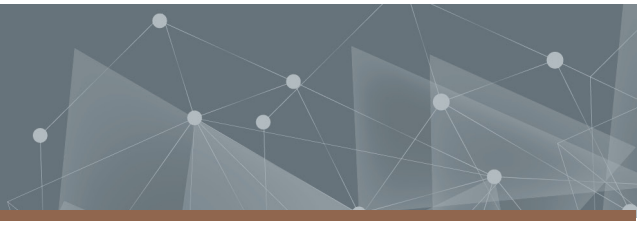
CHALMERS
UNIVERSITY OF TECHNOLOGY



Adaptive Control of Hydraulic Drive System

Real Time Steady-State Estimation Using Kalman Filter

ROBERT AGVIK
SOPHIE VÄNNMAN

DEPARTMENT OF MECHANICS AND MARITIME SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Adaptive Control Of Hydraulic Drive System

Real Time Steady-State Estimation Using Kalman Filter

ROBERT AGVIK
SOPHIE VÄNNMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics & Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2023

Adaptive Control Of Hydraulic Drive System
Real Time Steady-State Estimation Using Kalman Filter
Robert Agvik, Sophie Vännman

Supervisor: Marcus Broberg, CPAC Systems AB
Supervisor: Oskar Johansson, CPAC Systems AB
Examiner: Peter Forsberg, Mechanics and Maritime Sciences

Master's Thesis 2023
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Adaptive Control Of Hydraulic Drive System
Real Time Steady-State Estimation Using Kalman Filter
Robert Agvik, Sophie Vännman
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

This thesis investigates how an adaptive controller can be developed for a hydraulic drive system. The currently used controller consists of a feed-forward component and a PI component. The feed-forward component consists of tables describing the steady-states of the vehicle, where the desired value is acquired through linear interpolation between the table points. The problem being that the system behaviour changes with time, leading the table values to become inaccurate and as a consequence the controller cannot control the system to satisfaction.

The goal of this thesis is to solve this adaptivity problem and develop a controller that can handle the changes to the system over time. This is achieved by implementing a Kalman filter-based algorithm that updates the steady-state values of the feed-forward part of the controller to correct values, based on continuous measurements from the system. Relevant tables values are chosen as Kalman states which are updated based on measurements from the system and then reinserted into the tables. Two further development concepts are investigated. In the first the Kalman filter is updated with a batch of measurements and in the second a 3rd order polynomial is used to find interpolated values instead of linear interpolation.

The conclusion is that the above mentioned algorithm results in an efficient system controller that maintains a comfortable driving experience. It is shown in this thesis that updated table values provide better control performance and that this contributes to the adaptivity of the controller.

Keywords: Machine learning, Reinforcement learning, System control, Model-free Control, Model-Based Control, Adaptive Control, Kalman Filter, Extended Kalman Filter

Acknowledgements

First of all we would like to thank *CPAC Systems AB* for the opportunity to write this thesis. Thank you to our supervisors Marcus Broberg and Oskar Johansson for their valuable knowledge, help and time, especially regarding our many test runs. We would also like to thank our examiner, Peter Forsberg for his support and guidance. Lastly a big thank you to our unofficial supervisor Axel Ceder for his invaluable input and ideas.

Robert Agvik, Sophie Vännman, Gothenburg, June 2023

Thesis Advisors: Marcus Broberg and Oskar Johansson, CPAC Systems AB
Thesis Examiner: Peter Forsberg, Mechanics and Maritime Sciences

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CAN	Controller Area Network
DNN	Deep Neural Network
EKF	Extended Kalman Filter
LQR	Linear Quadratic Regulator
PCHIP	Piecewise Cubic Hermite Interpolating Polynomial
PD	Proportional Derivative
PILCO	Probabilistic Inference for Learning Control
RDPG	Recurrent Deterministic Policy Gradients
UKF	Unscented Kalman Filter

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

k	Index for sample
t	Index for time step

Parameters

i_{ctrl}	Control current
c_{ctrl}	Control pressure
K_p	Proportional control gain
K_i	Integral control gain
F_k	State transition model
B_k	Control-input model
$P_{k\|k}k$	Estimate covariance
$\hat{P}_{k\|k-1}$	Predicted estimate covariance
Q_k	Process noise covariance
H_k	Observation model
R_k	Observation noise covariance
S_k	Innovation covariance
K_k	Kalman gain
$P(x)$	Polynomial

Variables

$\hat{x}_{k\|k-1}$	Predicted state estimate
$x_{k\|k}$	Update state estimate i
$\tilde{y}_k$	Innovation
f'_k	Derivative at sample point

Contents

List of Figures

List of Tables

1
Introduction

The hydraulic drive system investigated in this thesis utilizes a hydraulic pump to generate a pressure that drive hydraulic motors that, in turn, actuates the wheels of the vehicle. A PI-controller with a feed forward block is implemented to regulate the pressure of the pump. However, the current control system is not adaptive over time which causes problems since there is considerable wear of the hydraulic pump and engine. The controller is therefore at risk of becoming inaccurate and could fail to control the system to satisfaction.

In a previous master thesis a machine learning based control system that uses a reinforcement learning algorithm has been developed to adapt to the changes of characteristics in the physical system [1]. The developed controller show promising results but is not stable when tested on the physical system and requires further development before it can replace the current control system.

The adaptivity problem remains to be solved and is further investigated in this thesis. The goal is to develop a control system that successfully controls the hydraulic drive system while maintaining stability.

1.1 Background

This thesis investigates control of a hydraulic drive system, which consists of a axial piston variable pump, mounted on the main engine, and a radial piston motor. The pump has several pistons mounted on a swashplate with a controllable angle. The main engine will drive the rotation of the swashplate and as the pistons rotate they move in and out of the cylinders, generating a pressure that then drives the radial piston motor. The volumetric flow is controlled by adjusting the swashplate angle of the pump, which is in turn actuated by two hydraulic cylinders with spring return to the center position. Electrically controlled hydraulic valves actuate the cylinders [1]. The pump generates a hydraulic flow, that drives the pistons in the hydraulic motors and thereby actuate the wheels of the vehicle. To control this system, a work pressure reference is provided to the controller, which outputs an electric current as control signal.

Figure 1.1 show a schematic description of the system where the *work pressure* is defined as the relative pressure between A and B.

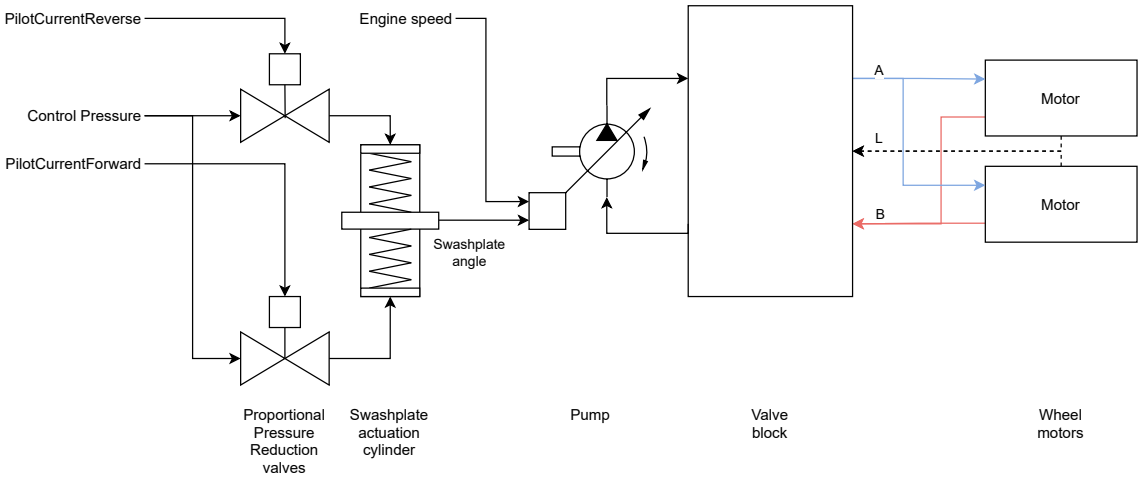


Figure 1.1: Hydraulic system overview [1].

The current applied to the hydraulic valves generate a specific control pressure. The characteristics are shown in figure 1.2.

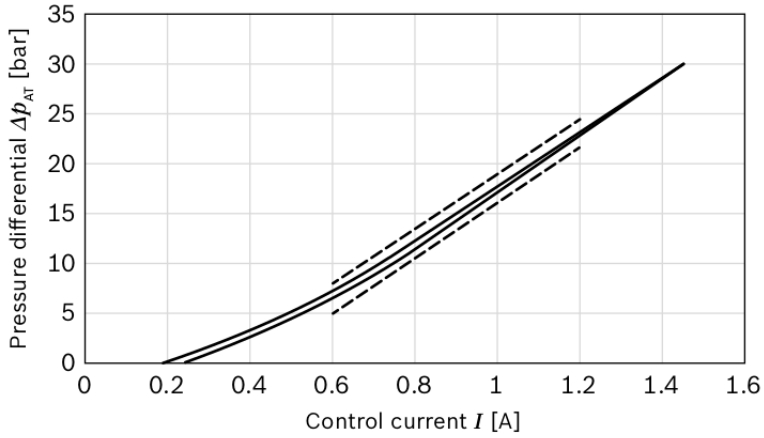


Figure 1.2: Characteristics curve of hydraulic valves with tolerance band. [2]

1.1.1 Current controller

The existing controller is a proportional-integral-regulator (PI) combined with static feed-forward block. A schematic representation can be seen in figure 1.3.

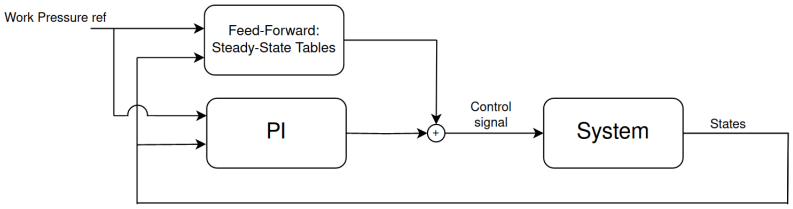


Figure 1.3: Schematic view of the current hydraulic drive controller.

The static feed-forward part of the controller consists of a number of tables that correspond to specific reduction ratios. The reduction ratio is the angular velocity ratio between the hydraulic pump and the wheels. The tables provide information about which control pressure is necessary to acquire a specific work pressure given an engine speed at steady-state, see fig 1.4. To acquire the control pressure value for any data point, trilinear interpolation is used along the axes engine speed, work pressure and reduction ratio. The tables were acquired through calibration runs a few years ago. They were done by driving the vehicle at specific reduction ratios, engine speeds and work pressures, for a certain amount of time, ensuring that steady-state has been reached. The data points in the tables are carefully chosen to ensure that the system behaviour between them can be approximated as linear. The tables used in the current controller are referred to as "current tables".

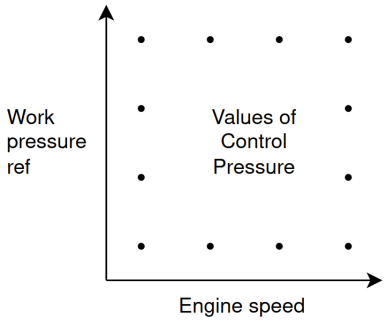


Figure 1.4: Schematic representation of one of the tables used in feed-forward part of controller.

The PI-part of the controller handles system dynamics and compensates for inaccuracies in the table values. The PI controller has proportional and integral control actions, described by K_p and K_i. The proportional control action adjusts the output based on the difference between the desired reference and the current measured value while the integral control action integrates the error over time to eliminate steady-state errors.

The two control components are added to each other and describe the desired control pressure, c_{ctrl}. The control current, i_{ctrl}, needed to obtain a control pressure is found through equation (1.1), which is based on the characteristics seen in figure 1.2.

$$i_{ctrl} = \frac{c_{ctrl} + 10}{27.5} \tag{1.1}$$

1.2 Relevant research articles and literature review directions

Previous work has been done in the field of system control using machine learning and AI and on how hybrid control, using both machine learning and traditional control methods, could be implemented. Some examples will be presented in this literature review.

1.2.1 Master thesis on front wheel drive control

A master thesis has been done previously at CPAC Systems AB by Oskar Johansson and Benjamin Lundgren, "Adaptive Model-Free Control Applied to Truck Front Wheel Drive, Real time control with reinforcement learning utilising recurrent deterministic policy gradient" [1]. The purpose being to solve the problem of adaptability, where the front wheel drive controller becomes inaccurate with time. The goal of the thesis was to emulate an existing, dynamical controller with a recurrent neural network and supervised learning and to improve its performance by using reinforcement learning.

