might not have converged within the required time. A forgetting factor of $\lambda = 0.8$ is more realistic compared to this theoretical value. However, during the real-time experiments, it has been found that the forgetting factor needs to be increased in order to reduce the influence of the noise, thus a good compromise was found by using a forgetting factor of $\lambda = 0.9$.
From this discussion the idea arises that the estimates might be improved by using an adaptive scheme for the forgetting factor, rendering it time-dependent $\lambda(t)$. Reference [27, p. 379] states this idea and suggests that a low forgetting factor might be used as soon as a sudden change is sensed, e.g. the leg touching the ground, resulting in a quick convergence. As soon as the estimates have converged, the forgetting factor can then be increased again in order to improve robustness against noise and disturbances.

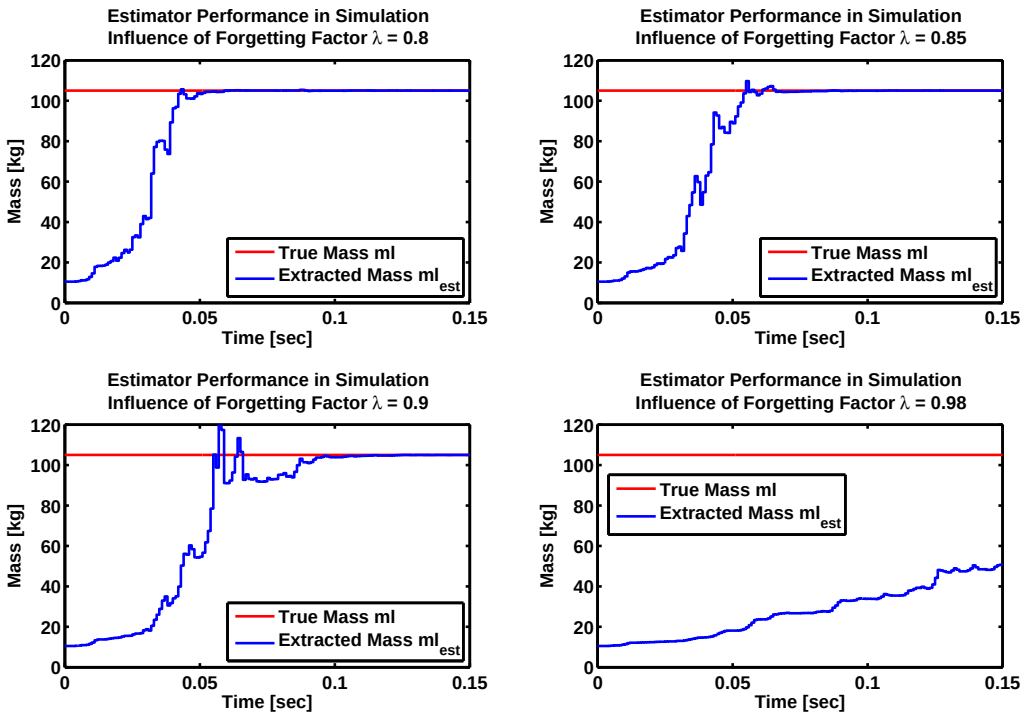

Figure 6.7: Comparison of estimated mass $m_{l\,est}$ extracted from the system estimates against the true mass $m_{l\,true}$, which is shown as a red line. The estimated mass using a forgetting factor of $\lambda = 0.8$ has converged after just 50 ms, whereas for a forgetting factor of $\lambda = 0.9$ almost 90 ms are needed. Notice the very slow convergence speed in case of a very high forgetting factor $\lambda = 0.98$.

Chapter 7

Adaptive Control Synthesis

This chapter explains the details of the LQR controller introduced in section 5.1. The influence of the weighting matrices is discussed, followed by a study of the impact of the system variations on the controller gains. Simulation results assess the performance of the LQR controller. This chapter introduces the basic equations, the implementation can be found in section 8.2.

7.1 LQR Controller Layout

The chosen controller is a LQR (linear quadratic regulator) controller with feedforward term and integrator. Figure 7.1 shows the LQR controller layout. The inputs are the reference $r[k]$ and measurements $y[k]$ shown in red, at time step k. These signals are first normalized by the blue triangles using the normalization matrix T defined by (3.52). The feedforward term Γ (top orange triangle) acts directly on the reference signals $r[k]$ for increased transient response. The LQR gain K (middle orange triangle) acts on the tracking error $e[k] = r[k] - y[k]$. A discrete-time integrator with anti-reset windup (green blocks) is implemented, preventing integrator overflow when the system reaches the actuator limits. The integrator gain K_I (bottom orange triangle) is extended by an additional integrator gain xi not affected by the LQR weighting matrices. This tuning factor can be used to tune the integrator action without having to change the weighting matrices, thus this gain is system independent. These three parts are then summed up and denormalized by the matrix V defined by (3.53). The total controller output $u[k]$ is then applied to the system.

7.2 LQR Controller Equations

The LQR controller is a linear state feedback controller whose gain K is calculated using the solution of the discrete-time algebraic Riccati equation (DARE) for some system matrix and some weighting matrices. The discrete-time system is:

$$x[n+1] = \mathcal{A} \cdot x[n] + \mathcal{B} \cdot u[n] \tag{7.1}$$

where x denotes the state, u is the input and n is the current discrete-time step. The two matrices $\mathcal{A}$ and $\mathcal{B}$ describe the linear system. The resulting linear state feedback controller is:

$$u[n] = -\mathcal{K} \cdot x[n] \tag{7.2}$$

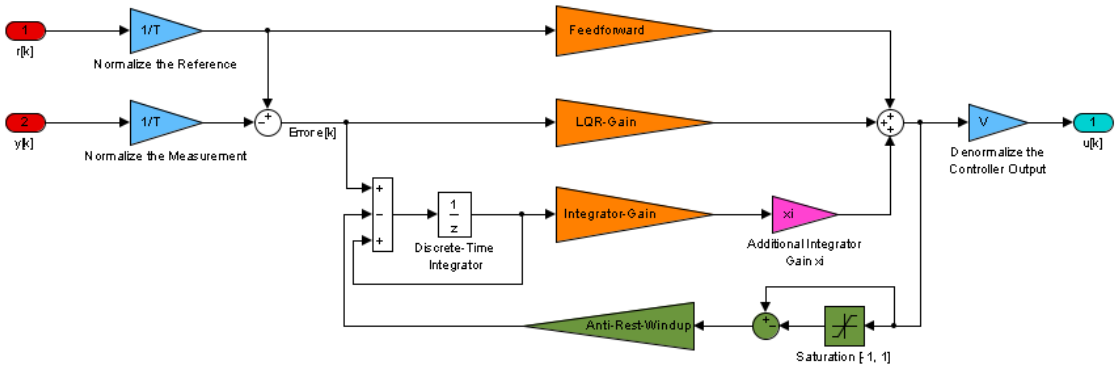

Figure 7.1: Layout of the LQR controller with LQR gain K, feedforward term Γ and integrator gain K_I all shown in orange. The inputs are the reference $r[k]$ and measurements $y[k]$ shown in red, at time step k. These signals are first normalized by the blue triangles. The discrete-time integrator, with additional gain xi (magenta), is implement with an anti-reset windup (green blocks) preventing integrator overflow when the system reaches the actuator limits. The total controller action is then denormalized by the right blue triangle. The controller output $u[k]$ is then fed to the actuator of the system.

where $\mathcal{K}$ is the LQR-gain matrix. The linear quadratic regulator (LQR) minimizes the cost function [2]:

$$J(u) = \sum_{n=0}^{\infty} x[n]^T \cdot Q \cdot x[n] + u[n]^T \cdot R \cdot u[n] + 2 \cdot x[n]^T \cdot N \cdot u[n] \tag{7.3}$$

where Q is the weighting matrix for the states, R is the weighting matrix for the inputs and N is the weighting matrix for the cross-coupling of state and input. The finite-time solution to this optimization problem is the DARE equation [7]:

$$\mathcal{A}^T \cdot S \cdot \mathcal{A} - S - \left(\mathcal{A}^T \cdot S \cdot \mathcal{B} + N\right) \cdot \left(\mathcal{B}^T \cdot S \cdot \mathcal{B} + R\right)^{-1} \cdot \left(\mathcal{B}^T \cdot S \cdot \mathcal{A} + N^T\right) + Q = 0 \tag{7.4}$$

where the Riccati matrix S is its solution. The LQR-gain matrix is the minimizer of this equation and is calculated as:

$$\mathcal{K} = \left(\mathcal{B}^T \cdot S \cdot \mathcal{B} + R\right)^{-1} \left(\mathcal{B}^T \cdot S\right) \tag{7.5}$$

The input $u[n]$ is the *optimal control* signal that regulates the state to zero while minimizing the cost function $J(u)$. We are, however, interested in tracking the reference, thus the input $x[n]$ of equation (7.2) is substituted by the tracking error, that is:

$$u[n] = -\mathcal{K} \cdot x[n] = -\mathcal{K} \cdot (r[k] - y[k]) = -\mathcal{K} \cdot e[k] \tag{7.6}$$

Hence, the tracking error is regulated to zero.

7.2.1 Augmented LQR Equations for Calculating the Integrator Gain

The integrator gain K_I is calculated by augmenting the system (7.1) by additional states for the integrators (ρ = number of additional states) according to [22]:

$$\begin{aligned} \tilde{\mathcal{A}} &= \begin{bmatrix} I & -\mathcal{C} \\ \mathcal{O} & \mathcal{A} \end{bmatrix} \\ \tilde{\mathcal{B}} &= \begin{bmatrix} \mathcal{O} \\ \mathcal{B} \end{bmatrix} \end{aligned} \tag{7.7}$$

where I is the diagonal unity matrix and $\mathcal{O}$ is the zero matrix. These augmented system matrices are then used by the above equations to give the LQR-gain $\mathcal{K}$, which then contains both the gain for the states and the gain for the integrator, that is:

$$\mathcal{K} = \begin{bmatrix} K_I & K \end{bmatrix} \tag{7.8}$$

The weighting matrices used in equation (7.3) need to be augmented accordingly to accommodate for the additional integrator states.

7.2.2 Specifying the LQR Controller for the Hydraulic System

The same specifications, as introduced in subsection 6.2.1 for the estimator, are used for the LQR controller since it relies on the estimated system matrices $\mathcal{A}$ and $\mathcal{B}$. Hence, the same three states along with three reference values are used. There is one controlled input to the valve of the hydraulic system. To get rid of steady-state force tracking errors an additional integrator $\rho = 1$ for the force channel is introduced.

7.2.3 Influence of the Weighting Matrices

The three weighting matrices Q, R and N are the tuning factors of the LQR controller; they define the relative weight of the state, input and cross-coupling of state and input in the cost function given by equation (7.3). These weighting matrices are typically defined as diagonal matrices, however, a dense matrix can sometimes give some improvements in control performance. For simplicity, we assume diagonal matrices and additionally set the cross-coupling weight N to zero. Hence, we end up with a total number of $n + \rho + n_u + xi = 6$ parameters to tune. These were tuned using simulation by comparing the control performance for various reference signals like steps or multisines. The following weighting matrices were found to give the best compromise between tracking performance and minimum input amplitude.

$$Q = \begin{bmatrix} 2.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5000 \end{bmatrix} \tag{7.9}$$

$$R = 0.1 \tag{7.10}$$

$$xi = 6 \tag{7.11}$$

The weight on the force channel (last row of Q) is several magnitudes higher than the weight for the position x_p and velocity $\dot{x}_p$ since we are particularly interested in good force tracking. The low weights on the position and velocity contributions can be interpreted in the sense that the controller is allowed to use these states to improve the force tracking. That is, the velocity of the cylinder is allowed to be increased to give better force tracking. This relationship is of course directly related to the velocity compensation concept [11], where an additional velocity depended gain is used to improve force tracking. Since our hydraulic system test set-up has springs, a change in position directly relates to force, hence the position is also used to improve force tracking. After the tracking performance was satisfactory, the weight for the input energy was increased with the goal of minimizing the occurrence of controller outputs reaching the actuator limits. The weighting for the integrator state and its additional gain were tuned to give zero steady-state tracking error.

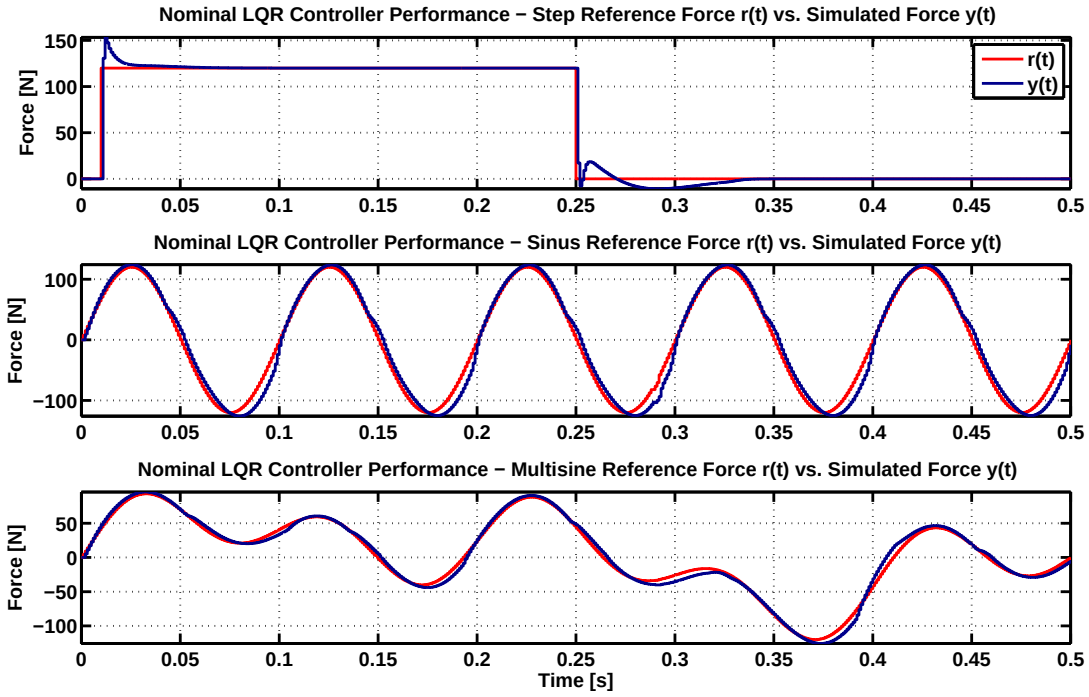

Figure 7.2: Nominal force tracking performance of the LQR controller with feedforward and integrator. Shown are on top a step, in the middle a sine (frequency $f = 10\,\text{Hz}$) and at the bottom a multisine (frequencies $f = [2, 5, 10\,]Hz$) force reference signal. The controller was designed using the weighting matrices (7.9)-(7.11) for the system parameters of the real-time hardware FC1D as given in appendix section A.4.

7.3 LQR Controller Performance in Simulation

The force tracking performance is shown in figure 7.2 where on top a step, in the middle a sine (frequency $f = 10\,\text{Hz}$) and on the bottom a multisine (frequencies $f = [2, 5, 10\,]Hz$) are shown. As can be seen, the force tracking is very good. There is a slight overshoot in the step response due to the high feedforward gain. The integrator eliminates the steady-state error of the step response. There is a trade-off between good step response without overshoot and tracking of the sinusoidal reference signals without phase delay. A high gain for the force tracking error and feedforward decrease the phase delay, but result in overshoot in the step response. Figure 7.3 shows the controller output signals associated with the three responses shown in figure 7.2. All three contributions (blue, green and red) as well as the total control output (black) are shown for the three different reference signals. Note that the main contribution to the total controller output stems from the LQR-gain K for the states. The integrator part K_I is only used to reduce steady state error. The feedforward part Γ helps to reduce the phase delay and increase transient response. The controller output is shown in the unit milliampere and the maximum valve input is $u_{v\,max} = 10\,\text{mA}$.

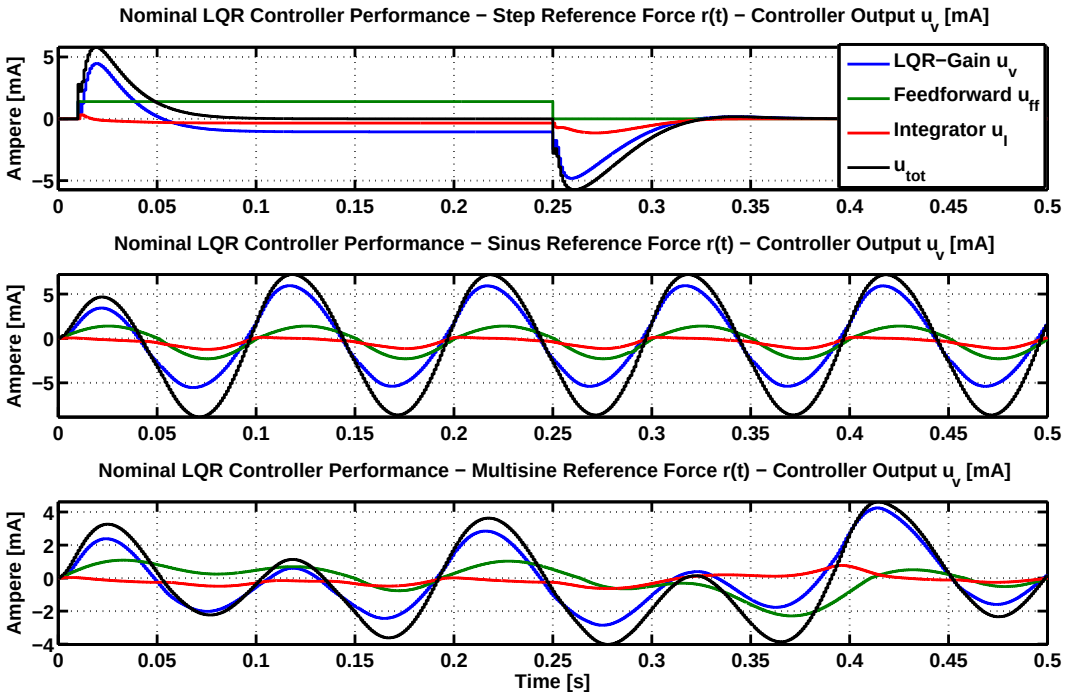

Figure 7.3: LQR controller output signals for the nominal force tracking performance shown in figure 7.2. All three contributions (blue, green and red) as well as the total control output (black) are shown for the three different reference signals. Note that the main contribution to the total controller output stems from the LQR-gain K. The integrator part K_I is only used to reduce steady state error. The feedforward part Γ helps to reduce the phase delay. The controller output is shown in unit milliampere and the maximum valve input is $u_{v\,max} = 10\,\mathrm{mA}$.

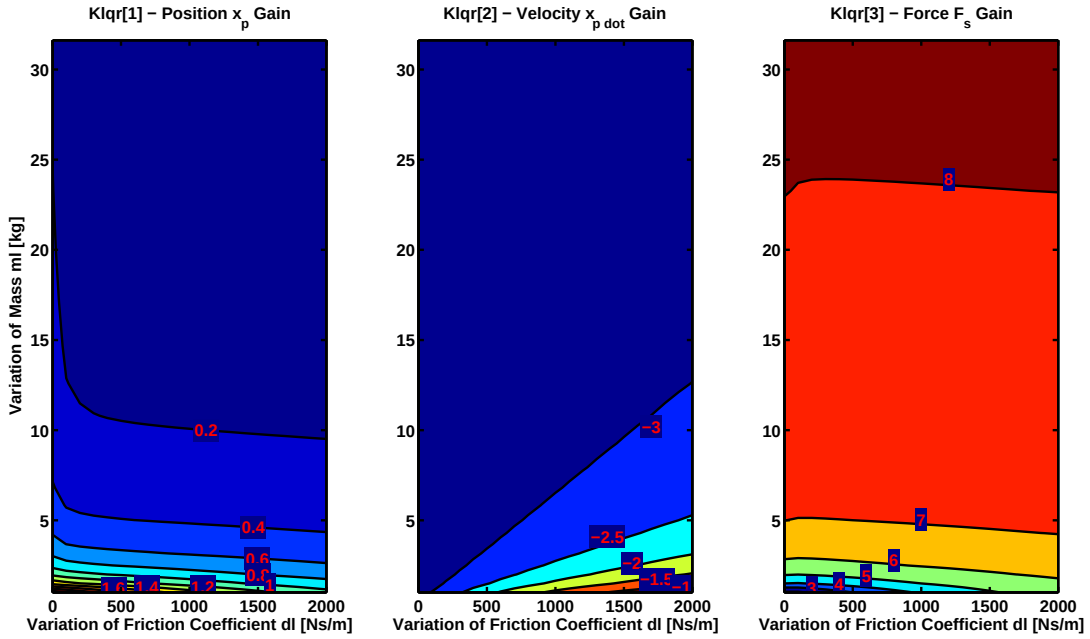

Figure 7.4: LQR controller gain map for simultaneous variations of the system parameters, that is, the load mass m_l and friction coefficient d_l. The weighting matrices defined in subsection 7.2.3 are used. Only the gains for the states are shown. Integrator and feedforward gains are not shown. Increasing mass reduces the gain for the position channel (first column), but increases the gains for both the velocity and force channel (second and third column). Increasing friction coefficient means that the velocity gain can be decreased because of the higher friction forces.

7.4 LQR Controller Gain Maps

This section studies the influence of system parameter variations on the LQR controller gains. These are analysed by varying two parameters at a time while keeping the other fixed at the respective nominal value. The parameters are: the load mass m_l, the friction coefficient d_l and the spring stiffness C. The goal of this study is to give insights on how and by how much the gains of the controller change when the underlying system changes thereby visualizing the impact of the three parameters on the controller gains. These are calculated using the weighting matrices defined in subsection 7.2.3. The controller gains, due to variations of both load mass m_l and friction coefficient d_l, are shown in figure 7.4. The first column shows the gain for the position channel, the second for the velocity channel and the third for the force channel. Only the LQR gains K for the states are shown, no integrator and feedforward gains.

When examining the gain map, it can be seen that increasing the friction coefficient has a negligible effect on the position gain and lighter mass requires a higher gain. The position gain mainly depends on the spring stiffness, hence the relatively small variations. The velocity gain (second column), however, depends on the friction coefficient since higher coefficients result in higher friction forces, hence less control input is required. The gains also increase (in absolute value) for increased mass since higher mass again means more velocity compensation. Note that the gains are negative, which, multiplied by the negative load velocity feedback, results in control commands that are actually positive. This is of course exactly the concept of load velocity feedback as outlined in [11]. The LQR controller has inherent velocity compensation since it uses the insights from the system model and

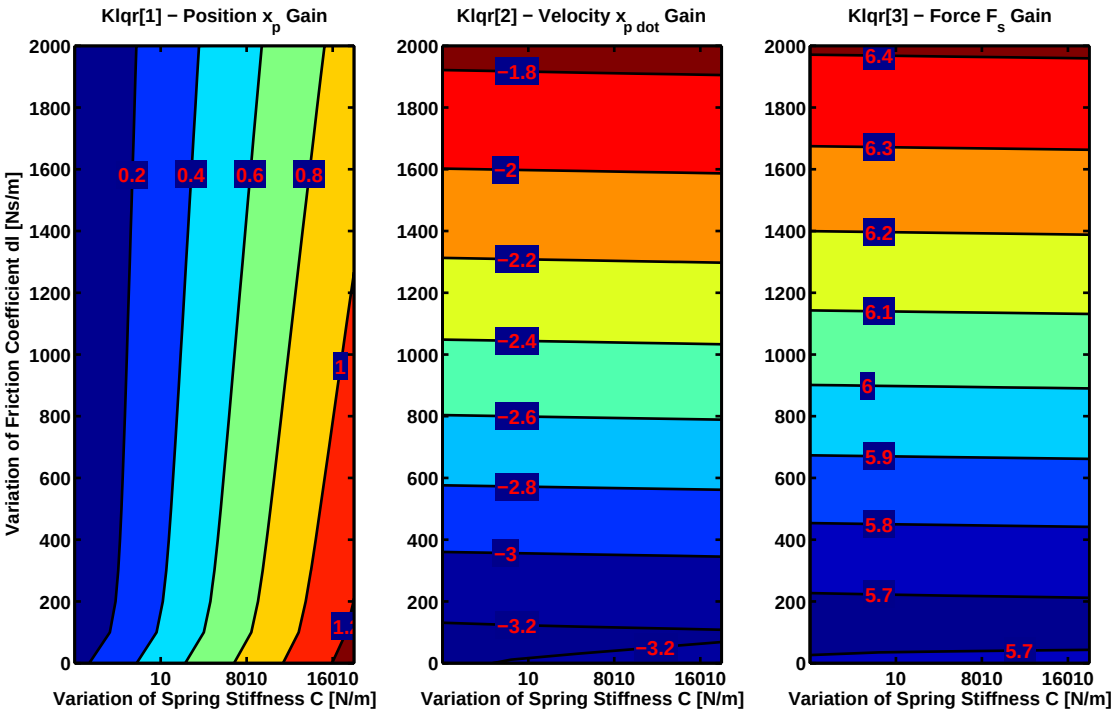

Figure 7.5: LQR controller gain map for simultaneous variations of the system parameters, that is, the friction coefficient d_l and spring stiffness C. The weighting matrices defined in 7.2.3 are used. Only the gains for the states are shown. Integrator and feedforward gains are not shown. The position gain depends heavily on the stiffness since it needs to compensate for the higher spring forces. The friction coefficient largely affects the gains for both velocity and force.

because it is allowed to do so due to the specifically chosen weighting matrices defined in subsection 7.2.3. The controller gain for the force (third column) is heavily load mass dependent. The gain is increased for higher masses, but decreases only slightly for a higher friction coefficient.

Figure 7.5 shows the variation of the controller gain for varying the parameter friction coefficient d_l and spring stiffness C. Increasing the spring stiffness C only affects the position gain, as is readily visible when comparing column one to two and three. The position gain depends heavily on the stiffness since it needs to compensate for the higher spring forces. Friction forces, as can be seen, have a negligible effect on the position gain. However, the friction coefficient largely affects the gains for both velocity and force. Velocity directly depends on the friction forces, hence the gains change appropriately. Variation in the friction coefficient has a higher effect for lighter load masses, as can be seen from the third column of figure 7.4.

The variation in controller gains for varying the parameters load mass m_l and spring stiffness C is shown in appendix by figure C.3.

7.5 Simulation of Adaptive Control Framework: Controller and Estimator

So far the estimator and controller were analysed in isolation; let us now simulate the full adaptive control framework as outlined in chapter 5. This simulation uses the RLS

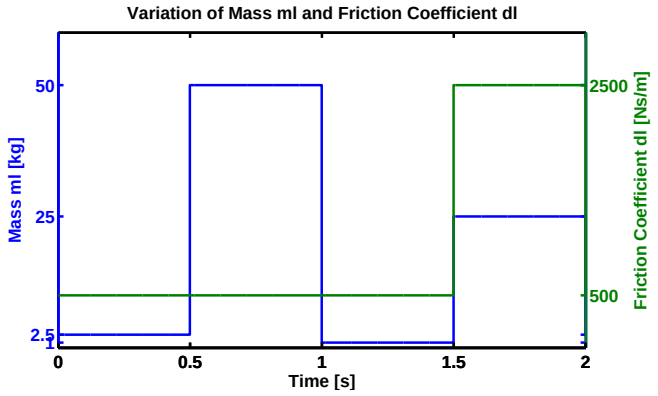

Figure 7.6: Variation of the system parameters load mass m_l and load friction coefficient d_l for simulating the full adaptive control framework. The sequence again tries to mimic a walking cycle of the robot HyQ. The simulation uses the parameters identified for the real-time FC1D test set-up; they are listed in appendix A.4. More realistic values are used for this simulation compared to the ones shown in figure 6.1.

estimator as defined in subsection 6.2.1 and the LQR controller discussed above with the weighting matrices defined in subsection 7.2.3. The simulated hydraulic system uses the parameters identified for the real-time FC1D test set-up; they are listed in appendix A.4. The system parameters load mass m_l and load friction coefficient d_l are changed according to figure 7.6. The input signal to the controller is a force reference, which is a multisine with frequencies $f = [2, 5, 10]Hz$ and amplitude 160 N. The initial LQR controller is derived for a system with very low mass $m_l = 0.5\,\text{kg}$ to mimic a lightweight leg. The new controller gains are limited and monitored by the update logic, which is discussed in detail in subsection 8.2.1.

The goal of this simulation is to show the impact of the adaption on the force tracking performance, hence two figures are shown: one without activated adaption and the second with adaption. Figure 7.7 shows the force tracking without activated adaption, that is, the initial controller gains remain constant during the simulation. As can be clearly seen, the tracking performance is disappointing. The closed-loop system even becomes unstable for high mass situations (second and fourth quarter). Only when the system is near the nominal system (first and third quarter) the controller was designed for, the response is fairly good. The adaption is then activated and shown in figure 7.8. It uses the exact same simulation and settings.

It is immediately evident that the force tracking has improved significantly for high mass and/or high damping situations as well as improved the force tracking of the stable set of parameters (third quarter). At the beginning of the first quarter of figure 7.8, the controller is not changed, thus the gains for a system with load mass $m_l = 0.5\,\text{kg}$ are used. This results in a phase delay, as is clearly seen by the blue (reference signal $r(t)$) and green (measured force $y(t)$) force signals. At around 0.4 s, the system is identified and the update logic allows the controller to update its gains. The system response immediately improves, although still being in the transient phase as the next system change to a load mass of $m_l = 50\,\text{kg}$ occurs (second quarter). During the first 80 ms, the gains are updated several times while still being in the transient phase. After this phase, the system is correctly identified and the proper controller gains are calculated, as can be seen when comparing the controller gains shown in the second subplot with the controller gain map shown in figure 7.4. Reading

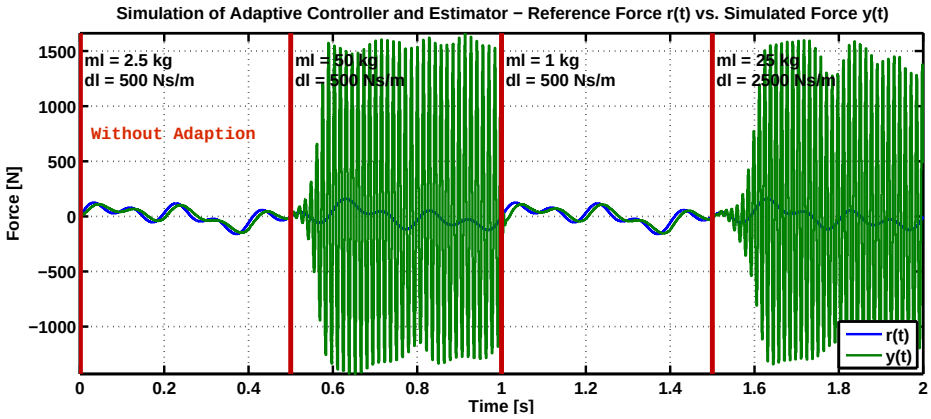

Figure 7.7: Time series plot of the simulation **without** adaption with force reference signal shown in blue and the controlled system force in green. The four sections correspond to the system changes shown in figure 7.6. Note that the fixed gain controller is not able to cope with the large system changes of section two and four, the response even becomes unstable. The force tracking is reasonable only when the system remains around the nominal system the controller was designed for (section one and three).

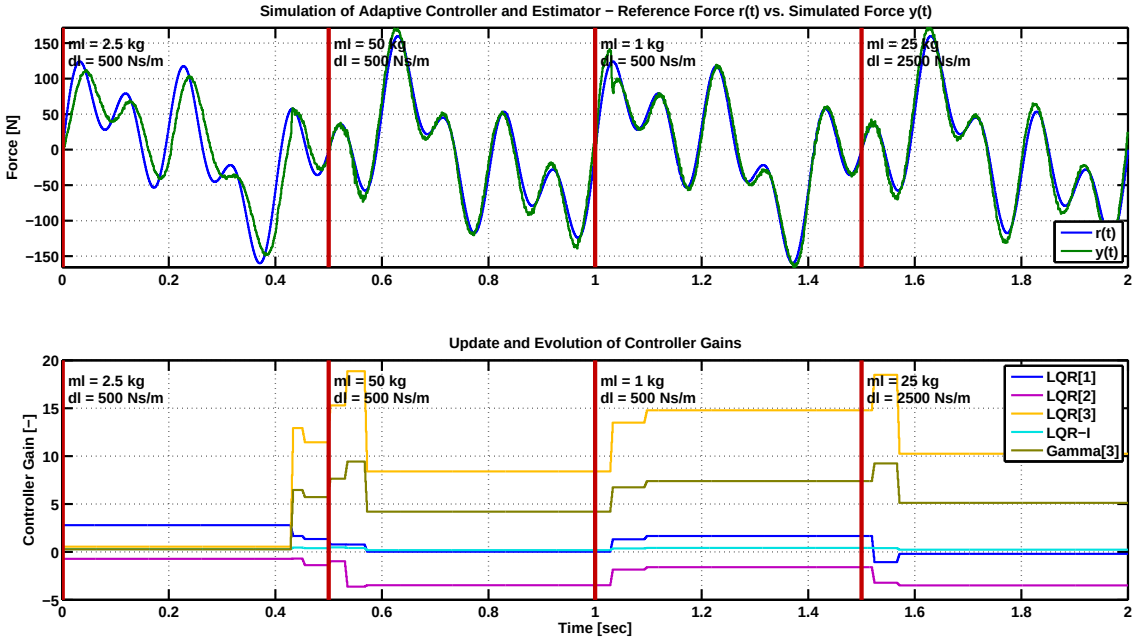

Figure 7.8: Time series plot of the simulation **with** activated adaption as outlined in chapter 5. The second subplot shows the evolution of the controller gains. Note that the system response improves as soon as the correct gains for the first section are found and continues to be quite good for the rest of the sections despite some severe changes in the system dynamics (shown in figure 7.6). Hence, the effectiveness of the outlined adaptive control framework is demonstrated in simulation.

out the gains from the controller map (figure 7.4) for $m_l = 50\,\text{kg}$ and $d_l = 500\,\text{Ns/m}$ gives for the LQR-gain $K_{map} \approx [0.2, -3, 8]$ corresponding well with the "online" calculated gains within the second quarter, which are $K \approx [0, -4, 8]$. The force tracking is very good with minimal overshoot and no phase delay. During the third quarter, the system has again very low mass, but the transient phase is relatively short - the correct gains are found quickly. Now comparing again the gains during the third quarter with the controller gain map (figure 7.4) reveals that they do not match.
This is due to the fact that the system was not correctly identified by the estimator, which can be readily seen when examining the pole locus plot shown in figure 7.9. The third column corresponds to the third quarter and, as can be clearly seen, the system is identified as a system with only real poles (yellow symbols) instead of the true complex poles shown by the blue symbols. Hence, the controller *thinks* that there is no oscillating behavior to deal with, consequently this results in higher controller gains than expected. When examining again the force tracking shown in figure 7.8, one can see that the tracking performance is very good, despite the *wrongly* identified system. Figure 4.8 reveals that a system with light mass has a transfer function with very little oscillating dynamics and a low resonance peak centered at high frequencies (near the sampling frequency). When the excitation signal contains no high frequency components, the systems complex conjugate poles are not sufficiently excited, hence the difficulties for the estimator to correctly identify these complex conjugate poles. This observation also means that a real system is indeed a valid approximation for such a system with light mass, hence the good force tracking, although, at first sight, the *wrong* controller gains are used. This approximation is of course only valid around low to middle frequencies; as soon as the system is operated at higher frequencies, the dynamics need to be characterized by complex poles. These observations can also be seen when considering the associated Bode plot shown in appendix C.4 by figure C.4. The tracking performance and evolution of the controller gains of the fourth quarter behave again as expected. The gains are a little higher since the friction is increased by a factor of 5.
All in all, we can summarize that the force tracking with activated adaption performs very well and shows promising results for implementing the proposed framework on the real-time hardware. The linear estimator does quite well (with the exception of the system with light mass) in identifying the system, as can be seen when examining the estimated poles (shown in colors from black to yellow in figure 7.9) versus the true poles shown in blue. The initial poles of the respective sections are shown in green. The chosen multisine force reference signal apparently excites enough frequencies without exciting the nonlinearities of the system such that the estimator can identify all the relevant dynamics of the system. The multisine reference signal was specifically chosen to mimic a real-world reference signal for moving the legs of the robot HyQ.