In the thesis an Artificial Neural Network, trained with supervised learning, was used to emulate the existing controller. Then the reinforcement learning algorithm Recurrent Deterministic Policy Gradients (RDPG) is implemented to improve the controller and make it adaptive.

The conclusion of the thesis was that emulating the existing controller was feasible but the results were relatively poor. Using reinforcement learning to improve the performance of the artificial neural network showed promising results. It was found that the learning rate of the critic network affected whether the algorithm would converge or not. When testing the method against the physical system the learning would diverge, therefore further testing and tuning is necessary to allow the system to converge [1].

1.2.2 Machine learning in control

Novel research has been done on how machine learning can be used to control complex nonlinear systems. D. Kim et. al. published a review on machine learning methods in soft robotics [4]. Soft robots are, compared to rigid robots, flexible, adaptive and deformable, and exhibit complex behaviours caused by nonlinearities and hysteresis. This creates problems when trying to model, calibrate and control these robots, therefore methods using machine learning have been researched in recent years. The review showed that machine learning can effectively solve problems related to soft robots and is therefore critical in the field. Several limitations remain, which are due to the large quantity of data needed to train the network, manufacturing errors in sensors and actuators and problems related to real-time applicability [4].

Reinforcement learning is a useful tool in artificial domains, but the research made is difficult to use in practice [5]. In the article "Challenges of Real-World Reinforcement Learning" G. Dulac-Arnold et. al. present the main challenges connected to implementations of reinforcement learning in real-world scenarios. The challenges are related to, among other things, learning rates in relations to the amount of samples, safety constraints, the demand from operators to have interpretable or explainable policies and unknown or large delays in sensors or actuators.

To summarize the research on machine learning and system control, it can be said that machine learning can be a very useful tool, especially when it comes to nonlinear, complex systems that are difficult to control with traditional methods [4, 6].

With this being said, there are some important challenges that need to be addressed in order to make it work in a real-world scenario.

1.2.3 Black-box problem and interpretability

A controller for a safety critical system needs to be proven safe. Therefore it is relevant to look at research exploring the downsides of black-box algorithms and how interpretable AI can be designed and used to improve the transparency of the decision-making process.

There are inherent issues linked to using deep neural networks (DNN). It is often impossible to trace back how their many parameters are interconnected and combined to make decisions [7]. The conclusion is, that the DNN can be considered a black-box: there is no way of making sense how it makes decisions and is referred to as the black-box problem. This poses problems in safety-critical systems, such as some control systems and in system's making high stakes decisions, for example in health care and criminal justice [8].

To solve this black-box problem, there has been a surge of work done on "explainable AI", which is when a second model is designed to explain what happens in the black-box [8]. According to Cynthia Rudin, this is problematic since the explanations are often unreliable and misleading. Instead she suggests that inherently interpretable AI should be used. It is difficult to give interpretability an all-purpose definition since it is highly domain-specific [8]. According to Mattias Wahde et al., an important principle to apply, to achieve safe and interpretable conversational AI is to use only interpretable primitives, which aligns with Cynthia Rudin's conclusion [9].

1.2.4 Hybrid control

In the reinforcement learning based control system developed by O. Johanssons and B. Lundgren, there were problems during the initial training as the control policy deteriorated at first before eventually improving [1]. A similar problem is described by I. Koryakovskiy et. al. and J Yoo et. al. in their articles: the reinforcement learning approach causes problem in the initial stages of training as it requires several iterations of trial and error before learning the control policy, which causes problem when it could potentially damage the systems being controlled [10, 11].

I. Koryakovskiy et. al. present a solution to solve this problem in robots, where they combine reinforcement learning with model-based control [11]. They suggest a combination of nonlinear model predictive control and reinforcement learning, where the reinforcement learning is used to compensate for differences between the internal model and the real model. The model-based aspect of the algorithms improve safety, without it the reinforcement learning algorithm would make the robot explore behaviours that would for example cause it to fall down and cause irreparable damage to the robot itself [11].

Similarly J Yoo et. al. suggest combining a PD-controller or an LQR-controller with PILCO, a model-based reinforcement learning algorithm, where they let the

classic controllers (PD and LQR) complement the reinforcement learning policy [10]. Testing showed promising results regarding both control performance and convergence.

F. L. Lewis et. al. show in their article how optimal, adaptive controllers can be designed using reinforcement learning [12]. Optimal controllers usually use knowledge of the system dynamics to solve Hamilton–Jacobi–Bellman equations, which is difficult (or impossible) when the system is nonlinear. Adaptive controllers solve optimal design equations after identifying the system parameters using system identification techniques [13]. In the article it is shown that a new class of adaptive controllers can be designed using reinforcement learning. These controllers have an actor-critic structure, typical for reinforcement learning, that solves Hamilton-Jacobi-Bellman equations online, i.e. under usage, in order to learn optimal control policies without knowing the system dynamics [12].

1.2.5 The Kalman filter

In 1960, Rudolph E. Kálmán published his paper describing the Kalman filter [14]. It has since then been used in many engineering fields such as for trajectory, state and parameter estimation for control, diagnosis, data merging and signal processing, to name a few examples [15]. Famously, the Kalman filter was used in the Apollo Project to estimate the trajectories of the spacecraft to and from the moon [16]. According to Auger et al.'s review of the applications of the Kalman filter in industrial settings, the main drawback of the filter has been the computational load required. With the availability of low-cost and more elaborate processors the authors believe that the Kalman filter will likely become more commonly used in the future.

Huang Z. et al. show that an Extended Kalman Filter (EKF) can successfully estimate the dynamic states of a power system [17], using a full dynamic system model combined with Kalman filtering techniques. The authors expect that this dynamic estimation of states will transform power systems operations. The EKF has also been successful at estimating the states of a permanent magnet synchronous motor and that the implementation can be used in real-time despite the heavy computational requirements [18].

Kanieski et al. have in their paper implemented a control system for power quality conditioning devices, that includes a Kalman-filter algorithm [19]. The Kalman filter is used to generate correct references for the controller even under grid perturbations.

1.3 Purpose

The objective of this thesis is to investigate how the existing hydraulic drive PI-controller can be replaced with an adaptive controller. It will investigate how reinforcement learning can be used and combined with traditional control methods to improve adaptability and stability. Where and how is modelling of the system's dynamics necessary or useful?

Does this approach allow for more interpretability than a machine learning based controller and if so how?

1.3.1 Goals

1. Develop an adaptive controller, that can handle changes to the system over time.
2. Obtain stability and adaptivity such that the controller provides a comfortable driving experience and works for multiple scenarios.

1.3.2 Limitations

The controller has to be executable in real-time on a personal computer in order to control the physical system. The system has a fixed cycle rate which will be used for the controller developed in this thesis.

2
Theory

The following section describes keys topics that are touched upon in this thesis. Firstly the reinforcement learning algorithm PILCO will be described, followed by an introduction to adaptive control, a description of the kalman filter and its implementation and finally description of different kind of interpolation methods.

2.1 PILCO

The PILCO algorithm, "Probabilistic Inference for Learning Control", is a model-based reinforcement learning algorithm that is useful to control complex systems in continuous-time [20]. Even though it is model-based, it does not require the system dynamics to be known. The algorithm has shown to be successful in continous-time control applications [10, 11] .

The algorithms learns a probabilistic model, a Gaussian process, that describes the dynamics of the controlled system [20]. The Gaussian process is iteratively updated based on observed data from the system. This model is then used to evaluate and optimize a controller, also referred to as control policy. The algorithm is described in **Algorithm 1**.

Algorithm 1 PILCO

Initialization : Sample controller parameters θ
Apply random control signals and record data
Repeat :
    Learn probabilistic (GP) dynamics model using all data
    Repeat :
        Approximate inference for policy equation
        Gradient-based policy improvement
        Update policy parameters θ
    Until convergence; **return** θ^*
    Set $\pi \leftarrow \pi(\theta^*)$
    Apply π^* to system and record data
Until task learned

Assume that there is a dynamical system described as in equation (2.1). The system dynamics are described by the unknown function $f(x_t, u_t)$ and w_t is Gaussian noise. When using the PILCO algorithm, the system dynamics are assumed to follow a Gaussian process model, it can therefore be described by a mean and variance. This

process model is learnt by the algorithm using a training set containing previous states and corresponding control inputs.

$$x_{t+1} = f(x_t, u_t) + w_t \tag{2.1}$$

The goal of this algorithm is to find an optimal control policy $\pi(x, \theta)$ that controls the system toward a reference. This is done by optimizing the parameter θ so that it minimizes the cost function $J^{\pi}(\theta)$ as in equation (2.2). Which cost function is chosen depends on the system and application but it is here described by $l(x_t, u_t)$.

$$\min\ J^{\Pi}(\theta) = \sum_{t=0}^{T} E[l(x_t, u_t)] \tag{2.2}$$

2.2 Adaptive control

An adaptive controller will adjust its parameters over time in order to cope with changes in the physical system being controlled [13]. This could be changes in operating conditions, disturbances or changing uncertainties. Adaptive control can be very useful for ever changing and uncertain systems where designing a fixed control strategy can be difficult. Adaptive control is done based on real-time feedback from the system, often obtained from sensors or other measurements. The adjustment of control parameters can be based on a pre-defined algorithm, or it can be learned through machine learning techniques. Adaptive controllers are commonly used in many engineering applications, such as automotive control and robotics [21].

2.3 Kalman Filter

The Kalman filter is a recursive, under specific assumptions, optimal data processing algorithm [22]. Its purpose being to produce optimal state estimates of a modeled process using noisy measurement [14]. The filter considers known system dynamics, statistical descriptions of process noise, available measurements and their error, when estimating the future state [22].

Under the assumption that the state transition model can be described by linear equations and that the process and measurement noise are described by independent gaussian distributions, the Kalman filter will produce optimal state estimates [22]. These assumptions might not be commonly found in the real world, but the filter can still deliver reasonable and useful estimates. For non-linear systems, the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) can be useful tools to estimate states [23, 24].

2.3.1 Implementation in discrete time

The Kalman filter consists of two part: the prediction step and the update step. In the prediction step, the filter uses the previous estimate of the system to predict the coming (a priori) state estimate $\hat{x}_{k|k-1}$. The predicted (a priori) estimate covariance

$\hat{P}_{k|k-1}$ is also calculated. This is done according to equation (2.3) and (2.4), where F_k is the state transition model, B_k the control input model, Q_k describes the covariance of the process noise.

$$\hat{x}_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k \tag{2.3}$$

$$\hat{P}_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.4}$$

In the update step, the innovation $\tilde{y}_k$ and its covariance S_k are calculated as in equation (2.5) and (2.6), where z_k is the measurement and R_k the corresponding noise covariance . The innovation is the difference between the predicted state and the measured state. The future state $x_{k|k}$ is then estimated as in equation (2.8), using the innovation and the Kalman gain K_k. The kalman gain is calculated as in equation (2.7), it is needed to take the measurement and prediction uncertainties into account when estimating the new state.

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \tag{2.5}$$

$$S_k = H_k \hat{P}_{k|k-1} H_k^T + R_k \tag{2.6}$$

$$K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1} \tag{2.7}$$

$$x_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.8}$$

$$P_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1} \tag{2.9}$$

2.4 Interpolation

Given a set a of data points, interpolation is a mathematical method that can be used to estimates values in between the known datapoints. Interpolation can be linear, polynomial or spline, to name a few, and it can be done in several dimensions. The method of interpolation affects the properties of the estimated value and should be carefully considered to maximize the accuracy of the estimation. Extrapolation is a related concept that is used to estimate values outside of the known samples.

2.4.1 Linear interpolation

Given two known data points (x_0, y_0) and (x_1, y_1), linear interpolation will estimate points to lay on the straight line going through both data points [25]. Linear interpolation guarantees a continuous function but the first derivative may take a discrete step at the datapoints. This linear polynomial is described by equation (2.10).

$$P(x) = \frac{x_1 - x}{x_1 - x_0} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \tag{2.10}$$

2.4.2 Trilinear interpolation

Trilinear interpolation is used to estimate a function value at a point in 3D-space, that is surrounded by eight known datapoints that are corners of a cube C_{x_i,y_j,z_k}, $i, j, k \in \{0, 1\}$. See figure 2.1 for a visual representation of the interpolation point and the known data points. The value is calculated by linear interpolation in three steps, one for each dimension. In this explanation the order is x,y,z but the order does not affect the resulting value.

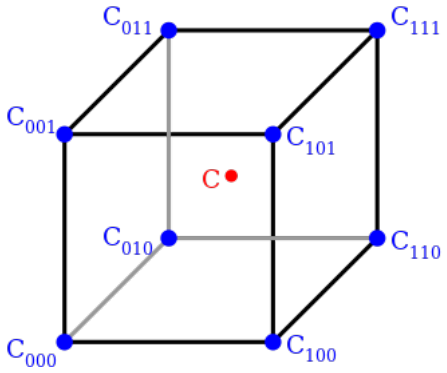


Figure 2.1: Interpolation point C (red), surrounded by known data points C_{xyz} (blue). [3]

In the first step, linear interpolation is made in the x-direction, resulting in 4 values. The second step is to linear interpolate between the y-value, resulting in two values. The third and last step is to linear interpolate between the samples on the z-axis, resulting in the desired value. A visual representation is seen in figure 2.2.

$$\text{step 1:} \qquad C_{y_j,z_k} = C_{x_0,y_j,z_k}\frac{x_1 - x}{x_1 - x_0} + C_{x_1,y_j,z_k}\frac{x - x_0}{x_1 - x_0} \qquad j, k \in \{0, 1\} \qquad (2.11)$$

$$\text{step 2:} \qquad C_{z_k} = C_{y_0,z_k}\frac{y_1 - y}{y_1 - y_0} + C_{y_1,z_k}\frac{y - y_0}{y_1 - y_0} \qquad k \in \{0, 1\} \qquad (2.12)$$

$$\text{step 3:} \qquad C = C_{z_0}\frac{z_1 - z}{z_1 - z_0} + C_{z_1}\frac{z - z_0}{z_1 - z_0} \qquad (2.13)$$

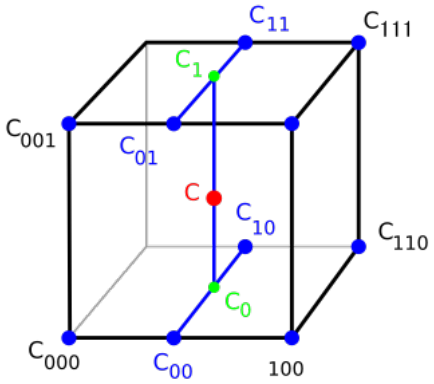


Figure 2.2: Interpolation point C (red), surrounded by known data points C_{xyz}(blue). [3]

2.4.3 Piecewise cubic hermite interpolating polynomial

Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) is a type of function used to interpolate data points. PCHIP preserves the monotonicity, which means that if the data points are increasing or decreasing, so will the function and there will be no overshoot.

The algorithm generates a spline consisting of third degree polynomials $P(x)$, defined as in equation (2.14) and (2.15).

$$P(x) = At^3 + Bt^2 + Ct + D \tag{2.14}$$

$$t = \frac{x - x_k}{x_{k+1} - x_k} \tag{2.15}$$

Between every pair of succeeding points, x_k, x_{k+1}, the polynomial has to pass through the data points while the first derivative remains continuous. However, the second derivative may be discontinuous at the sample points. $P(x)$ has to meet the conditions (2.16)-(2.20) [26].

$$P(x_k) = y_k \tag{2.16}$$

$$P(x_{k+1}) = y_{k+1} \tag{2.17}$$

$$P'(x_k) = f'_k \tag{2.18}$$

$$P'(x_{k+1}) = f'_{k+1} \tag{2.19}$$

$$\tag{2.20}$$

The parameters of the polynomials between succeeding points, see equation (2.14), are then defined as in equations (2.21)-(2.25).

$$t = \frac{x - x_k}{x_{k+1} - x_k} \tag{2.21}$$

$$A = 2f_k - 2f_{k+1} + f'_k + f'_{k+1} \tag{2.22}$$

$$B = -3f_k + 3f_{k+1} - 2f'_k - f'_{k+1} \tag{2.23}$$

$$C = f'_k \tag{2.24}$$

$$D = f_k \tag{2.25}$$

3
Methods

This section describes the methods chosen to implement and verify a controller for the hydraulic drive system investigated in this thesis. The intention is to develop an adaptive controller as wear on the system causes the current controller to become suboptimal over time.

An initial attempt was made to mathematically model the hydraulic system, since that would be a helpful tool both in a simulator and when implementing a controller. Due to lack of information about the physical components in the system, such as the size of the pistons in the pump and motor and information about leakage and loses in the system, the attempts were unsuccessful. The conclusion being that the goals of this thesis need to be achieved without a mathematical model.

A simulator is implemented to allow for easier testing, the reinforcement learning algorithm PILCO is investigated and a reinforcement learning based algorithm that uses a Kalman filter is implemented.

3.1 Simulator

A simulator is implemented to allow for testing and evaluation of the controller's performance, without having to have access to the real system. For this specific application the simulator needs to return a realistic work pressure given an engine speed, reduction ratio and a control current. This is done by using the same tables as in the current controller described in section 1.1.1. The behaviour of the control pressure is assumed to be linear between these data points, therefore trilinear interpolation is used to acquire a work pressure given the previously mentioned parameters. The acquired work pressure is filtered in a first order low pass filter in order to simulate dynamics of the physical system.

In order to get a more realistic simulation regarding the dynamics of the system, a neural network was trained on real data. The goal being to provide work pressure values given historic engine speed, reduction ratio and control current values. This simulator showed fairly good results but the training data was limited which made the simulator less reliable. The training was done using supervised learning, implemented by O. Johansson et. al. [1].

3.2 PILCO

PILCO controllers show good results on continuous-time model-free control applications [11, 10], which makes it a relevant algorithm for the application of this thesis. The system model, learned by the PILCO algorithm, can be updated based on observations of the system. This would allow for adaptivity of the controller, which is why this algorithm is initially chosen as approach to achieve the purpose of this thesis.

The PILCO algorithm is implemented by using code written by Nikitas Rontsis and Kyriakos Polymenakos [27]. It is a Python implementation of the original Matlab implementation done by Marc Deisenroth et al. in 2013. The PILCO implementation generates an optimal controller, which is applied to the simulator of the system and its performance is evaluated.

As suggested by Deisenroth et al. the saturating cost is used as cost function in the algorithm [20]. The chosen states that were deemed as relevant for the model to allow for control were engine speed, current work pressure, reduction ratio and control current. These states were chosen since they are continuously measured and they could potentially say something about the wear in the system, since the three states affect the system in different ways.

In the original implementation by N. Rontsis et al. a Mujoco environment is used as simulator to train the algorithm. Since the hydraulic system considered in this thesis cannot be described by an existing mujoco environment, the simulator is used instead. Relevant functions are written to allow for similar use to the Mujoco environment to minimize the changes to the original code.

The machine-learning based simulator is used since it describes historic dependency between states, as it takes states a few samples back as input when calculating the new work pressure state.

3.3 Controller 1 : update of controller tables

The adaptivity of the currently used controller, described in section 1.1.1, could be improved if the tables used for the feed-forward part of the controller stayed accurate over time and evolved with the changing system. In order to achieve this, a continous update of the table values is necessary. This is done by adding an updater-block to the current controller, that takes measured states a input: engine speed, reduction ratio and workpressure and controller output, and then returns updated tables to the controller. A schematic overview can be seen in figure 3.1. The controller described in this section will be referred to as "Controller 1".

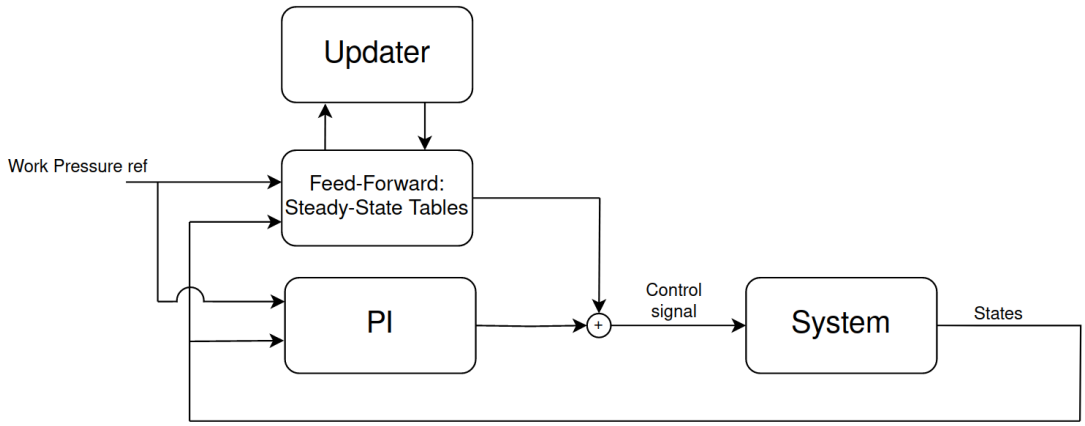


Figure 3.1: New hydraulic drive controller with updater block.

The main idea is to compare the expected control pressure, i.e. the control pressure returned from the feed-forward block, and the actual control pressure delivered from the controller. The expected control pressure is acquired by taking the measured work pressure, the engine speed driving the pump and the current reduction ratio, and then do trilinear interpolation between the known table values to get the expected value at that specific point. Note that the tables are only accurate at steady-state, therefore a guard is required to ensure that the tables are only updated based on measurements at steady-state.

To exemplify, assume that the reduction ratio and engine speed are fixed, figure 3.2 shows the relationship between work pressure and control pressure. The red dots show how work pressure and control pressure are related according to the tables, described in section 1.1.1. The green line shows the actual relationship between work and control pressure in the physical system. The purple dot shows the measured control pressure for a set engine speed, reduction ratio and measured work pressure, it appears on the green line since that is the true control pressure needed in the physical system. The purple dot show the expected control pressure value, acquired by doing linear interpolation between the two known table values. The blue dots show what the table values should be, i.e. the values that should be found by the updating algorithm. After comparing the expected and actual control pressure values (green and purple dot) the algorithm should make a decision on how to update the table values.

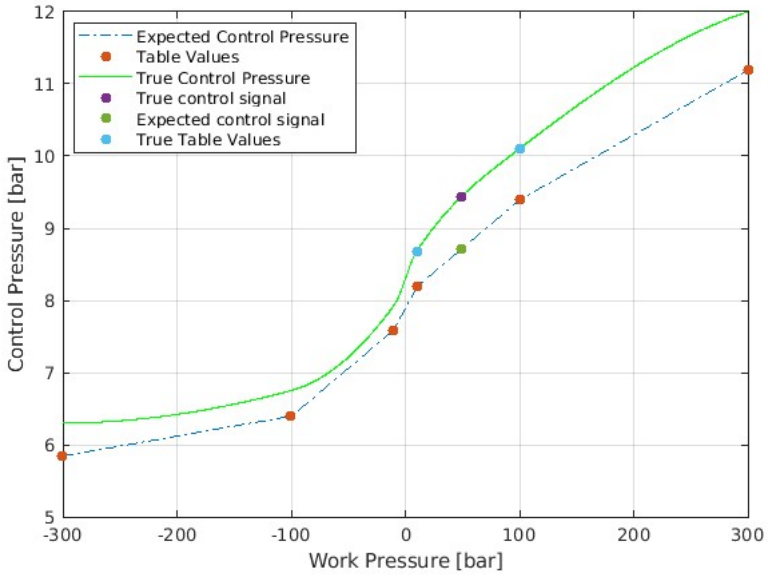


Figure 3.2: Update of table values.

3.3.1 Kalman filter for state estimation

The update of table values is done using a Kalman filter. This method is chosen because the update step of the Kalman filter take into account the state innovation and the state uncertainty, which is relevant for this application. It is desirable to use an algorithm that will update the states both with regards to the new measurement and the previous state, that also takes into account the measurement noise and uncertainty of the estimate. Kalman filters are also commonly used for state estimation. In this case it could be argued that the control pressure values found in the tables, are subject to steady-state estimation.

The Kalman filter states x are the eight control pressure values corresponding to the eight surrounding points in the tables. Figure 3.3 shows a visual representation of the Kalman states and the measurement. In figure 3.3, the blue dots represent points for which the tables contain a control pressure value. For any control pressure measurement, a red dot in the figure, there will be a known engine speed, work pressure and reduction ratio, which places the measurement in this 3D-space. Each measurement will be surrounded by 8 points, as seen in figure 3.3. These are the eight points whose control pressure value will be updated by the Kalman filter at each iteration.