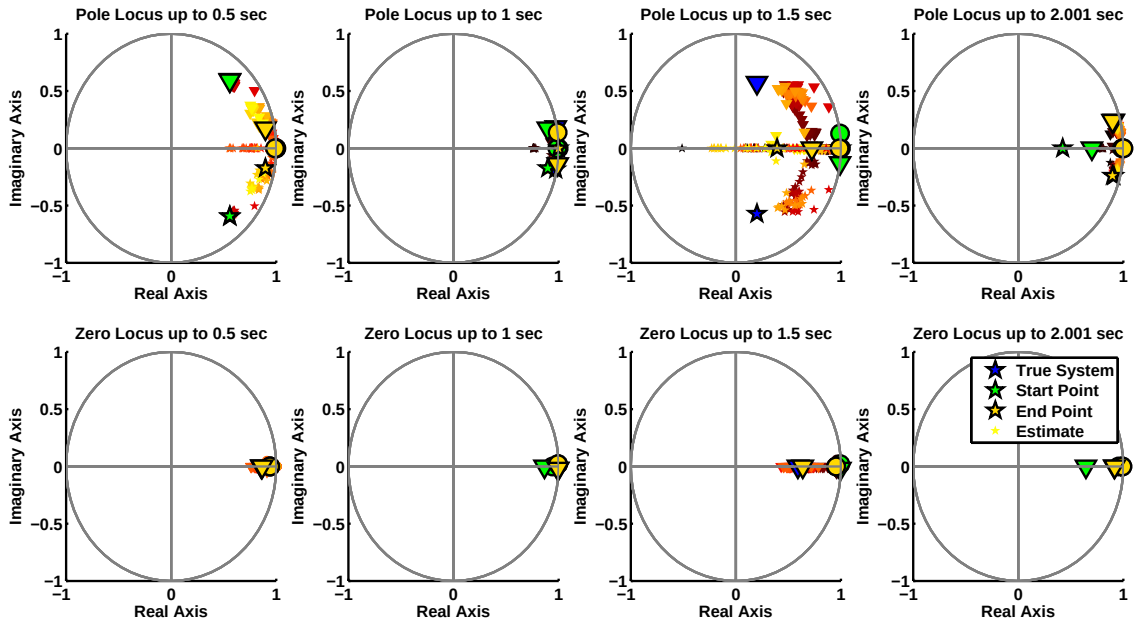

Figure 7.9: Pole and zero locus for the adaptive controller and estimator simulation **with** activated adaption. This figure corresponds to figure 7.8. Shown are the estimated poles (in colors from black to yellow) versus the true poles, shown in blue. The initial poles of the respective sections are shown in green. Note the accurate identification of the poles with the exception of the third section, where real poles instead of complex poles are identified.

Chapter 8

Implementation on the Real-Time FC1D Hardware

This chapter explains in detail the implementation of the adaptive controller framework on the real-time FC1D hardware using the software framework SL [34]. A particular advantage of the SL software package is that the same code can be used both in simulation and on the real-time hardware, which enables thorough and easy testing of new code and functions before running them on the hardware. More details about the FC1D test set-up can be found in section 2.1 and in appendix D. This chapter contains many implementation-specific details and serves as a documentation of the added functionality. It is thus intended for a reader interested in implementation or using hardware and code.
The implemented adaptive control framework, as outlined in chapter 5, consists of estimating the current system and, if needed, updating the controller to achieve persistent control performance. Figure 8.1 shows the adaption cycle with more implementation-specific details. The adaption cycle starts with reading-out the sensors (dark blue shape at top right), then processes the measurements and feeds them to the estimator (green shapes). The update logic (dark purple shape) receives the latest system estimate and, if necessary, calculates new controller gains (red shape). The updated controller and associated system description are stored within the bright purple block for later use. The controller can be changed in a feedforward sense through the yellow shape (top left) depending, for example, on the gait pattern. Finally, the new controller is applied to the system and a desired motion is performed and the cycle starts again.

8.1 State Space Estimator Implementation

The estimator algorithm, as explained in chapter 6, is implemented in C++ programming language using the concept of classes. Figure 8.2 shows the basic estimator software layout. A base class is used which implements various functions such as the base constructor, all set- and get-functions, private functions (such as normalization and linearization), as well as flag- and error-handling. The actual estimation algorithm is defined within the subclass which only implements the pure virtual recursive update function and inherits all other functionalities from the base class. With this set-up multiple estimator implementations, differing only in their recursive estimation algorithms, but using the same base functions and access structure, can be used at the same time or interchanged easily. These two classes are implemented using C++ programming language. The SL environment is written in C

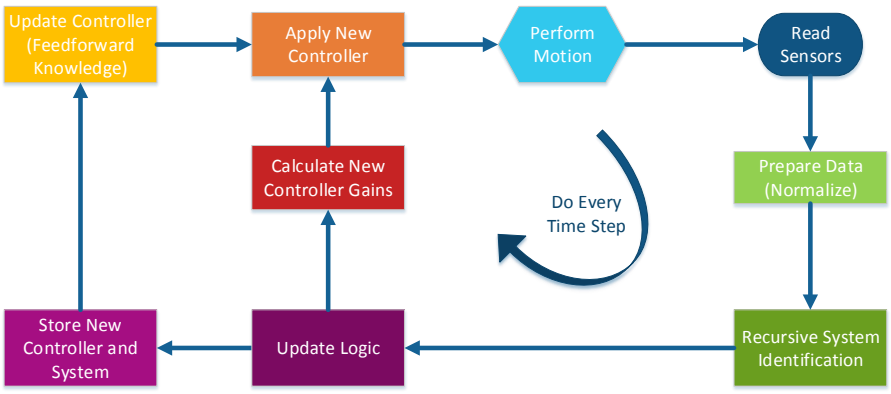

Figure 8.1: Flow chart of the adaptive control framework highlighting implementation details. The real-time cycle starts with reading the sensors (dark blue shape at top right), then processes the measurements and feeds them to the estimator (green shapes). The update logic (dark purple shape) receives the latest system estimate and, if necessary, calculates new controller gains (red shape) and stores (bright purple) controller and system description for later use. The controller can be changed in a feedforward sense depending, for example, on the gait pattern through the yellow shape (top left). Finally, the new controller is applied to the system and a desired motion is performed and the cycle starts again.

code, hence a wrapper function is needed to be able to access these C++ classes. All matrix calculations are performed using the Eigen C++ library [20], current version 3.2.1. The flow of information, as shown in Figure 8.2, is as follows: From the SL environment, a function of the wrapper is called which itself calls the corresponding function of the inherited estimator class, transferring necessary data to it. The inherited class uses the functionality provided by the base class to prepare the data (for example normalization and linearization functionalities) and finally runs its actual recursive estimation algorithm. The results can be returned to the SL environment through the wrapper via get-functions. Below the base class (subsection 8.1.1), inherited class (subsection 8.1.2) and the wrapper functions (subsection 8.1.3) are explained in more detail. Not all details are illustrated, but the most important ones.

8.1.1 State Space Estimator - Base Class

The base class provides the input and output functionalities, parameter settings, type definitions, as well as error handling. All these functionalities are inherited by the subclass. The three most important functionalities are explained in detail: these are the constructor, the recursive update structure, and the estimate access function.

State Space Estimator - Base Class - Constructor

Figure 8.3 depicts the flow chart of the constructor layout of the inherited and the base class. On top (red shape), an object of the inherited class is created, for example by calling: *myEstimator =State_ Space_ Estimator_ RLS(Theta_ 0,size_ of_ Theta,normalize_ on ,lambda,linearize_ on)*. Its constructor calls the base class constructor with the respective arguments and performs checks on the correctness of the parameters (e.g. parameter lambda) dedicated only for the inherited class. The base class constructor (dark blue)

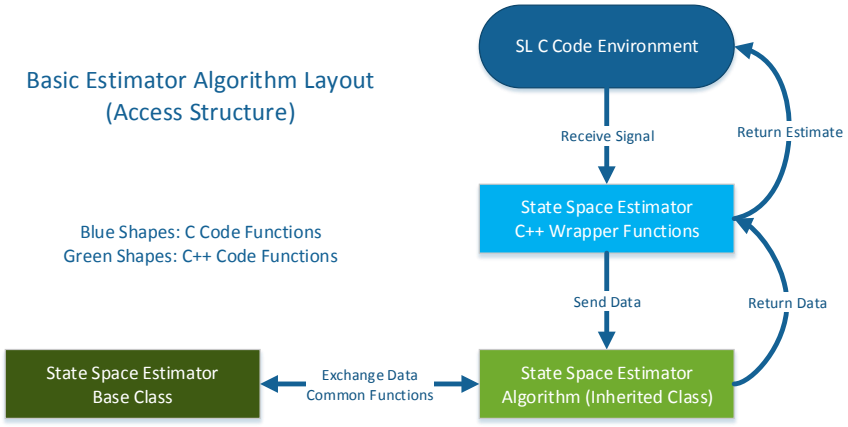

Figure 8.2: Estimator software layout: Functions within the SL C code environment can access the estimator algorithm via a C++ wrapper. This wrapper then sends the desired commands to the actual estimator algorithm which inherits basic functions from the estimator base class.

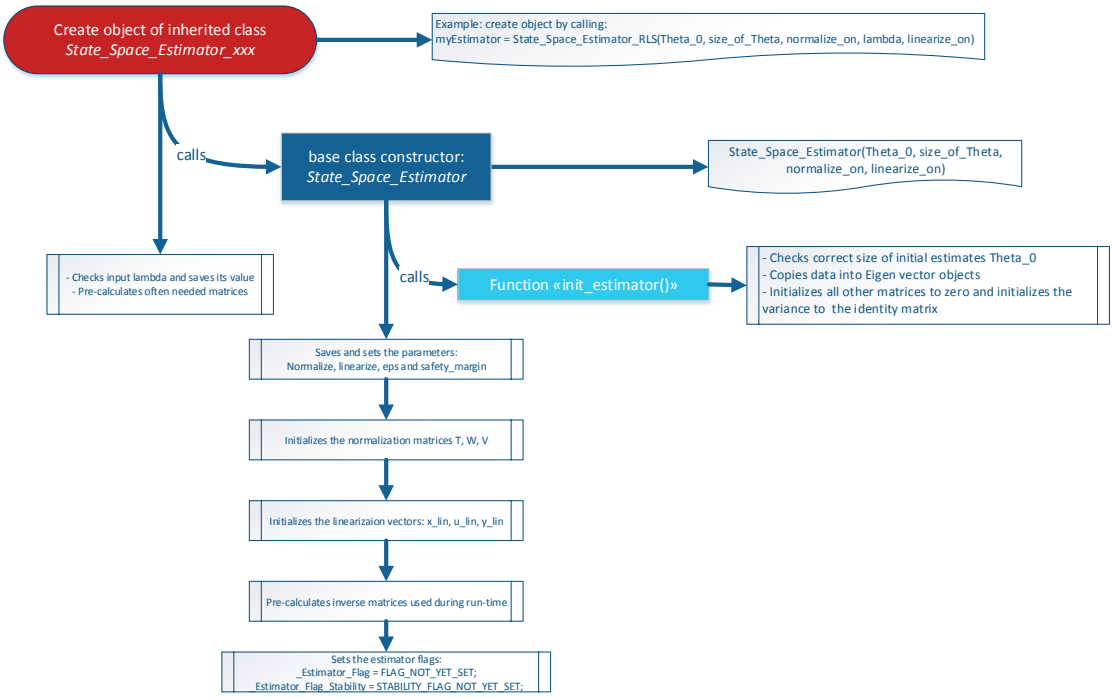

Figure 8.3: Layout of the constructor of the inherited and base class of the state space estimator implementation. On top (red shape), the constructor of the inherited class is called, and then it calls the base class constructor and checks some additional parameters for correctness. The base class constructor initializes the estimates (light blue) and checks the passed parameters for correctness. If everything is correct, a flag is set to enable further use of the created estimator object.

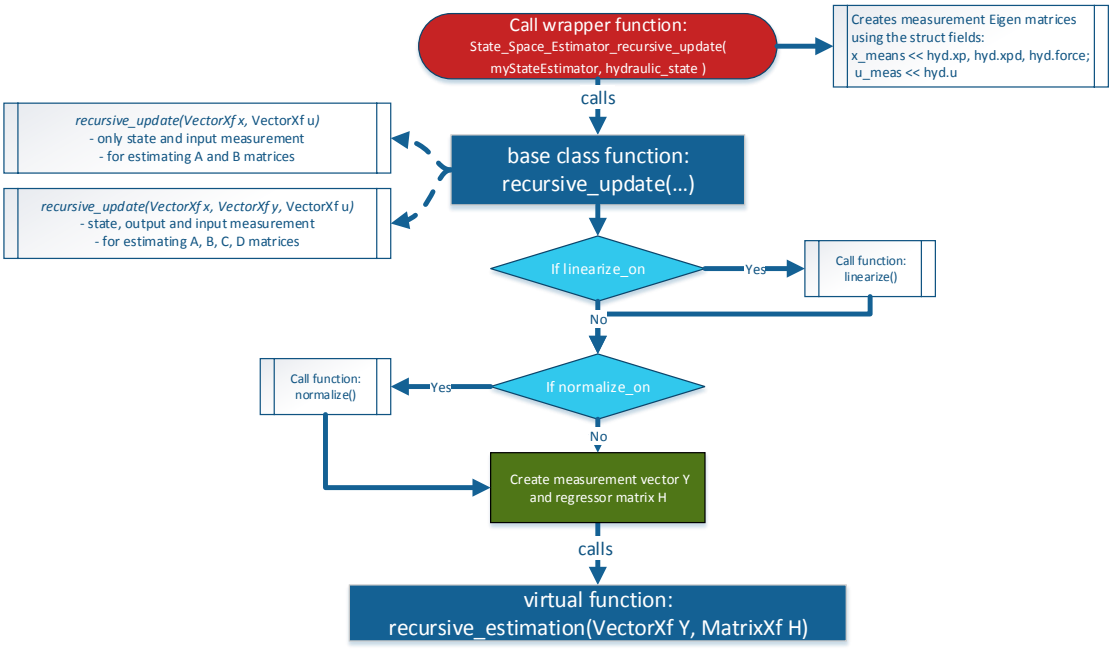

Figure 8.4: Layout of the recursive update structure of the inherited and base class of the state space estimator implementation. On top (red shape), the recursive update function of the estimator wrapper is called which creates the appropriate matrices and calls the base class function (dark blue) performing the specified features (linearization and normalization). After creating the measurement vector Y and the regression matrix H, the pure virtual recursive estimation function is called.

then calls the function *init_estimator()* which checks the parameters for correctness and initializes the storage variables, and sets the initial estimates according to the passed arguments. The constructor function also initializes the normalization and linearization features, if activated, and pre-calculates often needed matrices in order to decrease computational demand. If no errors occurred, appropriate flags are set to enable the use of the created estimator object. The parameter limits and logic are currently hard-coded and cannot be easily changed; yet, these need hardly to be changed. However, the parameter values are defined by a text file within the user folder of the SL framework. The parameters are then read by the function *read_estimator_settings()* within the top level file *SL_user_sensor_proc_unix.c* and are passed to the object's constructor within the initialization phase of the top level code. With this structure the parameters for the estimator can be changed without having to recompile to code.

State Space Estimator - Base Class - Recursive Update Structure

The recursive update function is composed of three parts: It begins (red shape) with the call of the function *State_Space_Estimator_recursive_update(myStateEstimator,hydraulic_state)* of the estimator wrapper, as shown in figure 8.4, with the first argument being a pointer to the estimator object and the second argument being of type *struct* containing the latest measurements of the hydraulic system. This wrapper function first extracts the necessary measurements from the hydraulic_state object and creates the appropriate Eigen matrices, then calls the *recursive_update()* function of the base class. Depending on the input

arguments and parameters defined during initialization, a different implementation of this function is called. The difference is that the case of additional output measurement y is handled explicitly, again in order to decrease computational complexity. Depending on the options, the functions *linearize()* and *normalize()* are called (light blue). The measurement vector Y and the regression matrix H are then created (green shape) using the equations defined in section 6.2. The last step consists of calling the pure virtual function implemented by the inherited class; as can be seen, the measurement vector and regression matrix remain the same regardless of the actual implementation of the recursive estimation algorithm.

State Space Estimator - Base Class - Get Estimates

The functionality for extracting the estimates from the estimator object, as shown in figure 8.5, follows the same structure as the recursive update structure explained above. On top (red shape), the *State_Space_Estimator_get_estimate_ss()* function of the estimator wrapper is called with the first argument being a pointer to the estimator object and the second argument being a pointer to the double array storage within the top level function. The wrapper function then calls the appropriate base class function *get_estimate_ss()* (dark blue) depending on the parameters set during initialization of the estimator. The two functions differ in the number of output storage matrices that need to be created to hold the estimates. The correct (number and size) state space matrices are then extracted from the parameter vector Θ using the function *create_state_space_from_theta()* (blue shape). If activated, the state space matrices are mapped to the stable discrete-time region by partitioning the matrices using eigenvalues and eigenvectors. This is done within the function *map_to_stable_system()*, whose logic is also shown in the figure 8.5. The eigenvalues are then scaled in order to give stable transition matrices. If no errors occurred, the Eigen matrices are copied to the supplied Eigen matrix references by the base class. The wrapper function copies the elements of these matrices to the double array passed by the pointer (second input argument).

8.1.2 State Space Estimator - Inherited Class

This subclass implements only the pure virtual recursive update function and inherits all functions and member variables from the base class. With this software layout, an arbitrary estimation algorithm can be used to implement the actual recursive update functionality. Currently, due to time constraints, only the recursive least squares algorithm, as explained in subsection 6.1.1, is implemented and fully tested. However, any other algorithm, such as a Kalman filter interpretation or an instrumental variable method, as outlined in section B.3, can easily be used. Multiple estimators can be used in parallel at the same time to compare their performance or for estimating different parts of the system: For example, one unique estimator for each foot of the robot HyQ.

Pure Virtual Recursive Estimation Function Using RLS Algorithm

This function implements the actual recursive update algorithm. It is defined as a pure virtual function within the base class in order to dictate the type and number of arguments. This way a consistent function structure is assured and interchangeability is simplified. Figure 8.6 shows the implementation using the RLS algorithm defined in subsection 6.1.1.

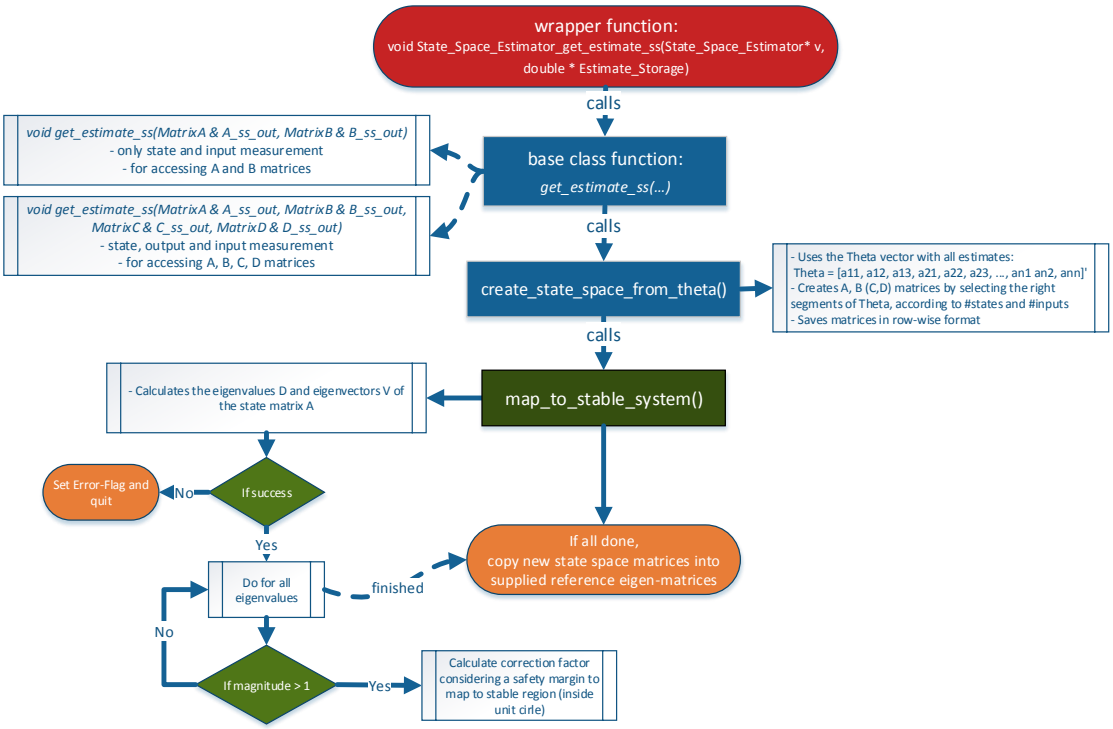

Figure 8.5: Layout of the *get-estimate* functionality of the state space estimator implementation. On top (red shape), the *State_Space_Estimator_get_estimate_ss()* function of the estimator wrapper is called which then calls the appropriate base class function (dark blue) depending on the parameters set during initialization. The correct state space matrices are then extracted from the parameter vector Θ. The state space matrices are mapped to the stable discrete-time region, if activated. If no errors occurred, the Eigen matrices are copied to the supplied Eigen matrix references by the base class. The wrapper function copies the elements of these matrices to the double array passed by the pointer (second input argument).

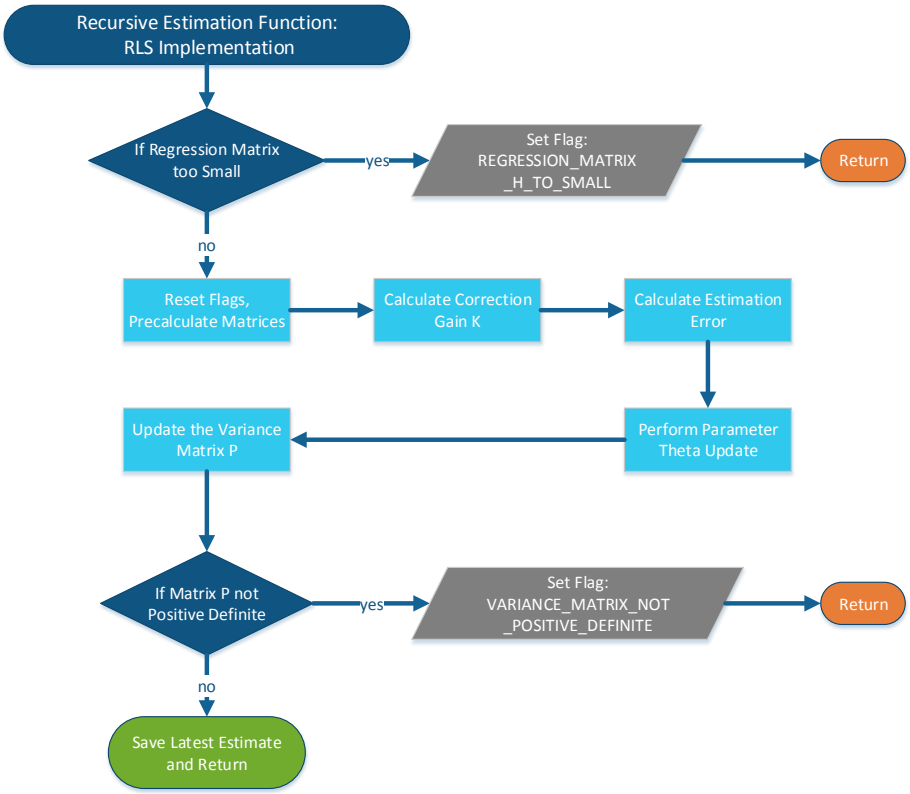

Figure 8.6: Implementation of the pure virtual recursive update function using the RLS algorithm as outlined in subsection 6.1.1.

The blocks refer to the equations and features mentioned there. The first step (blue rhombus) is to check if the regression matrix H is sufficiently rich, meaning checking if estimator wind-up occurs, see subsection 6.1.3 for details. If estimator wind-up occurs, the algorithm terminates, else it continues by pre-calculating the transpose of the regression matrix H since this matrix is often used below (bright blue shapes). The next four bright blue shapes each calculate a particular equation of the algorithm. As a final step, the variance matrix P is checked on its positive definiteness (blue rhombus). If the check is true, the algorithm returns, else it writes a flag and returns.

8.1.3 Estimator Wrapper Functions

As mentioned before, the wrapper for the estimator allows to access the C++ functions of the implementation. This wrapper is not needed when working in a full C++ environment. The wrapper functions, as mentioned above, provide many functionalities such as creation and initialization of estimator objects, update functions, and access functions. The diagram F.1 in appendix F shows all functions along with a description and an example on how to call that particular function. This diagram is intended for documentation purposes, hence its high information density.

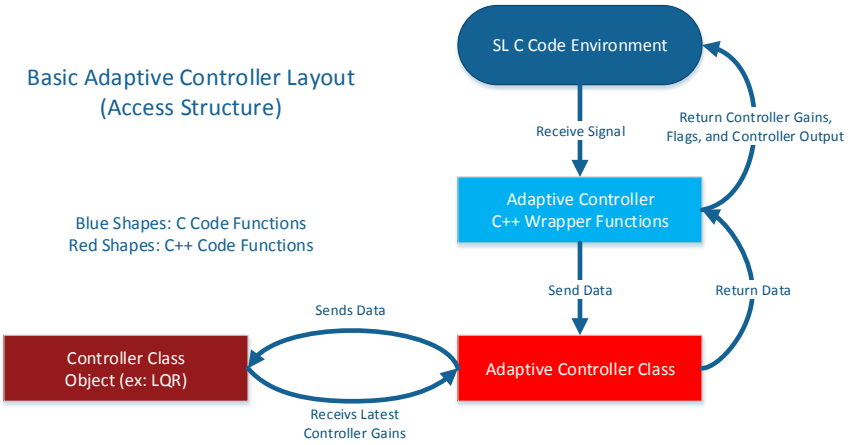

Figure 8.7: Controller software layout: Functions within the SL C code environment can access the adaptive controller class via a C++ wrapper. This wrapper gathers the necessary data from the estimator class, converts C class objects into Eigen matrices and then sends the desired commands to the adaptive controller class. The controller subclass provides the update functionality and the latest controller gains.

8.1.4 State Space Estimator - Flags and Status Information

Every function of the base and inherited class writes flags in case an intended function has successfully finished and its functionality can be used again or in case an error such that an unexpected behavior can be prevented and appropriate actions can be taken. A full table listing all possible flags along with their meaning is given in section F.2 of appendix F.

8.2 Adaptive Controller Implementation

The implementation of the controller framework follows a similar structure as the estimator implementation discussed in section 8.1. There is again an additional layer necessary to access the C++ code from within the SL environment. The wrapper functions gather necessary data from the estimator object, convert SL data objects into Eigen matrices, and call the desired functions of the adaptive controller class. This class provides many features in order to be used as a basic controller framework. These functions, among others, are the constructor with parameter checks, type definitions of Eigen matrices, set- and get-functions, controller reset-functions, an update logic, and controller output calculation. These functions also provide diagnostics via flags of both the base class and the controller subclass. The adaptive controller class only requires an object of a controller subclass which calculates the latest controller gains. The implemented controller subclass can be universal; for example a LQR or a pole-placement controller could be used. Due to time constraints, only a LQR with integrator and feedforward terms was implemented. Below the adaptive controller class is discussed in more detail, highlighting the two most important functions which are the update logic and the controller output calculation. Next, the implementation of the LQR algorithm of the controller subclass is discussed, highlighting the calculation of the controller gains using an iterative approach to solve the Riccati equation. Finally, the controller wrapper is briefly mentioned.

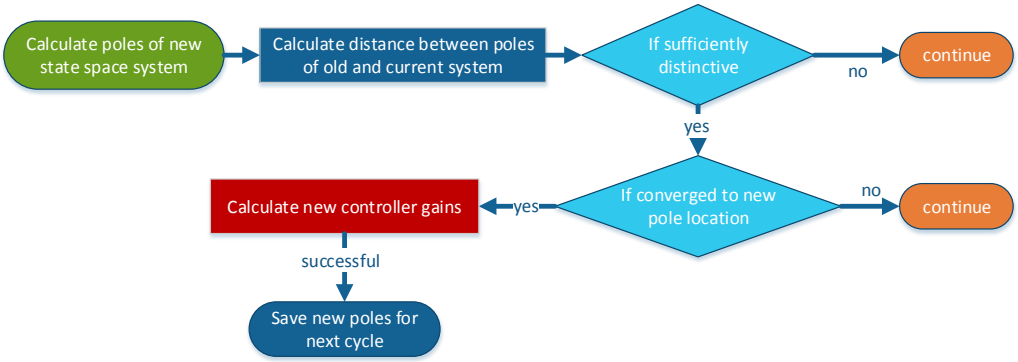

Figure 8.8: Flow chart of the adaptive controller base class update logic for determining when to update the controller gains. The poles of the newly estimated state space system are compared to the poles the currently used controller was calculated with. When these poles are far enough apart and when the estimation has converged, the controller gains are updated and the current poles are saved for the next cycle.

8.2.1 Adaptive Controller - Base Class

The constructor of the controller base class, similar to the estimator base class, provides the creation and initialization of the adaptive controller object along with the necessary checks of the parameters on correctness, defines the controller settings and pre-calculates often needed matrices. The weighting matrices are also initialized and it is verified that these meet the requirements of the LQR controller calculation algorithm. These parameters and weighting matrices are defined in a text file within the user folder of the SL framework. This file is then read by the function *read_lqr_weighting_matrices()* within the top level file *SL_user_sensor_proc_unix.c* and the variables are then passed to the controller constructor within the initialization phase of the top level code. Additional base class functions are three set-functions permitting to change the weighting matrices (function *set_weighting_matrices()*), controller parameters (function *set_controller_parameters()*), and allow to manually set the controller gains through the function *set_controller_gains()*. Hence, the adaptive controller object does not need to be reinitialized and these changes can be made while the system is running via the user input terminal. A reset function *reset_controller()* is also implemented such that the user can reset the adaptive controller via the input terminal in case any unexpected error occur.

Adaptive Controller - Base Class Update Logic

The update logic of the adaptive controller base class is intended for determining when to update the controller gains. Different approaches were discussed in section 5.3, where the chosen implementation based on pole (or eigenvalue) calculation was motivated. Figure 8.8 shows the flow chart of the adaptive controller update logic where only the most important blocks are shown. The flow chart with full implementation details is given in appendix F.3 by figure F.2. The two main goals of this update logic are: First, to make sure that the controller gains are not updated while the estimator is still converging, hence unexpected behavior can result. Secondly, to minimize the computational burden since solving the Riccati equation via an iteration is complex (see section 8.2.2). A simplified flow chart of

the update algorithm is shown in figure 8.8. The update logic starts on the left (green shape) by calculating the poles of the newly estimated state space system. These are then compared to the poles the current controller was calculated with. When these poles are far enough apart (top light blue triangle) and when the location of the new poles has converged (bottom light blue triangle), meaning that the estimation has converged, permission is given to update the controller gains. The current poles are saved for the next cycle. Within the red block, the new controller gains are calculated using the implemented subclass; in our case, this block is provided by the LQR controller implementation, which will be discussed below in subsection 8.2.2.
This update logic needs three parameters, as can be seen in the more detailed figure F.2. These parameters are used to determine the minimum distance the poles need to be apart, else the controller gains are not updated. The allowable distance will be larger for controllers that are more robust to system changes and smaller for optimizing best control performance and vice versa, respectively. The second parameter defines how close successive poles need to be in order to be considered converged. The time these poles remain converged is defined by the third parameter. These three parameters depend on the system dynamics (meaning how drastically the system can change), on the controller since its robustness defines how quickly the gains need to be updated. And finally on the estimator since its convergence speed and accuracy determines the time the poles need to be converged.