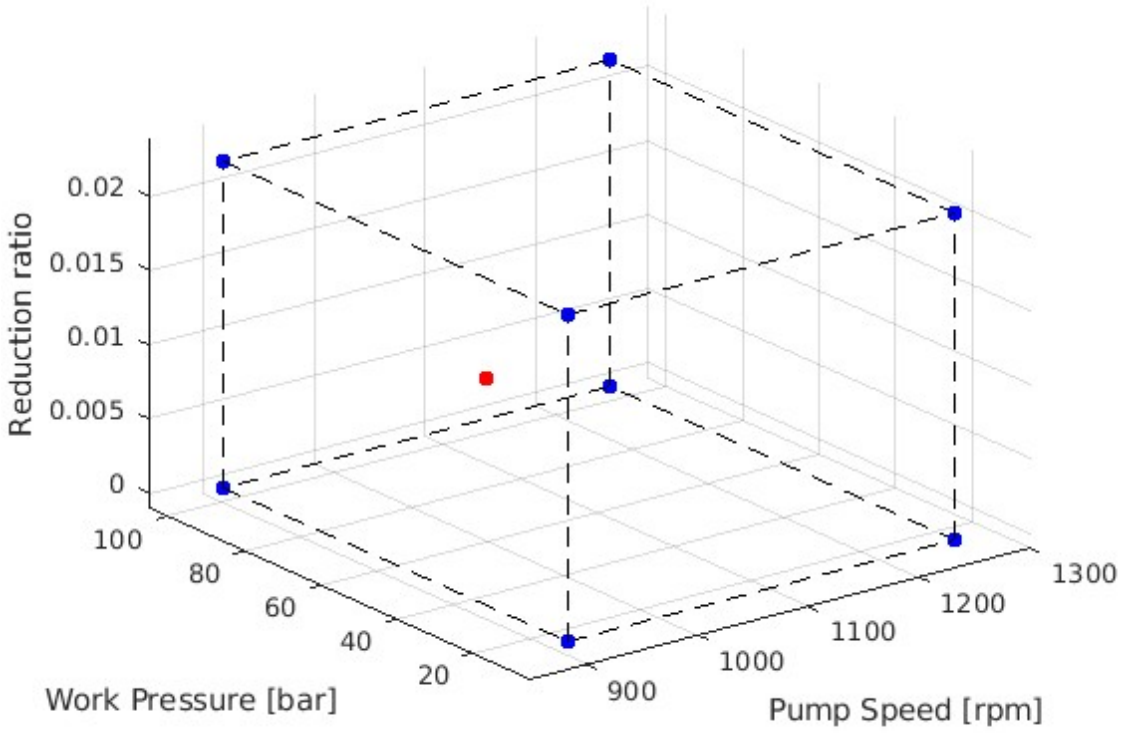


Figure 3.3: For any control pressure measurement there are 8 points surrounding it with corresponding table values.

For each Kalman iteration, the algorithm extracts the eight Kalman states from the tables. The Kalman prediction and update is done, as described in Section 2.3, then the updated states are reinserted into the tables. The covariance matrix P is handled in a similar way. A large covariance matrix is defined that contains information about the covariance of all possible states, i.e. all table values. Elements related to the specific states of the iteration are then extracted to create a smaller covariance matrix that is used for the Kalman prediciton and update. The values in the updated covariance matrix are then reinserted into the large matrix. Note that, since only the eight values closest to a measured control pressure are updated with this algorithm, the whole table will only be updated if relevant measurements, covering the whole span of the tables are acquired.

In the prediction step a naive prediction is done, meaning that the state transition matrix is an eye matrix. This means that the predicted states are assumed to be equal to the previous state. Such an assumption is possible because the physical process, the aging of the hydraulic system, is very slow so the changes are assumed to be described within the system noise. The covariance matrix P is assumed to be very low initially, i.e. it is assumed that the table values are correct. This is necessary to avoid initial problem with unreasonably large updates.

In the update step the observation model-matrix H describes the trilinear interpolation of the Kalman states. The matrix H describe how each state contributes to the expected control pressure value. In the innovation step, the component $H\hat{x}$,

i.e. the matrix multiplication between H and the predicted state, correspond to the expected control pressure at a given point. This is subtracted from the measured control pressure, z_k, to form the innovation.

The Kalman filter can only handle linear systems, which in this case causes problems when the physical system exhibits non-linear behaviours, such as when the work-pressure reference is changed. This was confirmed with the simulator, non-linear behaviour was introduced in the simulator, which led to poor results for this implementation of the Kalman filter. To handle this, a guard was introduced to ensure that the system reached steady-state before measurements were used to update table values. The guard prevents the tables to update if the work pressure, its reference, the engine speed or the reduction ratio change too much. A reasonable threshold for each parameter was obtained by looking at data from step-responses from the physical system.

3.3.1.1 Observation matrix

The observation matrix, H, used in the Kalman filter for this application describe how the eight table points surrounding the measured point contribute to the control pressure value at the measurement. The observation matrix times the states result in the expected control pressure value. The matrix is calculated using trilinear interpolation, explained in section 2.4.2, where the first axis of interpolation is the reduction ratio, the second work pressure and the third pump speed.

Any state value C_{x_i,y_i,z_i}, will contribute to the expected value along the x-, y- and z-axis with a factor γ_{x_i}, γ_{y_i} and γ_{z_i} respectively, which is calculated as in (3.1)-(3.4), where $i \in \{0, 1\}$.

$$\gamma_{x_i} = \frac{x_i - x}{x_i - x_{1-i}} \tag{3.1}$$

$$\gamma_{y_i} = \frac{y_i - y}{y_i - y_{1-i}} \tag{3.2}$$

$$\gamma_{z_i} = \frac{z_i - z}{z_i - z_{1-i}} \tag{3.3}$$

$$\tag{3.4}$$

The expected value C_{exp}, is then calculated as the sum of every state value times it contribution factors, see equation (3.5).

$$C_{exp} = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} C_{x_i,y_j,z_k}\gamma_{x_i}\gamma_{y_j}\gamma_{z_k} \tag{3.5}$$

As the table values are the Kalman filter states, the observation matrix can be written as in equation (3.6)

$$H = [\cdots,\ \gamma_{x_i}\gamma_{y_j}\gamma_{z_k},\ \cdots] \qquad i, j, k \in \{0, 1\} \tag{3.6}$$

As the states and contribution from each state varies over time, the observation matrix is therefore time dependent and requires to be recalculated in every time step.

3.3.1.2 Tuning of parameters

Kalman has two main parameters that need tuning to fit the application of the filter, the process noise covariance Q_k and the measurement noise covariance R_k. For many Kalman filter application, the measurement noise covariance can be found by looking at actual measurements and estimating how much it varies in steady-state. This is not possible in this case, because expected control pressure is not per se a measurement. It is not affected by the noise caused by a sensor, but it is still relevant to tell the filter how much it should trust the controller's control pressure compared to the transition model used to predict the future state.

Both covariance matrices are given initial guesses and then tuned while doing tests in the physical system to find the best fitting values. The process covariance needed to be relatively small compared to the noise covariance in order to avoid to large update steps of table values in an erroneous direction. The found values are found in (3.7) and (3.8). Note that the diagonal values in the Q matrix are larger because every state is assumed to mostly affect itself and that the non diagonal elements are non-zero as it can be assumed that the table values has some small covariance.

$$Q = 10^{-4} \begin{bmatrix} 101 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 101 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 101 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 101 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 101 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 101 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 101 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 101 \end{bmatrix} \tag{3.7}$$

$$R = 50 \tag{3.8}$$

3.4 Controller 2: batch

In initial testing, Controller 1 that uses a Kalman filter to update the tables, showed some undesirable behaviour where large overshoots caused large fluctuations in the table values. In order to minimize this problem the controller was altered so that the Kalman update was made on a batch of observations, instead of only one measurement. The idea being that this would lead to more careful updates of the states and therefore less jumpy tables. This altered controller is referred to a "Controller 2".

3.4.1 Extended observation matrix

In order to update the filter with a batch of data, the observation vector, z_k, is extended. As a consequence, the observation matrix H also has to be extended, as well as the innovation $\tilde{y_k}$.

$$\tilde{y_k} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \end{bmatrix} - \begin{bmatrix} H_0 \\ H_1 \\ \vdots \end{bmatrix} \hat{x}_{k|k-1} \tag{3.9}$$

Three different options regarding the implementation of this controller are presented below.

- Only extend the batch with samples that utilize the same state vector.
  - \+ Easy to implement.
  - − Does not benefit from samples around a table value, only in one octant.
- Extend the state vector to include all the control pressure values in the table.
  - \+ Benefits from samples around each and every table value.
  - − Significant overhead as the H-matrix will consist of zeros where none of the samples affect the table values.
- A dynamic state vector that only consist of relevant control pressure values from the tables.
  - \+ Less overhead than including all table values and each state can benefit from samples in the eight octants around it.
  - − Increased complexity to implement.

The second approach, extending the state vector to include all control pressure values in the tables, is chosen. Mainly because it allows for more samples to contribute to the estimation while keeping the implementation simple. Tests on the simulator show that the overhead in the observation matrix did not affect the calculation time in a manner that affected the performance of the controller. Updating with a batch size of 100 samples is feasible to maintain update frequency of the control signal.

3.4.1.1 Prediction step covariance

In the prediction step of the Kalman filter, the predicted estimate covariance is affected by the covariance of the process noise, see equation (2.4). If the covariance noise is applied to all the table values, the state covariance will over time increase infinitely for the table values that are unaffected by the innovation step. As the reference work pressure is fixed most of the time when driving, the major part of the states will not be affected by a measurement. To avoid this problem the state covariance for unaffected table values, the covariance of the process noise is zero for all values that are not affected by a measurement.

3.5 Controller 3: nonlinear interpolation

If the system exhibits highly non-linear behaviours between the data points in the tables, linear interpolation will yield poor results. This is highlighted in figure 3.4,

where the purple x's show which table values controller 1 will eventually find given the measurement. The blue dots represent the true control pressure value at those points, which do not align perfectly with the estimated values.

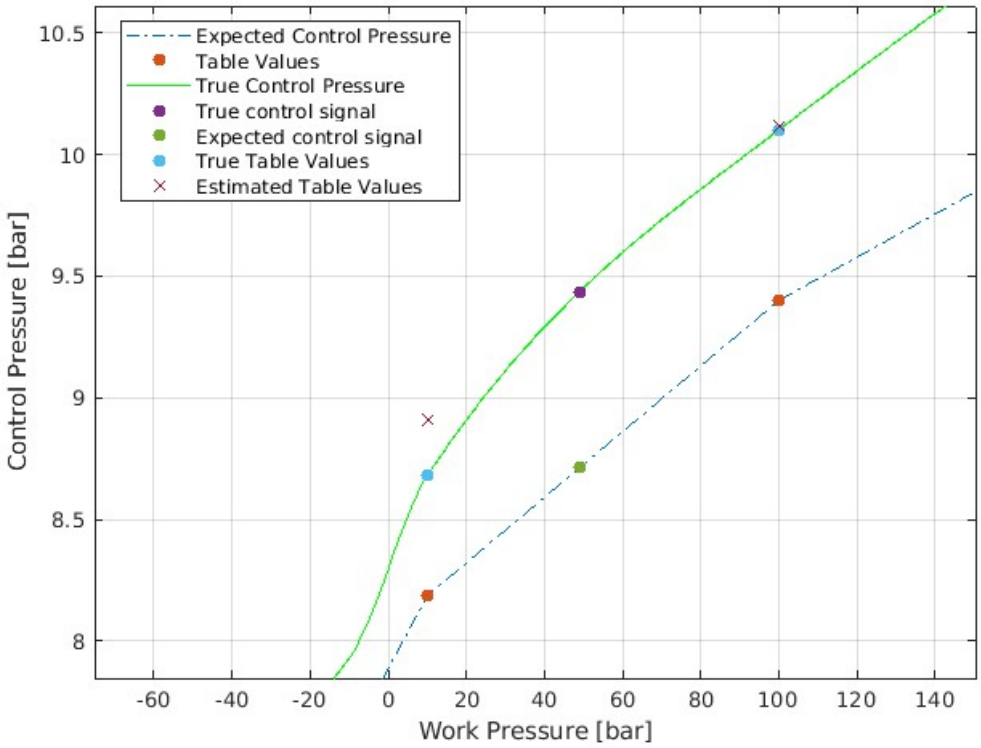


Figure 3.4: Kalman estimation of states

To solve this problem a third controller is developed that uses a higher order interpolation method than controller 1 and 2. PCHIP is concluded to be a suitable choice since it generates a reasonably smooth graph, which can be seen in figure 3.5 where linear interpolation, polynomial interpolation and PCHIP are compared. The controller described in this section, is referred to as "Controller 3".

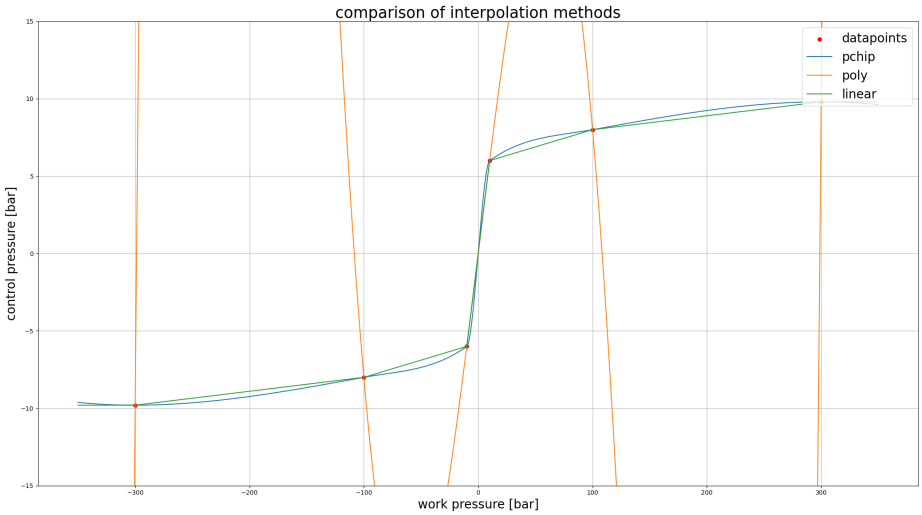


Figure 3.5: Comparison between PCHIP, linear and polynomial interpolation of the table values with zero reduction ratio and pump speed of 900 rpm

When designing a PCHIP function, the derivative in the sample points can be defined in different ways [28]. For this application the "scipy.interpolate.PchipInterpolator" is chosen as interpolator [29].

The slope d_k, between two sample points, x_k and x_{k+1} is described by $d_k = (y_{k+1} - y_k)/h_k$, where $h_k = x_{k+1} - x_k$. The derivative f'_k in a sample point x_k is then defined as the weighted harmonic mean, described in equation 3.10, where $w_1 = 2h_k + h_{k-1}$ and $w_2 = h_k + 2h_{k-1}$.

$$\frac{w_1 + w_2}{f'_k} = \frac{w_1}{d_{k-1}} + \frac{w_2}{d_k} \tag{3.10}$$

If the sign of d_k and d_{k-1} are different or either of them is equal to zero, then the derivative is described by $f'_k = 0$. At the endpoints, the derivative is set to the noncentered, shape-preserving, three-point formula, seen in equation 3.11 [30].

$$f'_0 = ((2x_0 + x_1)d_0 - x_0 d_1)/(x_0 + x_1) \tag{3.11}$$

PCHIP generates an interpolated value from one variable. The method to generate values from three variables is conceptually similar to trilinear interpolation but PCHIP interpolation is used instead of linear interpolation, thus the calculations are made in three steps. All table data points are used to generate the splines used for interpolation, compared to the linear case where only the eight enclosing points are used. As a result, the first step results in $6 \cdot 7 = 42$ points, the second in 7 points and the last step provides the desired interpolated value, C_{PCHIP}.

3.5.1 Adding PCHIP spline in observation matrix

A non-linear observation function can be used in an EKF. If PCHIP were to be used as observation function it would significantly increase the the complexity and

computational load of the algorithm, since the derivative of the observation function is needed to calculate the innovation [31]. To avoid unfeasible complex calculations, the observation matrix remains as linear interpolation but with the addition of a bias term, C_{bias}, that is the difference between the linearly interpolated control pressure ($H_k\hat{x}_{k|k-1}$) and the PCHIP interpolated control pressure, C_{PCHIP} in the desired point. The state vector is correspondingly extended with a constant 1 and the estimate covariance matrix, P, and the process noise covariance, Q, are padded with zeros to account for the new state.

$$C_{bias} = C_{PCHIP} - H_k\hat{x}_{k|k-1} \tag{3.12}$$

$$H_{spline} = [H \quad C_{bias}] \tag{3.13}$$

$$x_{spline} = \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{3.14}$$

$$P_{spline} = \begin{bmatrix} P & 0 \\ 0 & 0 \end{bmatrix} \tag{3.15}$$

$$Q_{spline} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \tag{3.16}$$

3.6 Tests in simulator

For testing of the controllers implemented in this thesis the simulator based on empirical calibration table is used. The simulator is mostly used to show proof of concepts while developing the controllers and that the communication via CAN is working properly to prepare for testing on the physical system. The following tests is preformed to investigate properties of the controllers.

1. A general performance test to compare behaviour and investigate the convergence time of the table values. A basic scenario of a few states involved is designed to analyze the behaviour in a feasible manner.
2. A test to measure calculation time for the different controllers. The test is required as the tables updates online and the calculation time must be within the given control frequency. The test is useful to deduce a reasonable batch size.

3.7 Physical testing

To investigate the performance of the controllers developed in this thesis, tests are conducted on the physical system. The first test investigates if the controllers can successfully find and update its table values, this will show if the adaptability problem has been solved. Other tests aim to show and highlight how the performance of the new controllers differ from the current controller as well as each other. This is done by looking at the size of the contribution from the PI part of the current controller compared to the contribution to the new controller.

3.7.1 Test 1: find table

The first test aims to investigate if the controllers can successfully find correct steady-state tables. A calibration script is written, going through a serie of workpressure and engine speed points with a constant time interval, to ensure steady state. Table values are continuously logged. These table values will then be compared to the reference table. This test is conducted on all three implemented controllers. The found tables are then compared to the reference table found by doing a calibration run with the current controller. The specifics of the tests are described in Appendix A.

3.7.2 Test 2: work pressure step-response

For this test the engine speed is kept constant at 800 rpm and a step is taken, up and then down, by the work pressure reference. To ensure that steady-state is reached after each step, there is 15 seconds between each step. For this test both the work pressure and the PI-controller's contribution to the control signal are measured in order to be able to evaluate the control performance.

In figure 4.5 it can be seen that for positive work pressures, all table values are very similar, while they differ much more for negative work pressures. To evaluate how the system is affected by this fact, two different work pressure step-response tests are conducted. One on positive work pressure values, where the old and up-to-date table values are very similar, and one with negative work pressure values, where tables are significantly different. The tests are referred to as "positive work pressure step response" and "negative work pressure step response", respectively. The tests are conducted on the controllers both with old and up-to-date table values.

Specific information about the test can be found in Appendix A.

3.7.3 Test 3: engine speed step-response

For this test the work pressure reference is kept constant at 100bar, the engine speed takes a step up and down and then followed by a sinus wave. The steps are unrealistic behaviours for a true driving scenario, therefor a second test including a sinus wave is done to show a more realistic driving pattern. For this test both the work pressure and the PI-controller's contribution to the control signal are measured in order to be able to evaluate the control performance.

Specific information about the test can be found in Appendix A.

4

Results

The following chapter accounts for all results found in this thesis. Starting with the results from the reinforcement learning algortihm PILCO, followed by results from tests, both in simulator and in truck, related to the three controllers using Kalman filter.

4.1 PILCO

The implementation of the PILCO algorithm got poor results. The controller generated could not successfully control against either simulator. It showed signs of trying to control the system as it converged to a work pressure value, but it was not the desired one. When trying to identify the reason it was found that many of the examples provided in the code could not be controlled successfully either. The potential reasons for this failure are further discussed in section 5.1, but these results led to this algorithm being abandoned.

4.2 Simulations of Kalman controllers

Initial testing of Controller 1 with the simulator show promising results. The controller can successfully update controller table values to the correct values expected from the simulator. This can be done even if the initial table used in the controller is far from the correct values. As long as the PI-controller can generate a correct control signal for the desired action, the table values will eventually converge to the true values. Controller 2 and 3 was also tested against the simulator and generated similar behaviour.

4.2.1 State convergence

State convergence is investigated in the simulator under different scenarios. In the first test the work pressure reference is set to 55 bar with a varying noise component, a fixed engine speed and reduction ratio. There are two table points that will be updated in this scenario, where the work pressure is 10 and 100 bar.

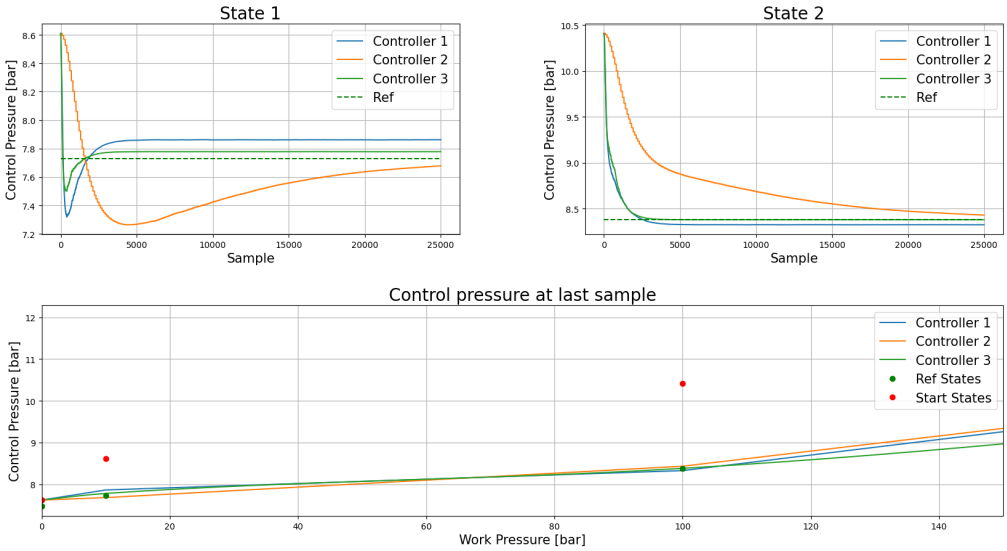


Figure 4.1: State convergence with work pressure reference at 55 bar with normal distributed noise, standard deviation = 5.67.

Figure 4.1 show how the states converge given a noise with a standard deviation of 5.67 bar, which is a reasonable value based on data from the physical system. The plots show that all controllers manage to successfully estimate the state values as they converge to the correct values. Controller 1 and 3 have similar performance while Controller 2 has a slower and more careful convergence. It is also visible that it is taking "steps" which might be a cause for problems in the real system since it will lead to discontinuous behaviour in the control signal.

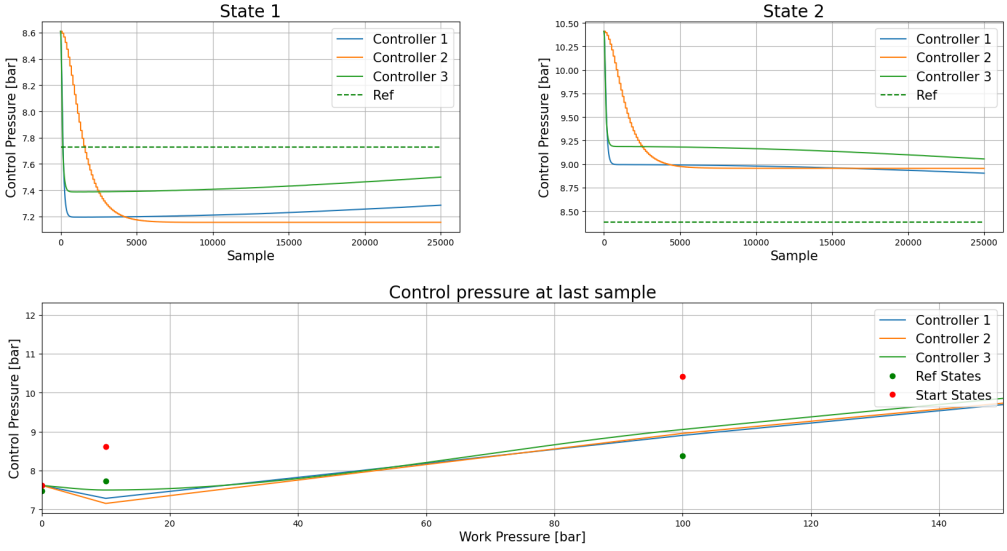


Figure 4.2: State convergence with work pressure reference at 55 bar with normal distributed noise, standard deviation = 0.1.

Figure 4.2 show the state convergence with a noise with a standard deviation of 0.1 bar. For this test, none of the controllers manage to converge to the correct values within the time span. After the initial step, where both states overshoots, the state values are considered accurate by the algorithm since the line going through the two states passes close to the measured point, which leads to very slow convergence. This is a potential problem with the chosen algorithm and interpolation methods but these two tests show that the presence of noise minimizes the issue. Is is unlikely that this becomes a problem in the physical system since there is a significant amount of noise.

Next the convergence with a varying work pressure reference is investigated. The work pressure varies back and forth between 10 and 100 bar. In figure 4.3, each step takes 10 samples and in figure 4.4 every step takes 1000 samples .