8.2.2 Adaptive Controller - Subclass Implemented as a LQR Controller

This subclass implements the actual LQR controller algorithm. It contains all functions necessary to calculate the latest controller gains. The adaptive controller base class augments the system matrices to incorporate the integrators and also takes care of the feedforward calculations. This subclass is therefore a pure LQR controller implementation, written as a templated class on the two parameters (number of states and number of inputs). Thus, the implementation is completely flexible since all variables, e.g. Eigen matrices, are defined at compile time. Changing these two parameters does hence not require a recompilation of the controller subclass. Once the subclass object within the adaptive controller base class (see subsection 8.2.1) is initialized, it can be used by the update logic, as shown above in figure 8.8, to calculate new controller gains.
The LQR framework is well known and taught at many universities; this thesis uses knowledge gathered from the lecture "Discrete Time Control Systems" [22] taught at ETH Zurich and information from the books [2] and [7] about optimal control to implement the Riccati equation, discussed below.

Iteratively Solving the Riccati Equation

Solving equation (7.4) is a well known problem and a vast amount of publications are devoted to various aspects of this problem. One of the most cited publications is probably [3], which serves as the foundation for the function *dare* implemented in MATLAB [30]. A simple and not particular robust nor efficient method is, as outlined in [3], to iteratively solve equation (7.4) until it converges. There is, however, little need for a sophisticated algorithm since solving the Riccati equation is essentially the last step within the adaptive control framework which anyhow results in a greater or lesser extent of inaccuracy. Thus, calculating a precise answer on a imprecise foundation does not make much sense. Additionally, there is a stringent limitation on the available computational power, thus faster

and less precise algorithms are preferred. Therefore, a simple iterative approach is chosen by introducing a time notation on the solution S_k and solving equation (7.4) for S_{k+1}, giving the following iterative scheme:

$$S_{k+1} = A^T \cdot S_k \cdot A - \left(A^T \cdot S_k \cdot B + N\right) \cdot \left(B^T \cdot S_k \cdot B + R\right)^{-1} \cdot \left(B^T \cdot S_k \cdot A + N^T\right) + Q \quad (8.1)$$

where k denotes the iteration step and the initial solution S_0 is calculated using:

$$S_0 = -N \cdot R^{-1} N^T + Q \quad (8.2)$$

Note that equation (8.2) can be pre-calculated at compile time or anytime the weighting matrices are changed. The abort criterion is based on the relative error using a fixed precision $RTOL$, as:

$$||S_{k+1} - S_k|| < RTOL \quad (8.3)$$

where the error between these two matrices is calculated using the Frobenius norm [38, page 552]:

$$||A||_F = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\mathrm{tr}(AA^*)} = \sqrt{\mathrm{tr}(A^*A)} \quad (8.4)$$

where $tr(\dots)$ is the sum of the diagonal elements and A^* is the complex conjugate transpose of A. However, the actual implementation does not calculate the square root in order to save computation time. Hence, the fixed precision $RTOL$ is defined in terms of $||A||_F^2$. An absolute iteration limit (counter) is also implemented in case the iteration does not converge.

There are some conditions that must be fulfilled in order to calculate the solution of the Riccati equation. First, the state space system formed by the matrices A and B must be stabilizable. Since we know that our system is always controllable regardless of the system changes, we use the stronger condition and require that the pair (A, B) is controllable. If this condition is not fulfilled, the function immediately returns with an error flag and keeps the old controller gains. Secondly, the weighting matrices must fulfill some conditions, these are:

$$\begin{aligned} R &> 0 \\ Q - NR^{-1}N^T &\geq 0 \end{aligned} \quad (8.5)$$

The function *positive_ definite()* performs the necessary checks to ensure that the weighting matrices fulfill this second condition, otherwise the calculation is aborted and an error flag is set. In addition to these requirements; the update logic "naturally" allows the update of the controller gains only when the estimates are reliable, hence the matrices A and B will be "well-behaved" most of the time, thus little numerical issues can be expected. It has to be mentioned though that using normalization for both the estimator and controller greatly improve the numerical stability of this iterative approach.

Figure 8.9 compares the performances of the iterative algorithm (left hand axis) versus the *true* solution of the Riccati equation (right hand axis) calculated by the MATLAB function *dare()* for an arbitrary state space system.

As can be seen, the relative error decreases almost quadratically and is already low after just 100 iterations. There is, of course, a trade-off between accuracy and computational demand. However, it makes no sense to require a precise calculation, or in other words: The precision of this algorithm shall only be as high as the accuracy of the estimation

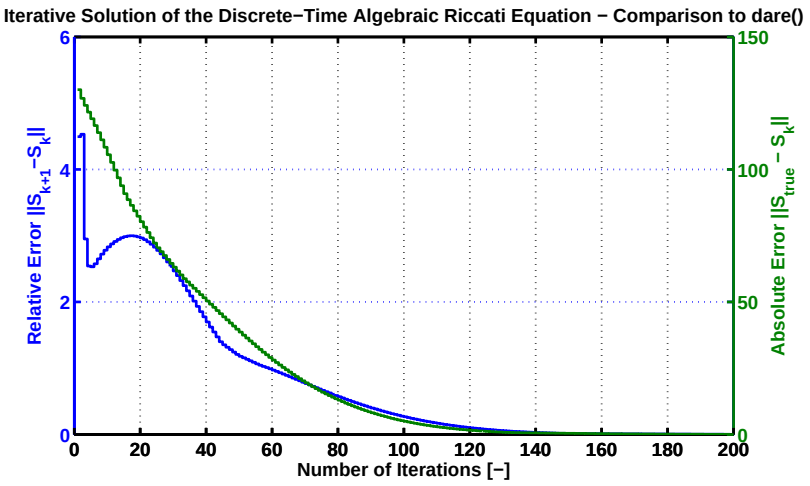

Figure 8.9: Calculating the Riccati equation - comparison of the iterative solution vs. MATLAB function. The left hand axis (blue) shows the relative error between successive solutions S_{k+1} and the right hand axis (green) shows the absolute error of the iteration to the true Riccati solution S_{true} computed using the MATLAB function *dare()*. The difference between the matrices is calculated using equation (8.4).

algorithm. Therefore, a rather high relative tolerance value was currently selected, that is: $RTOL = 0.4$, although higher values might still give reasonable results. An iteration that is not able to converge within the counter limit or a calculation that fails poses no problem to the controller since its gains remain unchanged and the Riccati equation can be solved again during the next time step.

8.2.3 Adaptive Controller Wrapper Functions

Again a wrapper is used to hide the C++ code and to provide a convenient interface for the user working within the SL framework. The adaptive controller wrapper makes sure that the standard C++ data types are converted correctly into Eigen objects. The wrapper calls the corresponding functions of the estimator and gets other data needed by the adaptive controller base class. The wrapper provides the following four basic groups of features, more details are shown in appendix F by figure F.3:

- **Constructor and Destructor**: Accepts pointers to double arrays, converting them to Eigen objects, and calls the constructor of the adaptive controller class.
- **Update Functions**: Accepts pointer to the estimator object and controller object and updates the controller gains stored within the adaptive controller base class.
- **Get Functions**: Functions for accessing the latest controller gains, reading the latest flags and information as well as calculating and accessing the latest controller output when supplying the latest measurement and reference data.
- **Set Functions**: Provide functionality to initialize and change the weighting matrices, controller parameters and gains. Additionally, a reset function (for resetting the integrator within the controller base class) and an initializing function for the update logic are implemented.

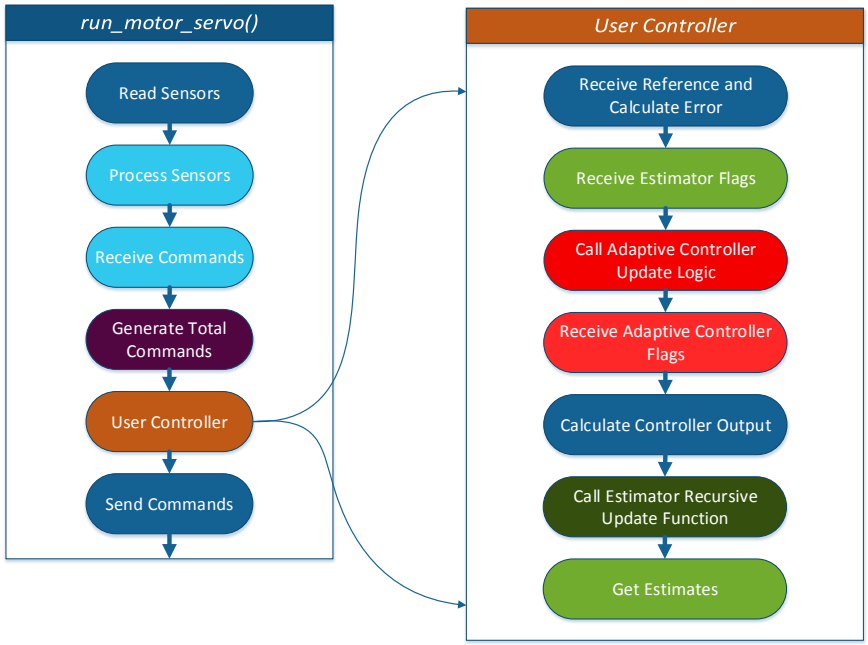

Figure 8.10: The FC1D software layout - illustration of the program sequence that the SL framework provides. The adaptive controller is integrated into this structure. This layout is used both in simulation and on the real-time hardware. The same layouts are used on the HyQ and other test set-ups.

8.3 Integration of the Adaptive Control Framework into the SL Framework

The two previous sections discussed the implementation of the estimation and control part of the adaptive control framework as introduced in chapter 5. This section briefly discusses how this framework is integrated into the existing SL framework. Currently, the adaptive control framework is implemented for the FC1D test set-up; however, it can be easily implemented for other test set-ups, for example the HyL leg set-up or the full HyQ robot. The estimator or controller classes remain the same, regardless of implementation. The software routine of the FC1D test set-up is shown in figure 8.10, where not every detail, but the relevant steps of the program sequence are illustrated. The main function (not shown here) calls a initialization routine which creates instances the estimator and controller objects, initializes the sensors and so forth. After the robot is fully initialized (or the FC1D set-up), a loop starts which runs the motor servo function *run_motor_servo()*, shown on the left side of figure 8.10.

The function does the following every time step: First, the sensors are read and then processed (e.g. filtering) followed by reading out the commands from the user terminal or some higher level tasks, e.g. commands from the vision algorithm. The sensor measurements and commands are then used to calculate the total control command which is passed to the user controller function. This function is responsible for the low-level control based on the total control commands created by the higher-level functions. The controller output is then sent to the low-level device, e.g. the servo-valve.

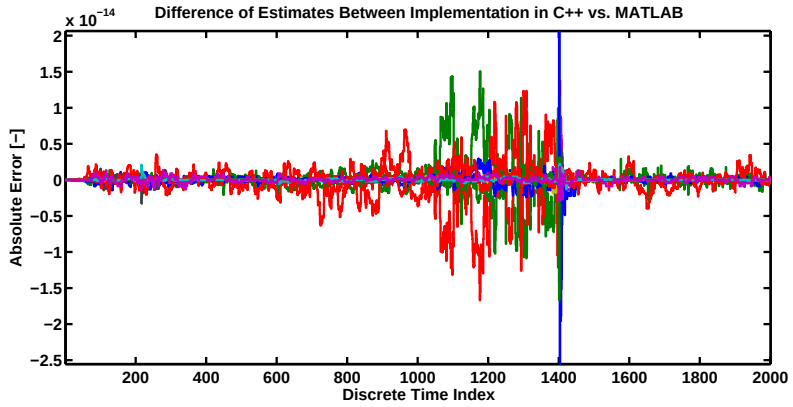

Figure 8.11: Difference of estimates between the implementation in C++ code versus MATLAB code. The C++ code was executed using mex-files. Each color represents an element of the difference vector, however individual differences are not of importance here. As can be seen, the maximum absolute error, that is, the difference between the components of the unknown parameter vector, is below $< 2 \cdot 10^{-14}$.

The user controller function is shown in more detail on the right hand side of figure 8.10. The relevant reference signals are first extracted from the passed commands and the error is calculated (using the sensor measurements). The latest flags of the estimator object are read out. The adaptive controller update logic (second red shape) is only called when these flags do not indicate any issues within the estimator, else the old controller gains are kept. Again the flags of the controller object are read out to make sure the calculation was successful. These flags are also saved to a log file for later analysis. The next blue shape then uses these, possibly new, controller gains to calculate the control output. For the FC1D test set-up, the controller output is in terms of valve opening factor, that is, the controller output is limited to: $u \in [-1, 1]$. As a last step, the recursive update function of the estimator object is called and the latest estimates are read out and saved for the next cycle.

8.4 Verification of the C++ Code

Implementing new functions or programming in general can be error prone and tedious. Therefore, the above functions are extensively tested and verified in order to make sure that absolutely no errors are introduced. MATLAB allows to run C++ code from its m-file environment through so called mex-files which are nothing else than interface functions. Hence, two mex-files were written for both the estimator and controller code, as outlined above. The MATLAB code is assumed to be correct and hence the C++ code is compared to the outputs of the MATLAB code. Signals of the simulations, e.g. from the simulation shown in section 6.3, are recorded at each time step and then passed to the mex-files. These handle the data exchange, call the corresponding C++ functions (e.g. the estimator or controller class) and feed back the results, that is, the estimates or controller gains. These outputs are then compared to the values the MATLAB functions have calculated. Figure 8.11 depicts the error between the estimates calculated by the MATLAB functions versus the estimates calculated by the C++ code of a simulation similar to the one shown in section 6.3.

As can be seen, the absolute error of each element of the unknown parameter vector Θ (see equation (6.35)) is below $< 2 \cdot 10^{-14}$. The machine precision of MATLAB is $\approx 2.2 \cdot 10^{-16}$. We can conclude that there are no implementation errors and that either code (C++ or MATLAB) will result in reliable estimates. Similar verifications were performed for checking the calculation of the Riccati solution. However, there is a trade-off between accuracy and computational demand. The C++ code is currently optimized for computational efficiency.

Chapter 9

Experiments on the FC1D Test Set-Up

The main results of the experiments are presented in this chapter. First, the estimator's performance for different algorithms and different input signals is discussed, followed by experiments validating the controller and a comparison against a currently implemented controller on HyQ. Finally, some important parameters of the real-time FC1D test set-up are also identified. Some limiting aspects of the system are discussed as well.
Unfortunately, experiments assessing the performance of the estimator in conjunction with the controller, as outlined in chapter 5 and demonstrated using simulation in section 7.5, could not be done because the hardware of the test set-up was not able to run both the estimator and controller functions at the same time in real-time mode. A more powerful embedded computer would be needed or alternative algorithms have to be selected with computational restrictions in mind.

9.1 Estimation Experiments on the FC1D Test Set-Up

An extensive amount (over 60) of experiments on the real-time FC1D test set-up were carried out in order to assess the performance of the recursive linear estimator (see equation (6.21)) for various conditions: different input signal frequencies, different input signal amplitudes, and different types of input signals (e.g. chirp or sine signals). Some experiments even helped to uncover some flaws of the test set-up itself, for example, wrong sensor positioning and calibration, improper length of connecting rod, and unfavorable signal filter settings.
Figure 9.1 shows the load cell force transfer function of three such experiments. The (normalized) transfer functions are calculated (post-processed) from the state space matrices created using the estimated parameter vector Θ (see equation (6.35)) at each time step. The initial guess for the transfer function[1] is drawn in magenta and the nominal system (calculated using non-causal estimation techniques) is drawn in blue. The colors of the estimates change from black to yellow for each time step, whereas the last estimate within the time window is drawn in green. The input signals are chirp sine waves with frequencies ranging from $0\,\mathrm{Hz} - 30\,\mathrm{Hz}$ and amplitudes of $5\,\%$, $10\,\%$ and $20\,\%$ of the nominal valve opening. Each experiment has a duration of $8\,\mathrm{s}$. The forgetting factor is $\lambda = 0.92$.

[1] The initial guess (magenta line) is intentionally chosen to be different from the truth (blue line) in order to visualize convergence of the estimator.

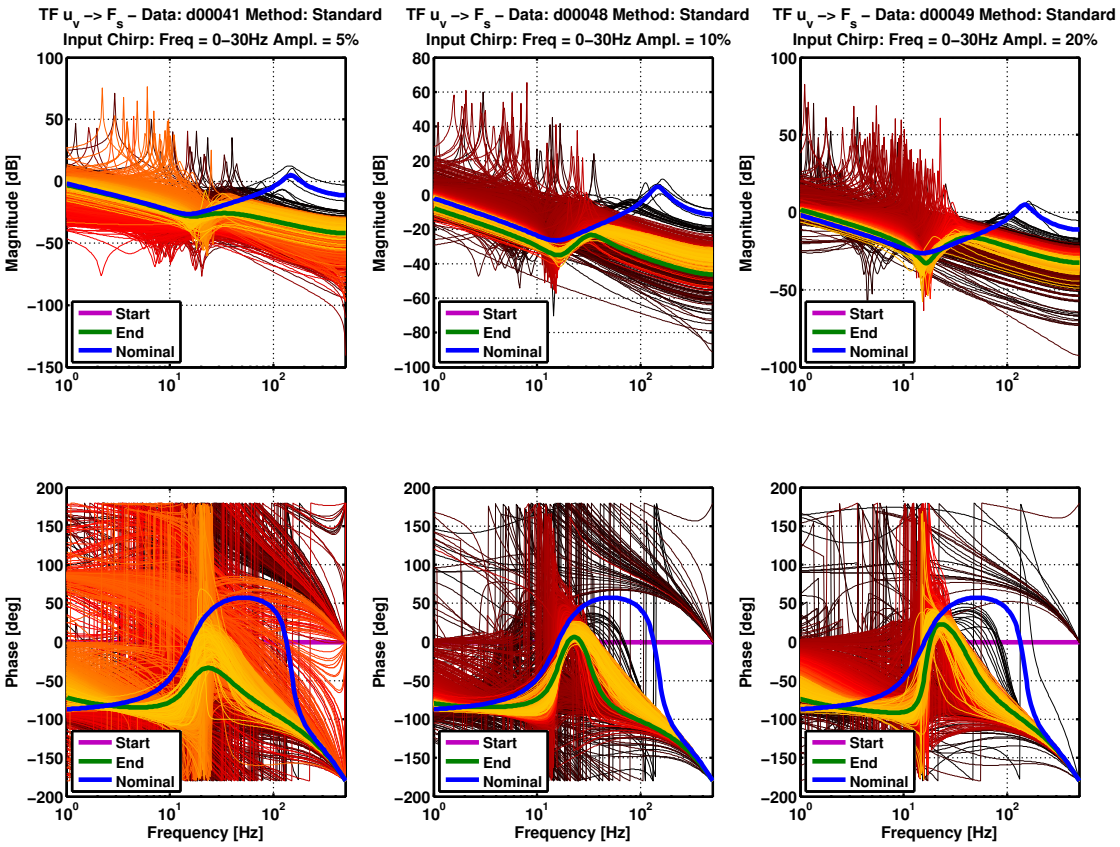

Figure 9.1: Bode plot of the load cell force transfer functions estimates (normalized) for each time step using real-time FC1D test set-up. The initial guess is drawn in magenta (bold) and the nominal transfer function is drawn in blue. The colors of the estimates change from black to yellow for each time step, whereas the last estimate within the time window is drawn in green. The input signals are chirp sine waves with frequencies ranging from $0\,\mathrm{Hz} - 30\,\mathrm{Hz}$ and amplitudes of $5\,\%$, $10\,\%$ and $20\,\%$ of the nominal valve opening. Each experiment has a duration of $8\,\mathrm{s}$. The forgetting factor is $\lambda = 0.92$. The standard estimation function introduced so far is used for this figure.

Several things are worth to be pointed out:

1. The standard estimation function introduced so far is used for this figure.
2. The estimates diverge quickly from no initial knowledge about the system (the phase of the nominal system (magenta line) is zero) to intermediate estimates drawn in black. Within this first phase of the chirp signal, where the frequency content is still low, the system is not fully excited, hence the estimated transfer functions contain fictive resonance peaks at low frequencies. These peaks are probably due to the nonlinearity inherent to the hydraulic system and noise. These peaks will render the adaptive control framework incapable of controlling the system.
3. As soon as the chirp signal excites the system enough, these resonance peaks disappear and the low frequency part of the hydraulic system is identified by the estimator. The estimates change color from red to yellow with increasing time.
4. Even towards the end of the chirp signal, the estimator is only able to give a rough approximation of the transfer function of the nominal system (blue line). This behavior is unavoidable since the dynamics of the system change and the resonance peak disappears for large valve openings as was analyzed and shown in figure 4.6.
5. The transfer function of the nominal system (blue) was calculated using the model given by equation 3.51 .
6. When comparing the experiments on the left against those on the right, one can observe that the amplitude of the chirp input directly affects the degree of excitation. The performance of the estimator increases with better excited systems.

To summarize we can say that the linear estimator is able to partially identify the essential dynamics of the hydraulic system not only in simulation but also on the real-time hardware. However, these undesirable peaks cannot be accepted and improvements to this standard algorithm are necessary. Three different algorithms are discussed to cope with the problem.

9.1.1 Adaptive Forgetting Factor

It has already been observed that the estimator converges quickly but diverges again because of nonlinearities and noise. One solution to this problem is to use an adaptive forgetting factor (AFF) algorithm which uses the knowledge about the estimation error and noise characteristics of the measurements. The algorithm works as follows [8]:

1. First, the a priori prediction error is calculated: $e[k] = y[k] - H[n] \cdot \Theta$
2. The a propri error signal $e[k]$ can be expressed as a noise-free component corrupted by a white Gaussian noise: $e[k] = e_{f,k} + v_k$. The noise-free a priori error signal can be estimated using knowledge about the standard deviation of the measurements σ_v:

$$\hat{e}_k = sign(e_k) \cdot \max\left(|e_k| - \sqrt{c_1 \cdot \sigma_v^2}, 0\right) \tag{9.1}$$

3. The time average of this error is now called σ_k and it is basically the filtered version of $\hat{e}_k$. The filter is a first-order moving average filter where β is the pole defining the time characteristics:

$$\sigma_k = \beta \cdot \sigma_{k-1} + (1 - \beta) \cdot \hat{e}_k^2 \tag{9.2}$$

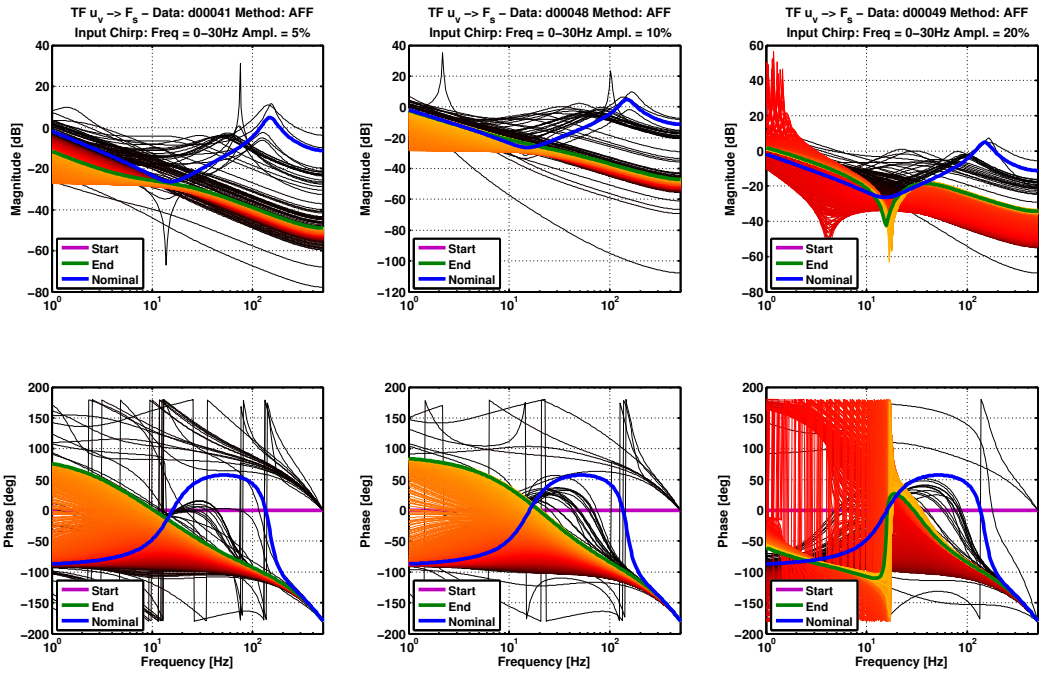

Figure 9.2: Bode plot of the load cell force transfer functions estimates (normalized) for each time step using the real-time FC1D test set-up with activated adaptive forgetting factor (AFF). Color scheme and layout is the same as in figure 9.1. Clearly, the performance has improved in the sense that less inappropriate resonance peaks appear at low frequencies. The transition of the estimates from one time step to the next is smoother.

4. The forgetting factor for each measurement equation can now be calculated using:

$$\lambda_k = 1 - \frac{2 \cdot \sigma_k}{M \cdot (\sigma_k + c_1 \cdot \sigma_v^2)} \tag{9.3}$$

   where M is the number of unknown parameters. The variables c_1 and β can be viewed as "tuning" factors of the algorithm. Increasing the parameter c_1 gives an insensitive and slow adaption. Increasing β gives a smoother evolution of the forgetting factor λ but also slows down the adaption. The standard deviation σ_v can be calculated using measurements; these values need to be normalized using equation 3.52. The range of permissible values for λ is usually limited to the range: $[0.8, 1]$.

A more elaborate derivation can be found, for example, in [8] or [33]. The normalized standard deviation of the measurement noise was identified using raw sensor readings.
Figure 9.2 shows the same experiments as in figure 9.1 but with activated adaptive forgetting factor.
Several things are worth to be pointed out:

1. Clearly, the performance has improved in the sense that less inappropriate resonance peaks appear at low frequencies. The phase appears to be much more steady and distinct.

2. The gradient of estimates is much smoother from one time step to the next and consequently slower.

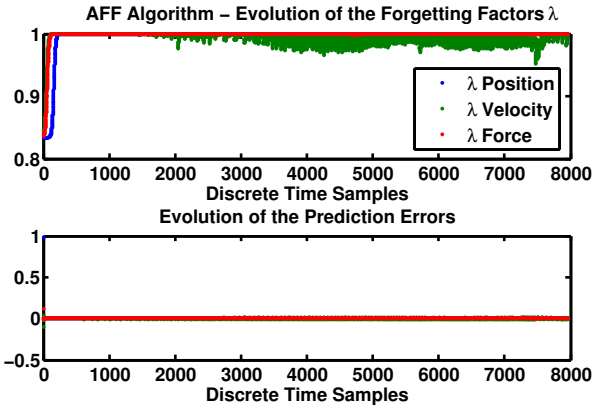

Figure 9.3: Plot of the individual forgetting factors λ_i calculated by the adaptive forgetting factor (AFF) algorithm for the experiment $d00049$ shown in the right column of figure 9.2.

3. Interestingly, the frequency estimates resemble straight lines over a substantial period of time of the experiment. This behavior is probably caused by too little excitation of the system and is amplified by the fact that the AFF algorithm is less sensitive to noise, hence only the low frequency part of the system is identified. This is clearly visible in the transfer functions shown in the left column and less visible in the right column. The estimator is able to identify more high frequency parts of the system because the magnitude of the input is larger - the system is better excited.

Figure 9.3 shows the progression of the individual forgetting factors for the experiment $d00049$ shown in the right column of figure 9.2. The prediction error of the estimator decreases promptly and stays low in accordance to a steep increase of all three forgetting factors. They remain largely at 1 (meaning less forgetting), but the velocity related forgetting factor is influenced slightly by the noise. This behavior directly depends on the choice of parameters, hence for different experiments the adaption can show a completely different behavior.
In conclusion, we can say that the AFF algorithm does improve the system estimates, but the algorithm has difficulties when the system is not excited enough. The next section discusses two methods for extending the AFF algorithm.

9.1.2 Persistence of Excitation

As we have seen so far, input signals containing only limited frequency content or which are of low amplitude or both do not excite the system enough for the estimator to give an accurate estimate. Two commonly used methods are presented, the first is based on the calculation of the so called persistence of excitation matrix and checks whether it contains enough information. The second method calculates the frequency content of the input signals to the estimator and checks whether there are enough distinct frequencies with adequate amplitude available.
A vast amount of literature about the persistence of excitation (PE) algorithm exists, but only a few are concerned with real-time implementation issues. The references [1] and [9] give a brief description of the algorithm. The PE condition is derived using a convergence analysis of the least squares algorithm. The resulting condition for the regressor $H[k]$ is

essentially a measure of how much energy or information $H[k]$ contains, the condition is:

$$\alpha \cdot I \leq \sum_{i=k-N}^{k} H[i]H^T[i] \leq \beta \cdot I \tag{9.4}$$

where α is a lower limit, β is an upper limit, k is the current time step, N is the length of the sequence in consideration and I denotes the identity matrix of appropriate size. The condition requires that the sum of the matrix product over some interval N shall be larger than some parameter α but bounded by some parameter β. The literature lacks information about the choice of these parameters because the derivation only requires that condition 9.4 must hold for some parameters. The problem is that there will most likely be some parameters for which this condition is fulfilled. However, on a system operating in real-time, it is infeasible to calculate large amounts of parameter combinations just to check if a condition is fulfilled. Hence, these parameters are treated as tuning parameters such that the estimator's performance is maximized. Condition 9.4 is checked by calculating the singular values of the matrix summation; the maximum singular value must be lower than β and the minimum must be larger than α. If the condition is fulfilled, the estimator is permitted to update the estimates using all measurements within the interval N.
Figure 9.4 compares the AFF algorithm (left column) against the combination of AFF and PE algorithm (middle column). The right column shows the combination of AFF and FFT algorithm, which will be discussed below. The evolution of the forgetting factors for the left column is shown in appendix G by figure G.1. The forgetting factors for the middle column are shown in figure 9.5. The corresponding singular values are shown in figure 9.6.

Several aspects are worth to be discuss:

1. Figure 9.5 clearly shows that the estimation algorithm is switched on and off several times due to the permission given by the PE condition. A first activation occurs after 3000 samples lasting for roughly 650 samples. A second activation occurs near the end of the chirp input signal around sample 5100 till 6300. The AFF algorithm keeps the forgetting factors high due to relatively small prediction errors.

2. The minimum and maximum singular values are shown in figure 9.6 along with the bottom row displaying the permission signal for the estimator. Several features can be observed, the first being that during the first few hundred samples the minimum singular value is larger than the lower limit, but the maximum singular value violates the limit. The maximum value then dives under the limit and remains there until around sample 6000, hence only the minimum value is decisive. It exceeds the lower limit twice, hence the estimator is activated twice.

3. There are essentially three regions where the system is excited enough, that is at the beginning of the experiment for the first few hundred samples, in the middle and towards the end. Unfortunately, the PE algorithm misses the first region. This observation can be very well seen in figure 9.4, where it is apparent that the black lines are nonexistent within the middle column (compared to the left column).

4. Considering again the middle column of figure 9.4, where resonance peaks are identified occurring at frequencies lower than 10 Hz. These systems are calculated by the estimator during the first activation sequence as displayed in figure 9.5. It is not

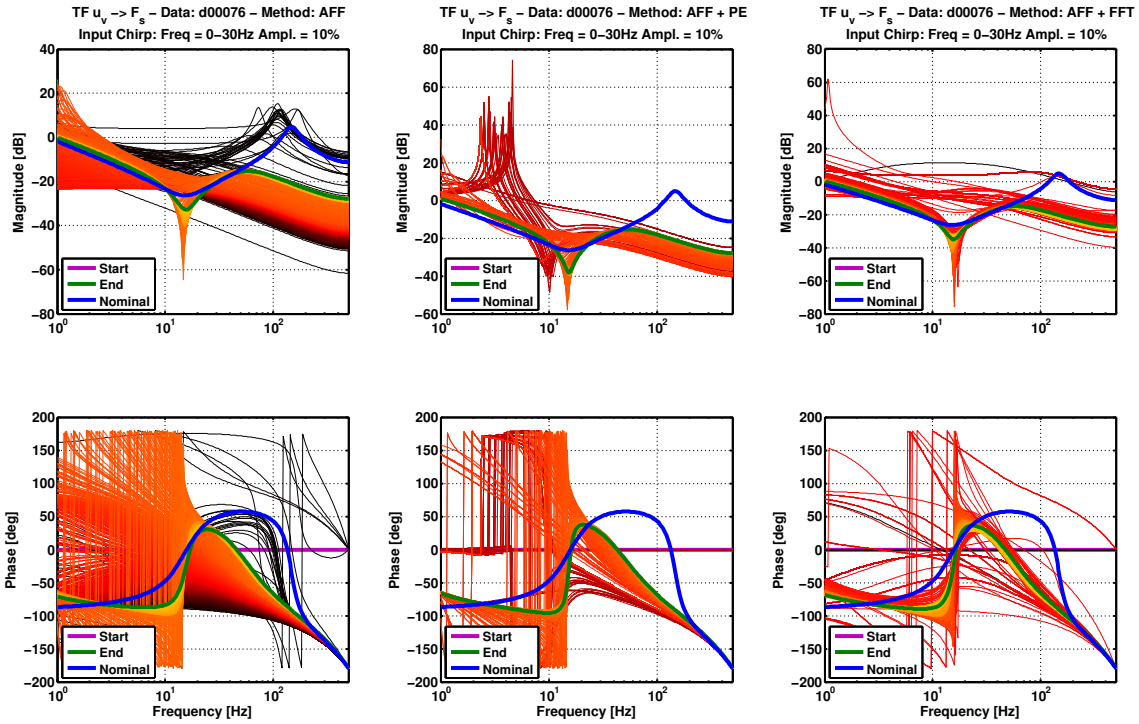

Figure 9.4: Bode plot of transfer function estimates (normalized) for each time step using FC1D test set-up. Shown are three different algorithms for the experiment *d*00076 with chirp input signal. Color scheme and layout are the same as in figure 9.1. The left column shows the AFF algorithm, the middle the combination of AFF and PE algorithms, and the right column the combination of AFF and FFT algorithms. The PE algorithm does lead to some improvements, however, the simpler FFT method on the right performs even better.