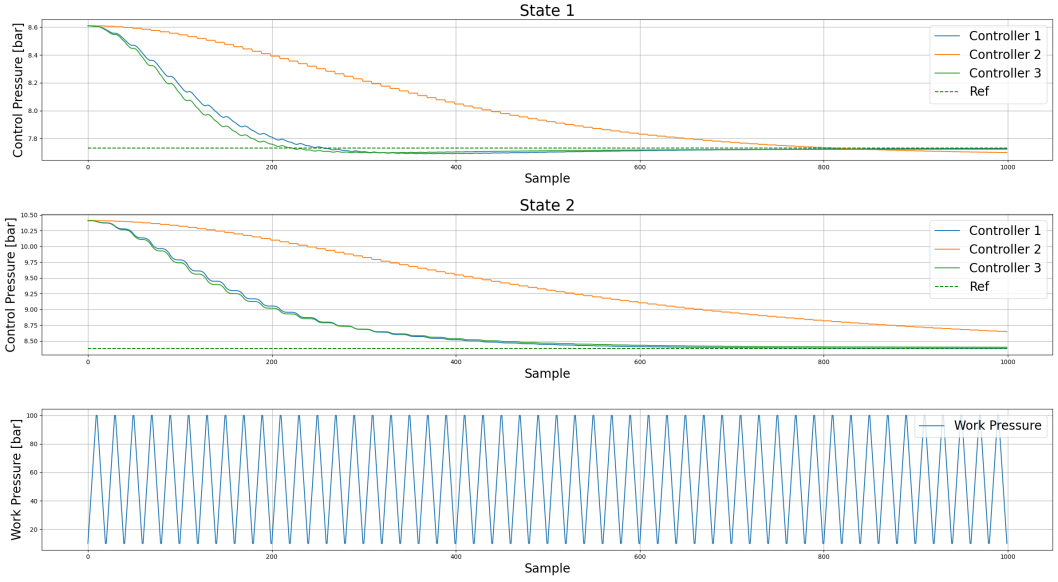


Figure 4.3: State convergence with high frequency changes in work pressure reference (10 samples/step).

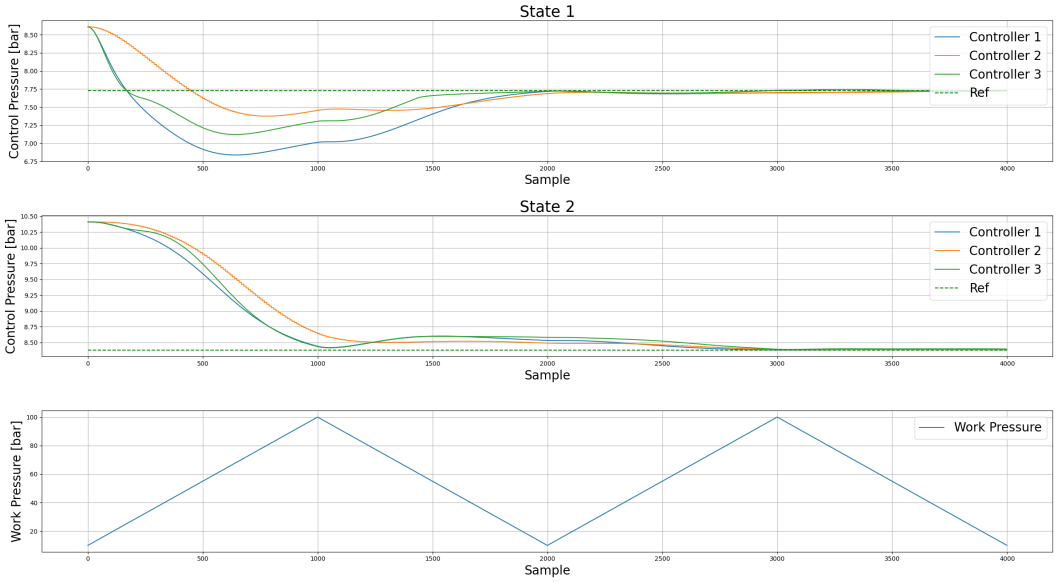


Figure 4.4: State convergence with low frequency changes in work pressure reference (1000 samples/step).

The test results, seen in figure 4.3 and 4.4, show that variations in the work pressure reference affects the convergence of a state. The convergence is faster for the test with more reference variations, seen in figure 4.3, but also more careful with less overshoot. This is expected behaviour since the difference between each sample is larger and therefor there is an even contribution to both states so they both converge at approximately the same rate.

For slower changes in work pressure, the contribution to each state is more uneven. The first 500 samples contribute mainly to changes in the first state and less so to changes in the second state, and for the next 500 to 1500 states it is the other way around. This behaviour is visible in the plots, as the first state has a high derivative for the first 500 samples, while the second state has a very slow convergence rate and then the convergence of the first state slows down while the convergence of the second state accelerates. At sample 1000, the derivative of state 1 is 0 since there is no contribution to that state at that sample.

4.2.2 Calculation time

The calculation time for each controller should be below the period time of the system cycle. Table 4.1 show the average time necessary to calculate a new update in percentage of the system cycle time, where Controller 2 has a batchsize of 100 samples.

	Controller 1	**Controller 2, batch=100**	**Controller 3**
% of cycle time	7,5	96	17,5

Table 4.1: Calculation times

The calculation time for Controller 2 varies with the batch size. The algorithm always waits to have a full batch before updating its table values, therefore a larger batch leads to less frequent updates of the table values. A batch size larger than 100 samples can not be calculated within the control frequency, indicating that larger batch sizes require another approach to successfully control the system, further discussed in Section 5.2.1.

4.3 Physical testing

The results from the tests done on the physical system are presented in this section. The test results will be used to evaluate the performance and adaptivity of the controllers and to compare them with the current controller.

4.3.1 Update of controller tables

In this test the controllers are initiated with the current tables that were acquired a few years ago through a calibration run. The same calibration run is done again at a specific reduction ratio, to see how the table have changed in these years, this new table is referred to as "reference table". In figure 4.5 the orange surface represents the initial table in the controllers, the blue surface the new reference, the green and red surface show the tables found by Controller 1 and 3 after running the test described in Appendix A.1.

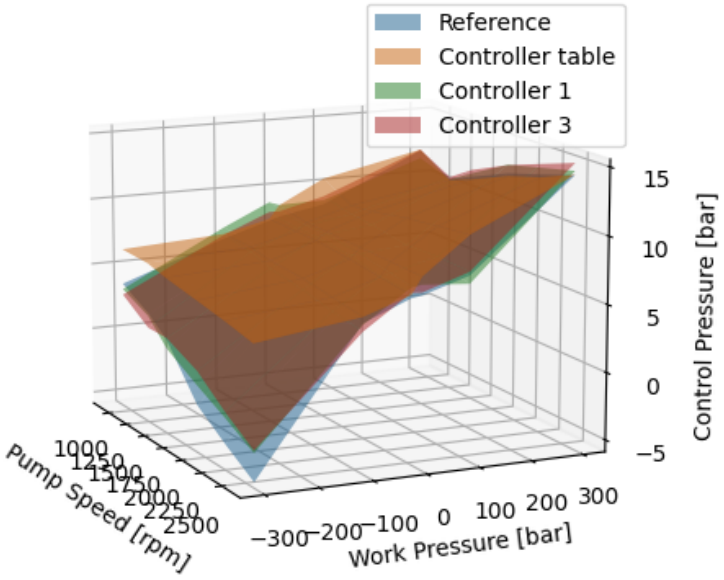


Figure 4.5: Table values found by Controller 1 and 3.

These results show that the controller can successfully find the correct reference table while running the calibration script. Both controllers successfully updated their table values to match the reference. The standard deviation from the reference table values is 0.73 bar and 0.76 bar for Controller 1 and 3 respectively.

The standard deviation could suggest that Controller 1 is more accurate at estimating the steady-state. It should be noted that the steady-state value is affected by system parameters whose effects are not evaluated in this thesis, for example the oil temperature. For this reason, these results are not enough to prove that Controller 1 always performs better estimations. Further testing in a better controlled environment is necessary to evaluate the performance of the two controllers.

For Controller 2 the control performance was poor which led to jerky driving behaviour and therefore this test could not be completed, which is why it is not evaluated.

4.3.2 Performance tests

In the following section the performance oriented test results are presented. This include the work pressure step responses and the engine speed variance test, see Appendix A.2 and A.3. All three controllers are evaluated initially but Controller 1 and 3 showed better performance and are therefore more thoroughly evaluated.

4.3.2.1 Work pressure step-response

The two work pressure step-response tests are conducted on the physical system, the tests are more thoroughly described in section 3.7.2. First the results from the positive step-response test are presented, followed by the results from the negative step-response test.

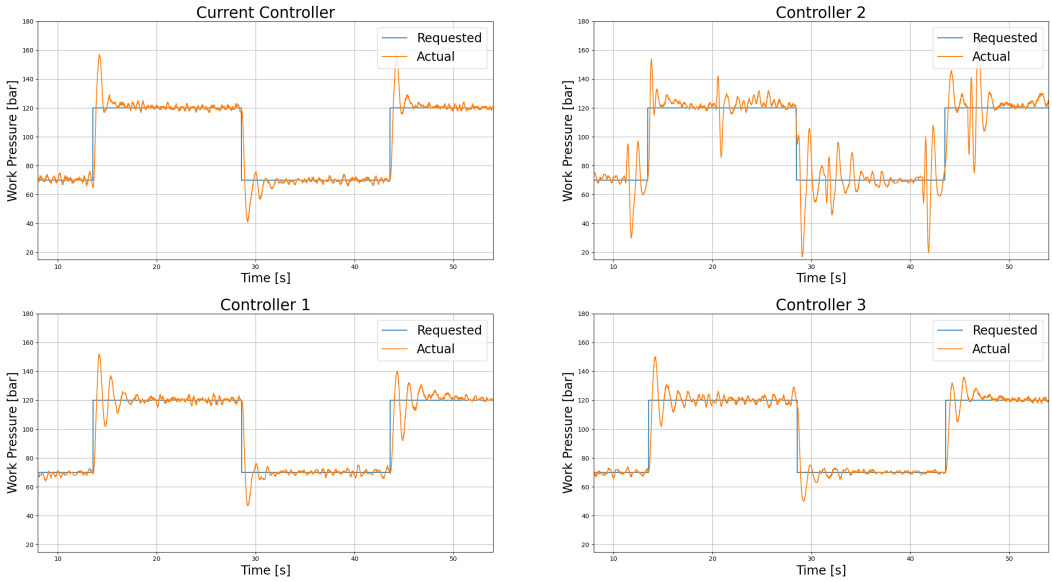


Figure 4.6: Positive work pressure step-response test results.

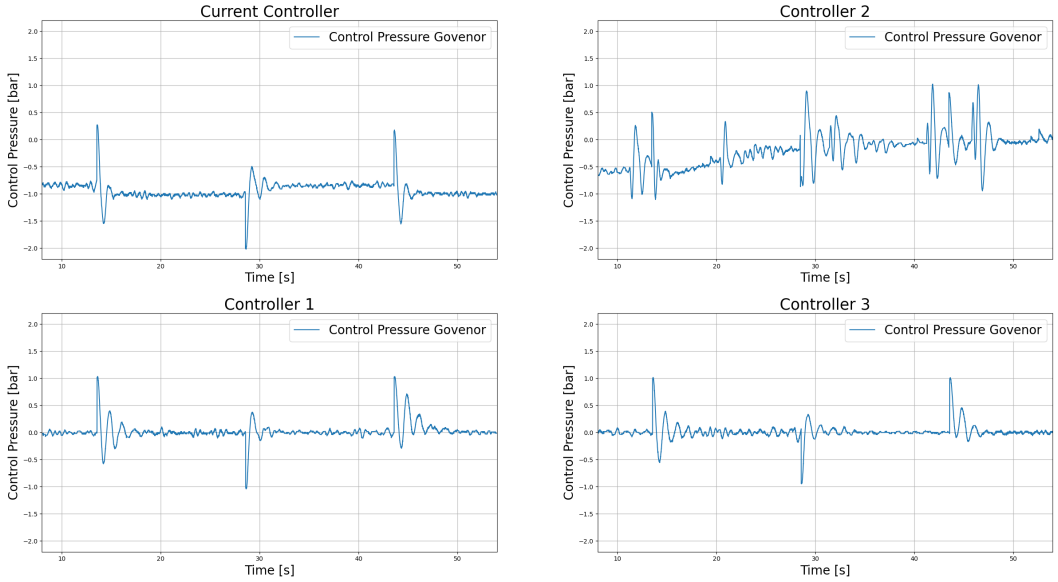


Figure 4.7: Control signal contribution from the PI-part of the controller for positive work pressure step-response test.