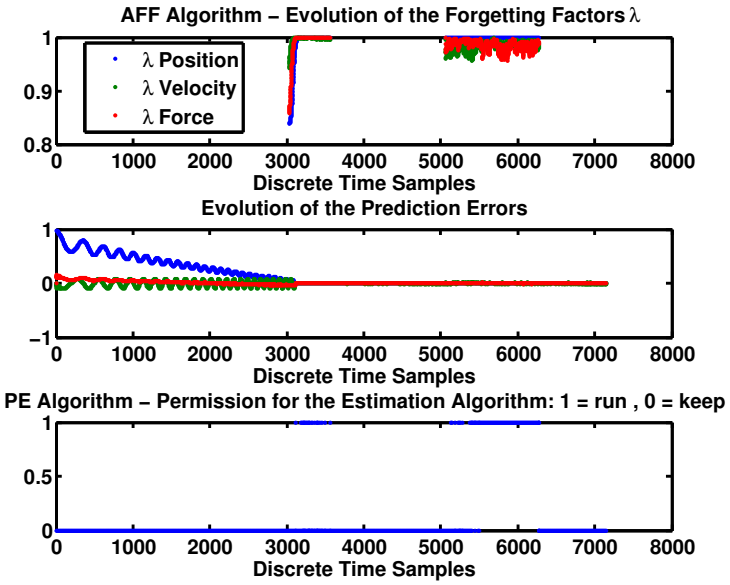

Figure 9.5: Plot of the individual forgetting factors λ_i calculated by the AFF algorithm for the combination of AFF and PE algorithms for the experiment $d00076$ shown in the middle column of figure 9.4.

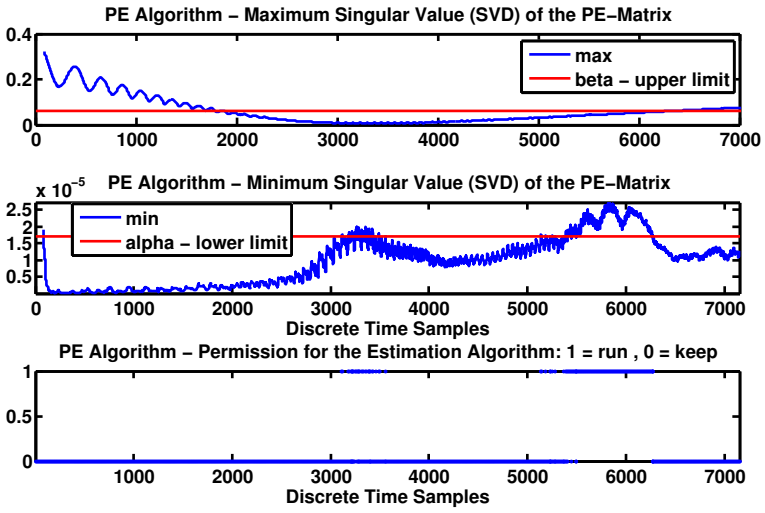

Figure 9.6: Plot of the minimum and maximum singular values calculated by the PE algorithm for the experiment $d00076$ shown in the middle column of figure 9.4. The top row shows the maximum values, the middle row the minimum, and the bottom row shows the permission for the estimator to update.

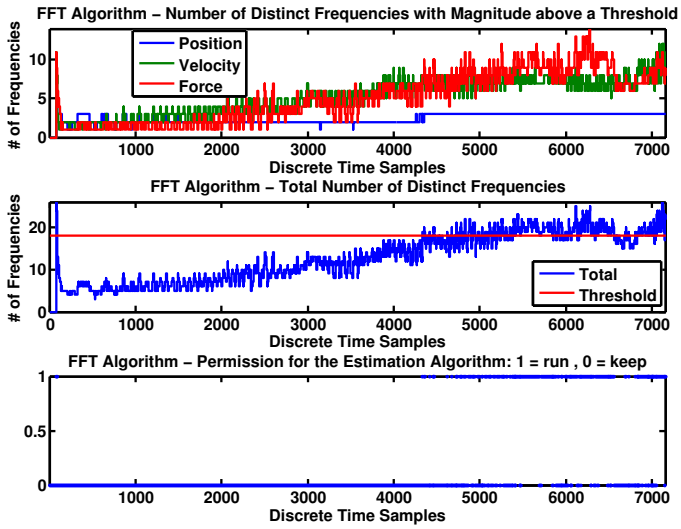

Figure 9.7: Plot of the distinct number of frequencies calculated by the FFT algorithm for the combination of AFF and FFT algorithms for the experiment *d*00076 shown in the right column of figure 9.4. Permission to update is given if the total number of distinct frequencies succeeds a threshold (red line).

clear why these peaks occur, it might have to do with the forgetting factors being low, hence the estimator is sensitive to noise. These peaks also appeared previously in other experimental data, e.g. in figure 9.1. These peaks are of course not tolerable and need to be eliminated in order to yield a robust adaption scheme.

5. One significant drawback of the PE algorithm is that the quality of the PE condition cannot be judged. It is hence not possible to eliminate the activation of the estimator around sample 3000 without compromising the activation after sample 5000.

Another method is the above mentioned frequency based algorithm (FFT) which calculates the frequency content of the input signals to the estimator and only gives permission when enough distinct frequencies are excited over some interval N. Thus the FFT condition is:

$$f_{min} < \sum_{i=k-N}^{k} \left[\sum_{\omega} mag\left(fft(Y)\right) > M_{min}) \right] \tag{9.5}$$

where mag is a function calculating the magnitude of the frequency response calculated by the function fft of the input measurement vector Y. The summation over all frequencies ω gives the number of frequencies where their magnitude is larger than some threshold M_{min}. The total number of frequencies over the interval N needs to be larger than a threshold f_{min}. If the condition is fulfilled, the estimator is given permission to update. The combination of AFF and FFT algorithm is shown on the right of figure 9.4 and the number of distinct frequencies as well as the permission to update are shown in figure 9.7. The corresponding figure for the forgetting factors is shown in the appendix by figure G.2. Again, several aspects are worth to be mentioned:

1. Consider figure 9.7, where it is expected that the number of significant frequencies rises contiguously with the frequency increase of the chirp input signal. However, at

the very beginning of the experiment, a high number of excited frequencies occurs. The algorithm probably catches the rich dynamics when the pressure builds up and the piston starts to move. A similar behavior is also present in figure 9.6 for the PE algorithm.

2. Unfortunately, this burst of excitation is not long enough or the algorithm does not take full advantage of it as can be seen in figure G.2, where the prediction error is drastically reduced, however not enough, yielding inaccurate estimates.

3. In contrary to the behavior of the PE algorithm, the FFT algorithm does not show any increase in excitation in the middle of the experiment. The observed behavior is probably unique to the singular value decomposition.

4. The FFT algorithm has, albeit, a similar disadvantage in setting a suitable threshold for the minimum number of distinct frequencies necessary for a good estimate. There is again a trade-off between catching short bursts of excitation (such as at the beginning of the experiment) and robustness against noise and disturbances influencing the excitation of individual frequencies.

9.1.3 Experiment for Different System Weights

Let us now study the performance of the estimator when the mass of the system is changed. The mass of the system is changed by adding weights used in gymnastics on top of the cart where they are slid over a threaded rod and secured by two nuts. The estimator, along with the previously described enhancements, is run for experiments with $+5\,\mathrm{kg}$ and $+15\,\mathrm{kg}$ of additional mass. The estimates for the two experiments ($d00089$, $d00091$) are shown in figure 9.8 in the middle and right column. The left column shows the estimates for a system without additional mass. The magenta line shows an arbitrarily chosen initial system whose purpose is to show adaption speed and convergence. The magenta line basically imitates an estimator state before a system change has occurred. The blue system is again calculated based on a nominal system without mass and an operating point around zero.
Again, several things are worth to be mentioned:

1. Comparing the early estimates (black lines) between the three experiments one can readily see that additional weight directly influences the system's behavior and its estimates during the low frequency part of the input signal. The system estimates on the left (no additional weight) remain longer around the nominal system response.

2. Adding additional weight shifts the resonance peak to lower frequencies, as has been studied and shown in figure 4.8.

3. Comparing the final estimates (green line) between the three experiments, it is apparent that adding weight has an influence on at least the friction term since the anti-resonance around $10\,\mathrm{Hz}$ is significantly suppressed. This behavior is not captured by the linear system discussed in chapter 2. The influence of bearing compression due to weight and other nonlinear influences are not included in the model and are probably responsible for this effect, which is especially visible for the heavy weight situation shown in the right column.

4. Interestingly, adding weight also reduces the influence of the valve opening on the system behavior. Comparing the left versus the right column, it can be observed

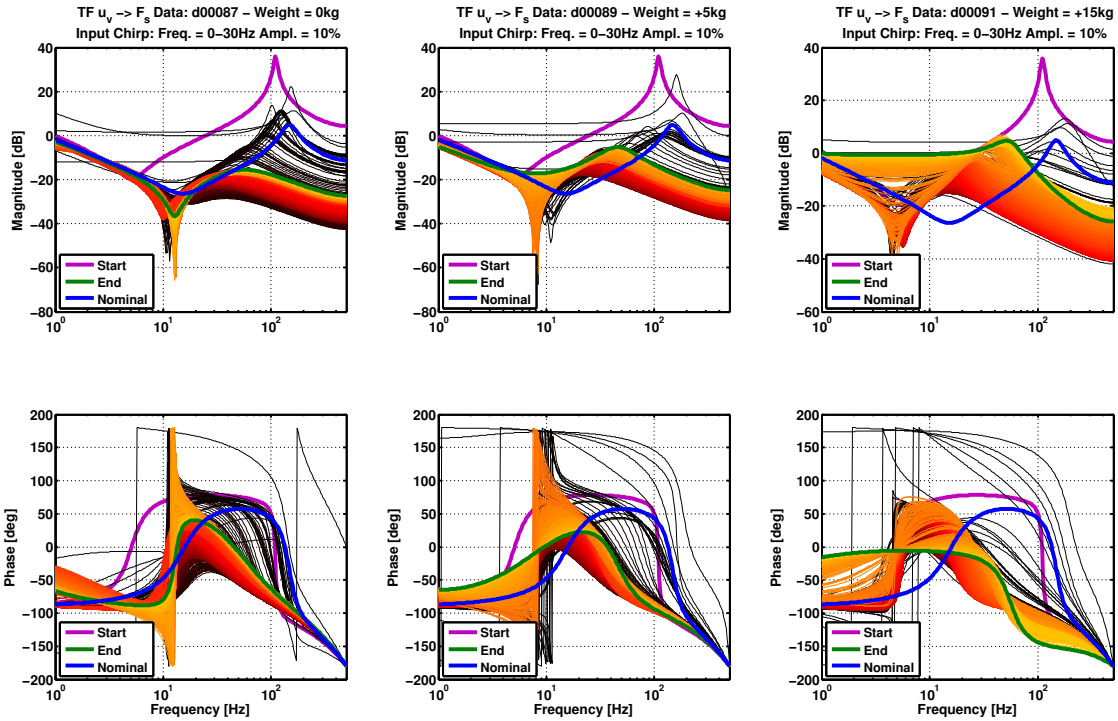

Figure 9.8: Bode plot of transfer function estimates (normalized) for each time step using the FC1D test set-up. Shown are three experiments for additional weights $0, +5, +15\,\mathrm{kg}$ using a chirp input signal. Color scheme and layout are the same as in figure 9.1. The left column shows no additional weight, the middle an addition of $+5\,\mathrm{kg}$ and the right column of $+15\,\mathrm{kg}$. All experiments are run using the enhanced AFF algorithm.

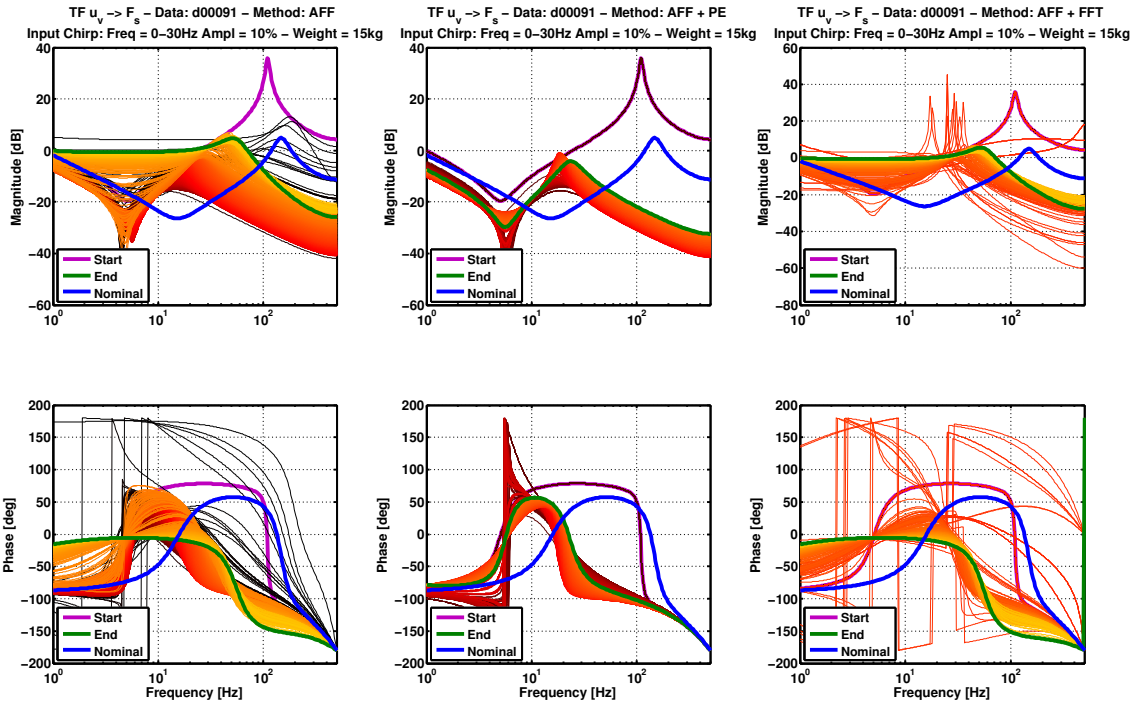

Figure 9.9: Bode plot of transfer function estimates (normalized) for each time step using the FC1D test set-up. Shown are three estimation methods for additional weight +15 kg using a chirp input signal. Color scheme and layout are the same as in figure 9.1. The left column shows the AFF algorithm, the middle the combination of AFF and PE algorithms, and the right column the combination of AFF and FFT algorithms. The PE and FFT algorithms are both sensitive to the measurement signals and do not yield significantly better results. Hence, the simpler AFF method on the right performs better.

that, for estimates towards the end of the experiments, the resonance peak is heavily suppressed. Less so in case of added weight, where a resonance peak around 40 Hz is still visible. Again, this nonlinearity was not captured by the linear model.

5. The transfer function in magenta can basically be viewed as the last estimate before a system change occurred. One can observe that the estimator is able to react very quickly to these changes. The time during which the prediction error of the model is high is kept at a minimum. In general, this property is beneficial for an adaptive controller.

6. These three experiments demonstrated the linear estimator's ability to react quickly to system changes and showed its ability to describe the linearized behavior of the nonlinear system at the current operating point.

The performance of the PE and FFT methods for the experiment *d*00091 is shown in figure 9.9. Similar figures for the other two experiments are shown in the appendix G, figure G.3 for experiment *d*00087 and figure G.4 for experiment *d*00089. The AFF algorithm is shown in the left column of figure 9.9, the PE method in the middle and the FFT algorithm on the right. Observe how the PE algorithm turns the estimation on during the middle of the experiment and off again towards the end. The FFT algorithm activates the estimation only towards the end of the experiment when the frequency content is high. Only the

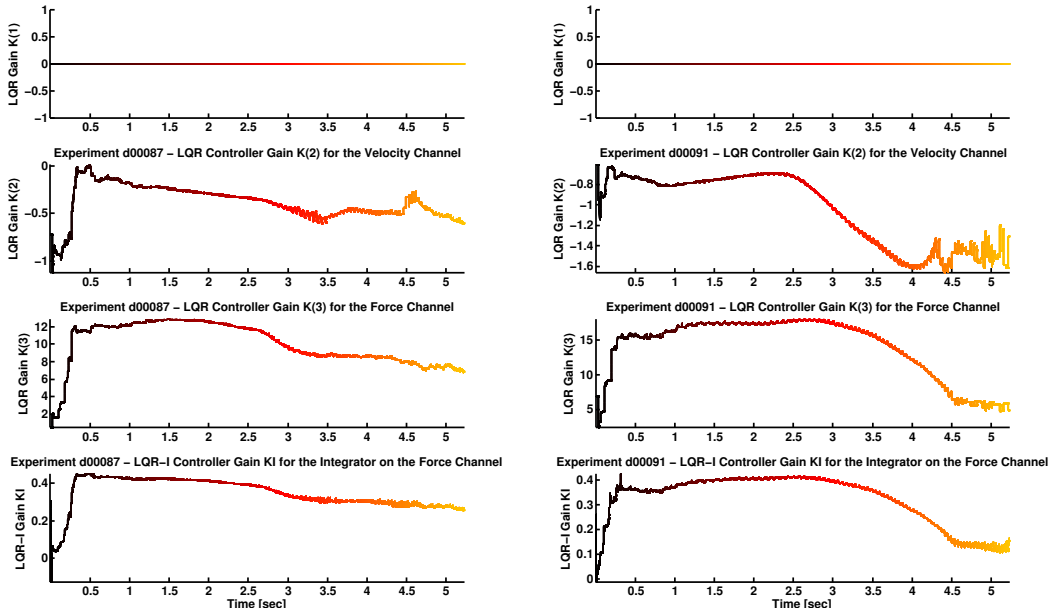

Figure 9.10: Evolution of the controller gains for each time step using the estimated system descriptions derived with the FC1D test set-up. On the left side the evolution of the "compliant" LQR controller gains is shown for experiment *d*00087 and on the right for *d*00091 (an additional weight of +15 kg). The system estimates were calculated using the AFF algorithm and are shown in figure 9.9 and 9.8, respectively. Color scheme and layout are the same as in figure 9.1.

AFF algorithm captures the system for the full experiment. The PE and FFT algorithms are very much dependent on tuning parameters and behave differently for each situation and input signal. These enhancements are hence not robust in the sense of an universal applicability. The AFF algorithm delivers an acceptable overall performance with minimal tuning and its parameters do not depend on the input signal but on hardware properties which are fairly constant and known. Similar observations can be drawn from figures G.3 and G.4.

9.1.4 Controller Gains Calculated Based on System Estimates

The controller gains are calculated based on the estimated system and weighting matrices. The calculation is deterministic in the sense that the results do not differ whether they are calculated in real-time or off-line using the same system estimates. Unfortunately, the hardware of the test set-up was not able to run both the estimator and controller functions at the same time in real-time mode, thus the controller gains are calculated off-line for demonstration purpose on the system estimates obtained above. Figure 9.10 shows the evolution of the controller gains for experiment *d*00087 and on the right for *d*00091, which has an additional weight +15 kg. The system estimates were calculated using the AFF algorithm and are shown in figure 9.9 and 9.8, respectively. Controller gains for the "compliant" LQR are calculated, that is without position gains, see section 9.2.4 for more details. Additional figures for the evolution of the controller gains for the experiment *d*00087 based on estimates calculated with the PE and FFT method are shown in appendix G.
Observe how the controller gains increase steadily and reach steady-state after about 0.3 s. The evolution of the controller gains is relatively smooth due to the AFF algorithm. It can be observed how the system changes during the experiment and towards the end of

the sequence where the controller gains have changed accordingly. The evolution is less smooth and it can be questioned if these fast changes at around $4 - 6\,\mathrm{s}$ will not lead to instability when applied to the real system. Unfortunately, the test set-up was not able to run the estimation and controller part at the same time, hence no experiments with the full adaptive framework could be performed. It can be expected that some logic is needed in order to prevent jumps and fast changes of controller gains, which would destabilize the system.

9.1.5 Concluding Remarks on the Estimation Experiments

The standard recursive least squares algorithm, as introduced in chapter 6, has its limitations especially when exposed to nonlinear fast changing systems such as the hydraulic system considered in this work. Even the three presented methods enhancing the capability of the estimator were not enough to handle the nonlinearities. However, the adaptive forgetting factor (AFF) algorithm was able to give satisfactory performance for many different systems and input signals without adjusting its parameters. The PE and FFT algorithms are only suited for one specific experiment because of their system and signal dependent parameters. Clearly, one would need to employ a true nonlinear estimator in order to capture the dynamics. Part of the problem is inherent to the system in use since its dynamics change rapidly for different valve openings. The estimator is essentially confronted with "different" systems for each valve opening, hence it is not clear which of these systems the estimator should describe. The linear estimator in use tries to mix all these different systems into one representation. At the most, the linear estimator gives a description of the linearized behavior of the nonlinear system around the current operating point. Only a nonlinear estimator would be able to handle the nonlinear characteristics. In addition, it is not clear which operating point of the system or which of these systems should be used for designing a controller.

9.2 Controller Experiments on the FC1D Test Set-Up

A vast amount of additional experiments for the adaptive controller (see section 7.2) and the state-of-the-art feedback linearization controller were carried out. These experiments assessed the performance of the LQR framework for different force reference signals (various amplitudes and frequencies) and for different variations in added load mass. Two different implementations of the LQR were also tested and compared, that is, a full-state LQR controller (herein referred to as "non-compliant") and a partial-state LQR controller (herein referred to as "compliant") without using the position state measurement. Part of the performed experiments were done for finding the most suitable cut-off frequency of the digital filters, tuning the parameters of the feedback linearization controller and so forth. Only the experiments with the latest software version are shown here, additional figures are shown in appendix G.

9.2.1 LQR Controller Performance for a Sine Wave with Different Frequencies

The following experiments where performed using the LQR controller designed for the nominal system, with load mass $m_l = 2.5\,\mathrm{kg}$, applied to the test set-up configured at nominal weight ($2.5\,\mathrm{kg}$). The old test set-up was used along with the non-compliant LQR

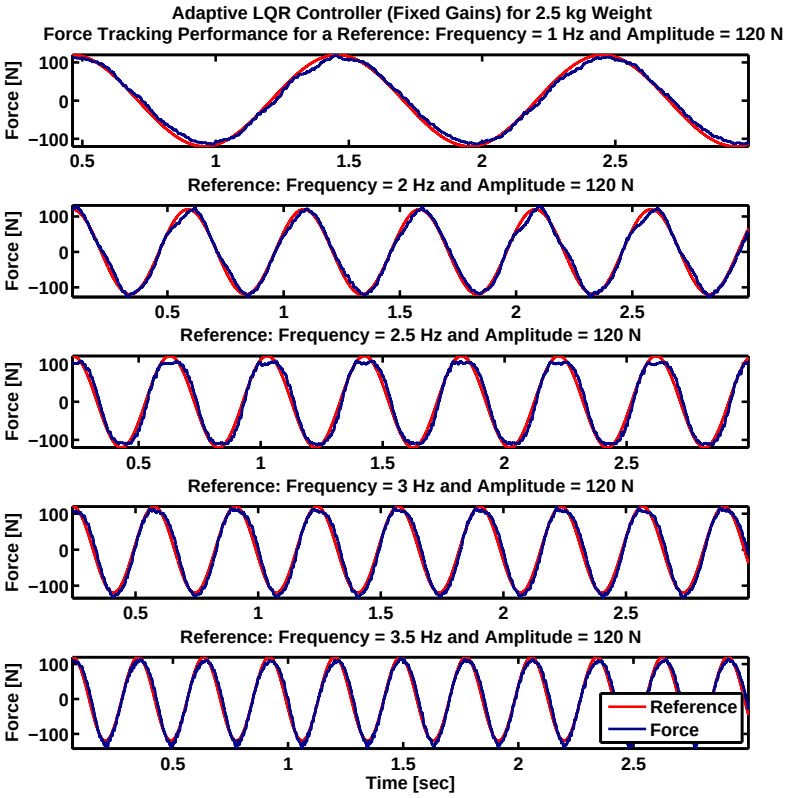

Figure 9.11: Force tracking performance of the adaptive LQR controller (fixed gain) for different sine wave reference signals using various frequencies performed on the old test set-up. This LQR controller also uses the position channel to control the force. As can be seen, the force tracking is rather good, regardless of reference signal frequency.

controller version. Five sinusoidal reference signals were commanded each with an amplitude of 120 N but different frequencies ranging from 1 Hz to 3.5 Hz. Figure 9.11 shows the performance of the force tracking capabilities of the LQR controller (fixed gain). Next, experiments for sine wave reference signals with different amplitudes were carried out, where three sinusoidal reference signals were commanded each with a different amplitude ranging from 60 N to 260 N and frequency of 1 Hz (except for the first signal using 0.5 Hz). Figure 9.12 shows the performance of the force tracking capabilities of the LQR controller. When examining these two figures, it can be seen that the force tracking performance is quite good, even for signals with a higher frequency of 3.5 Hz. Hence, we can conclude that the LQR controller is able to give satisfactory force tracking performance when designed for the correct system.

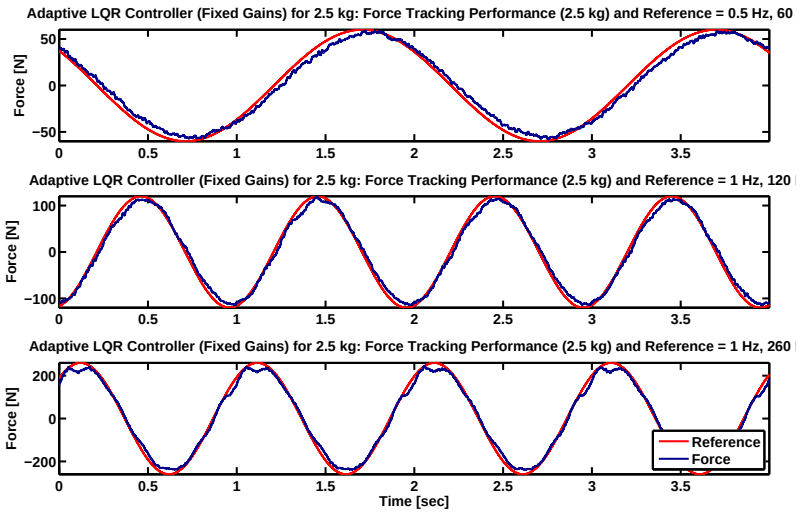

Figure 9.12: Force tracking performance of the adaptive LQR controller (fixed gain) for different reference signal amplitudes. The force tracking, as can be seen, is quite good, regardless of reference signal amplitude.

9.2.2 LQR Controller Performance for Step Signals

Experiments using a step in the force reference were performed to study the rise time and overshoot characteristics of the "compliant" LQR controller. The new test set-up in nominal configuration was used for these experiments. The LQR augmented by an integrator component was used, as detailed in section 7.2. Figure 9.13 shows two experiments (top row) commanding a step from 0 N to 400 N and a step from 400 N to −400 N giving a total difference in the reference force of 800 N. Note that the controller is able to reach the step very quickly albeit with some overshoot. The trade-off between overshoot and rise time can be adjusted using the weighting matrices for the LQR calculation. The steady-state force tracking is very good and is reached fast. The individual contributions of the different components of the LQR controller are shown in the second row in figure 9.13 just below the force signal plot. The total LQR controller output u_{tot} is the summation of the contributions from the velocity u_{vel}, force u_{force} and feedforward $u_{forceff}$ terms. The integrator part of the controller is not shown. Note that the total controller output is shown in normalized form. From the characteristics of these internal controller signals one can see that the velocity channel contributes the most to the total signal. In other words, velocity compensation is essential for good force tracking. The contribution of the force channel is similar but smaller in magnitude. The feedforward term is very small, though could be increased via some logic depending on the reference signal encountered during operation. Note how the rise of the force curve is steep and steady without ripples. As can be seen, the force tracking characteristics do not change much when commanding positive steps or negative steps. Some difference is visible for the negative step, the overshoot is less and the curve seams to be less steep. This is probably due to the asymmetric cylinder used, which affects the force-pressure relationship[2].

9.2.3 LQR Controller Performance for Chirp Signals

A chirp reference signal is useful because the controller's phase lag can be studied. Figure 9.13 shows (third and fourth rows) two experiments commanding a chirp reference. The experiment on the left commands a chirp signal with 400 N amplitude and $0 - 3$ Hz frequency range. The experiment on the right uses the same chirp signal but for frequencies in the range: $0 - 5$ Hz. The test set-up and controller are again the same as before (see subsection 9.2.2). Note that the controller, for the chirp signal on the left, is able to follow the force reference without loosing phase and with little amplitude distortion. In contrast, the controller is not able to follow the higher frequencies of the chirp signal on the right, hence a slight phase lag appears for frequencies above approximately 3.5 Hz.
The fourth row of figure 9.13 shows again the individual controller contributions. Otherwise invisible system characteristics directly influencing the controller's behavior can now readily be seen. The level of stick friction of the new test set-up is quite significant as can be seen by the distinct ripples of the velocity related controller output signal u_{vel} within the first few seconds of the chirp signal on the left of figure 9.13. These ripples occur when the velocity of the cylinder approaches zero and the cylinder does not have enough momentum to overcome the stick friction force. The cylinder will start to stick but is accelerated quickly again by the force term of the controller. As can be seen, the force term of the controller u_{force} leads ahead of the velocity signal (in other words: the velocity signal u_{vel}

[2] The negative side, that is the side of the cylinder with the rod, produces less force for the same pressure because its area is smaller compared to the other side without rod. See chapter 2.1 for details.

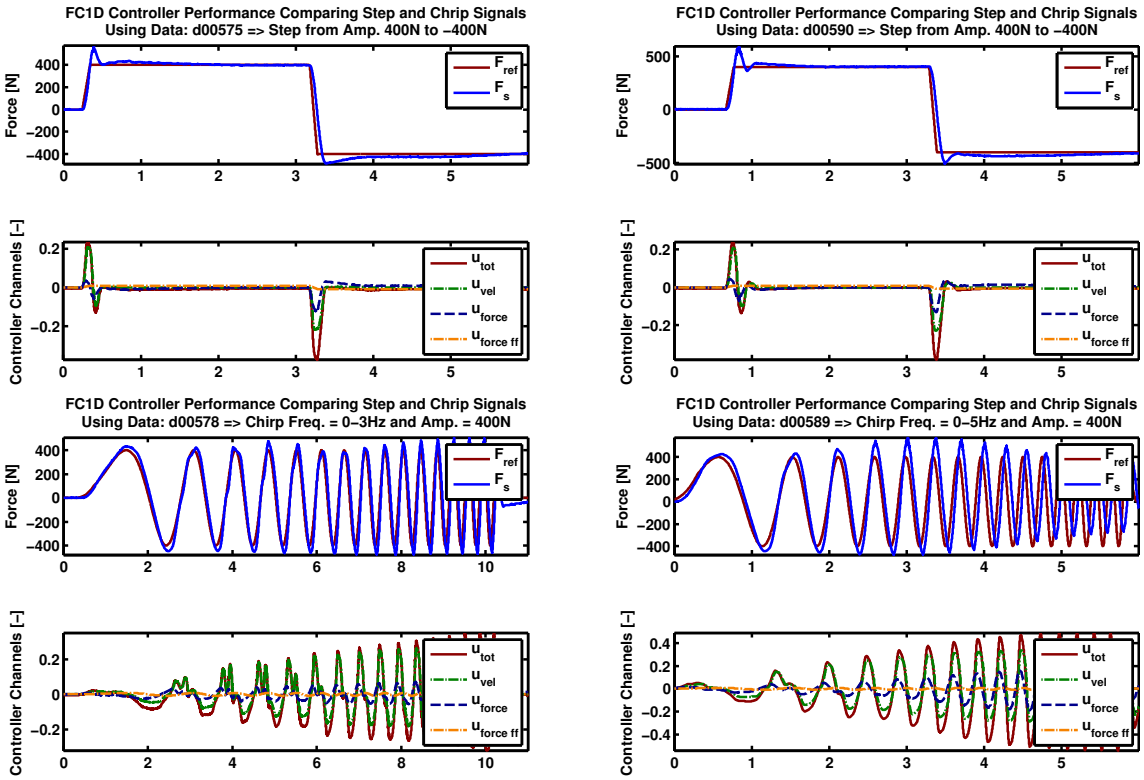

Figure 9.13: Force tracking performance of the adaptive LQR controller (fixed gain) for step signals, performed on the new test set-up using the "compliant" LQR controller version. The two experiments show a step from 0 N to 400 N and a step from 400 N to −400 N giving a total difference in the reference force of 800 N. Note that the controller is able to very quickly reach the step value albeit with some overshoot.

lags behind). This sticking and moving pattern vanishes as soon as the cylinder has enough momentum to overcome the stick friction component, as is observed towards the end of the chirp signal. The experiment on the right side shows only little sign of the just described phenomena because the frequencies are higher and hence the momentum and velocity are larger preventing the cylinder from sticking.

Note also that the controller following the force reference starts to overshoot towards the middle of the chirp signal experiment on the right side of figure 9.13. The overshoot mutates into an undershoot towards the end of the experiment. The overshoot visible between $2-4\,\mathrm{s}$ is probably due to the decrease in friction related forces according to the well known Stribeck friction curve [25, page 71], which is velocity dependent. As the velocity increases, the fricition related forces decrease, hence too much input is applied by the LQR controller unaware of this nonlinear dependency. The friction related forces eventually increase again for increasing velocity. The mutation into the slight undershoot visible towards the end of the experiment is probably due to the change in system dynamics, namely the decrease in input to force magnitude for increased valve opening and frequency. The sensitivity of the load cell force transfer function has been studied in chapter 4, where figure 4.6 readily reveals the basic cause. As the controller demands more input to follow the increased frequency of the force reference, the valve input increases accordingly. Hence, the system's force response mutates from the blue curve to something in between the green and red curve, as shown by figure 4.6. The relevant frequency range is also shifted from at around $1\,\mathrm{Hz}$ towards $5-10\,\mathrm{Hz}$. The magnitude of the force response decreases by about $10\,\mathrm{dB}$, hence intensifying the undershoot tendency. Again, the LQR controller is unaware of these dynamic changes in the behavior of the system because of the linear state space model chosen around the nominal operating point.