	Current Controller	**Controller 1**	**Controller 2**	**Controller 3**
Average error [bar]	3.05	3.50	7.29	3.15

Table 4.2: Average error for positive work pressure step-response test.

Figure 4.6 show the control performance for all four controllers initiated with the current controller tables. The average error for each controller is presented in table 4.3. The current controller had the best performance as it had the smallest average error, the reason for this is further discussed in section 5.2. Controller 1 and 3 could both control the work pressure while the driving experience remained comfortable. This was not the case for Controller 2, the driving experience was noticeably worse than for the other controllers tested. It made the vehicle behaviour jumpy and uncomfortable.

Figure 4.7 show the control current contribution from the PI-part of each controller for the test seen in figure 4.7. Note that the contribution in the current controller has a bias, which is expected since the table values less accurate which is compensated for by the PI-controller. Controller 1 and 3 have more overshoot and slower convergence time at reference steps than the current controller.

Control current in Controller 2 has an initial bias, which is likely related to the slower rate of update of the table values. It eventually converges to zero, which points towards the table values converging to their correct value. Both the control current and the work pressure are much more noisy and there are spikes in the signal that are not related to changes in reference. This will be further discussed in section 5.2.1.

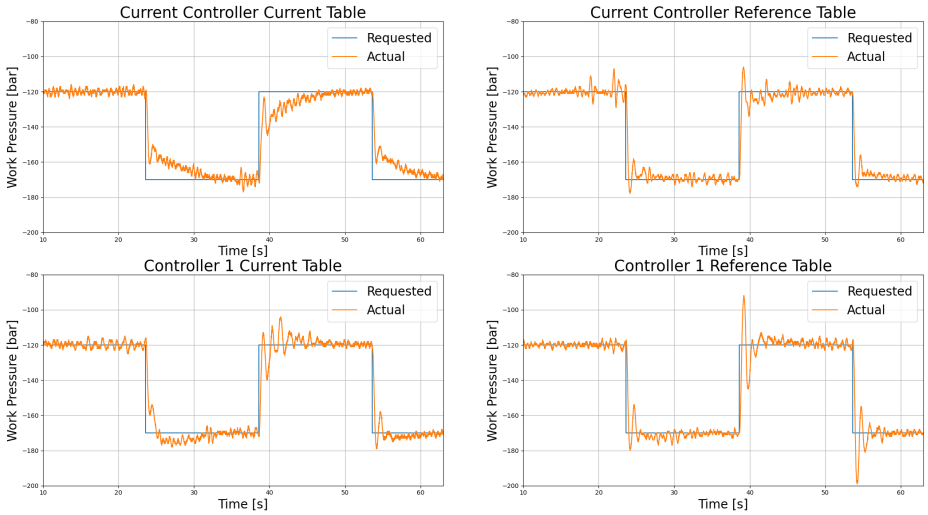


Figure 4.8: Negative step response test, comparison between current controller and Controller 1 with reference and current controller table values.

	Current Controller		**Controller 1**	
	Current tables	**Reference tables**	**Current tables**	**Reference tables**
Average error [bar]	4.87	2.68	3.14	2.87

Table 4.3: Standard deviations for negative work pressure step-response test.

Figure 4.8 show the negative step-responses for the currently used controller and Controller 1, both with current table values and the new reference table. The driving experience with all controllers is comfortable. The average error is highest for the currently used controller and lowest for the current controller with new table values. Controller 1 with the current controller's table has noticeably better performance than the current controller with the same tables while being only marginally worse than Controller 1 with reference tables. This shows that updated tables leads to better performance and that the updating of table values is fast and efficient.

These results show that updated table values are important to maintain satisfying performance of the controller, that the updating is quick enough that initial errors in table values do not have a considerable effect on the performance and finally that good performance can be achieved when using a controller that is not updated live but that has up-to-date tables.

4.3.2.2 Engine speed variance testing

The affect of changes in engine speed are investigated for all three implemented controllers. The results from the step-response test, described in section 3.7.3, can be seen in figure 4.9 and table 4.4. Controller 1 performs best as the work pressure

manages to follow the reference even when there are changes in engine speed, compared to the current controller and Controller 3 where there are disturbances when the engine speed takes a step.

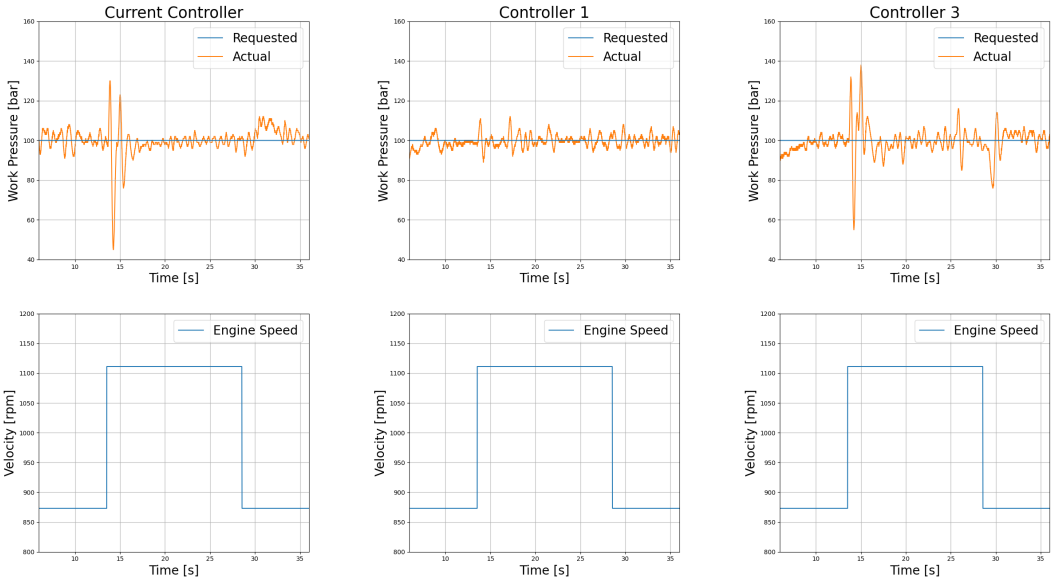


Figure 4.9: Engine speed step response.

	Current Controller	**Controller 1**	**Controller 3**
Average Error [bar]	3.97	2.45	4.60

Table 4.4: Average error from engine speed step test.

The second engine speed test, where the engine speed follows a sinus wave while the work pressure reference remains fixed is further described in section 3.7.3. The results can be seen in figure 4.10 and table 4.5. In this test Controller 3 shows the best results as the error has the smallest average error of the three controllers. The current controller did not manage to keep the work pressure at 100 bar as the acceleration changed and it is possible to see that the work pressure follow a sinus wave. Controller 1 had no bias but is very noisy and it has the largest average error. A continuously changing engine speed is a more realistic driving scenario than a step, which makes these results more relevant when evaluating which controller is better fitted for the application. These results are further discussed in section 5.2.

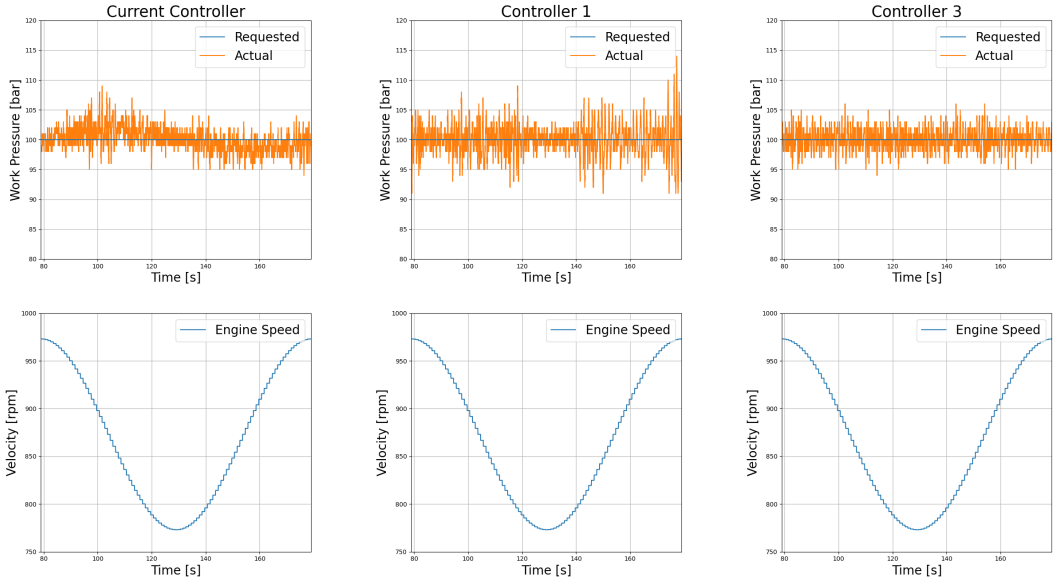


Figure 4.10: Engine speed sine response.

	Current Controller	**Controller 1**	**Controller 3**
Average error [bar]	1.56	1.99	1.33

Table 4.5: Average error from engine speed sine test.

5
Discussion

The purpose of this thesis is to implement a stable and adaptive controller for a hydraulic drive system. This resulted in three controllers that use state-estimation to deliver an appropriate control signal. This section will discuss the results, the performance of the controllers and future work.

5.1 PILCO

The poor results by the PILCO controller found were difficult to explain. The controller shows signs of trying to control the system as it converges to a value but it is not the correct one. A simple system with only two states was implemented, where the first state consists of the control signal added to the previous state and the second state is the derivative of the first state, i.e. the control signal. This system could not be successfully controlled by the algorithm.

The fact that most examples and very simple dynamics could not be controlled by the PILCO generated controller points towards some kind of implementation error in the code used. The complexity of the algorithm and code made it difficult to debug and pinpoint where the problem lays. The problem could be related either to the implementation of the cost function or rewards used to optimize the controller, that could explain why the controller seems to try to do something, even though it is unsuccessful.

The choice of states is also considered as a potential source of error, as the states chosen do include the derivatives of the states. It is potentially easier for the algorithm to learn the time dependency of the states if the derivatives are included as states. In the simple system implemented and tested the derivative is a state and it still doesn't allow for successful control meaning that if this is a problem, it is not the only one.

5.2 Controller analysis

The results from the tests on the physical testing show that Controller 1 and 3 both manage to successfully and comfortably control the vehicle while keeping their tables up-to-date. Controller 2 can control the system and can successfully update its table but the driving experience is uncomfortable and jerky and the control current exhibits strange behaviors, as can be seen in figure 4.7.

The results seen in figure 4.8 supports the idea that up-to-date table values results in better control and less requirements from the PI-part of the controller. When the system ages, the tables become less accurate and the PI-controller has to compensate for that. This means that the overall behaviour of the controller will change with time and that either the table values need to be updated or the PI parameters need to be re-calibrated to fit the system. With this new algorithm, the tables always give a true representation of the steady-state, which means that the PI-part never has to compensate for a bias and will only ever handle system dynamics.

When looking at the plots of the step-responses in figure 4.6, a slight difference in convergence time can be seen between the current controller and the Kalman-based controllers, where the later are slightly slower to converge. The differences likely depend on the tuning of the PI-part. With updated tables the feed-forward part of the controller delivers a more accurate control value, which means that the PI-controller should control the signal slightly slower than what is necessary when the feed-forward value has an offset caused by inaccurate table values. This explains why the current controller has the best control performance, the PI-parameters are tuned to fit that controller. The driving experience when using the controllers is satisfactory, so it could be argued that there is no need to remedy this, but for optimal performance it would be beneficial to tune the parameters K_p and K_i to fit the new algorithm. Even without tuning, the results presented in section 4.8 show that the new algorithm provides significantly better control of the system when the tables are inaccurate, compared to the currently used controller.

The difference in performance between Controller 1 and 3 is very limited, when looking both at simulated behaviours and testing in the physical system, the results are very similar. The purpose of Controller 3 was to investigate if a non-linear representation of the system would yield significant improvement to the state estimation. The engine speed sinus test showed slightly better results with Controller 3, with a smaller error standard deviation. The conclusion being that both controllers are well suited to control the system, but there is no significant result pointing towards one being much better suited than the other. Theoretically, it is possible that the differences in the states found by Controller 1 and 3 could be larger, especially if the non-linearities of the system become more prevalent and in that case further testing should be done to evaluate which is better suited.

The test results of Controller 1 for the engine speed step-response and the sine wave need further investigation. For step response, Controller 1 exhibits near perfect behaviour, it manages to perfectly follow the reference even though the engine speed has taken a step, while the noise remains low. In the sine test, Controller 1 is very noisy when the acceleration is large, which does not align with the test results from the other test. For this reason, it would be beneficial to do more extensive testing to determine the actual behaviour of the controller when the engine speed varies.