9.2.4 Comparison of the "Compliant" vs. "Non-Compliant" LQR Controller Versions

During testing it was observed that quite a high feedforward term was necessary to achieve adequate force tracking, which is not intuitive. Examining the individual contributions of the internal controller terms, it was found that the position channel was acting against the force and velocity channel, hence the feedforward term was needed to compensate for the position influence. The LQR controller was derived by setting the weighting matrices components for the position and velocity channel to a very small (or even zero) value such that the controller is allowed to use velocity compensation as much as needed. However, it also means that the controller tries to bring the system to its zero state using the "cheap" position channel. Figure 9.14 shows the force tracking and individual controller outputs for the "non-compliant" (on the left) and the "compliant" (on the right) LQR controller derived from and applied to the nominal system using the new test set-up. The full LQR controller is called "non-compliant" because it produces a valve command dependent on position measurements. The "compliant" LQR controller is derived by setting the gain for the position channel to zero, hence it uses more compliant measurements. This "compliant" LQR controller is not fully compliant because of the velocity feedback, feedforward term and added integrator, all of which are not purely dependent on the load cell force measurement. This terminology is only used to easily distinguish between the two LQR variants. Figure 9.14 shows a sine wave as a reference force signal with frequency $1\,\mathrm{Hz}$ and amplitude of $300\,\mathrm{N}$ and $200\,\mathrm{N}$, respectively.

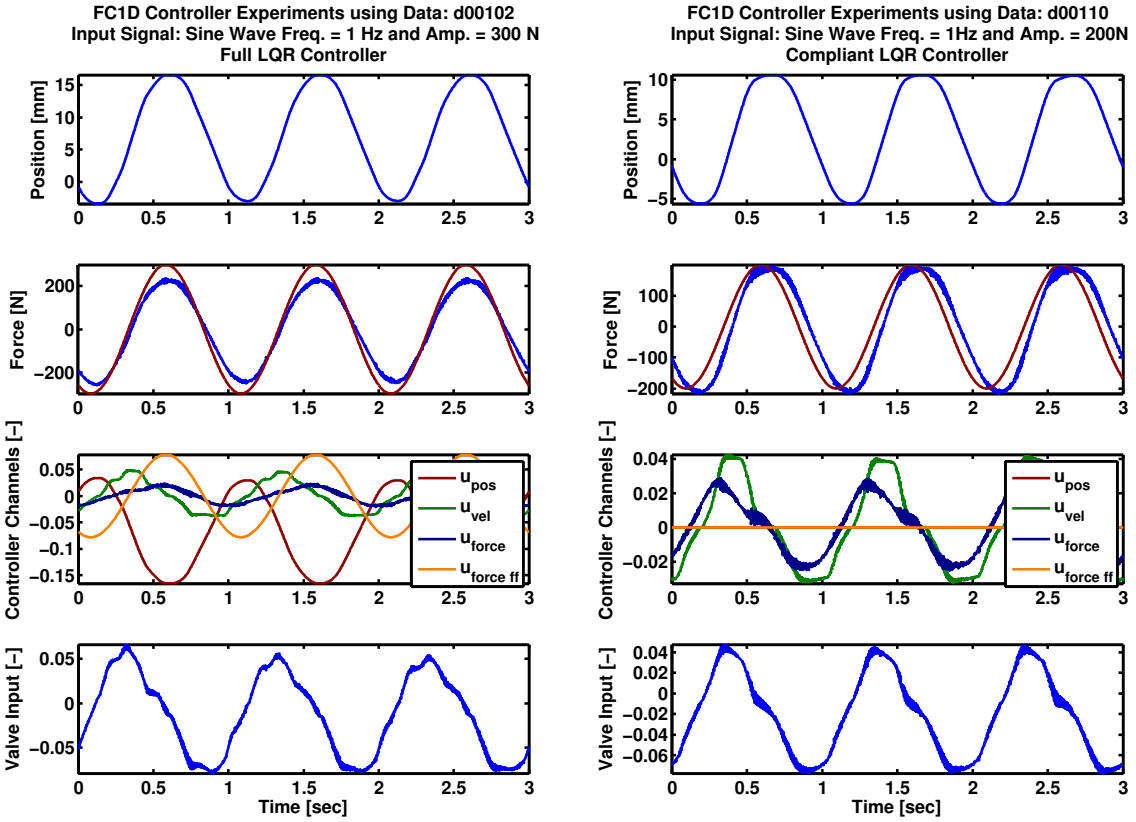

Figure 9.14: Comparison of the "compliant" vs. "non-compliant" LQR controller versions for a sine wave. Note how the position channel of the full LQR controller fights the velocity and force channels. A force feedforward term is needed to compensate for the position influence. In contrary, the "compliant" LQR (on the right) does not need position or feedforward terms to give good force tracking. However, a slight phase delay is visible.

As can be seen, the force tracking of the "non-compliant" LQR controller (left) is quite good. There is some small phase delay in the force tracking when using the "compliant" LQR controller (right). Considering the third row of figure 9.14, it is readily seen how the position channel u_{pos} (red) fights the velocity u_{vel} (green) and force u_{force} (blue) channels. Hence, a force feedforward term $u_{forceff}$ (orange) is needed to compensate for the position influence. On the right side, one can immediately see that there is little loss in tracking performance when eliminating the position channel, and hence the feedforward term can also be omitted[3].

Similar behavior can also be observed when looking at the force tracking of a step reference signal as shown in figure 9.15. The position channel of the "non-compliant" LQR controller is again acting against the velocity and force channel. The influence of the position channel is strong and prevents the force from reaching the commanded reference value. In contrary, the "compliant" LQR controller has a fast rise time and reaches steady-state behavior quickly. Note that the reference signal command for these experiments is a vector of three values: $\vec{x}_{ref} = (0, 0, f_{ref})^T$. The force tracking of the full LQR controller ("non-compliant"

[3]The feedforward term is still useful even for the "compliant" LQR variant because it can be used to compensate for some of the nonlinear dynamics of the system. Some of the shown controller experiments use a small amount of feedforward term.

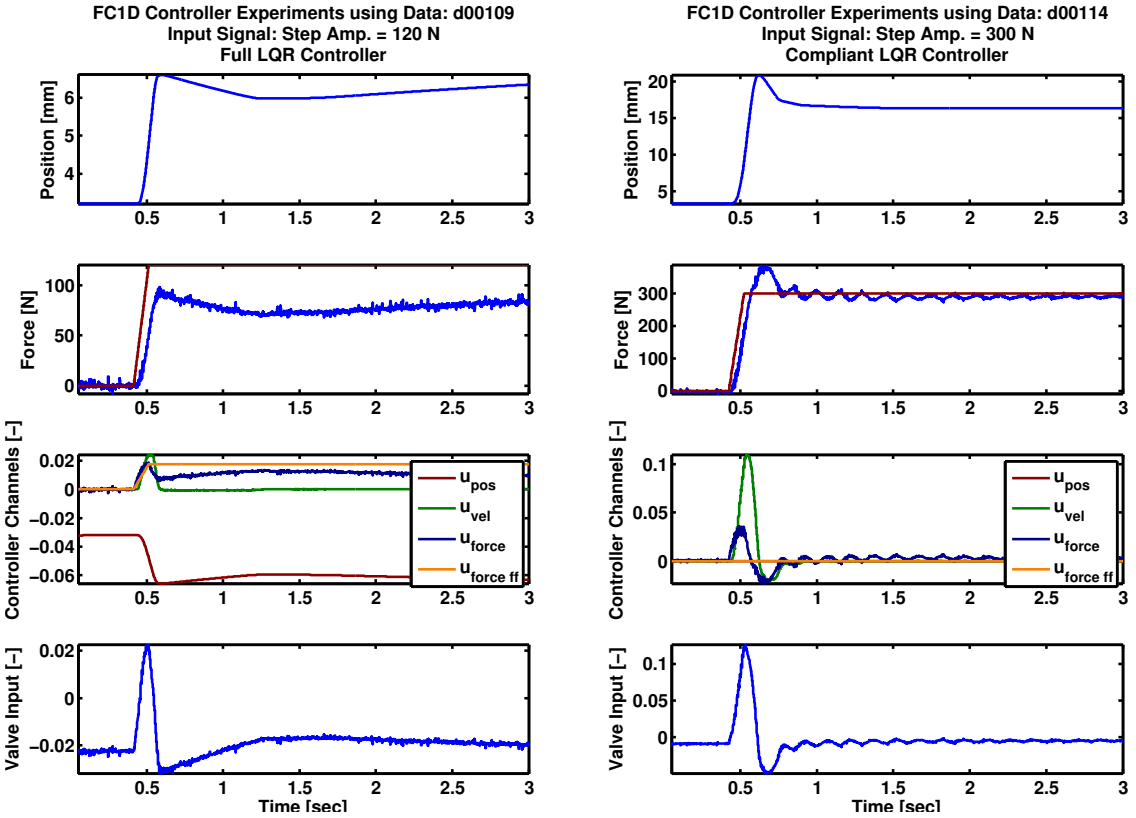

Figure 9.15: Comparison of the "compliant" vs. "non-compliant" LQR controller versions for a step reference signal. Note again how the position channel of the full LQR controller prevents the force from reaching the commanded steady-state force value. The added integrator term eventually drives the system to zero force tracking error. In contrary, the "compliant" LQR (on the right) does not need position or feedforward terms to give good force tracking. The rise time is very fast without relevant steady-state error.

variant) could be improved by using a clever position reference encoding the relationship between force and position due to the springs used on the new test set-up. This would probably eliminate some of the negative effects of the position channel on the force tracking.

9.2.5 Comparison with the Feedback Linearization (FL) Controller

Figure 9.16 compares the force tracking capabilities of the LQR controller to those of the state-of-the-art feedback linearization (FL) controller that currently runs on the robot HyQ. The FL controller was first implemented into the FC1D code and then tuned to give a satisfactory performance. The experiments were performed in the same fashion as those for the LQR controller. When comparing the responses, it can be seen that the force tracking performance is almost identical, hence both controllers are able to give satisfactory performance to a well-known system remaining close to the operating point the controller was designed for. Note that this LQR controller does not use feedback linearization nor an explicit velocity feedback compensation but is still able to give good force tracking since it uses state feedback. The performance of the FL controller might be improved by more

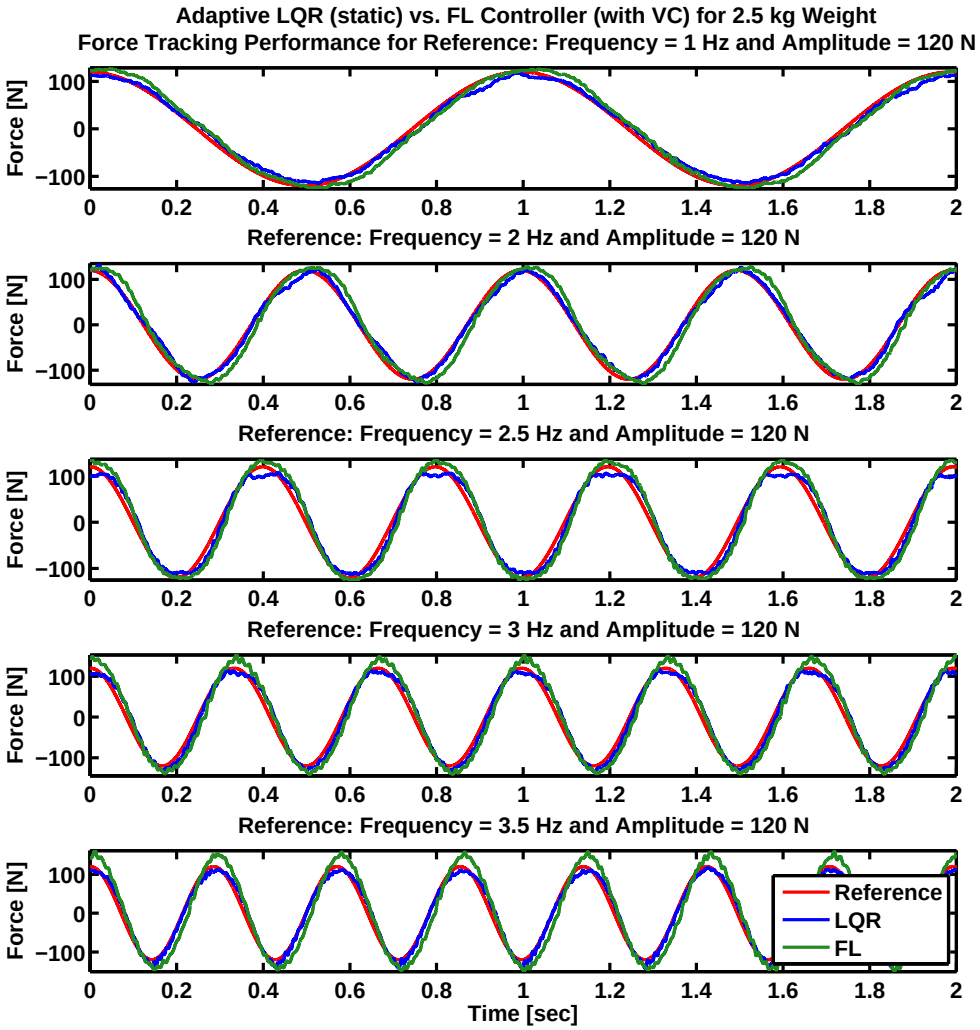

Figure 9.16: Comparison of the force tracking capabilities of the feedback linearization (FL) controller versus the LQR controller. The FL controller was tuned specifically to give a good response. Note that this LQR controller does not use feedback linearization nor an explicit velocity feedback compensation but is still able to give good force tracking since it uses state feedback.

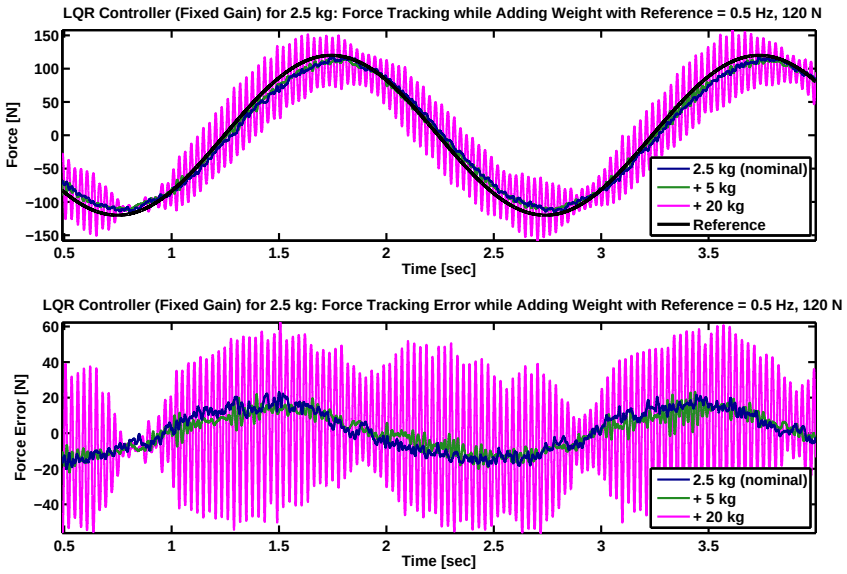

Figure 9.17: Force tracking performance of fixed gain LQR controller for a 0.5 Hz sinusoidal reference and amplitude 120 N. The LQR controller is designed for the nominal system; while the experiment is running, additional weight is added to the cart of the test set-up. Note that the LQR is able to cope with the 5 kg additional weight (green) due to the phase margin. However, 20 kg additional weight (magenta) are too much and the response becomes unstable.

extensive tuning, especially regarding the small overshoot present for the reference signal shown at the bottom of figure 9.16.

9.2.6 Control Performance for Different Amounts of Added Weight

The same LQR controller for the nominal system (load mass $m_l = 2.5\,\text{kg}$), as used above, is used again, but this time weight is added to the cart to alter the system dynamics. The test set-up is initialized with a nominal weight of 2.5 kg, then the controller is selected and a reference signal is commanded. While the controller tracks the reference, weight is added to the cart in 5 kg increments and the force tracking is observed. The idea of this test is to show the potential increase in force tracking performance that can be gained when using the right controller for the system. Also the robustness of the LQR controller is investigated. The impact on control performance for a sinusoidal reference signal with frequency 0.5 Hz and amplitude 120 N is shown in figure 9.17, whereas figure 9.18 shows the impact for a similar reference signal but with frequency 3.5 Hz.
When examining the above mentioned figures, it can be seen that the LQR controller (with fixed gains) is able to cope with some small amounts of added weight $\leq$ 10 kg without much decrease in tracking performance. However, when adding more weight, e.g. 20 kg (9 times more than the nominal weight), the controller is no longer able to track the reference - it even becomes unstable. A similar experiment using a LQR controller designed for a system with heavy weight +22 kg was also carried out. Weight was removed while the controller was tracking the reference. However, problems with the filter resulted in unreliable measurements. Nonetheless, a similar trend was observed as noted for the experiments above.
Experiments for step and ramp reference signals were also performed. These are shown in appendix G, where figure G.9 shows the tracking of a force ramp reference signal and

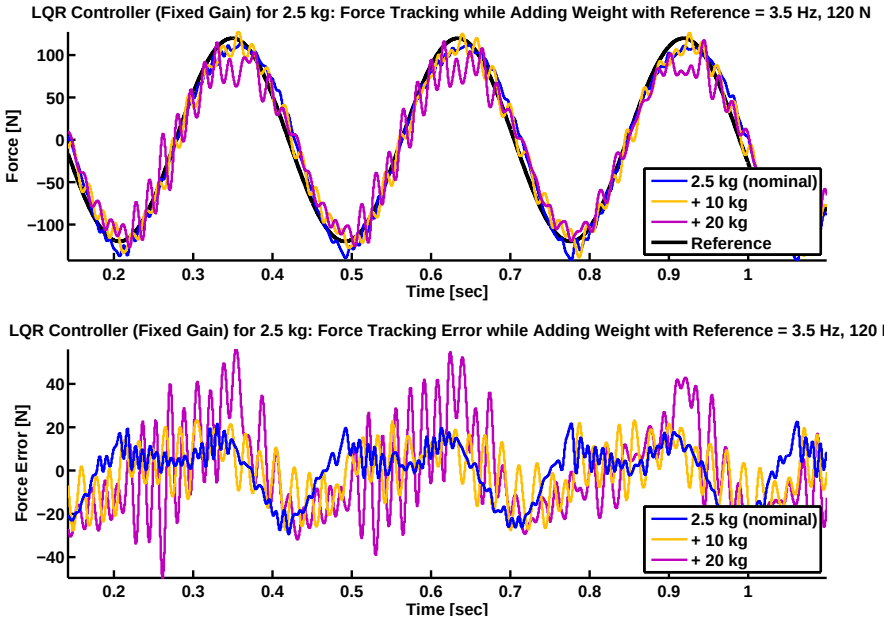

Figure 9.18: Force tracking performance of fixed gain LQR controller for a 3.5 Hz sinusoidal reference and an amplitude of 120 N. The LQR controller is designed for the nominal system; while the experiment is running, additional weight is added. Note that the LQR is barely able to cope with the 10 kg additional weight (gold) due to its phase margin. Additional 20 kg weight (magenta) are too much and the response becomes unstable, as has already been seen for the reference signal with lower frequency (see figure 9.17).

figure G.10 shows the tracking for a step reference signal. Two different amplitudes: 200 N and 400 N were commanded for both experiments, respectively. It can again be observed that the force tracking deteriorates for increased weight (+20 kg) when the controller gains remain fixed.

9.3 Identifying the Parameters of the FC1D Test Set-Up

9.3.1 Friction of the Load Mass and Cart

Friction is usually described as a combination of three terms (see for example [21]): the Stribeck friction, the Coulomb friction, and the viscous friction. The viscous friction coefficient of the load mass and cart of the FC1D (old) test set-up is estimated by moving the piston with constant velocity and measuring the force. The springs were removed from the test set-up in order to simplify the identification. Using the load cell force equation (2.22) and setting the acceleration and spring terms to zero, the following expression for identifying the viscous friction coefficient is obtained:

$$\hat{d}_l = \frac{|F_s|}{|v|} \tag{9.6}$$

where $|\dots|$ denotes the absolute value of the variable, v is the commanded constant velocity and $\hat{d}_l$ denotes the estimate. Several experiments were performed using different velocities in both directions of piston motion. The sensors were calibrated, offsets were removed and the mean of the corresponding data segments was used to calculate the estimates shown in figure 9.19.

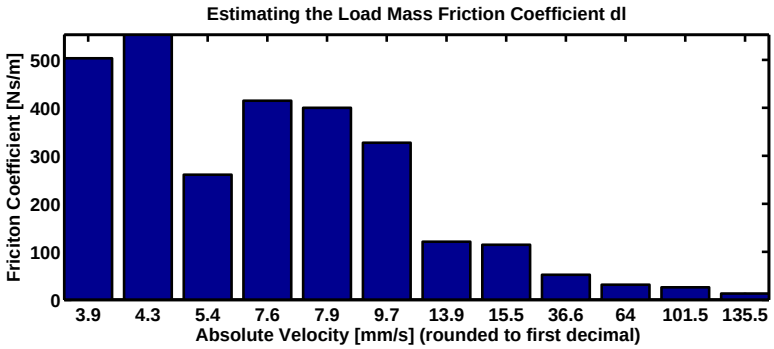

Figure 9.19: Estimation of the viscous friction coefficient of the load mass and cart of the FC1D (old) test set-up. The coefficient is estimated by moving the piston with constant velocity and measuring the force. The coefficient of viscous friction is then calculated using $\hat{d}_l = \frac{|F_s|}{|v|}$. The third bar probably is an outlier.

As can be seen, the viscous friction coefficient is quite heavily velocity dependent. It seems that there is Stribeck friction present at low velocities; however, not enough measurements are available to surely identify the Stribeck friction coefficient. The third bar probably is an outlier and too few experiments were captured in order to obtain reliable data.

9.3.2 Spring Stiffness

The FC1D test set-up uses four identical springs of 300 mm length; the mean coil diameter is 40 mm and the wire size 5 mm. According to the specification sheet, the spring stiffness is around 5360 N/m. A few experiments were performed to identify the spring stiffness of the FC1D test set-up. The piston was moved to fully extended or retracted position and the force at these locations was measured over some time window. The stiffness C of the springs is thus the measured force F_s divided by the displacement y from the middle position:

$$\hat{C} = \frac{|F_s|}{|y|} \tag{9.7}$$

where $|\ldots|$ denotes the absolute value of the variable and $\hat{C}$ denotes the estimate. Figure 9.20 shows the estimated stiffness per measurement along with the data sheet stiffness value and the mean over all measurements. The mean spring stiffness value for a single spring of about 5278 N/m is almost identical to the value listed in the specification sheet. The slight difference may be explained by aging effects, nonlinearities, manufacturing tolerances, and heavy use. Additionally, since the stroke of the piston is only 8 cm and the spring stiffness is very high, even the smallest calibration error can result in large variations of the estimated stiffness. The slightly lower estimated stiffness of the first two experiments (d00272 and d00273 in figure 9.20) may be due to the fact that they were performed on a different day with a slightly different sensor calibration.

9.3.3 Mass of Load and Cart

The cart was removed from the test set-up and was weighted on a scale, registering a total weight of $m_l = 2580$ gr. Standard weights commonly found in a gym were used to add weight to the cart, in increments of 5 kg. This brought the total possible weight of the cart up to 22.5 kg.

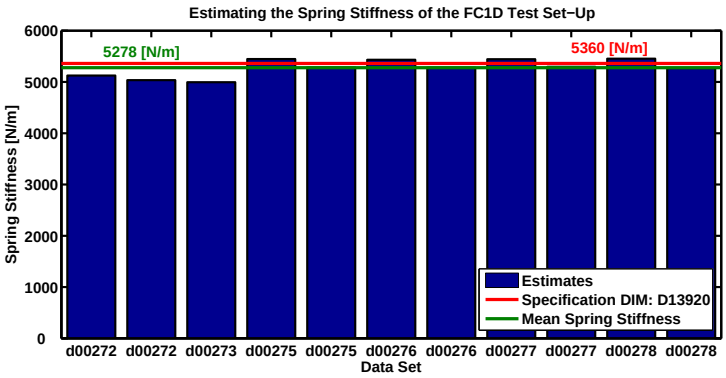

Figure 9.20: Estimation of the spring stiffness on the FC1D test set-up. Each data set contains measurements for the fully extended and retracted piston. Shown are the estimates, the spring stiffness according to the specification sheet and the mean stiffness over all measurements. The identified spring stiffness agrees very well with the specification data.

9.3.4 Valve Input Threshold

The valve deadband is "the null region associated with a spool overlap condition" [39]. Spool over- or underlap has an influence on the flow gain at zero. Overlap reduces the null flow gain and null leakage flow, whereas underlap increases null flow gain and valve null leakage [31]. For a valve with overlap there is a certain distance that the valve spool has to move until pressure starts to build up. Because of the sudden start of oil flow, a pressure peak is induced. For an underlap valve the valve spool cannot fully suppress the flow of oil, hence a certain leakage (null flow) will always be present. For highly dynamic motions, a valve with slight underlap is preferred because very little valve spool movement is necessary to change the flow of oil. In addition, no pressure peak will be induced because there is always some leakage flow resulting in a smoother pressure build-up. The input threshold is the amount of input that must be supplied to the circuit board (from the electronics on the circuit board through the amplifier to the valve) until a change in pressure is attained. Figure 9.21 shows the identified input threshold for the Moog valve E024 [32] assembly (including all electronics and associated nonlinearities) used on the FC1D test set-up. The threshold is identified by commanding a slowly increasing valve command while monitoring the pressure sensor readings. As soon as the pressure starts to drop (or increase), the threshold is surpassed (indicated by the green triangles). The identified threshold is about 0.6 % of the valve opening and is identical to the value given in the specification sheet of the valve [32], which lists a threshold of $0.5\,\% - 1\,\%$ (in percentage of the valve opening) depending on the supply pressure.

The valve deadband can have a significant influence on the control performance depending on the controller in use, the actual reference signal, and the current system state. Figure 9.22 shows another experiment where a step of amplitude 400 N was commanded. The top row shows an overview and the lower three rows show details for the selected range as indicated by the black ellipse. The start of the reference signal (second row on the right) is indicated by a magenta line. One can see that the valve input command (shown just below) immediately increases. However, there is a 10 ms time-delay until the valve opens enough permitting an increase in the load pressure. The green line indicates the point where the load pressure starts to increase, obviously this point of increase is exactly related to the amount of valve deadband, which is again, as shown by figure 9.21, at around 0.6 % of

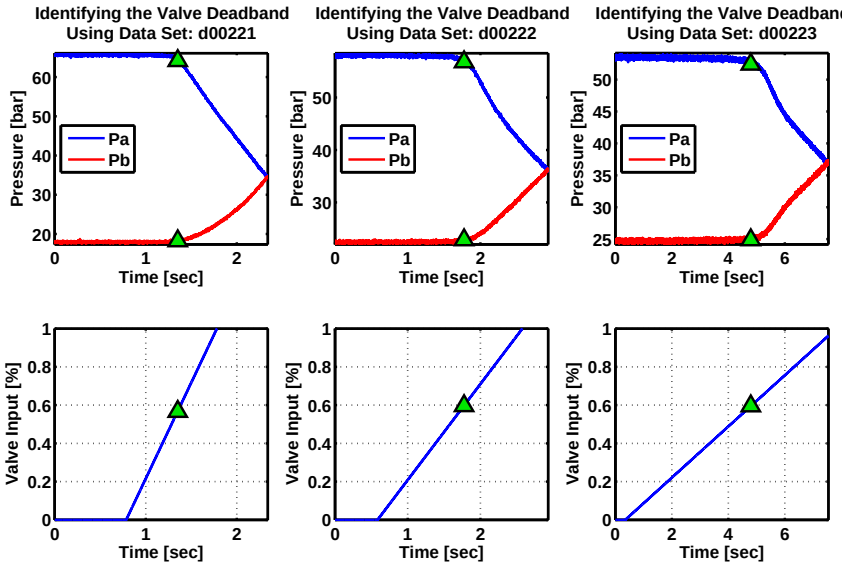

Figure 9.21: Identification of the valve threshold around zero inputs. A slowly increasing valve input is commanded and the pressure signals are monitored. As soon as the pressure drops (or increases), the threshold is surpassed and a green triangle is drawn in the figure. The identified valve threshold is about 0.6 % of the nominal valve opening.

the valve opening. The amount of time-delay depends on the rate of valve input increase or decrease. Valve inputs rising faster result in a lower time-delay, whereas slower inputs experience a longer time-delay. Some sort of a compensation scheme would be needed to overcome this time-delay issue.

9.4 Load Pressure Oscillation

While performing the various experiments, very fast vibrations were observed whose impact on the system could be heard rather clearly (sound wave). This section tries to find the cause of these high frequency oscillations. Figure 9.23 shows an experiment where a step of amplitude 400 N was commanded. The top row again shows an overview and the lower three rows show details for the selected range as indicated by the black ellipse. The comment (1) indicates the start of the valve command and points out that the system input is perfectly smooth. As the controller increases the valve input command, the valve deadband is overcome and the load pressure starts to build up. The velocity (second row) reveals that the piston begins to move with a slight delay with respect to the load pressure. The piston starts to move, as soon as the stick friction forces are overcome. As soon as the piston starts to move it pushes oil out of the opposite chamber (here chamber B) while increasing the volume of the pressurized chamber (here chamber A). The valve has not yet opened enough to support increasing flow of oil, hence the pressure drops (within chamber A) creating a suction effect which immediately increases the load pressure again. This increased pressure pushes more oil out of chamber B and so on. The load pressure oscillation has now started, indicated by the comment (2). This oscillation begins to be visible in the velocity signal as indicated by the green line and comment (3). The oscillation also penetrates the control signal (comment (4)) because the piston's velocity is a direct

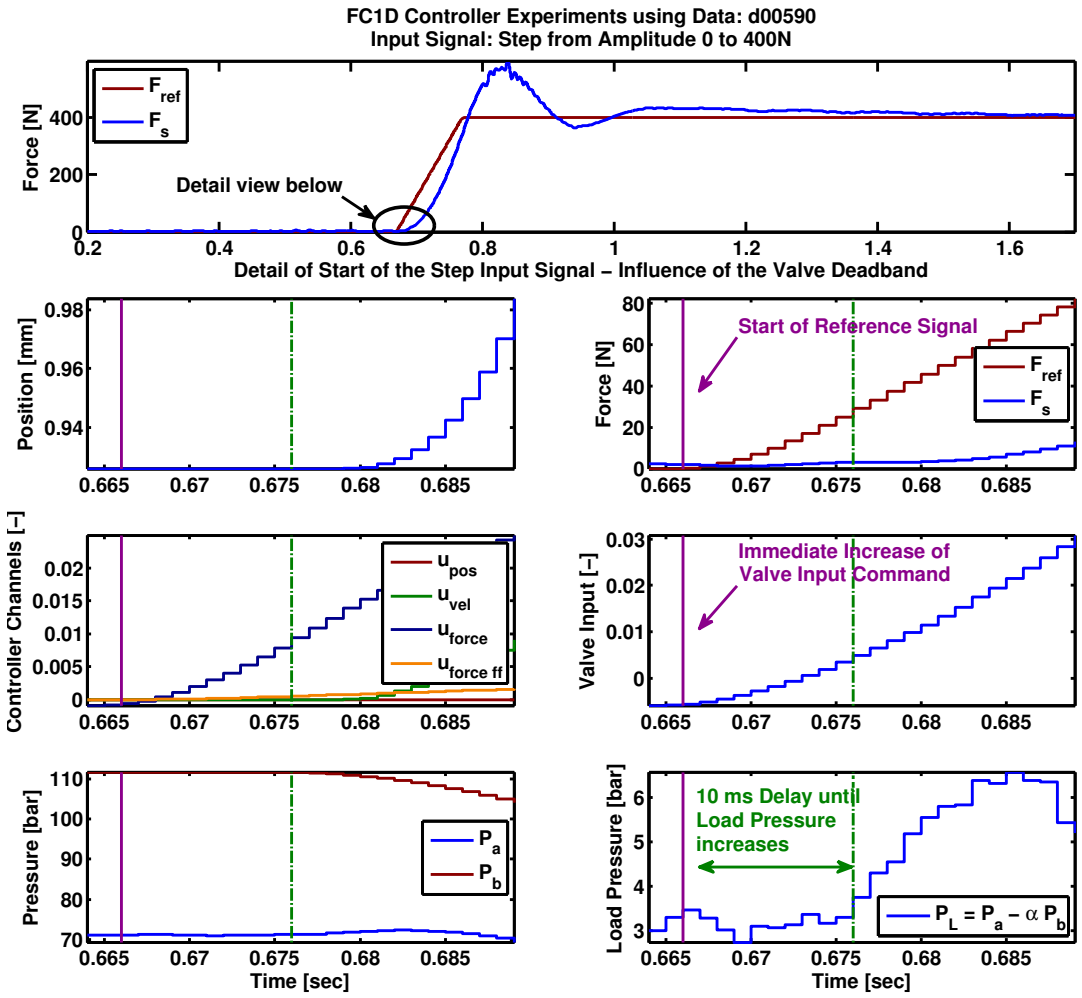

Figure 9.22: Influence of the valve deadband on control performance demonstrated using a force step command. The valve inputs begin to increase due to a increasing reference signal. The load pressure follows the commands with a 10 ms time-delay due to the valve deadband.

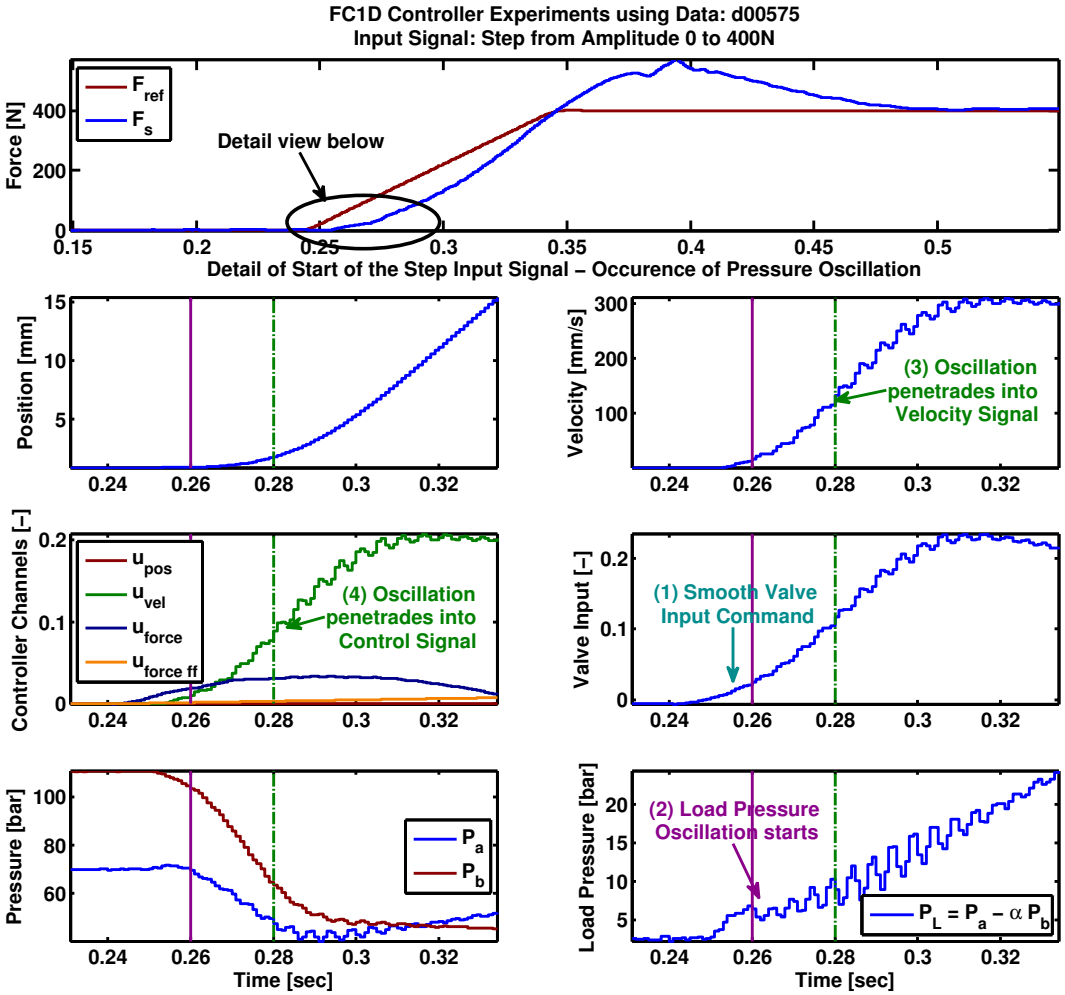

Figure 9.23: High frequency load pressure oscillations occurring due to pressure waves within the cylinder chambers. The valve input is smooth at first (comment (1)) until a sudden oscillation of the load pressure starts (comment (2)) penetrating into the velocity signal (comment (3)) and ultimately influencing the control signal (comment (4)). After about 55 ms the controller managed to dampen the oscillation.

feedback signal. The controller begins to counteract the oscillations and after about 55 ms the oscillations have disappeared.
It is not entirely clear if the just mentioned sequence really is the cause of these high frequency oscillations. The issue seems to be related to the begin of piston movement because the effect does not occur when the piston is in movement, for example during a sine or chirp reference command. The inertia of the piston and the magnitude of valve input command seem to somehow dampen these high frequency oscillations.

Chapter 10

Conclusion

In this thesis, an adaptive control framework was developed to improve the controller's torque tracking performance under various circumstances. The motivation of this thesis (see chapter 1) stems from the observation that the fixed gain low-level controller of the hydraulically actuated quadruped robot HyQ is not always able to deliver satisfactory torque tracking performance under changing load condition, e.g. when the robot is walking. The robot HyQ at the ETH Zurich is a four legged robot capable of versatile, robust and dynamic maneuvers. The robot is fully position and torque controlled, the latter enabling the use of many control laws that produce torque commands as outputs. Hence, *perfect*[1] torque tracking is desired to allow best performance of these control laws. Therefore, in this thesis an adaptive control framework (see chapter 5) was developed to cope with changing system dynamics by recursively estimating the current system dynamics (chapter 6) and updating the low-level controller when needed (chapter 7). The framework was implemented in C++ on a real-time one-dimensional force control (FC1D) test set-up (see chapter 8) with which experiments of both the estimator and controller were performed (see chapter 9).

10.1 Summary

First, the hydraulic system dynamics were studied and a nonlinear system was obtained (see chapter 2), where the influence of the load cell sensor, that is, the force measurement device, was examined more closely. A full linear and two simplified linear models were derived (see chapter 3) and extensively studied (see chapter 4). It was found that the simplified model based on the load cell force measurement accurately reproduces the force when compared to the nonlinear model, hence little model accuracy is lost due to the simplifications. Therefore, this simplified Fs model (given by the equations (3.51)) was used throughout this thesis; it has the additional advantage of relaying on a state vector which is fully measurable. Thus, full state feedback controllers are eligible. A "Degree of Nonlinearity" study (see subsection 4.3.1) for the nonlinear system as well as a parameter sensitivity study (see subsection 4.3.2) of the most important model parameters were carried out. It was found that the hydraulic system is heavily nonlinear due to the highly nonlinear pressure dynamics (see equations (3.21)), as can be seen in figure 4.7. Hence, this nonlinear behavior of the system might pose difficulties for an estimator trying to give accurate estimates of the underlying system.

[1] An engineer's interpretation of the word *perfect* is anticipated.

Secondly, the adaptive control framework was outlined and detailed in chapter 5. The framework is based on the concept of first estimating the system and then using this improved model knowledge to update the controller gains accordingly. Hence, it is modular; the choice of estimator and controller is virtually free. For the estimator, a very simple linear recursive least squares approach (see chapter 6) and a linear quadratic regulator (LQR) (see chapter 7) were chosen since the state vector is fully measurable and, more importantly, the LQR guarantees 60° of phase margin, hence good robustness against estimation errors. Simulation results assess the performance of both estimator and controller. It was shown that the estimator is able to identify system changes accurately and quickly for ideal[2] signals (see figure 6.2), that is, signals with high frequency content and low amplitude. However, the performance deteriorates as soon as non-ideal[3] signals are used, such as sinusoidal signals having low frequency content and/or high amplitude[4]. These non-ideal signals do not excite enough frequencies of the system and/or they excite the nonlinearities of the system, hence both cases pose a difficult task for an estimator.
The full adaptive control framework, where the estimator and controller work together, was simulated and it was shown (see figure 7.8) that the new adaptive controller is able to cope with abrupt and large system changes that can occur when the robot HyQ performs, for example, a walking trot.
The framework was implemented in C++ on a real-time FC1D test set-up in order to perform experiments. Details about the implementation are found in chapter 8 and the FC1D test set-up is described in appendix D.
Experiments (chapter 9) were performed to asses the performance of both the estimator and controller on the real-time hardware. It was found (see section 9.1) that the chosen estimator is able, for some ideal signals, to correctly identify (in real-time) the system dynamics. The performance of the controller was extensively verified (see section 9.2) and the controller performed at least as well as the *state-of-the-art* feedback linearization controller (with velocity compensation) currently used on HyQ. The potential increase in tracking performance that can be gained when using the right controller for the system was also shown. Preliminary experiments for the full adaptive control framework were also done, though it was found that the computational demand of the implementation was too high in order to let both estimator and controller (with activated adaption of the gains) be run simultaneously.

10.2 Discussion

The hydraulic system is highly nonlinear, as shown in subsection 4.3.1, hence the linear estimator, as discussed in chapter 6, has difficulties estimating the true system dynamics in case these nonlinearities are excited. The estimated transfer functions do not contain the resonance peak and the estimates have a higher variance, as can be seen in figure 6.5. Therefore, for an accurate system identification, another type of estimator might be needed.

[2]The term "ideal signals" classifies signals having high frequency content and low amplitude. Hence, exciting the system well, but not its nonlinearities.

[3]These are signals having low frequency content and high amplitude. Hence, not exciting the system sufficiently, but its nonlinearities.

[4]The qualitative meaning of amplitude is anticipated here since it does not matter if high amplitude signals are directly fed to the valve input or first as a force reference to the controller because both cases excite the nonlinearities of the system.

However, for control purposes it might be that the linear estimator, as discussed in section 6.1, still proves to be adequate because only the range of frequencies the controller is operated in needs to be identified correctly in order to achieve satisfactory force tracking performance. This observation can be made when considering the simulation results of the full adaptive control framework, as discussed in section 7.5. There, it is noted that the tracking performance within the third segment of figure 7.8 is very good, although the true system was not correctly identified, because a low frequency force reference signal was used. This discrepancy between true and identified system can readily be seen when considering figure C.4 given in appendix C.3. The transfer functions towards the end of the third segment (yellow lines) are essentially straight lines without any oscillation components. The blue transfer function represents the true system dynamics within this segment and clearly shows a resonance peak at high frequencies around $\approx 200\,\mathrm{Hz}$.
Important to realize is now that the controller is usually operated within a relatively small frequency range $\approx 0 - 30\,\mathrm{Hz}$[5]. Considering again figure C.4, it can readily be seen that the yellow transfer functions, within this restricted frequency range, provide indeed a good approximation of the true transfer function (blue). Hence, the controller gains are *correct* in the sense that they are calculated based on a system description that captures the relevant dynamics within the frequency range. Therefore, as can be seen in figure 7.8, the controller is able to deliver satisfactory force tracking within a reasonable frequency range. However, care must be taken since for systems with high mass, the resonance peak lies within or near the controlled frequency range, hence accurate identification of the resonance peak is of great importance, as can be seen in segment two of figure C.4.
During the LQR controller experiments, it was found that the filtering of the measurement signals has quite a high impact on the robustness and tracking capabilities of the LQR controller. The position of the piston is measured by a potentiometer, resulting in a relatively noisy measurement, hence the calculated velocity is even more noisy. This noisy signal is then directly used as a state feedback for the LQR controller. The corresponding velocity gain is quite high, as can be seen from the controller gain map in figure 7.4, hence the noise is amplified, resulting in a noisy controller output. We used a rather aggressive filter to get rid of the noise, hence introducing phase lag into the signals. Because the state measurement signals were then filtered differently, the different phase delays[6] reduced the robustness of the LQR drastically.
By using simulation as well as experiments, this thesis showed that the estimator is able to identify the system dynamics for some ideal signals and that the LQR controller is able to track a force reference almost perfectly and does so without any feedback linearization or velocity compensation concepts the *state-of-the-art* feedback linearization (FL) controller uses. Simulations also showed that the proposed adaptive control framework works very well and is indeed able to cope with large system changes.

10.3 Future Work

The most important next step is to show the performance of the full adaptive control framework using the FC1D test set-up. Unfortunately, the new FC1D set-up, whose design

[5] According to Victor Barasuol, the frequency of the balance reflex motion is at around 12 Hz and the normal trotting frequency is at around 2 Hz. Hence, assuming a worst-case frequency of 30 Hz is more than enough. However, no clear controller specifications were defined so far.

[6] The phase delay of the measurement essentially "eats" up the phase margin of the LQR controller, hence becoming less robust.

was part of the thesis, needs to be brought to fully operating condition first. Successful experiments proofing the framework's capabilities then might lead to its implementation on a leg test set-up (HyL) and ultimately to its implementation on the full HyQ robot. The ultimate goal is to sustainably improve the versatility and capability of the HyQ robot.
The discussion above clearly mentioned the problem of filtering and the associated phase delay of the signals. The role of the filtering should be analysed in more detail. Possible improvements may consist of adding an (analog) anti-aliasing filter before sampling the signals and to use oversampling to be able to filter the signals more aggressively.
Further experiments assessing the estimation performance may be needed, especially with regard to non-ideal signals. Other estimation algorithms might be worth considering, such as nonlinear estimators. These nonlinear estimators, as outlined in section 9.1.5, would be able to handle the inherent nonlinearities of the system and yield better estimates.
The adaptive controller behavior shall be stable and robust against switching dynamics, that is, when the robot switches from a stance phase to a flight phase. Further investigations are necessary to guarantee that the controller, applied to the real system, remains stable in such a switching event and that the estimator is able to converge reasonably fast within the stance phase to fully bolster the capabilities of the adaptive control framework.
One advantage of the proposed framework is that it is virtually model-free, hence, one should be able to use the exact same code on a different robot without redesign. An electronically actuated robot could, for example, be used to show the framework's versatility since only the weighting matrices and specifications of the measurement and input signals need to be newly defined.

Appendix A

Appendix: Nonlinear and Linear Model

A.1 General Equations for the Full Linear Model

This section extends the content given in subsection 3.1.1 of chapter 3 and calculates the Taylor series expansion of the set of nonlinear equations about an arbitrary operating point $x_{i\oslash}, u_{\oslash}, y_{i\oslash}$ (with i denoting the ith variable) and neglecting all terms of higher order than one. The nonlinear system (2.30) is thus approximated by the general linear system given in the state space form as:

$$\begin{aligned} \frac{d}{dt}\delta\vec{x}(t) &= \mathcal{A}\cdot\delta\vec{x}(t) + \mathcal{B}\cdot\delta u(t) \\ \delta\vec{y}(t) &= \mathcal{C}\cdot\delta\vec{x}(t) + \mathcal{D}\cdot\delta u(t) \end{aligned} \tag{A.1}$$

where

$$\mathcal{A} = \frac{\partial f}{\partial \vec{x}}\Big|_{\vec{x}=\vec{x}_{\oslash},u=u_{\oslash}} \qquad \mathcal{B} = \frac{\partial f}{\partial u}\Big|_{\vec{x}=\vec{x}_{\oslash},u=u_{\oslash}} \tag{A.2}$$

$$\mathcal{C} = \frac{\partial g}{\partial \vec{x}}\Big|_{\vec{x}=\vec{x}_{\oslash},u=u_{\oslash}} \qquad \mathcal{D} = \frac{\partial g}{\partial u}\Big|_{\vec{x}=\vec{x}_{\oslash},u=u_{\oslash}} \tag{A.3}$$

For the case $u_{v\oslash} > 0$, the nonlinear equations are given by (2.30a) and thus the matrices are given by:

$$\mathcal{A}^{+} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} & \frac{-\alpha A_p}{M} \\ \frac{\partial f_3^{+}}{\partial x_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} & \frac{\partial f_3^{+}}{\partial \dot{x}_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} & \frac{\partial f_3^{+}}{\partial p_a}\Big|_{\vec{x}_{\oslash},u_{\oslash}} & 0 \\ \frac{\partial f_4^{+}}{\partial x_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} & \frac{\partial f_4^{+}}{\partial \dot{x}_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} & 0 & \frac{\partial f_4^{+}}{\partial p_b}\Big|_{\vec{x}_{\oslash},u_{\oslash}} \end{bmatrix} \tag{A.4}$$

with

$$\frac{\partial f_3^{+}}{\partial x_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} = \frac{-\beta_e A p}{(V_{pl} + x_{p\oslash}A_p)^2}\left(K_v|u_{v\oslash}|\sqrt{p_s - p_{a\oslash}} - A_p\dot{x}_{p\oslash}\right) \tag{A.5}$$

$$\frac{\partial f_3^{+}}{\partial \dot{x}_p}\Big|_{\vec{x}_{\oslash},u_{\oslash}} = \frac{-\beta_e A p}{V_{pl} + x_{p\oslash}A_p} \tag{A.6}$$

$$\frac{\partial f_3^+}{\partial p_a}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e}{V_{pl}+x_{p\oslash}A_p}\left(\frac{-K_v|u_{v\oslash}|}{2\sqrt{p_s-p_{a\oslash}}}\right) \tag{A.7}$$

$$\frac{\partial f_4^+}{\partial x_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e\alpha Ap}{(V_{pl}+(L_c-x_{p\oslash})\alpha A_p)^2}\left(-K_v|u_{v\oslash}|\sqrt{p_{b\oslash}-p_t}+\alpha A_p\dot{x}_{p\oslash}\right) \tag{A.8}$$

$$\frac{\partial f_4^+}{\partial \dot{x}_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e\alpha Ap}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p} \tag{A.9}$$

$$\frac{\partial f_4^+}{\partial p_b}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p}\left(\frac{-K_v|u_{v\oslash}|}{2\sqrt{p_{b\oslash}-p_t}}\right) \tag{A.10}$$

$$\mathcal{B}^+ = \begin{bmatrix} 0 \\ 0 \\ \frac{\beta_e K_v\sqrt{p_s-p_{a\oslash}}}{V_{pl}+x_{p\oslash}A_p} \\ \frac{-\beta_e K_v\sqrt{p_{b\oslash}-p_t}}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p} \end{bmatrix} \tag{A.11}$$

For the case $u_{v\oslash} < 0$, the nonlinear equations are given by (2.30b) and thus the matrices are given by:

$$\mathcal{A}^- = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} & \frac{-\alpha A_p}{M} \\ \frac{\partial f_3^-}{\partial x_p}\Big|_{\vec{x}_\oslash,u_\oslash} & \frac{\partial f_3^-}{\partial \dot{x}_p}\Big|_{\vec{x}_\oslash,u_\oslash} & \frac{\partial f_3^-}{\partial p_a}\Big|_{\vec{x}_\oslash,u_\oslash} & 0 \\ \frac{\partial f_4^-}{\partial x_p}\Big|_{\vec{x}_\oslash,u_\oslash} & \frac{\partial f_4^-}{\partial \dot{x}_p}\Big|_{\vec{x}_\oslash,u_\oslash} & 0 & \frac{\partial f_4^-}{\partial p_b}\Big|_{\vec{x}_\oslash,u_\oslash} \end{bmatrix} \tag{A.12}$$

with

$$\frac{\partial f_3^-}{\partial x_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{-\beta_e Ap}{(V_{pl}+x_{p\oslash}A_p)^2}\left(-K_v|u_{v\oslash}|\sqrt{p_{a\oslash}-p_t}-A_p\dot{x}_{p\oslash}\right) \tag{A.13}$$

$$\frac{\partial f_3^-}{\partial \dot{x}_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{-\beta_e Ap}{V_{pl}+x_{p\oslash}A_p} \tag{A.14}$$

$$\frac{\partial f_3^-}{\partial p_a}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e}{V_{pl}+x_{p\oslash}A_p}\left(\frac{-K_v|u_{v\oslash}|}{2\sqrt{p_{a\oslash}-p_t}}\right) \tag{A.15}$$

$$\frac{\partial f_4^-}{\partial x_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e\alpha Ap}{(V_{pl}+(L_c-x_{p\oslash})\alpha A_p)^2}\left(K_v|u_{v\oslash}|\sqrt{p_s-p_{b\oslash}}+\alpha A_p\dot{x}_{p\oslash}\right) \tag{A.16}$$

$$\frac{\partial f_4^-}{\partial \dot{x}_p}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e\alpha Ap}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p} \tag{A.17}$$

$$\frac{\partial f_4^-}{\partial p_b}\Big|_{\vec{x}_\oslash,u_\oslash} = \frac{\beta_e}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p}\left(\frac{-K_v|u_{v\oslash}|}{2\sqrt{p_s-p_{b\oslash}}}\right) \tag{A.18}$$

$$\mathcal{B}^- = \begin{bmatrix} 0 \\ 0 \\ \frac{\beta_e K_v\sqrt{p_{a\oslash}-p_t}}{V_{pl}+x_{p\oslash}A_p} \\ \frac{-\beta_e K_v\sqrt{p_s-p_{b\oslash}}}{V_{pl}+(L_c-x_{p\oslash})\alpha A_p} \end{bmatrix} \tag{A.19}$$

The output matrices $\mathcal{C}$ and $\mathcal{D}$ remain the same as given in subsection 3.1.1 and stay the same regardless of the input sign. The matrices are again written here for completeness.

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -d_p & A_p & -\alpha A_p \end{bmatrix} \tag{A.20}$$

$$\mathcal{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.21}$$

where the matrices $\mathcal{B}, \mathcal{C}$ should not be confused with the total friction coefficient B and the total spring stiffness C of the system.

A.2 General Equations for the Simplified Linear Model

Using the load pressure dynamics equation (3.22) derived in subsection 3.2.1, we can write a general form of the simplified linear model in state space form for an arbitrary operating point $\oslash = (x_{p\oslash}, \dot{x}_{p\oslash}, p_{l\oslash}, u_{v\oslash})$. The state vector for the simplified system is defined as:

$$\tilde{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_p = \text{position of piston} \\ \dot{x}_p = \text{velocity of piston} \\ p_l = \text{load pressure} \end{pmatrix} \tag{A.22}$$

The state space form is defined by the equations (3.6) and (3.7). The matrices are given below and are written in terms of the coefficients defined within section 3.2.1.
The matrices for the simplified linear model for the case $u_{v\oslash} > 0$ are given by:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} \\ -K_{xpA} \cdot \sqrt{-p_{l\oslash} + p_s - \alpha p_t} + K_{xpB} \cdot \dot{x}_{p\oslash} & K_{\dot{x}_p} & K_{pl} / \sqrt{-p_{l\oslash} + p_s - \alpha p_t} \end{bmatrix} \tag{A.23}$$

$$\mathcal{B} = \begin{bmatrix} 0 \\ 0 \\ K_{uv} \cdot \sqrt{-p_{l\oslash} + p_s - \alpha p_t} \end{bmatrix} \tag{A.24}$$

And for the case $u_{v\oslash} < 0$:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-C}{M} & \frac{-B}{M} & \frac{A_p}{M} \\ K_{xpA} \cdot \sqrt{p_{l\oslash} + \alpha p_s - p_t} + K_{xpB} \cdot \dot{x}_{p\oslash} & K_{\dot{x}_p} & K_{pl} / \sqrt{p_{l\oslash} + \alpha p_s - p_t} \end{bmatrix} \tag{A.25}$$

$$\mathcal{B} = \begin{bmatrix} 0 \\ 0 \\ K_{uv} \cdot \sqrt{p_{l\oslash} + \alpha p_s - p_t} \end{bmatrix} \tag{A.26}$$

And the output matrices are the same for both cases:

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -d_p & A_p \end{bmatrix} \tag{A.27}$$

$$\mathcal{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.28}$$

A.3 Parameters of Nonlinear Cylinder and Load Mass Model

Parameter	Value	Description
Physical Parameters of Mass-Spring-Damper System		
m_l	$10\,\mathrm{kg}$	Mass of load
d_l	$700\,\mathrm{kg/s}$	Friction coefficient of linear bearings of the cart
k_1	$10740\,\mathrm{kg/s^2}$	Stiffness of spring 1 (open side)
k_2	$10740\,\mathrm{kg/s^2}$	Stiffness of spring 2 (cylinder side)
Cylinder specifications		
D_p	$0.016\,\mathrm{m}$	Piston diameter
D_r	$0.010\,\mathrm{m}$	Piston rod diameter
A_p	$2.0106 \cdot 10^{-4}\,\mathrm{m^2}$	Piston area of side without rod: $A_p = \pi \cdot D_p^2/4$
A_r	$1.2252 \cdot 10^{-4}\,\mathrm{m^2}$	Piston area of side with rod: $Ar = \pi \cdot D_p^2/4 - \pi \cdot D_r^2/4$
α	$0.6094\,-$	Factor for piston area with vs. without rod: $\alpha = A_r/A_p$
L_c	$0.08\,\mathrm{m}$	Piston stroke or length
m_p	$0.3141\,\mathrm{kg}$	The mass of the piston consists of the piston cylinder and rod: $m_p = A_p(\alpha Lc + 0.15) \cdot \rho_p$
ρ_p	$7860\,\mathrm{kg/m^3}$	Density of steel (material of piston and mass)
d_p	$700\,\mathrm{kg/s}$	Friction coefficient of piston assembly inside the cylinder
Hydraulic System		
β_e	$1.3 \cdot 10^9\,\mathrm{Pa}$	Effective bulk modulus of the oil in the hydraulic system
D_{pl}	$0.0038\,\mathrm{m}$	Inner oil pipe line diameter
L_{pl}	$0.3\,\mathrm{m}$	Pipe line length
V_{pl}	$3.4023 \cdot 10^{-6}\,\mathrm{m^3}$	Pipe line volume: $V_pl = L_{pl} \cdot \pi \cdot D_{pl}^2/4$
ρ_{oil}	$870\,\mathrm{kg/m^3}$	Density of oil
Valve Characteristics		
u_{vn}	$0.01\,\mathrm{A}$	Nominal input ampere
u_{max}	$0.01\,\mathrm{A}$	Maximum and minimum input ampere
q_n	$7.5/60000\,\mathrm{m^3/s}$	Nominal flow of oil
Δp_n	$70 \cdot 10^5\,\mathrm{Pa}$	Nominal valve pressure drop (across two ports)
K_v	$6.6815 \cdot 10^{-6}\,\frac{\mathrm{m^3/s}}{(\mathrm{A}\sqrt{\mathrm{N/m^2}})}$	Valve size gain: $K_v = \sqrt{2} \cdot q_n/(u_{vn} \cdot \sqrt{\Delta p_n})$
F_v	$250\,\mathrm{Hz}$	Cut-off frequency of the valve (MOOG data sheet page 14)
W_v	$1570\,\mathrm{rad/s}$	Valve spool angular frequency
D_v	0.5	Valve spool damping
Environment conditions		
p_s	$250 \cdot 10^5\,\mathrm{Pa}$	Supply pressure from hydraulic pump
p_t	$0\,\mathrm{Pa}$	Pressure of the reservoir
Normalization Values		
x_{norm}	$L_c\,\mathrm{m}$	Normalized piston position
$\dot{x}_{norm}$	$1.2\,\mathrm{m/s}$	Normalized piston velocity
$p_{a_{norm}}$	$p_s\,\mathrm{Pa}$	Normalized chamber pressure A
$p_{b_{norm}}$	$p_s\,\mathrm{Pa}$	Normalized chamber pressure B
F_{norm}	$5000\,\mathrm{N}$	Normalized force

Table A.1: Hydraulic cylinder and load mass parameters with value and description of the nonlinear (chapter 2) and linear models (chapter 3) used throughout this thesis.

A.4 Identified Parameters of the FC1D Test Set-Up

Parameter	Value	Description
Physical Parameters of Mass-Spring-Damper System		
m_l	$2.5\,\mathrm{kg}$	Mass of load
d_l	$500\,\mathrm{kg/s}$	Friction coefficient of linear bearings of the cart
Hydraulic System		
β_e	$5.5 \cdot 10^8\,\mathrm{Pa}$	Effective bulk modulus of the oil in the hydraulic system
L_{pl}	$0.6\,\mathrm{m}$	Pipe line length
V_{pl}	$1.0207 \cdot 10^{-5}\,\mathrm{m}^3$	Pipe line Volume: $V_pl = L_{pl} \cdot \pi \cdot D_{pl}^2/4$
Valve Characteristics		
K_v	$1.6704 \cdot 10^{-6}\ \frac{\mathrm{m^3/s}}{(\mathrm{A}\sqrt{\mathrm{N/m^2}})}$	Valve size gain: $K_v = (1/4) \cdot \sqrt{2} \cdot q_n/(u_{vn} \cdot \sqrt{\Delta p_n})$
Environment conditions		
p_s	$125 \cdot 10^5\,\mathrm{Pa}$	Supply pressure from hydraulic pump
p_t	$0\,\mathrm{Pa}$	Pressure of the reservoir

Table A.2: Parameters of the nominal model given by Table A.1 that needed to be adjusted to explain the measurements using the real-time FC1D test set-up at IIT. Note that these numbers have absolutely no meaning, they are just a result from trial and error to make the model fit the measurements. A proper grey-box identification should be done to get values with a proper physical meaning.

The following figure A.1 shows some real measurements where the load cell force transfer function was extracted from (using non-causal and nonlinear estimation techniques) and compared to the model adjusted by the parameters given in table A.2.

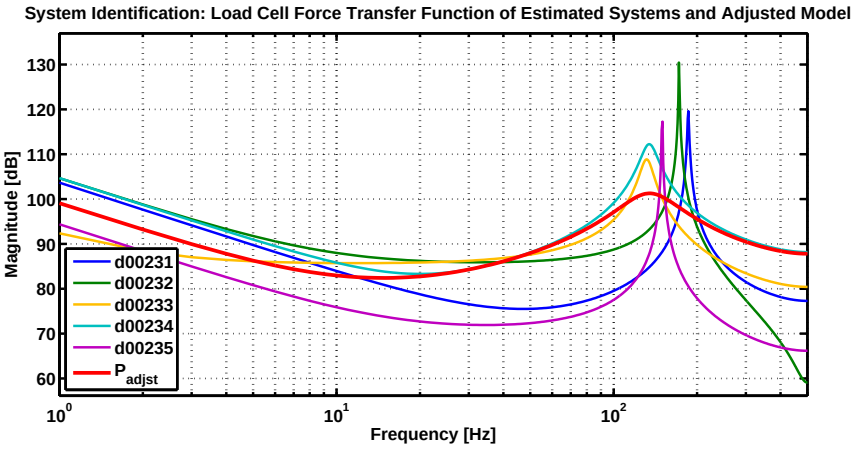

Figure A.1: Bode plot of measured transfer function compared to model with adjusted parameters according to table A.2.

A.5 Model Analysis - Additional Figures

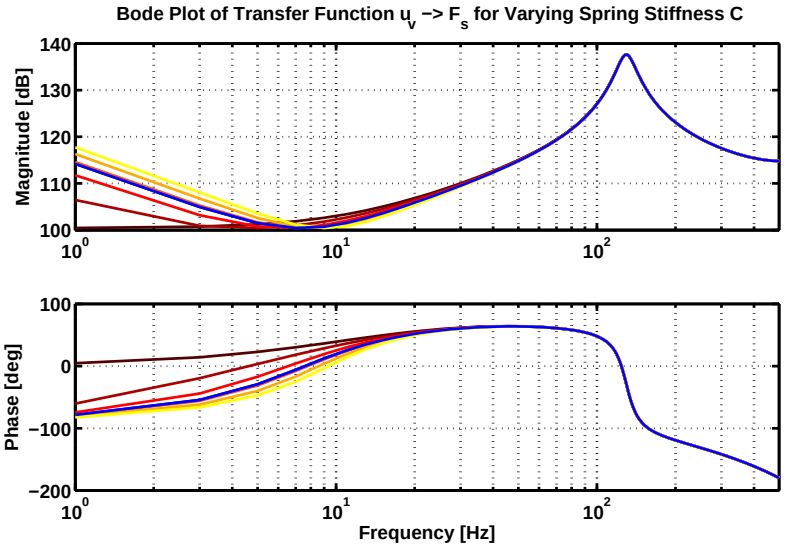

Figure A.2: This figure shows the effect on the shape of the force transfer function due to varying the spring stiffness C within the set: $C = k_1 + k_2 \in [10\,\mathrm{kg/s^2}, 35000\,\mathrm{kg/s^2}]$. The color changes from black to yellow with increasing stiffness. The changes in spring stiffness mainly affect the low frequency content of the system and leave the resonance peak unchanged.

Appendix B

Appendix: System Identification

B.1 Linear System Notation

B.1.1 Linear System Representation in Continuous Time

A general linear state space system in continuous time, as introduced in chapter 3, can be represented in the state space notation as follows:

$$\begin{aligned} \dot{x}(t) &= \mathcal{A}(t) \cdot x(t) + \mathcal{B}(t) \cdot u(t) \\ y(t) &= \mathcal{C}(t) \cdot x(t) + \mathcal{D}(t) \cdot u(t) \end{aligned} \tag{B.1}$$

where $x(t)$ is the state, $u(t)$ the system's input and $y(t)$ its output. The four matrices, potentially time-varying, describe the influence of the state and input on the dynamics of the states and outputs. This state space representation has the advantage that it provides full insight into the dynamics of the states $x(t)$. However, for system identification, this representation can be a drawback since those states will need to be measured or estimated. Thus, an alternative notation is the use of transfer functions, calculated from the state space representation as follows:

$$\frac{Y(s)}{U(s)} = \mathcal{C}(t) \cdot (s \cdot I - \mathcal{A}(t))^{-1} \cdot \mathcal{B}(t) + \mathcal{D}(t) \tag{B.2}$$

where s represents the complex Laplace variable. This function describes the relationship between input $u(t)$ and output $y(t)$, thus no insight into the dynamics of the states is possible. Moreover, hidden dynamics are not visible since the common poles and zeros cancel out. The transfer function can also be represented as a fraction of two functions, that is:

$$\frac{Y(s)}{U(s)} = \frac{b_m s^m + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} = \frac{b(s)}{a(s)} = b_m \frac{\prod_{j=1}^{m} (s - \zeta_j)}{\prod_{i=1}^{n} (s - \pi_i)} \tag{B.3}$$

where ζ_j are the *zeros* and π_i are the *poles* of the transfer function. This input-output relationship can also be written in the time domain as:

$$y^{(n)} + a_{n-1} \cdot y^{(n-1)} + \cdots + a_1 \cdot y^{(1)} + a_0 \cdot y = b_m \cdot u^{(m)} + \cdots + b_1 \cdot u^{(1)} + b_0 \cdot u \tag{B.4}$$

B.1.2 Linear System Representation in Discrete Time

In this work, we use the usual zero-order hold operation to convert the continuous-time system matrices to discrete time (see for example [22] or [18]), resulting in the following four matrices:

$$\begin{aligned} \mathcal{A}_d &= e^{\mathcal{A}\cdot Ts} & \mathcal{B}_d &= \int_0^{Ts} e^{\mathcal{A}\cdot T_s} d\tau \cdot \mathcal{B} \\ \mathcal{C}_d &= \mathcal{C} & \mathcal{D}_d &= \mathcal{D} \end{aligned} \tag{B.5}$$

where T_s is the sampling time and the matrices $(\mathcal{A}_d, \mathcal{B}_d, \mathcal{C}_d, \mathcal{D}_d)$ are the new discrete state matrices. Thus a discrete-time state space system can be written as:

$$\begin{aligned} x[n+1] &= \mathcal{A}_d[n] \cdot x[n] + \mathcal{B}_d[n] \cdot u[n] \\ y[n] &= \mathcal{C}_d[n] \cdot x[n] + \mathcal{D}_d[n] \cdot u[n] \end{aligned} \tag{B.6}$$

where n is the current time step. Again, a discrete-time transfer function can be defined by:

$$\frac{Y(z)}{U(z)} = \mathcal{C}_d[k] \cdot (z \cdot I - \mathcal{A}_d[k]))^{-1} \cdot \mathcal{D}_d[k] + \mathcal{D}_d[k] \tag{B.7}$$

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{B.8}$$

where z is the shift operator, the discrete time equivalent of the Laplace variable s. The coefficient a_0 is equal to 1 for all causal systems, thus only $(N + M + 1)$ coefficients are needed to fully describe the system. The highest output delay is defined by N and the highest input delay by M.

The input-output relationship can be written in discrete time using the discrete shift operator $q^{-1} \rightarrow x[n] \cdot q^{-1} = x[n-1]$:

$$a_0 \cdot y[n] + a_2 \cdot y[n-1] + \cdots + a_N \cdot y[n-N] = b_0 \cdot u[n] + b_1 \cdot u[n-1] \cdots + b_M \cdot u[n-M] \tag{B.9}$$

Thus this can equally be written as:

$$H(q) = \frac{Y(q)}{U(q)} = \frac{b_0 + b_1 q^{-1} + \cdots + b_M q^{-M}}{a_0 + a_1 q^{-1} + \cdots + a_N q^{-N}} \tag{B.10}$$

where q is the discrete-time shift operator.

B.2 Matrix Identities from Linear Algebra

Some identities that are used to derive the recursive least squares algorithm of subsection 6.1.1.

$$\frac{\partial trace(ABA^T)}{\partial A} = 2AB \quad \text{if } B = B^T \tag{B.11}$$

$$\frac{\partial trace(AB)}{\partial A} = B^T \tag{B.12}$$

and

$$trace(C) = trace(C^T) \tag{B.13}$$

thus

$$\frac{\partial trace(B^T A^T)}{\partial A} = B^T \tag{B.14}$$

The matrix inversion lemma is [27, page 364]:

$$\Big[A + BCD]^{-1} = A^{-1} - A^{-1}B\Big[DA^{-1}B + C^{-1}\Big]^{-1}DA^{-1} \tag{B.15}$$

B.3 Additional Interpretations of the RLS Estimator

B.3.1 RLS as Kalman Filter Interpretation

We do not assume that the parameter vector remains the same as above, but instead assume that it varies like a random walk:

$$\Theta[k+1] = \Theta[k] + \nu[k] \qquad R_\nu = E\Big[\nu[k]^T \nu[k]\Big] \qquad E[\nu[k]] = 0 \tag{B.16}$$

with our knowledge R_ν about the noise properties and using the knowledge about the system propagation $R[k]$.

$$\begin{aligned}
K[k] &= P[k-1] \cdot H^T[k] \cdot \Big(H[k] \cdot P[k-1] \cdot H^T[k] + R_\nu\Big)^{-1} \\
\epsilon[k] &= y[k] - H[k]\Theta[k-1] \\
\Theta[k] &= \Theta[k-1] + K[k] \cdot \epsilon[k] \\
P[k] &= P[k-1] - P[k-1] \cdot H^T[k] \cdot \\
&\qquad \Big(R_\nu + H[k] \cdot P[k-1] \cdot H^T[k]\Big)^{-1} \cdot H[k] \cdot P[k-1] + R[k]
\end{aligned} \tag{B.17}$$

B.3.2 RLS as Instrumental Variable Method

$$\begin{aligned}
K[k] &= P[k-1] \cdot \xi[k] \cdot \Big(H[k] \cdot P[k-1] \cdot \xi[k] + \lambda[k] \cdot I\Big)^{-1} \\
\epsilon[k] &= y[k] - H[k]\Theta[k-1] \\
\Theta[k] &= \Theta[k-1] + K[k] \cdot \epsilon[k] \\
P[k] &= \frac{1}{\lambda[k]}\Bigg[P[k-1] - P[k-1] \cdot \xi[k] \cdot \\
&\qquad \Big(\lambda[k] \cdot I + H[k] \cdot P[k-1] \cdot \xi[k]\Big)^{-1} \cdot H[k] \cdot P[k-1]\Bigg]
\end{aligned} \tag{B.18}$$

where ξ is the instrument, which can have various forms.

Appendix C

Appendix: Additional Simulation Results

C.1 Estimator Performance - Additional Simulation Results

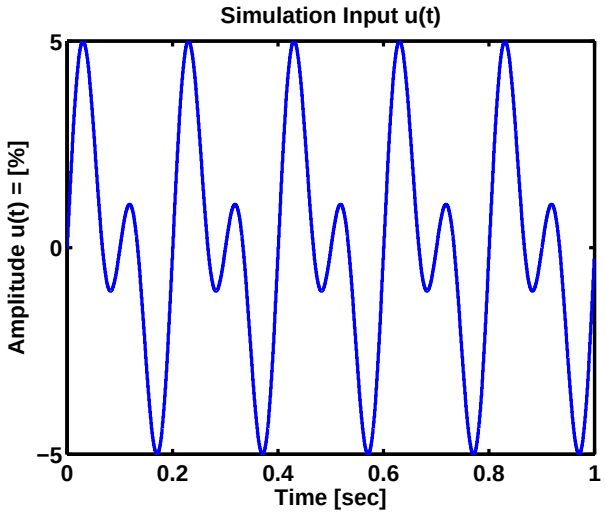

Figure C.1: Multisine input signal to the valve with amplitude of 5 % of the nominal valve opening and frequencies $f = [5, 10]Hz$. This signal is used for the Bode plot shown in figure C.2.

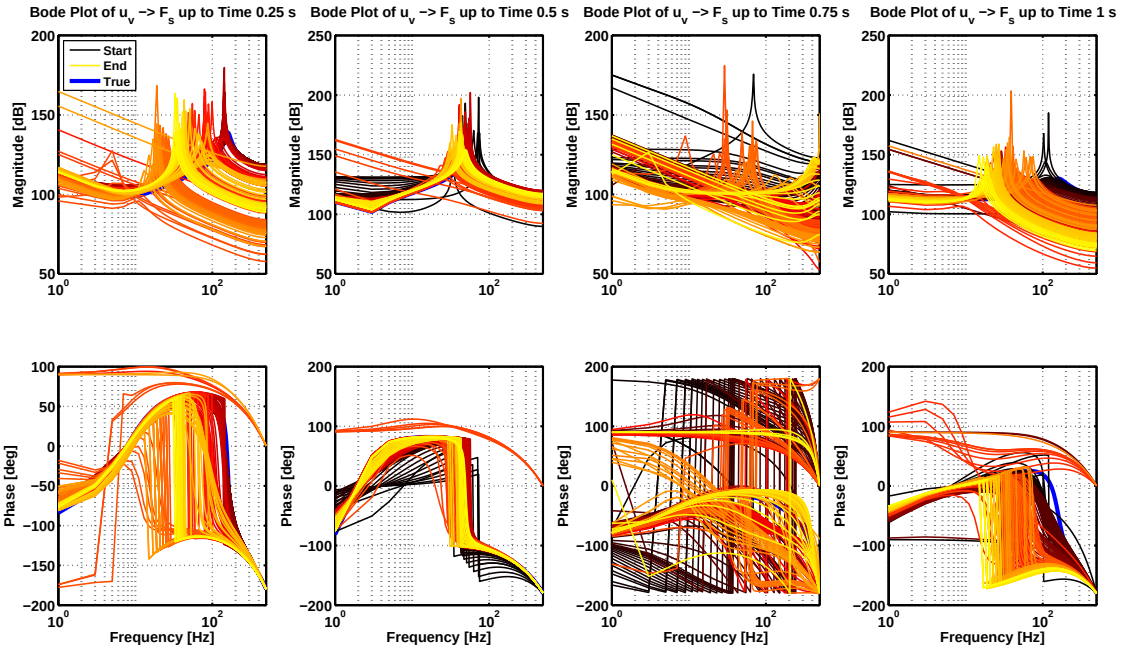

Figure C.2: Bode plot of the transfer function from valve input to measured load cell force output for each section of the figure 6.1. The true transfer function within the sections is drawn in blue, while the estimates are drawn in colors ranging from black to yellow with increasing time. The input signal (shown in figure C.1) is a multisine with an amplitude of 5 % of the nominal valve opening and frequencies $f = [5, 10]Hz$, thus not exciting enough frequencies of the system dynamics to enable a good estimation.

C.2 LQR Controller Gain Maps - Additional Figures

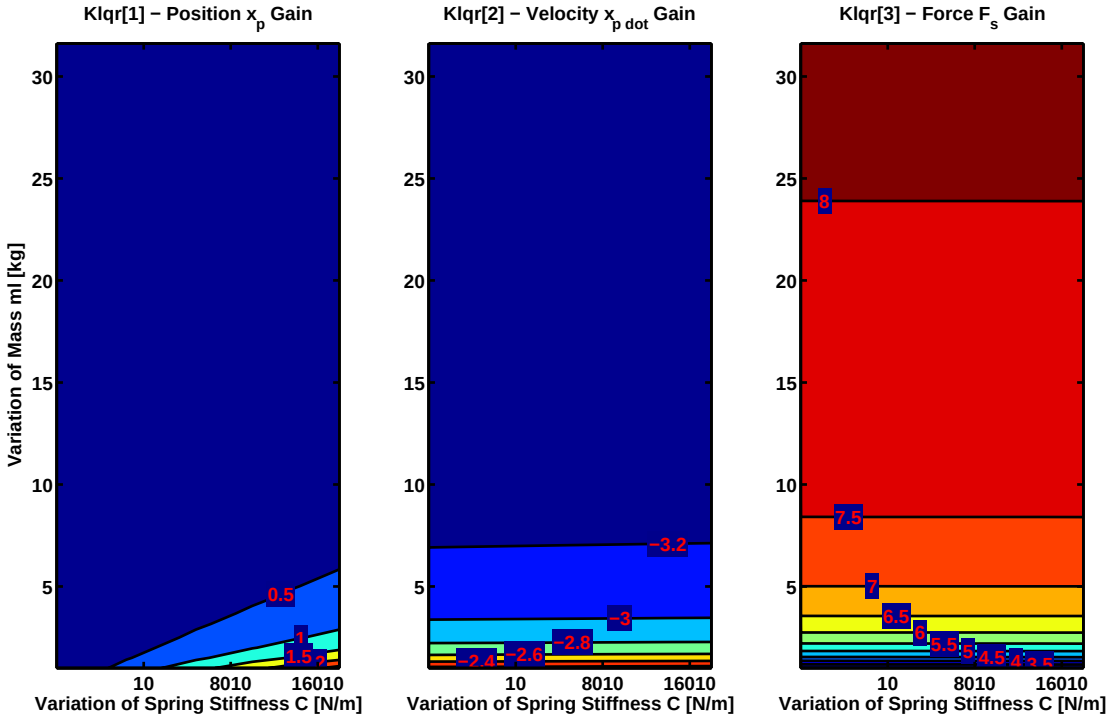

Figure C.3: LQR controller gain map for simultaneous variations of the system parameters, that is, the load mass m_l and spring stiffness C. The weighting matrices defined in subsection 7.2.3 are used. Only the gains for the states are shown. Integrator and feedforward gains are not shown. Increasing mass increases the gains for both the velocity and force channels (second and third column), but increases the gain for the position channel only slightly (first column). Increasing the spring stiffness does not affect the gains for the velocity and force; however, the position gain is increased to compensate for the higher spring forces.

C.3 Simulation of Adaptive Control Framework: Controller and Estimator - Additional Simulation Results

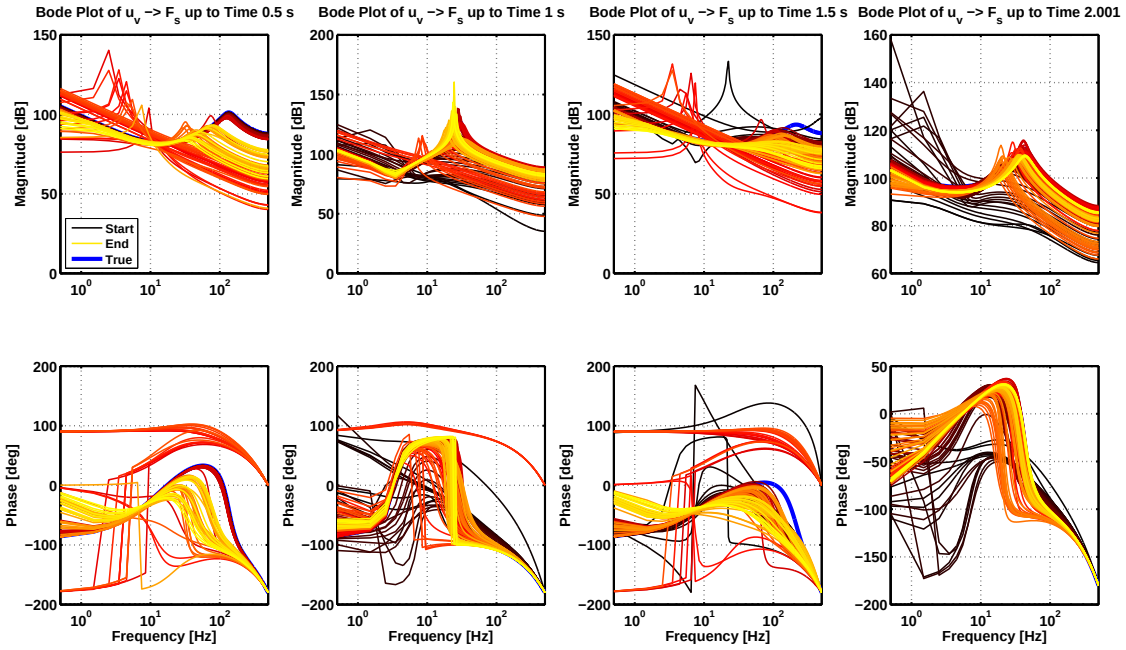

Figure C.4: Bode plot of the full adaptive control framework simulation shown in figure 7.8. The four columns correspond to the four sections. The true transfer function within the columns is drawn in blue, while the estimates are drawn in colors ranging from black to yellow with increasing time. The input signal is a multisine for the force reference input of the controller with an amplitude of 160 N of the nominal valve opening and frequencies $f = [2, 5, 10]Hz$. The system estimates in column one diverge since the multisine force signal does not excite enough high frequencies of the system. Note that the true system (blue) in column three is identified as a real system, hence the "straight-line" transfer functions (yellow). This *approximation*, as can be seen, is quite accurate for low to middle frequencies. The reason for this is again due to the low frequency content of the reference signal not able to excite the frequencies around the resonance peak of the system.

Appendix D

Appendix: New and Improved Design of the FC1D Test Set-Up

The new FC1D design is shown in figure D.1 and the almost finished assembled FC1D is shown in figure D.2. The physical description can be found in section 2.1. The new design is an improved version of the test set-up still in use at the IIT in Genoa. Many aspects were improved, for example, a new digital magnetic sensor for measuring the position is now used to get rid of the noisy measurements of the currently used sensor. Below the most important changes and their effect is listed:

1. The two end-position blocks were changed. With the new design, less material has to be cut. Two standard L-profiles are used on both sides and everything is mounted in between, thus making the test set-up more rigid. A series of pre-drilled holes can be used to fix the test set-up to a table or workbench.

2. The male/female shaft connecting the cylinder to load is replaced with a single shaft which is now arbitrarily fixable using two misumi axle holders. This feature is needed to guarantee that the piston can always be centered in the middle of its stroke for any pre-compression of the springs.

3. The "force-lines" of the cylinder rod and the pre-compression screw are now on the same height as the support axes such that no moments are generated. This is also critical for the force sensor since any moment results in imprecise measurements and may damage the sensor.

4. A new magnetic sensor for measuring the position of the piston has been added. A magnetic strip is glued to the side of the linear bearing housing and the sensor is added to the side of the assembly using a conventional slider mechanism. The position of the sensor in axial-direction is arbitrarily fixable to accommodate different initial positions due to the spring pre-compression feature.

5. The shape of the cover plastic has been changed; it now only covers the springs and axes. The cylinder assembly remains open and is easily accessible. The cover plate has now a slot in the middle such that the cable from the force sensor can be safely routed to the outside.

6. The complete test set-up is 2 cm wider than before to allow the pressure sensors to be mounted to the hydraulic pressure lines.

7. The shaft that the cylinder mounts to has been replaced by a standard misumi part and simplified the design of the attaching point.

8. The pre-compression screw has no fixture anymore, but the block has now a thread (M20) inside, it is thus arbitrarily fixable. Because of the thread, this screw should not move in axial direction during operation. The sole purpose of the pre-compression screw is to adjust the compressed length of the springs such that they are operated in their compressed phase. It must thus hold: $|x| < |L_0 - L_f|$ where x is the range of piston movement, L_0 is the pre-compressed length and L_f is the free length.

9. The clearance of the cover plate and as well as its thickness has been increased to make it robuster.

10. Standard screw sizes and similar types are used to reduce the number of different parts.

11. New spring guides are used that have plastic bearings pressed in to reduce the friction of the springs.

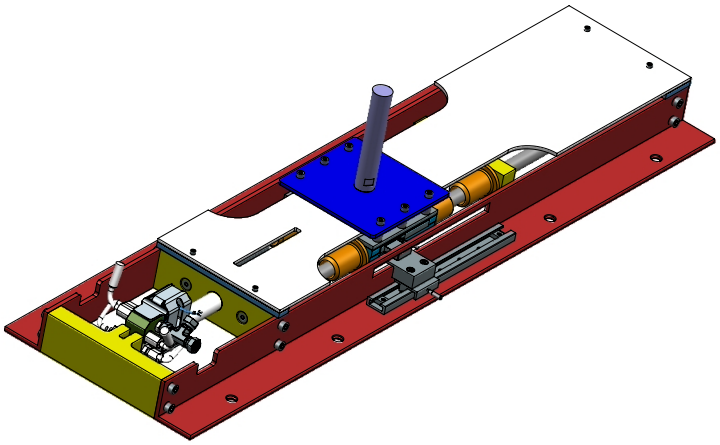

Figure D.1: CAD drawing of the newly designed FC1D test set-up.

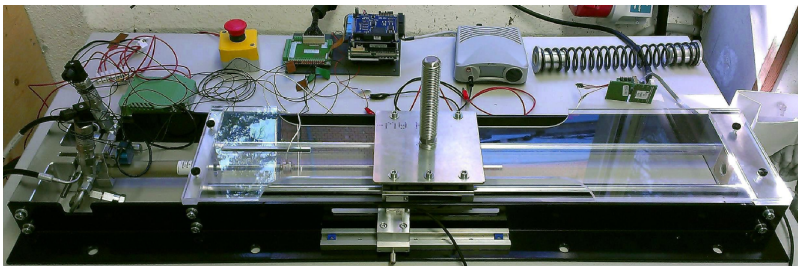

Figure D.2: Assembly picture of the newly designed FC1D test set-up.

Appendix E

Appendix: Overview of the Simulation

E.1 Simulation - Hydraulic Cylinder Model

The nonlinear system given by equation 3.5 is implemented as a Simulink model shown in figure E.1. The green shapes mark the inputs to the model and the red and magenta mark the outputs. The orange blocks denote continuous-time integrators. The light blue shapes denote the parameters of the load mass system. The inputs $1, 2, 3$ can be used to alter the mass, friction coefficient and spring stiffness of the load mass system while the simulation is running. All parameters are listed in Table A.1 and need to be loaded into the workspace before the simulation is run. The valve model, which is essentially a transfer function, can be switched on and off via a boolean (see section 2.4 for details). The Simulink model is implemented as a referenced model, hence all subsequent simulations use the very same model. The model interface is shown in figure E.2. All forces, that is, hydraulic force, friction force, spring force, total force, and load cell force are provided. Position, velocity and acceleration as well as pressure and flow are provided. The last output $u_v sp$ contains the input and output signals of the valve model. When the valve dynamics are deactivated, the signals are identical, else these signals can be used to visualize the impact of the valve dynamics on the input signal.

E.2 Simulation - Adaptive Controller Block

The controller block is shown in figure E.3, it takes the latest reference signal and measurement as inputs and calculates the corresponding control output using the specified fixed controller gains (within the block) or the gains from the additional inputs (LQR gain, LGR-I, feedforward). The control error and the individual parts of the control output are returned. The details of this LQR controller are shown in figure E.4. The input signals are first normalized by the light blue triangles on the left, then the signals are passed to the three parts of the LQR controller, that is, the feedforward part (Gamma) on top, the state feedback part below and the integrator part on the bottom. All three contributions are summed up for the anti-reset wind-up feature. Each control input is first de-normalized before being returned. The integrator is implemented as a discrete-time integrator using a specified sampling time. Currently the integrator acts only on the force channel, hence the selection block in front of the integrator.

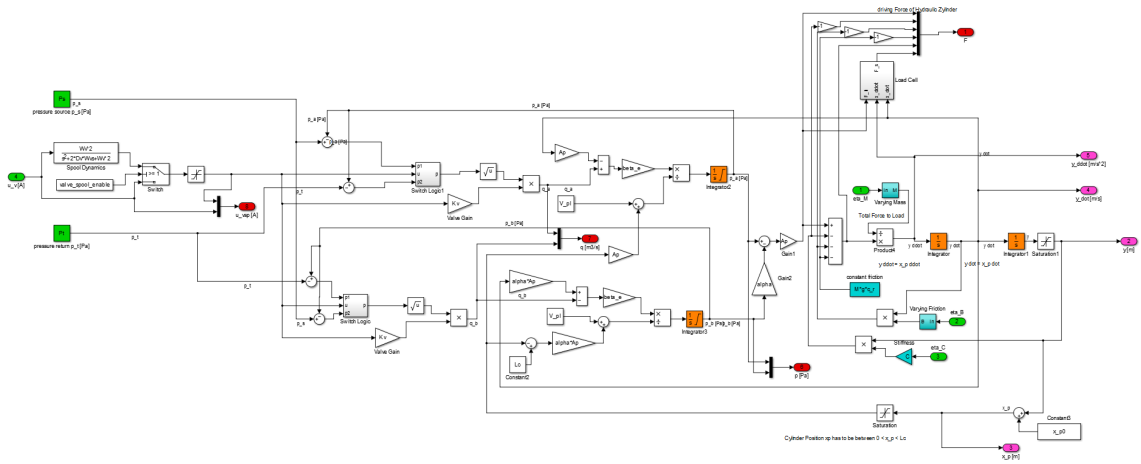

Figure E.1: Diagram of the hydraulic and load mass simulation.

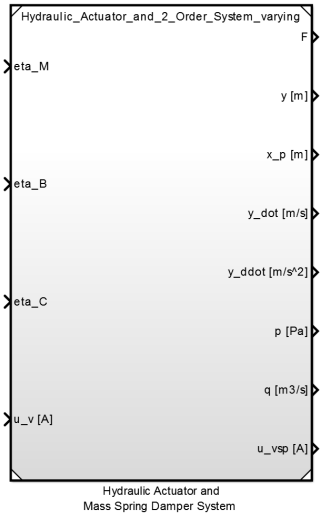

Figure E.2: Interface of the hydraulic and load mass simulation shown in figure E.1.

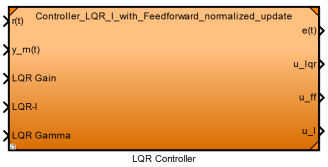

Figure E.3: Interface of the LQR-I controller block with feedforward term implemented in the Simulink simulation.

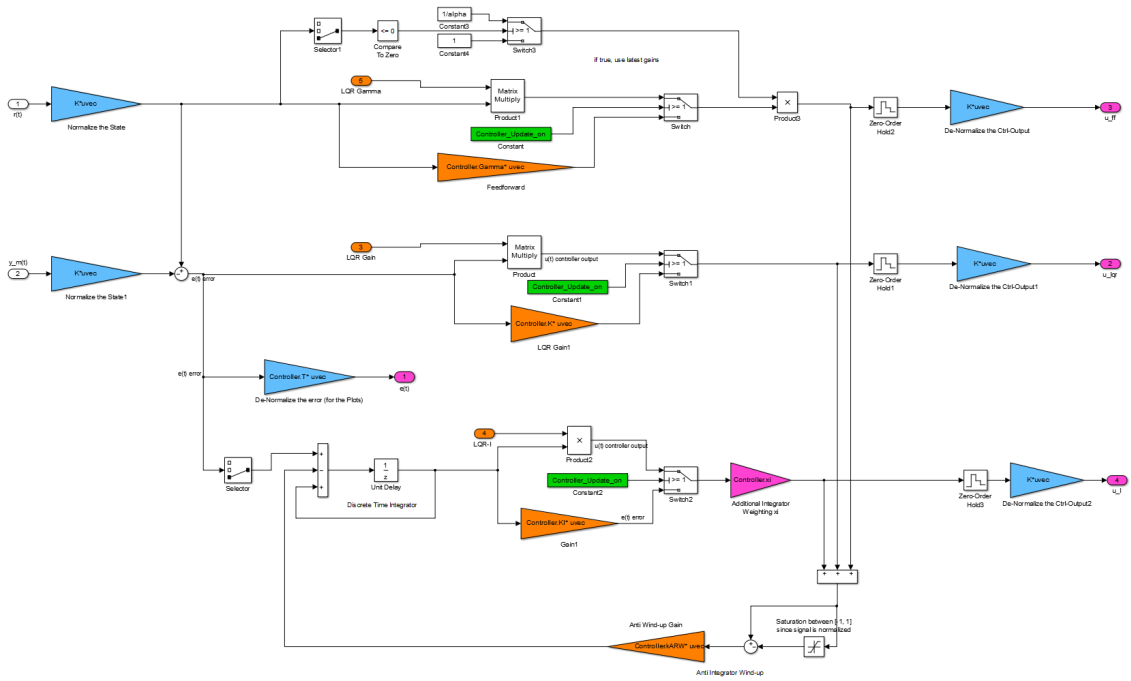

Figure E.4: Details of the LQR-I controller block with feedforward term. Shown are the normalization block and the three parts of the implemented controller. The controller gains can be changed via the inputs shown in orange.

E.3 Simulation - Estimation and Controller Block

Figure E.5 shows an example of how to interconnect these two systems along with the necessary features for logging and displaying the signals. This is the full simulation of the controller, hydraulic system, estimator, and update logic. There is only one single (direct) input, that is, the reference signal $r(t)$. Optionally there are three additional inputs (the other green shape in the middle of the figure) permitting to specify the values of the load mass parameters while the simulation is running. There are two memory blocks preventing the simulation to solve the algebraic loop, hence slowing down the simulation. These blocks only delay the signals by one simulation time step, not by the specified *sampling time* T_s. The blue blocks are responsible for saving the data to the workspace.

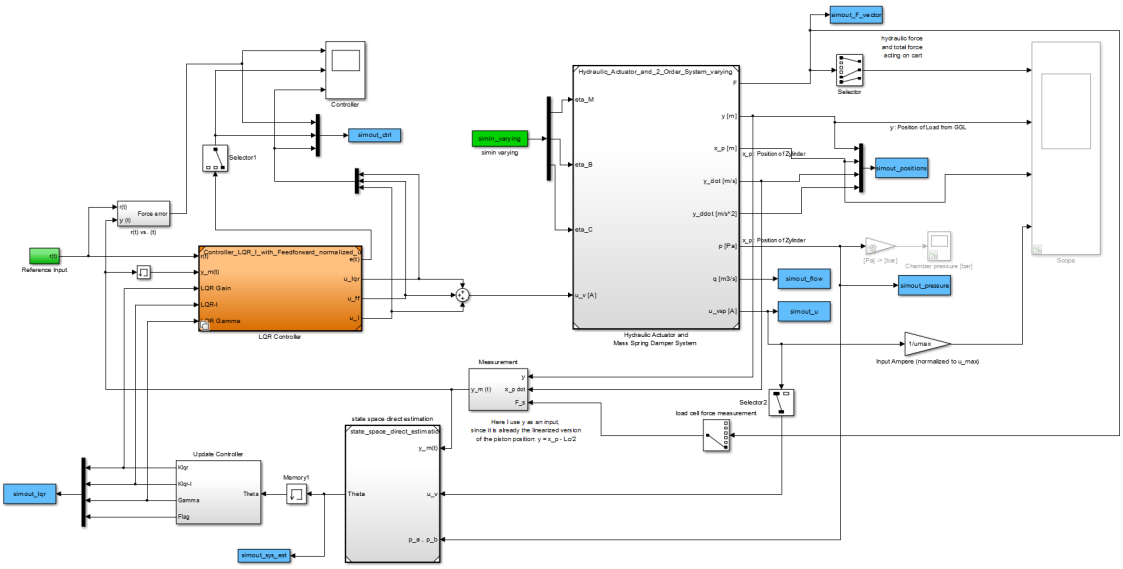

Figure E.5: This diagram shows the full simulation of the controller, hydraulic system, estimator, and update logic.

Appendix F

Appendix: Additional Implementation Details

F.1 Wrapper Functions of Estimator Implementation

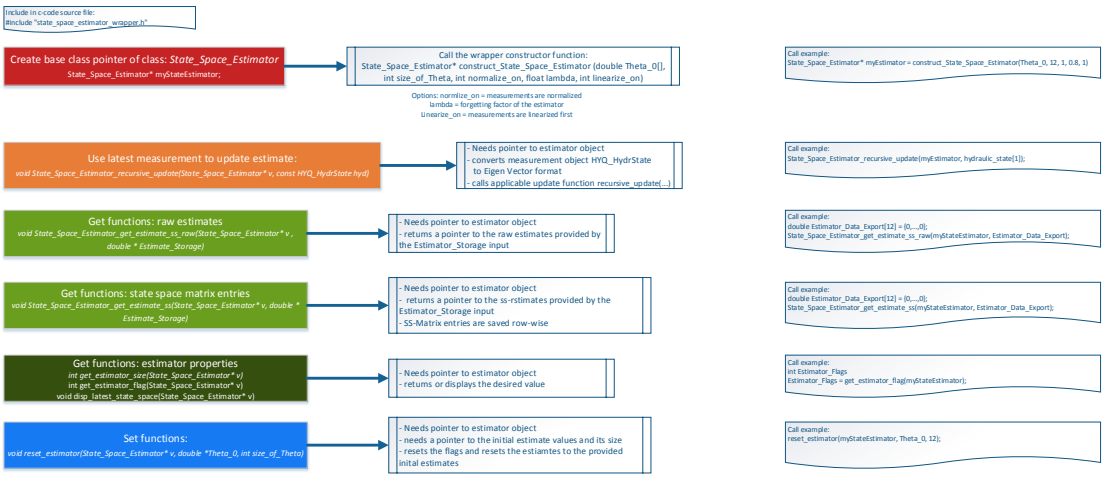

Figure F.1: List of all functions within the wrapper for the estimator implementation. For each function, there is a short description about the inputs and outputs and an example on how to call that particular function is also provided. This diagram is intended for documentation purposes.

F.2 Flags of Estimator Implementation

Implementation	Flag Name	Flag Number
Estimator Base Class	STABILITY_FLAG_NOT_YET_SET	1
	STABLE_SYS	2
	INSTABLE_SYS_MAP_TO_STABLE	3
	ERROR_WHILE_CALCULATING _EIGENVALUES_OF_SYSTEM_ESTIMATE	4
Estimator Subclass	FLAG_NOT_YET_SET	1
	ESTIMATION_ALL_OK	2
	REGRESSION_MATRIX_H_TO_SMALL	3
	VARIANCE_MATRIX_NOT_POSITIVE _DEFINITE	4
	DO_NOT_UPDATE_SINCE_INPUTS_TOO _SMALL	5

Table F.1: Table defining all flags of the estimator base class and subclass implementation. The implementation is discussed in section 8.1. The flag number can be used when there is no access to the enum type as defined within the C++ code. With this table, the number can be associated correctly.

F.3 Adaptive Controller Update Logic

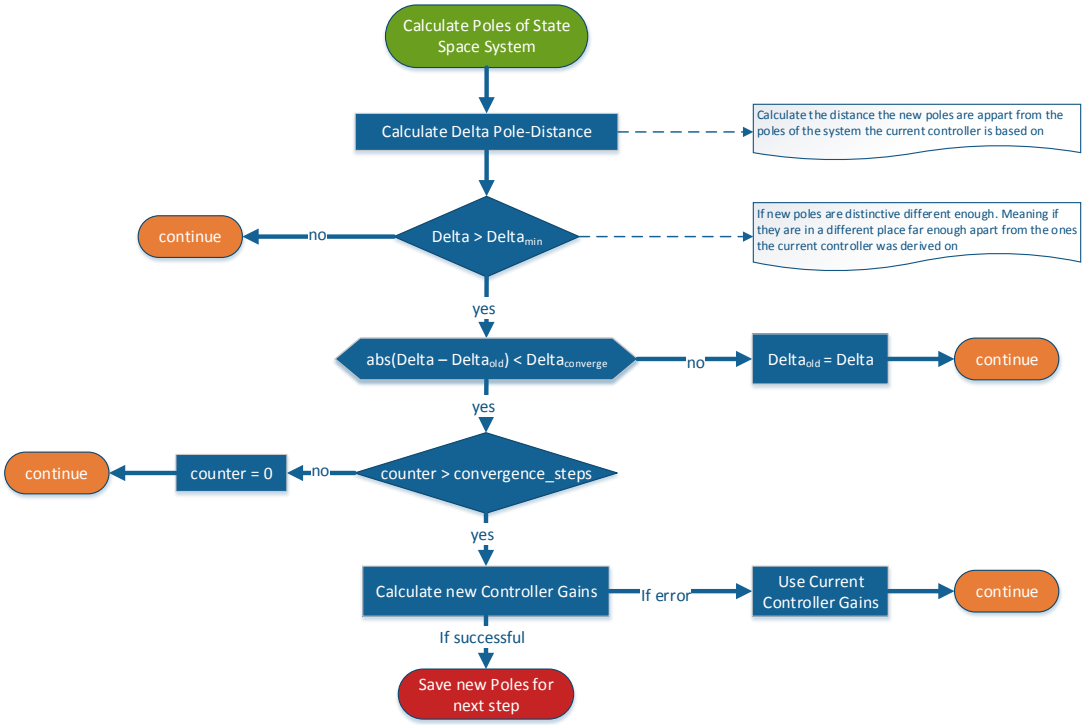

Figure F.2

F.4 Flags of Controller Implementation

Implementation	Flag Name	Flag Number
Controller Base Class	ADAPTIVE_CONTROLLER_INITIALIZED	0
	ADAPTIVE_CONTROLLER_OK	1
	WRONG_SAMPLING_TIME	2
	INTEGRATION_ACTION_FACTOR_XI_OUT _OF_BOUNDS	3
	WRONG_OPTION_FOR_FEEDFORWARD	4
	ERROR_WHILE_UPDATING_LQR_GAINS	5
	ERROR_WITHIN_SET_WEIGHTING _MATRICES_LQR	6
	NORMALIZE_OPTION_OUT_OF_BOUNDS	7
	INTEGRATION_SATURATION_SETTING _OUT_OF_BOUNDS	8
	SYSTEM_NOT_YET_CONVERGED	9
	SYSTEM_NOT_DIFFERENT_ENOUGH	10
Controller Subclass	INITIALIZATION_SUCCESSFUL	0
	LQR_CALCULATION_SUCCESSFUL	1
	CLOSED_LOOP_SYSTEM_NOT_STABLE	2
	SYSTEM_NOT_CONTROLLABLE	3
	SYSTEM_NOT_STABLE	4
	RICCATI_EQUATION_NOT_CONVERGING	5
	WEIGHTING_MATRICES_NOT_POSITIVE _DEFINITE	6
	POSITIVE_DEFINITE_CALCULATION _ERROR_EIGENVALUES_ARE_IMAGINARY	7

Table F.2: Table defining all flags of the controller base class and subclass implementation. The implementation is discussed in section 8.2. The flag number can be used when there is no access to the enum type as defined within the C++ code. With this table, the number can be associated correctly.

F.5 Additional Functions of the Controller Wrapper

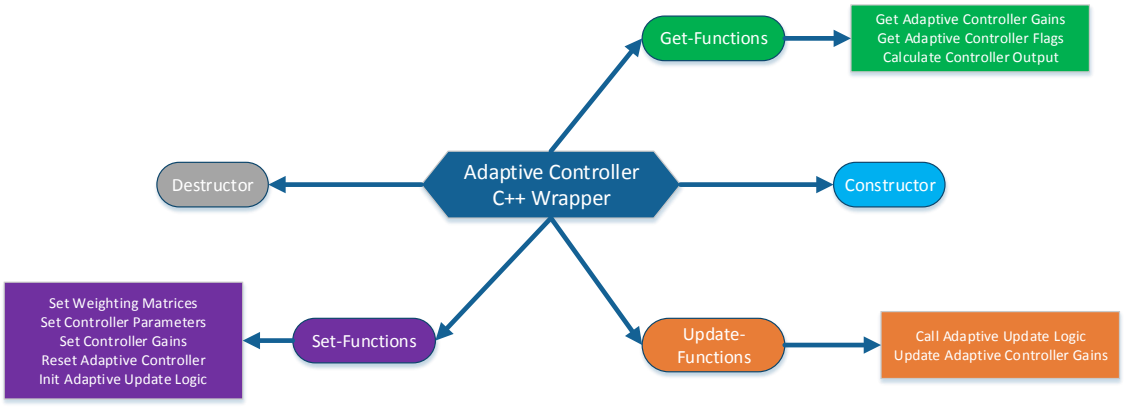

Figure F.3: Functions provided by the adaptive controller wrapper, used to enable access from the SL environment and to convert basic C++ data types into Eigen objects. Only basic function names are used in this figure.

Appendix G

Appendix: Additional Results from Experiments

G.1 Additional Experiments for Recursive Estimation Algorithm

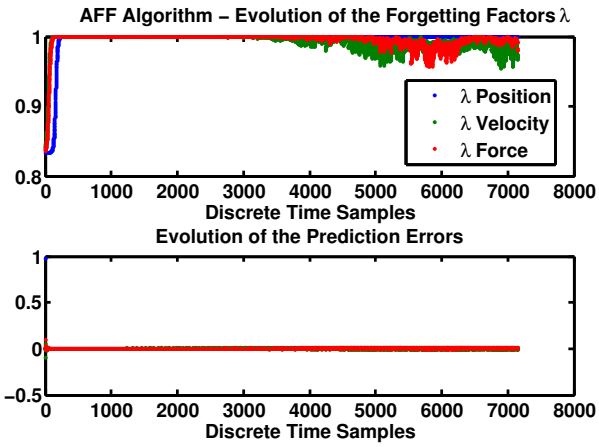

Figure G.1: Plot of the individual forgetting factors λ_i calculated by the adaptive forgetting factor (AFF) algorithm for the experiment $d00076$ shown on the left column of figure 9.4.

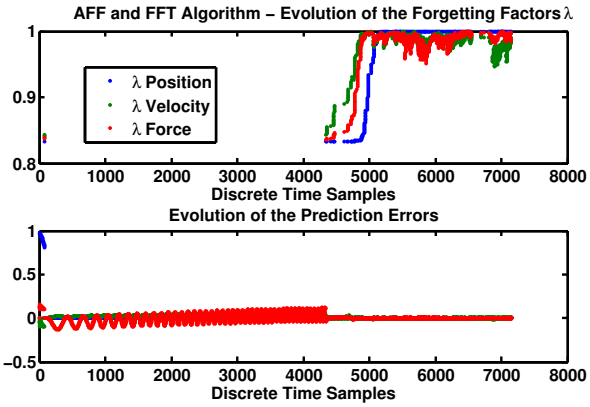

Figure G.2: Plot of the individual forgetting factors λ_i calculated by the adaptive forgetting factor (AFF) algorithm combined with the FFT algorithm for the experiment $d00076$ shown on the right column of figure 9.4.

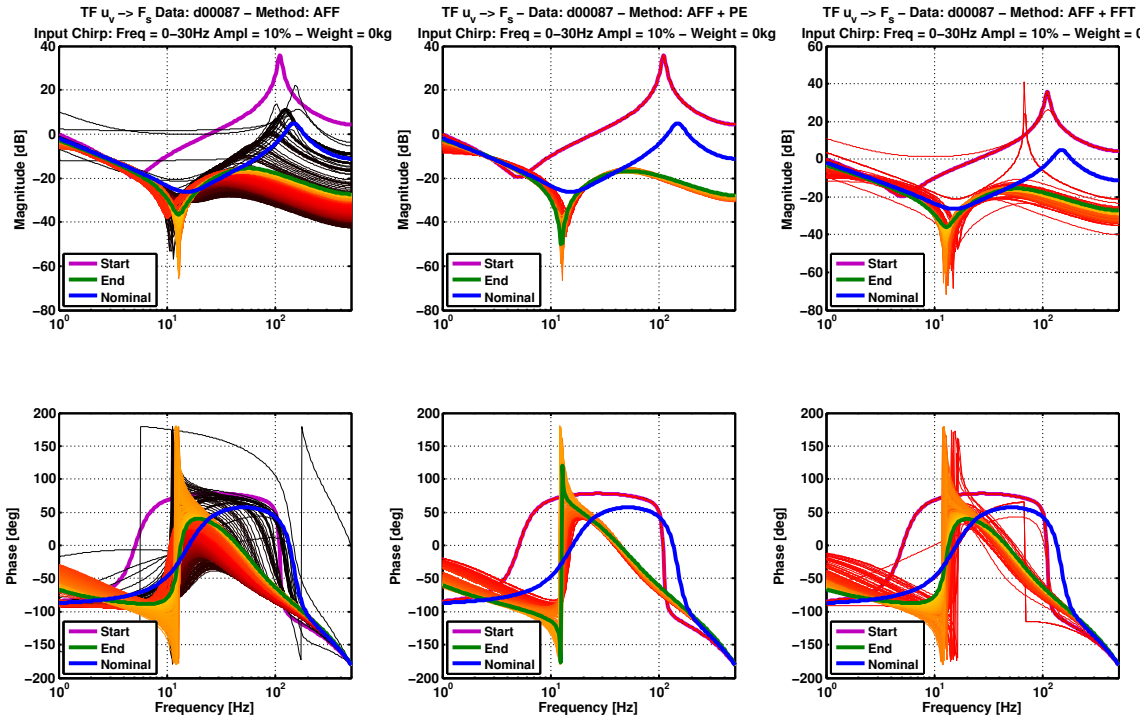

Figure G.3: Bode plot of transfer function estimates (normalized) for each time step using the FC1D test set-up. Shown are three estimation methods for nominal weight +0 kg using a chirp input signal. Color scheme and layout is the same as in figure 9.1. The left column shows the AFF algorithm, the middle the combination of AFF and PE algorithms and the right column the combination of AFF and FFT algorithms.

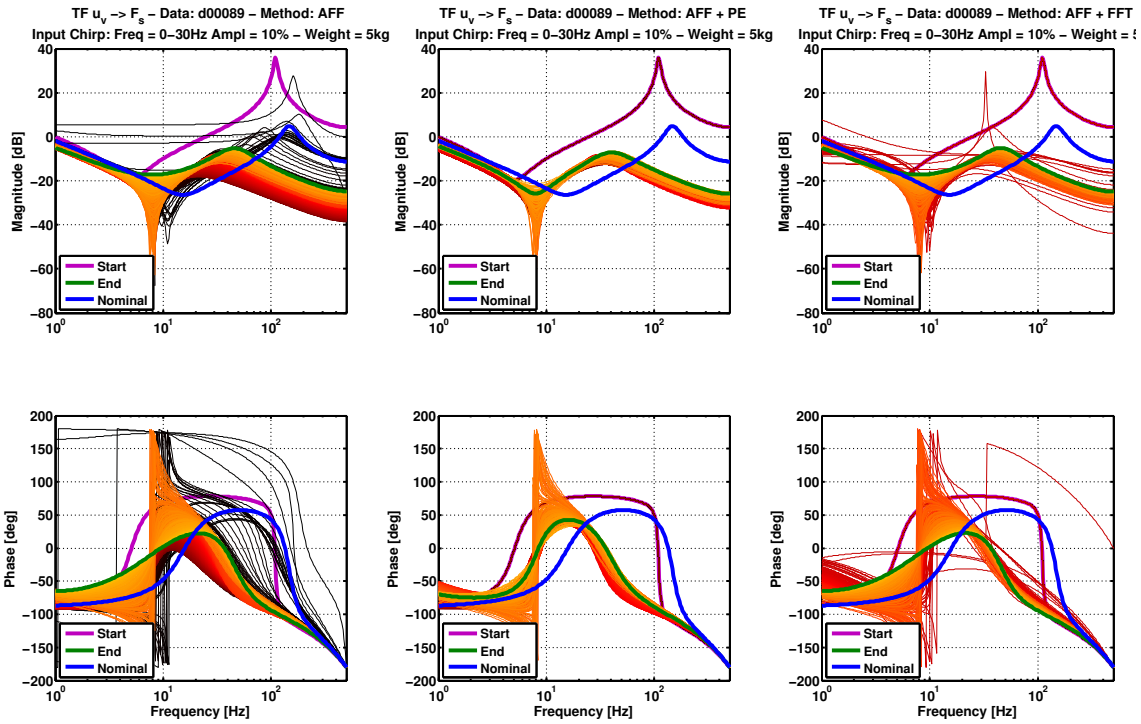

Figure G.4: Bode plot of transfer function estimates (normalized) for each time step using the FC1D test set-up. Shown are three estimation methods for additional weight +5 kg using a chirp input signal. Color scheme and layout is the same as in figure 9.1. The left column shows the AFF algorithm, the middle the combination of AFF and PE algorithms and the right column the combination of AFF and FFT algorithms. The PE and FFT algorithms are both sensitive to the measurement signals and do not yield to significantly better results.

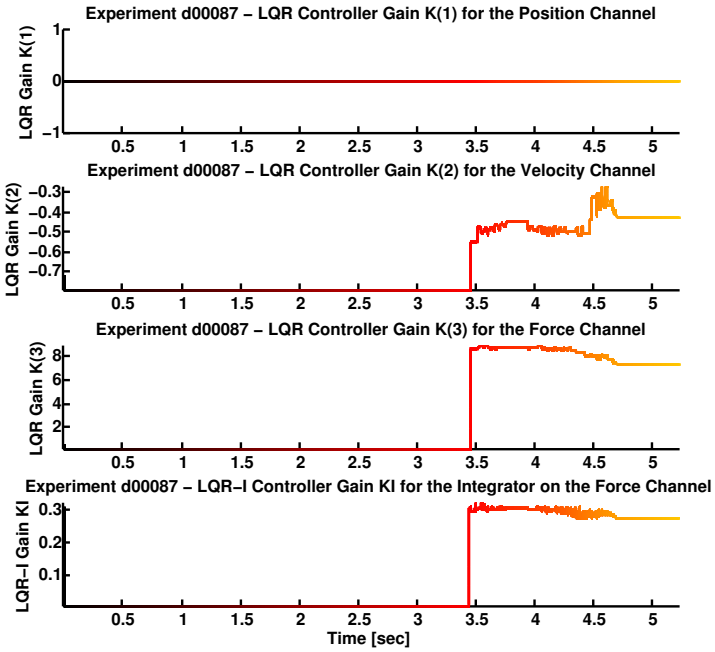

Figure G.5: Evolution of the controller gains for each time step using the estimated system descriptions derived with the FC1D test set-up. The evolution of the LQR controller gains is shown for the experiment *d*00087. The system estimates were calculated using the AFF and PE algorithm and are shown in figure 9.9 and 9.8, respectively. Color scheme and layout is the same as in figure 9.1.

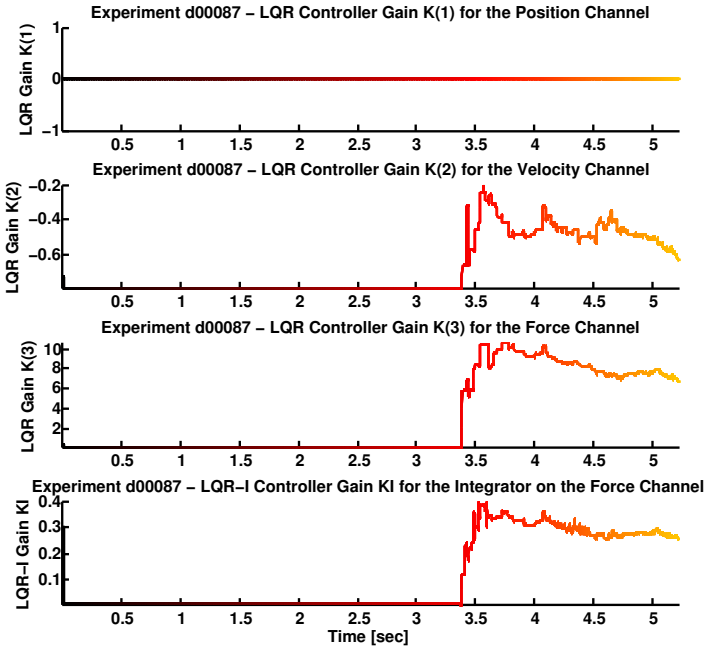

Figure G.6: Evolution of the controller gains for each time step using the estimated system descriptions derived with the FC1D test set-up. The evolution of the LQR controller gains is shown for the experiment *d*00087. The system estimates were calculated using the AFF and FFT algorithm and are shown in figure 9.9 and 9.8, respectively. Color scheme and layout is the same as in figure 9.1.

G.2 Additional Experiments for Adaptive Controller

G.2.1 Control Performance for Different Sine Waves

The following tests were performed using the "compliant" LQR controller designed for the nominal system, with load mass $m_l = 2.5\,\text{kg}$, applied to the new test set-up configured at nominal weight (2.5 kg). Four sinusoidal reference signals were commanded each with a different frequency ranging from 1 Hz to 5 Hz but equal amplitude of 200 N. Figure G.7 shows the force tracking performance of the LQR controller along with the individual components of the controller output signal u_{tot}.
Similar sine waves but with increased amplitude 400 N were commanded and are shown in figure G.8.

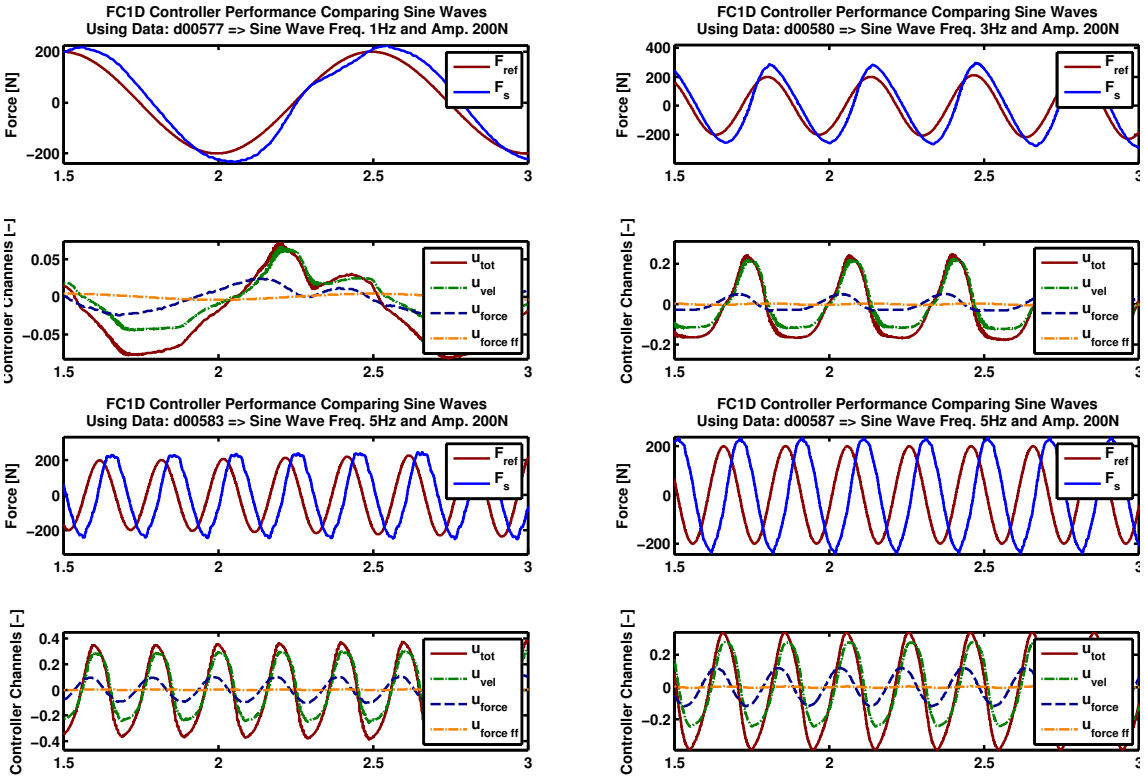

Figure G.7: Force tracking performance of the "compliant" LQR controller (fixed gain) for different sine wave signals of 200 N amplitude. The force tracking, as can be seen, is quite good, for sine waves below 5 Hz. A phase delay is present for frequencies higher than 5 Hz. Note also how the stick friction influence is particularly visible for the 1 Hz sine wave due to the relative slow velocity of the cylinder. As explained in detail within section 9.2.

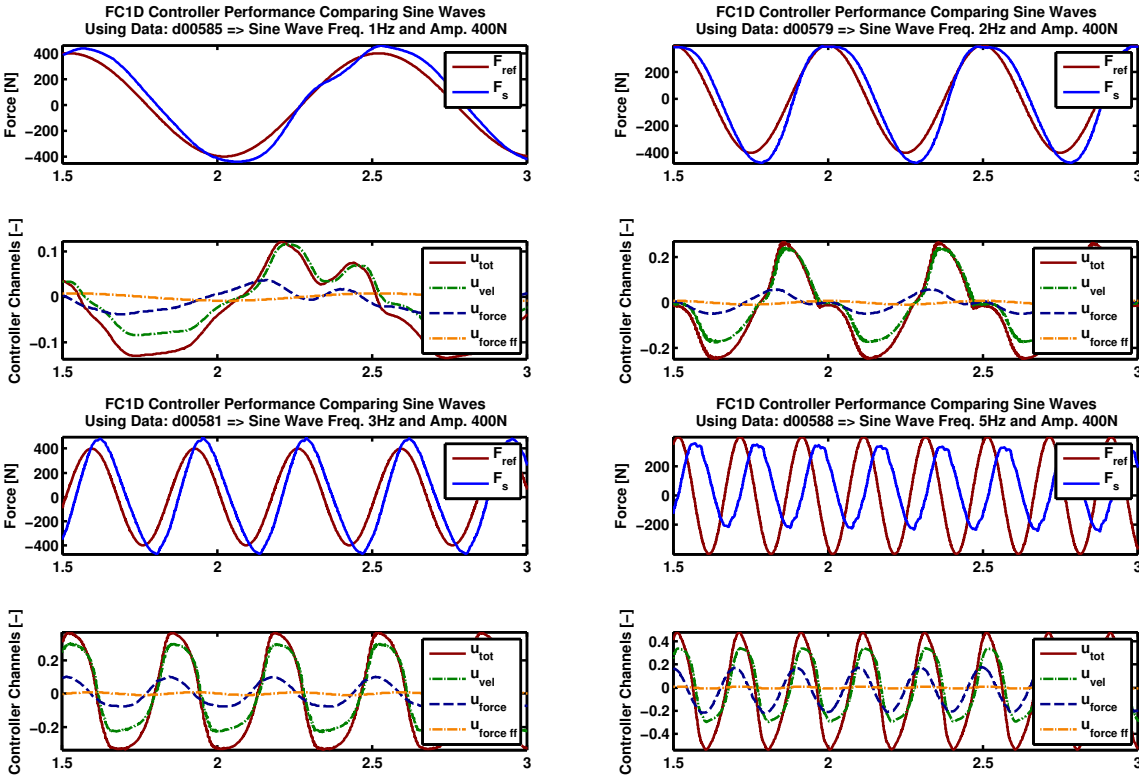

Figure G.8: Force tracking performance of the "compliant" LQR controller (fixed gain) for different sine wave signals of 400 N amplitude. Again, a phase delay is present for frequencies higher than 3 Hz. Note again that the stick friction influence is particularly visible for the 1 Hz sine wave and also for the 2 Hz sine, although with reduced impact. Note that this sticking behavior is more dependent on frequency, that is the velocity of the cylinder, than magnitude of the reference signal. See also section 9.2.

G.2.2 Control Performance for Different Load Mass

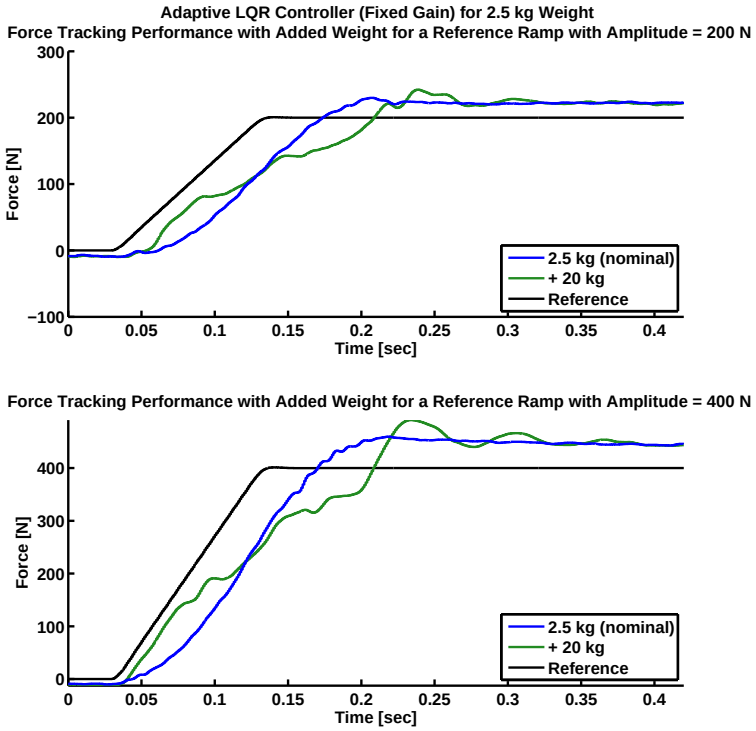

Figure G.9: Force tracking performance of the adaptive LQR controller (fixed gain) for a ramp reference signal.

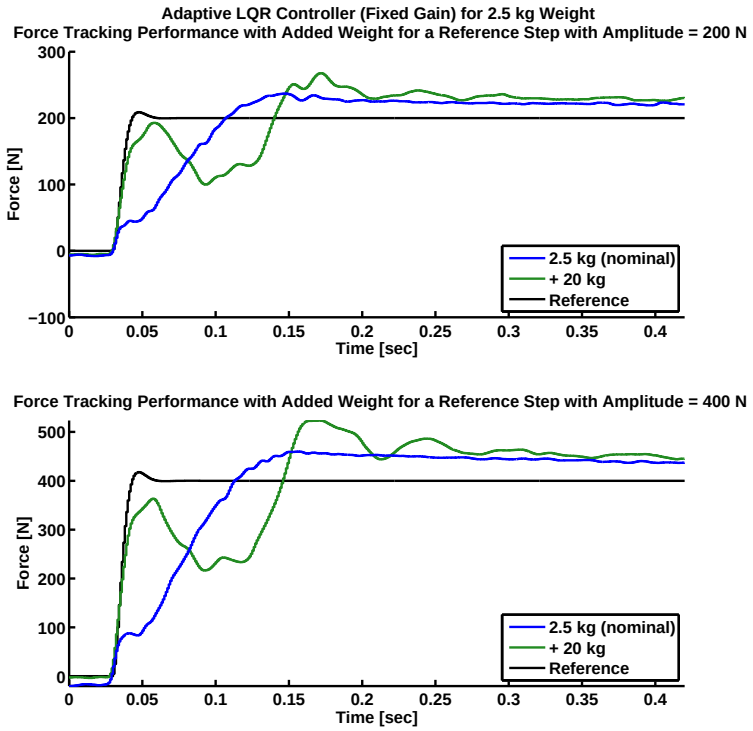

Figure G.10: Force tracking performance of the adaptive LQR controller (fixed gain) for a step reference signal.

Bibliography

[1] Mohammad Abu-Naser and Geoffrey A. Williamson. "Convergence Properties of Adaptive Estimators of Time-Varying Linear Systems using Basis Functions". In *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, pages pages 336–341, Sept 2006.

[2] Brian D.O. Anderson and John B. Moore. *Linear optimal control.* Prentice-Hall Englewood Cliffs, NJ, 1971.

[3] William. F. Arnold, III and Alan. J. Laub. "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations". In *Proceedings of the IEEE*, volume 72, pages 1746–1754, December 1984.

[4] Karl J. Åström and Björn Wittenmark. *"Adaptive Control"*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994. Good parts of the book are: Chapter 2 page 62: Real-Time Parameter Estimation, Chapter 11 page 473: "Practiacal Issues and Implementation" about the estimator wind-up problem and solutions. Chapter 11 page 477: Conditional updating.

[5] Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson de Pieri, and Darwin G. Caldwell. "A reactive controller framework for quadrupedal locomotion on challenging terrain". In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2554–2561, May 2013.

[6] Stéphane Bazeille, Victor Barasuol, Michele Focchi, Ioannis Havoutis, Marco Frigerio, Jonas Buchli, Darwin G. Caldwell, and Claudio Semini. "Quadruped robot trotting over irregular terrain assisted by stereo-vision". In *Intelligent Service Robotics*, volume 7, pages 67–77. Springer Berlin Heidelberg, 2014.

[7] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[8] Md. Zulfiquar Ali Bhotto and Andreas Antoniou. "New Improved Recursive Least-Squares Adaptive-Filtering Algorithms". *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 60(6):pages 1548–1558, June 2013.

[9] Robert R. Bitmead. "Persistence of excitation conditions and the convergence of adaptive schemes". *Information Theory, IEEE Transactions on*, 30(2):pages 183–191, Mar 1984.

[10] Thiago Boaventura. *"Hydraulic Compliance Control of the Quadruped Robot HyQ"*. Ph.d. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genoa, Italy, March 2013.

[11] Thiago Boaventura, Michele Focchi, Marco Frigerio, Jonas Buchli, Claudio Semini, Gustavo A. Medrano-Cerda, and Darwin G. Caldwell. "On the role of load motion compensation in high-performance force control". In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4066–4071, 2012.

[12] Thiago Boaventura, Claudio Semini, Jonas Buchli, Marco Frigerio, Michele Focchi, and Darwin G. Caldwell. "Dynamic torque control of a hydraulic quadruped robot". In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1889–1894, 2012.

[13] Jonas Buchli, Freek Stulp, Evangelos Theodorou, and Stefan Schaal. "Learning Variable Impedance Control". In *The International Journal of Robotics Research*, volume 30, pages 820–833. SAGE Publications, 2011.

[14] Burster Präzisionsmesstechnik GmbH & co kg 2012. *Subminiature Load Cell Tension and Compression - Technical Data Sheet*, 2012.

[15] John C. Doyle. "Guaranteed Margins for LQG Regulators". In *IEEE Transactions on Automatic Control*, volume 23, pages 756–757, August 1978.

[16] Michele Focchi. *"Strategies To Improve the Impedance Control Performance of a Quadruped Robot"*. Ph.d. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genoa, Italy, April 2013.

[17] Michele Focchi, Victor Barasuol, Ioannis Havoutis, Jonas Buchli, Claudio Semini, and Darwin G. Caldwell. "Local Reflex Generation for Obstacle Negotiation in Quadrupedal Locomotion". In *Int. Conf. on Climbing and Walking Robots (CLAWAR)*, 2013.

[18] Gene F. Franklin, Michael L. Workman, and Dave Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[19] Michael Green and John B. Moore. "Persistence of excitation in linear systems". In *Systems & Control Letters*, volume 7, pages 351–360, 1986.

[20] Gaël Guennebaud, Benoît Jacob, et al. Eigen c++ library v3. http://eigen.tuxfamily.org, 2010. Accessed March 2014.

[21] Ardéshir Guran, Friedrich Pfeiffer, and Karl Popp, editors. *Dynamics with Friction: Modeling, Analysis and Experiment*, volume 7 of *Series on stability, vibration and control of Systems*. World Scientific Pub., 2001.

[22] Prof. Dr. Lino Guzzella. *Discrete Time Control Systems*. IDSC at ETH Zürich, Spring 2013. Accessed March 2014, `http://www.idsc.ethz.ch/Courses/digital_control/exercises_digreg/Slides_DigReg_2013.pdf`.

[23] Marco Hutter, Christian Gehring, Michael Bloesch, Mark A. Hoepflinger, David C. Remy, and Roland Siegwart. "StarlETH: A Compliant Quadrupedal Robot for Fast, Efficient, and Versatile Locomotion". In *15th International Conference on Climbing and Walking Robot-CLAWAR 2012*, 2012.

[24] Istituto Italiano di Tecnologia (IIT) in Genova, Italy. *Website about the Development Project HyQ.* Accessed March 2014, `http://www.iit.it/en/advr-labs/dynamic-legged-systems/hydraulically-actuated-quadruped-hyq.html`.

[25] Mohieddine Jelali and Andreas Kroll. *Hydraulic Servo-systems: Modelling, Identification and Control.* Springer London, 2003.

[26] Claude Kaddissi, Jean-Pierre Kenné, and Maarouf Saad. "Indirect Adaptive Control of an Electro-Hydraulic Servo System Based on Nonlinear Backstepping". In *Industrial Electronics, 2006 IEEE International Symposium on*, volume 4, pages 3147–3153, July 2006.

[27] Ljung Lennart. *System Identification: Theory for the User.* PTR Prentice Hall, Upper Saddle River, NJ, 1999. Page 363–364, 366, 378.

[28] Y.F. Li and X.B. Chen. "On the Dynamic Behavior of a Force/Torque Sensor for Robots". In *Instrumentation and Measurement, IEEE Transactions on*, volume 47, pages 304–308, February 1998.

[29] Lennart Ljung and Tomas McKelvey. "A Least Squares Interpretation of Sub-Space Methods for System Identification". In *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, volume 1, pages 335–342, December 1996.

[30] The MathWorks, Inc. *Documentation about the dare function for solving the discrete-time algebraic Riccati equations.* Accessed March 2014, `http://www.mathworks.ch/ch/help/control/ref/dare.html`.

[31] Moog Inc., East Aurora, N. Y. 14052. *Type 30 Nozzle-Flapper Flow Control Servo-Valves.*

[32] Moog Inc., East Aurora, N. Y. 14052. *Moog E024 Sub Miniature Servo-Valve - Specification*, 2009.

[33] Constantin Paleologu, Jacob Benesty, and Silviu Ciochina. "A Robust Variable Forgetting Factor Recursive Least-Squares Algorithm for System Identification". *Signal Processing Letters, IEEE*, 15:pages 597–600, 2008.

[34] S. Schaal. The SL Simulation and Real-Time Control Software Package. Technical report, Computational Learning and Motor Control Laboratory at University of Southern California, Los Angeles, 2009. Accessed March 2014, `http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf`.

[35] Claudio Semini. *"HyQ–Design and Development of a Hydraulically Actuated Quadruped Robot".* Ph.d. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genoa, Italy, April 2010.

[36] Claudio Semini, Jonas Buchli, Marco Frigerio, Thiago Boaventura, Michele Focchi, Emanuele Guglielmino, Ferdinando Cannella, Nikos G. Tsagarakis, and Darwin G. Caldwell. "HyQ–A Dynamic Locomotion Research Platform". In *International Workshop on Bio-Inspired Robots, Nantes (France)*, 2011.

[37] Claudio Semini, Nikos G. Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G. Caldwell. "Design of HyQ – a hydraulically and electrically actuated quadruped robot". In *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, volume 225, pages 831–849, 2011.

[38] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*, volume 2. Wiley New York, 2007. Pages 381, 552.

[39] William J. Thayer. Specification Standards for Electrohydraulic Flow Control Servo-Valves. Technical Report 117, Moog Inc. Controls Division, East Aurora, N. Y. 14052, July 1959, revised June 1962.

[40] H. Unbehauen, P. Du, and U. Keuchel. "Application of a Digital Adaptive Controller to a Hydraulic System". In *Control, 1988. CONTROL 88., International Conference on*, pages 177–182, April 1988.

[41] Ganwen Zeng and Ahmad Hemami. "An Overview of Robot Force Control". In *Robotica*, volume 15, pages 473–482, September 1997.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Agile and Dexterous Robotics Lab
Institute of Robotics and Intelligent Systems
Prof. Dr. Jonas Buchli

Title of work:

Optimization of the Low-Level Torque Controller of the Quadruped Robot HyQ

Thesis type and date:

Master's Thesis, April 2014

Supervision:

Dr. Thiago Boaventura
Farbod Farshidian
Prof. Dr. Jonas Buchli

Student:

Name:	Sven Hubacher
E-mail:	sven.hubacher@alumni.ethz.ch
Legi-Nr.:	07-914-104
Semester:	3

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

`http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf`

Zurich, 29. 12. 2014: _____