5.2.1 Controller 2

Unfortunately Controller 2 could not be fully evaluated because of its control performance while testing on the physical system. The spikes seen in figure 4.6 when

doing the positive work pressure step-response test resulted in jumpy and uncomfortable behaviour. It was eventually discovered that the problem was caused by the calculation time of the update step being to large, which blocked the controller so no control signal was sent to the system. When the system does not receive any control signal for more that 5 cycles, the controller already implemented in the vehicle takes over, which led to the strange behaviour. This was partly expected as the simulation results indicated that the calculation time occasionally exceeded the cycle time but since it was only marginally longer it was expected to still be calculated within the 5 cycles before the current controller takes over.

Further work and testing is needed to implement and evaluate Controller 2. One suggestion is to let the calculations be done in another process than the controller. This would allow the calculation to take the time they need while the controller itself can still send control messages every system cycle. When that is done, the tests presented in this thesis can be run again to evaluate the performance of the final implementation. The simulation results show that this controller is slower at converging at has larger overshoot, suggesting it is not relevant to look further into this controller implementation since Controller 1 and 3 have better performance.

5.2.2 Future work

In order to make this algorithm ready for implementation and use in the physical system, it should be further tested in cases where the reduction ratio varies. As of now the algorithm has only been tested with fixed reduction ratios. As the interpolation along the reduction ratio-axis is done in the same way as along the engine speed and work pressure axes, it could be argued that it should work in a similar way. On the other hand, the knowledge about how the system behaves along that axis is sparse, so it is possible that the chosen interpolation methods does not fit the real system.

This thesis has shown that as time ages the hydraulic drive system, the steady-states used in this controller have changed significantly in the last years. The algorithm implemented in this thesis provides a way of handling these changes but if the steady-states change it is also very likely that the system dynamics do as well. The system dynamics are handled by the PI-part of the controller, which at this point is static, therefore frequent tuning is necessary to maintain the system behaviour. In order to obtain a truly adaptive controller, further testing is needed to evaluate how important this part is over time and how an adaptive PI-controller can be implemented.

Another aspect that should be further investigated is the chosen data points in the tables. As of now, the data points have been chosen by CPAC because the system exhibits behaviour that is close to linear between all points. It is possible that this will change with the aging of the system and that linear interpolation between those points will not provide a good representation of the system behaviour. That would significantly reduce the performance of the algorithm and it is therefore important to investigate the matter. One way of minimizing this problem would be to populate the tables with more points. This would probably require some modification of the

algorithm to get a update behaviour that fits the application.

Lastly further work could be done regarding the handling of the P matrix outside of the Kalman filter. A guard should be inserted to catch iterations where the covariance is unusually large and stop those updates from being reinserted in the tables. This would be an extra safety measure to avoid nonsense values. In this thesis this was not implemented because the filter always delivered reasonable update values and the need for this guard never occurred.

5.2.3 Extend system knowledge

Further investigation should be done on how the tables found in this thesis can be used to gain knowledge about the system. It might be possible to draw conclusions about which kind of driving causes the most wear on the system. In a shorter time perspective it could also show changes related to other parameters such as oil temperature. If the knowledge in this area is extended it could be possible to foresee when maintenance is needed, which could avoid both unnecessary maintenance and damage caused by it not being done in time.

5.2.4 Comparison to reinforcement learning based controller

In 2021 Oskar Johansson et. al. developped a machine and reinforcement learning based controller for this system [1]. It showed promising results but had stability and convergence issues, which require further work in order for it to be ready for implementation in the real system. The main difference between that controller and the one implemented in this thesis is the implementation complexity. The machine learning based controller uses different kind of neural networks, first for supervised learning and then for reinforcement learning, RDPG, to allow for adaptivity. This approach could probably yield good results if trained properly but it requires much more work in terms of tuning, training and training data collection. It is also quite difficult to get a good understanding of how the algorithm works and where potential problems occur. The controller in this thesis on the other hand is much simpler and as a consequence it is more interpretable, making it better suited for a safety critical application. It is also easier to debug and requires less work before being ready for final implementation in a real system, while showing satisfactory control performance.

5.3 Usage of the algorithm

The algorithm developed in this thesis could be used in different ways in a physical application. The results seen in section 4.3.2.1 suggest that a controller with updated tables but without continuous updates provides the best control performance. This points towards the algorithm being used as a calibration tool, where it is used once in a while to update the tables. This would require further investigation on how often it is suitable to do a calibration run. This would provide good enough control performance but require more maintenance work on the controller.

The algorithm could also be continuously used, meaning that the table values are constantly updated while driving. The results show that continuously updated tables provide good performance and it is believed to be improved if the PI-part is tuned to fit the system. This solution would show good enough performance while requiring minimal maintenance and tuning. The fast update rate points towards this algorithm being able to handle short-term changes to the system. One such example is changes in oil temperature which affects the control of the system on a daily basis.

6
Conclusion

The goal of this thesis is to develop a controller with satisfying stability and adaptability. In order to achieve these goals a Kalman filter-based algorithm is chosen to update tables representing the steady-states of the hydraulic drive system. The chosen algorithm can successfully control the system and maintaining a good driving experience. It can be claimed that the goals of the thesis have been obtained, since the controller is stable and the ability to handle changes to the system's steady-states points towards it being adaptive.

With the goal of improving the controller, two different approaches of further development are tested. The first approach involves using several data points to update the table values. This lead to worse control performance with slower convergence and more over-shoot in simulations. It could not be fully evaluated in the physical system due to poor performance. The second approach includes using a 3rd order spline instead of linear interpolation. It showed similar performance to the original implementation but results points towards it being slightly better at following a constantly changing engine speed. Further testing is necessary to confirm these results.

The time required to update the table-values to a correct values is very low, which shows that the algorithm is also fitted to compensate for temporary changes to the system, such as changes in oil temperature. The continuously updated tables also make it possible to learn more about the aging and behaviour of the system, which could make it possible to foresee when system maintenance is needed. Further testing at a wider range of reduction ratios, both fixed and varying, is necessary to confirm the functionality before implementing it in the real system.

Bibliography

[1] O. Johansson and B. Lundgren, "Adaptive model-free control applied to truck front wheel drive," Master's thesis, Chalmers University of Technology, 2021.

[2] B. Rexroth, "Proportional pressure reducing valve, direct operated, increasing characteristic curve, type dre05sk, data sheet."

[3] Wikipedia, "Trilinear interpolation," https://en.wikipedia.org/wiki/Trilinear_interpolation, accessed: 2023-05-03.

[4] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho1, and S. Jo, "Review of machine learning methods in soft robotics," *PLOS ONE*, 2021.

[5] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning 110*, 2021.

[6] C.-C. Chang, J. Tsai, J.-H. Lin, and Y.-M. Ooi, "Autonomous driving control using the ddpg and rdpg algorithms," *Applied Sciences*, vol. 11, p. 10659, 2021.

[7] B. Dickson, "The dangers of trusting black-box machine learning," *TechTalks*, 2020.

[8] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2019.

[9] M. Wahde and M. Virgolin, "The five is: Key principles for interpretable and safe conversational ai," *Chalmers University of Technology*, 2021.

[10] J. Yoo, D. Jang, H. J. Kim, and K. H. Johansson, "Hybrid reinforcement learning control for a micro quadrotor flight," *IEEE Control Systems Letters*, vol. 5, pp. 505–510, 2020.

[11] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuška, and W. Caarls, "Model-plant mismatch compensation using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2471 – 2477, 2018.

[12] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, pp. 76 – 105, 2020.

[13] K. J. Åstrom and B. Wittenmark, *Adaptive Control*. Mineola, New York: Dover Publications, 1995.

[14] G. F. Welch, *Kalman Filter*. Cham: Springer International Publishing, 2020, pp. 1–3. [Online]. Available: https://doi.org/10.1007/978-3-030-03243-2_716-1

[15] F. Auger, M. Hilairet, J. M. Guerrero, E. Monmasson, T. Orlowska-Kowalska, and S. Katsura, "Industrial applications of the kalman filter: A review," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 12, pp. 5458–5471, 2013.

[16] M. S. Grewal and A. P. Andrews, "Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 69–78, 2010.

[17] Z. Huang, K. Schneider, and J. Nieplocha, "Feasibility studies of applying kalman filter techniques to power system dynamic state estimation," in *2007 International Power Engineering Conference (IPEC 2007)*, 2007, pp. 376–382.

[18] R. Dhaouadi, N. Mohan, and L. Norum, "Design and implementation of an extended kalman filter for the state estimation of a permanent magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 6, no. 3, pp. 491–497, 1991.

[19] J. M. Kanieski, R. Cardoso, H. Pinheiro, and H. A. Gründling, "Kalman filter-based control system for power quality conditioning devices," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5214–5227, 2013.

[20] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 37, 2015.

[21] G. Tao, *Adaptive Control, Design and Analysis*. Hoboken, New Jersey: John Wiley Sons, Inc., 2003.

[22] P. S. Maybeck, *The Kalman Filter: An Introduction to Concepts*. New York, NY: Springer New York, 1990, pp. 194–204. [Online]. Available: https://doi.org/10.1007/978-1-4613-8997-2_15

[23] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43.46, pp. 3736–3741, 2004.

[24] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[25] Wikipedia, "Linear interpolation," https://en.wikipedia.org/wiki/Linear_interpolation, accessed: 2023-05-23.

[26] F. N. Fritsch and J. Butland, "A method for constructing local monotone piecewise cubic interpolants," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, no. 2, pp. 300–304, 1984. [Online]. Available: https://doi.org/10.1137/0905021

[27] N. Rontsis and K. Polymenakos, "Github: Pilco," https://github.com/nrontsis/PILCO, accessed: 2023-05-03.

[28] MathWorks, "pchip," https://se.mathworks.com/help/matlab/ref/pchip.html, accessed: 2023-05-09.

[29] T. S. community, "scipy.interpolate.pchipinterpolator," https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.PchipInterpolator.html, accessed: 2023-05-09.

[30] MathWorks, "Interpolation," https://se.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/interp.pdf, accessed: 2023-05-09.

[31] K. Fujii, "Extended kalman filter," *Refernce Manual*, vol. 14, 2013.

A

Performance Tests

A.1 Test 1: Find Table

$$span\_wp = \begin{bmatrix} [55 & 75 & 100 & 130 & 160 & 190 \\ 220 & 250 & 280 & 300 & 280 & 250 \\ 220 & 190 & 160 & 130 & 100 & 80 \\ 50 & 20 & 10 & 0 & -10 & -40 \\ -70 & -80 & -100 & -130 & -160 & -190 \\ -220 & -250 & -280 & -300 & -280 & -250 \\ -220 & -190 & -160 & -130 & -100 & -80 \\ -50 & -20 & -10 & 0 & 10 & 20] \end{bmatrix}$$

$span\_engineSpeed = [900.0, 1250.0, 1500.0, 1900.0, 2250.0, 2600.0]/1.26$

for es in $span\_engineSpeed$ **do**
  $engine\_speed \leftarrow es$
  for wp in $span\_wp$ **do**
    $work\_pressure\_reference \leftarrow wp$
    Wait 2s;

A.2 Test 2: Work pressure step-response

Positive step-response test

$engine\_speed \leftarrow 800$
$work\_pressure\_reference \leftarrow 70$
Wait 25s
$work\_pressure\_reference \leftarrow 120$
Wait 15s
$work\_pressure\_reference \leftarrow 70$
Wait 15s
$work\_pressure\_reference \leftarrow 120$
Wait 15s
$work\_pressure\_reference \leftarrow 70$
Wait 15s

Negative step-response test

$engine\_speed \leftarrow 800$
$work\_pressure\_reference \leftarrow -120$
Wait 25s

$work\_pressure\_reference \leftarrow -170$
Wait 15s
$work\_pressure\_reference \leftarrow -120$
Wait 15s
$work\_pressure\_reference \leftarrow -170$
Wait 15s
$work\_pressure\_reference \leftarrow -120$
Wait 15s

A.3 Test 3: Engine speed step-response

$work\_pressure\_reference \leftarrow 55$
$engine\_speed \leftarrow 1100/1.26$
Wait 15s
$engine\_speed \leftarrow 1400/1.26$
Wait 15s
$engine\_speed \leftarrow 1100/1.26$
Wait 25s

$fs = 100$
$T = 1$
np.arange(0, T, 1/fs)
$f = 1$
$x = \text{np.sin}(2\text{np.pi}ft)100 + 1100/1.26$

for j in range(2) **do**
    for i in range(len(x)) **do**
        $engine\_speed \leftarrow x[i]$
        Wait 1s;

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY