Design and Optimization of Controllers for an Electro-Hydraulic System

Examensarbete utfört i reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Simon André

LiTH-ISY-EX--14/4783--SE

Linköping 2014
Design and Optimization of Controllers for an Electro-Hydraulic System

Examensarbete utfört i reglerteknik vid Tekniska högskolan vid Linköpings universitet av

Simon André

LiTH-ISY-EX--14/4783--SE

Handledare: Isak Nielsen
ISY, Linköpings universitet

Carl Stålfors
Scania CV AB

Examinator: Daniel Axehill
ISY, Linköpings universitet

Linköping, 13 juni 2014
Electro-Hydraulic (EH) systems are commonly used in the industry for applications that require high power-weight ratios and large driving forces. The EH system studied in this master thesis have recently been upgraded with new hardware components and as a part of this upgrade a new controller was requested. The system consists of a controller that computes a control signal for an electric motor. The motor drives a gear pump that generates a flow of hydraulic fluid. The flow is then directed to a cylinder. The movements of a piston in the cylinder is affected by the flow and the piston position can be measured. The measured piston position is then fed back to the controller and the control loop is complete. The system was previously controlled using a Proportional-Integral-Derivative (PID) controller and the purpose of this thesis is to compare the old controller with alternative control strategies suitable for this application. The evaluation of the controllers is based on both software and hardware simulations and results in a recommendation for final implementation of the best suited controller. The control strategies chosen for investigation are: a retuned PID controller, a PID controller with feed forward from reference, a PID based cascade controller, a Linear Quadratic (LQ) controller, and a Model Predictive Controller (MPC). To synthesize the controllers an approximate model of the system is formed and implemented in the software environment MATLAB Simulink. The model is tuned to fit recorded data and provides a decent estimation of the actual system. The proposed control strategies are then simulated and evaluated in Simulink with the model posing as the real system. These simulations resulted in the elimination of the cascade controller as a possible candidate since it proved unstable for large steps in the reference signal. The remaining four controllers were all selected for simulation on the real hardware system. Unfortunately the MPC was never successfully implemented on the hardware due to some unknown compatibility error and hence eliminated as a possible candidate. The three remaining control strategies, PID, PID with feed forward from reference and the LQ controller, were all successfully implemented and simulated on hardware. The results from the hardware simulations compared to simulations made with the old controller, as well as the results from the software simulations, were then evaluated. Depending on the purpose one of two control strategies is recommended for this application. The LQ controller achieved the best overall performance and is presented as the control strategy best suited for this application.
Abstract

Electro-Hydraulic (EH) systems are commonly used in the industry for applications that require high power-weight ratios and large driving forces. The EH system studied in this master thesis have recently been upgraded with new hardware components and as a part of this upgrade a new controller was requested. The system consists of a controller that computes a control signal for an electric motor. The motor drives a gear pump that generates a flow of hydraulic fluid. The flow is then directed to a cylinder. The movements of a piston in the cylinder is affected by the flow and the piston position can be measured. The measured piston position is then fed back to the controller and the control loop is complete.

The system was previously controlled using a Proportional-Integral-Derivative (PID) controller and the purpose of this thesis is to compare the old controller with alternative control strategies suitable for this application. The evaluation of the controllers is based on both software and hardware simulations and results in a recommendation for final implementation of the best suited controller. The control strategies chosen for investigation are: a retuned PID controller, a PID controller with feed forward from reference, a PID based cascade controller, a Linear Quadratic (LQ) controller, and a Model Predictive Controller (MPC). To synthesize the controllers an approximate model of the system is formed and implemented in the software environment MATLAB Simulink. The model is tuned to fit recorded data and provides a decent estimation of the actual system. The proposed control strategies are then simulated and evaluated in Simulink with the model posing as the real system. These simulations resulted in the elimination of the cascade controller as a possible candidate since it proved unstable for large steps in the reference signal. The remaining four controllers were all selected for simulation on the real hardware system. Unfortunately the MPC was never successfully implemented on the hardware due to some unknown compatibility error and hence eliminated as a possible candidate. The three remaining control strategies, PID, PID with feed forward from reference and the LQ controller, were all successfully implemented and simulated on hardware. The results from the hardware simulations compared to simulations made with the old controller, as well as the results from the software simulations, were then evaluated. Depending on the purpose one of two control strategies is recommended for this application. The LQ controller achieved the best overall performance and is presented as the control strategy best suited for this application.
Acknowledgments

I would like to express my gratitude to my supervisor Isak Nielsen for all his help with fast answers to my questions as well as his comments and remarks on the report. I would also like to thank my other supervisor Carl Stålfors for his quick responses and his help to solve many of the problems I have come across during this thesis. A special thanks go to Markus Svensson, Lars Broander, Mengxi Wu and Christofer Öman, for all your help, support and technical advices. I would also like to thank my mother, father, brother and sister for all your love and support throughout my life and for always being there for me. Finally I would like to express my thanks to my fiancée for putting up with me through all these years and hopefully many more to come, you are forever my source of inspiration and happiness, I love you with all my heart.

Linköping, June 2014

Simon André
# Contents

Notation ix  

1 Introduction 1  
1.1 Related Work ........................................... 1  
1.2 Purpose .................................................. 3  
1.3 System Description ....................................... 3  
1.4 Goals for the Controller .................................. 4  
1.5 Problem Formulation ..................................... 5  
1.6 Limitations ............................................... 6  
1.7 Method .................................................. 7  

2 Modeling 9  
2.1 Hardware Limitations .................................... 9  
2.2 Electric Motor ............................................ 10  
2.3 Gear Pump and Hydraulic Cylinder ....................... 13  
2.4 Combined Model .......................................... 17  
2.5 State-Space Model ....................................... 19  

3 Control Strategies 25  
3.1 Deciding on Control Strategies .......................... 25  
3.1.1 Retuned PID Controller ................................ 25  
3.1.2 Feed Forward from Reference ......................... 26  
3.1.3 Cascade PID .......................................... 28  
3.1.4 Linear Quadratic Controller ........................... 29  
3.1.5 Model Predictive Controller ........................... 30  
3.2 Controller Evaluation ..................................... 32  
3.2.1 Step Responses ........................................ 33  
3.2.2 Chirp as Input ......................................... 33  
3.2.3 Disturbances .......................................... 34  
3.2.4 Robustness Analysis ................................... 35  

4 Results 37  
4.1 Controller Performance .................................... 37
4.1.1 Step responses ........................................ 37
4.1.2 Chirp as Input ........................................ 42
4.1.3 Disturbances ........................................... 48
4.1.4 Robustness Analysis ................................. 54
4.2 Discussion ................................................. 56

5 Conclusions .................................................. 59

Bibliography ..................................................... 61
**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EH</td>
<td>Electro-Hydraulic (system)</td>
</tr>
<tr>
<td>LQ</td>
<td>Linear quadratic (controller)</td>
</tr>
<tr>
<td>MPC</td>
<td>Model predictive controller</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional, integral, differential (controller)</td>
</tr>
</tbody>
</table>
## Nomenclature

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Mean area of $A_1$ and $A_2$</td>
</tr>
<tr>
<td>$A_i$</td>
<td>Area of piston for each chamber, $i=1,2$</td>
</tr>
<tr>
<td>$C_{factor}$</td>
<td>Constant factor used in the hydraulics model</td>
</tr>
<tr>
<td>$C_{lp}$</td>
<td>Pump leakage coefficient</td>
</tr>
<tr>
<td>$C_{pos}$</td>
<td>Constant gain used in the hydraulics model</td>
</tr>
<tr>
<td>$D_p$</td>
<td>Pump displacement coefficient</td>
</tr>
<tr>
<td>$G_{cyl}$</td>
<td>Process model of hydraulics</td>
</tr>
<tr>
<td>$G_m$</td>
<td>Process model used for controller synthesis</td>
</tr>
<tr>
<td>$G_{motor}$</td>
<td>Process model of electric motor</td>
</tr>
<tr>
<td>$K_D$</td>
<td>Gain for the PID controller</td>
</tr>
<tr>
<td>$K_I$</td>
<td>Gain for the PID controller</td>
</tr>
<tr>
<td>$K_m$</td>
<td>Gain used in the electric motor model</td>
</tr>
<tr>
<td>$K_P$</td>
<td>Gain for the PID controller</td>
</tr>
<tr>
<td>$L$</td>
<td>Length of the cylinder</td>
</tr>
<tr>
<td>$M_{max}$</td>
<td>Maximum engine torque of the electric motor</td>
</tr>
<tr>
<td>$N$</td>
<td>Prediction horizon</td>
</tr>
<tr>
<td>$P_{max}$</td>
<td>Maximum load pressure of the system</td>
</tr>
<tr>
<td>$Q_i$</td>
<td>Weights used in problem formulations, $i=1,2,3$</td>
</tr>
<tr>
<td>$Q_{kalm}$</td>
<td>Dynamics weight for the observer gain synthesis</td>
</tr>
<tr>
<td>$R_{kalm}$</td>
<td>Measurement weight for the observer gain synthesis</td>
</tr>
<tr>
<td>$t_i$</td>
<td>Time constants for chirp signal</td>
</tr>
<tr>
<td>$T$</td>
<td>Time constant for three parameter model</td>
</tr>
<tr>
<td>$T_D$</td>
<td>Time constant for low pass filter</td>
</tr>
<tr>
<td>$T_L$</td>
<td>Time delay constant for three parameter model</td>
</tr>
<tr>
<td>$T_s$</td>
<td>Sample time</td>
</tr>
<tr>
<td>$V$</td>
<td>Volume of cylinder</td>
</tr>
<tr>
<td>$V_i$</td>
<td>Volume of cylinder chamber, $i=1,2$</td>
</tr>
<tr>
<td>$x_p$</td>
<td>Piston position displacement</td>
</tr>
<tr>
<td>$\dot{x}_p$</td>
<td>Piston movement speed</td>
</tr>
<tr>
<td>$y$</td>
<td>Measured piston position</td>
</tr>
<tr>
<td>$y_{dist}$</td>
<td>Disturbance on measured piston position</td>
</tr>
<tr>
<td>$y_{ref}$</td>
<td>Reference (or target) piston position</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Factor used when evaluating controllers</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Accuracy of sensor measurements</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Factor used for choosing external loads</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Rate limit of electric motor</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>Flow through hydraulic pump</td>
</tr>
<tr>
<td>$\Phi_i$</td>
<td>Flow in/out of cylinder chambers, $i=1,2$</td>
</tr>
<tr>
<td>$\psi(t)$</td>
<td>Frequency function</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Motor speed</td>
</tr>
<tr>
<td>$\omega_{max}$</td>
<td>Maximum speed of the motor</td>
</tr>
</tbody>
</table>
1

Introduction

Electro-Hydraulic (EH) systems are widely used since they offer high power-weight ratio and large driving forces. The systems are however influenced by highly nonlinear effects caused by oil viscosity, friction and leakage as mentioned in Chiang et al. [2014]. These nonlinearities, and the need for high precision control, put high demands on finding a precise model and robust controller. This thesis will mainly be focused on finding a control strategy for an Electro-Hydraulic system, hence the models used will be based on approximate linear models. A thesis running parallel to this one will focus on finding a more precise model of the same system. The expectation for these projects are that these theses can help each other achieve a precise controller and a model that provides a good representation of the actual system. The time units and value axes in the plots and figures of this thesis have been modified in order not to disclose company secrets.

1.1 Related Work

Since EH systems have been widely used in industry, substantial research have been made on the topic. For most applications a basic Proportional-Integral-Derivative (PID) controller is used to track a given reference signal, as for instance in Skarpetis and Koumboulis [2013] and Lei et al. [2011]. These controllers perform decently in terms of reference tracking but at the expense of a quite oscillatory control signal, which in turn could increase actuator wear and shorten the life expectancy of the system.

Another popular choice for controlling EH systems is to use fuzzy PID controllers, like for instance in Chiang et al. [2014] and Truong and Ahn [2009]. These controllers utilizes both the general PID controller as well as more loosely specified
controls of the type “Use high throttle when going up a hill”. In other words a fuzzy controller performs similar to as what an operator would be expected to do. In order to construct a well performing fuzzy control, an extensive knowledge of the system behavior is needed since “control laws”, like the one mentioned earlier, has to be interpreted as a specific mode with various numbers and gains. A fuzzy PID controller is similar to a parametrized controller where the controller gains are changeable and dependant on, for instance, the displacement of an object or its movement speed.

Another approach is to build the control strategy on a model and utilise this information when choosing the control signal. Creating a model that describes a system requires an extensive knowledge of the system, including the specific behavior of each included subsystem. Generally each subsystem is modeled on its own and then a full model is constructed based on the submodels and approximations, as for instance in Gnesi et al. [2013]. The constructed model can then be used when choosing the control signal by, for instance, using a linear quadratic (LQ) controller, Micheal et al. [2013], or by using a model predictive controller (MPC), Marusak and Kuntanapreeda [2011]. These type of model based controllers generally achieves a high reference tracking performance, which can be seen in the previous referrals, with the possibility to choose a smooth control signal by tuning the parameters. These advantages do however come at the expense of higher required computational power, at least for the MPC, Enqvist et al. [2010], and sensitivity to model errors, since they depend on a model to estimate future states which, as mentioned in Glad and Ljung [2004], is almost never perfect. The computational requirement might be a problem for this application where the computational power of the control unit is limited. Hence, the computational power requirement is an important aspect to keep in mind since this puts limits on the possible complexity of the controllers.

Other types of control strategies have been investigated, e.g. a sliding mode control in Chen et al. [2005]. A nonlinear backstepping controller was presented in Kaddissi et al. [2007]. A force controller utilizing quantitative feedback theory was presented in Niksefat and Sepehri [2000]. These control strategies have all presented well performing closed loop systems. They all require an extensive knowledge of the analytical descriptions of the system, hence they will not be further investigated in this report. The reports will however be used as basis for the modelling parts of this thesis. For this purpose the work presented by Alleyne and Liu [2000] will also be used since it presents a model of a similar system.

Most EH systems are controlled using a PID controller since it provides an acceptable result in most cases, with a low demand on computational power. When an accurate model of the system exists the model based controllers can utilize predictions of future states and possibly achieve a higher performance compared to controllers solely dependent on a feedback. The remainder of this thesis will focus on investigating different types of control strategies and determine which strategy that performs best in a system where accuracy and stability are key performances.
1.2 Purpose

The EH system investigated in this thesis has previously been controlled using a PID controller. The components of the system have recently been upgraded and as a part of this upgrade the effectiveness of the preconfigured PID was questioned. The goal of this thesis is hence to investigate different types of control structures, implement the most promising ones on hardware and choose a controller for final implementation. The controller that is chosen for the final implementation must satisfy the following criteria:

- it must have a small overshoot
- it must have a limited steady-state error
- it must be stable and non-oscillatory for a wide variety of loads
- it should have the fastest rise time of the proposed controllers when performing a step response, given that the previous demands are fulfilled
- it must be possible to implement on a pre-existing control unit.

Whether these criteria are achieved or not will primarily be validated using software simulations in MATLAB Simulink and verified by implementing the promising control strategies on the real hardware system.

1.3 System Description

The EH system in this application is used to control the position of a piston that is connected to a load. Figure 1.1 shows a principal outline of the system set up, whereas Figure 1.2 provides a block diagram of the system layout as well as the labels of some of the signals defined in the system. The complete set up consists of:

- a controller
- a power unit, consisting of an electric motor and a gear pump
- an asymmetrical hydraulic cylinder, including the hydraulic hoses as well as the piston and a displacement sensor.

Figure 1.4 provides a slightly more detailed description of the gear pump while Figure 1.3 shows the defined parameters of the cylinder.

The controller computes the desired number of revolutions per minute for the electric motor. The motor drives one of the gearwheels in the gear pump which means that the rotational speed of the gearwheels are the same as the speed of the motor. Furthermore the displacement of the pump generates a flow that is directly proportional to the rotational speed of the gearwheels in the pump, which then implies that the flow generated by the pump is proportional to the rotational speed of the motor. A simplified figure illustrating how the motor speed is connected to the flow can be seen in Figure 1.5. The fluid is then directed to the
Figure 1.1: Principal outline of the system configuration showing the controller, power unit, the assymetrical cylinder and the connected load.

Figure 1.2: Representative block diagram of the system. The blocks represent the components of the system and the arrows the signals. Dashed lines represent non measured signals whereas solid lines represent measured signals. The diagram shows how the blocks are connected as well as the signal that forms the connection.

cylinder via the hoses. This flow causes a pressure increase which applies a force on the piston and thereby moves it. The piston rod is connected to the load and hence the load moves in the same way as the piston. The displacement of the piston is measured by a sensor and fed back to the controller. The existing system is currently controlled by a PID controller. Although the basic PID controller provides decent results, there is still room for improvements.

1.4 Goals for the Controller

The piston position is measured in millimetres and the sensor has an accuracy of $\gamma$ mm, where $\gamma < 1$ mm. $T_s$ is the sample time for this application.

Primary goals:

- steady-state error should be less than $3\gamma$ (preferably equal to zero)
1.5 Problem Formulation

The system requires a robust and non-oscillatory control for a wide variety of loads, hence these are the primary goals for the controller. Precision is of high importance hence the controller has to be designed to limit overshoots and remove steady-state errors. A note of consideration when choosing the controller is that the existing control unit has a limited amount of memory and computational power. These limitations may challenge the implementation of more com-
Figure 1.5: An overview of the internal dependencies of the system. Blue arrows represent the flow and the green arrow is the positive rotational direction of the electric motor. This figure illustrates how the load $M$ is affected by the electric motor. The red arrow is the same leakage illustrated in Figure 1.4.

For more complex control strategies a stated optimization problem is required. Since the demands on the system require fast reference tracking while using a smooth control signal, a quadratic optimization problem penalizing these factors were chosen and the representation looks as follows:

$$
\begin{align*}
\text{minimize} & \quad \sum_{t=1}^{\infty} (C x(t) - r(t))^T Q_1 (C x(t) - r(t)) + u^T(t) Q_2 u(t) \\
\text{subject to} & \quad x(t+1) = f(x,u,t), \quad t = 0, \ldots, N-1 \\
& \quad U = [-\omega_{max}, \omega_{max}] 
\end{align*}
$$

In this minimization problem reference tracking is regarded the main priority, hence $Q_1$ is chosen to be significantly higher than $Q_2$ since oscillations in the control signal is only a secondary objective. For simpler controllers like the PID controller, a tuning process will be performed in order to find a compromise between fast control and limited overshoot.

1.6 Limitations

This thesis only stretches over a limited period of time, hence there is only room to investigate a limited amount of controllers. This time limit might also affect the tuning possibilities of the controllers, hence there might exist a set of param-
eters for the controllers presented that slightly enhances their performance.

1.7 Method

The outline of the methodology of this thesis is as follows:

- Modelling and software implementation
- Software simulation
- Testing
- Evaluation
- Hardware implementation and validation on real system
- Comparison between controllers and the pre-existing PID on the real system

The models will be fitted to recorded data and created using linear approximations combined with nonlinear elements. The software implementation is done using MATLAB Simulink. When the models and the proposed controllers have been implemented they are simulated in the software environment. Simulations are performed to evaluate their performance by comparing estimations with recorded data. The most promising controllers are then implemented on hardware using Real-Time Workshop. New hardware simulations will be recorded for each controller and form the basis for the comparison of the results with those of the pre-existing PID controller.

A description of the methodology written in pseudo code is:

Initialize models:
Find approximate models using logged data and calculations;
for each model do
  Implement the model in Simulink;
  Set tuning condition to not OK;
  while tuning condition is not OK do
    Tune parameters by software simulation;
    Validate model using logged data;
    Evaluation:
    if model seems reasonable then
      Set tuning condition to OK;
    end
  end
end
Compare models;
Choose the best model;

When a suitable model has been determined, the procedure of finding a controller is started. The method for this is very similar to when determining the
model and looks as follows.

**Initialize controllers:**
Decide on reasonable controller strategies based on the chosen model;
for each controller do
  Implement the controller in Simulink;
  Set tuning condition to not OK;
  while tuning condition is not OK do
    Tune parameters by software simulation;
    Validate controller using model and logged data;
    Evaluation:
    if controller is not improving then
      Set tuning condition to OK;
    end
  end
end
Compare controllers;
Implement the most promising on hardware;
Compare and validate versus existing PID controller;

Since another master thesis is focusing on finding an accurate model of the system, the models created in this thesis will be approximate and suitable for controller synthesis. The main objective with these approximative models is to give a reasonable description of how the system behaves in various situations. These approximate models do however possibly introduce model errors which have to be taken into consideration both when designing and validating the controllers.

The models and controllers created for this thesis will be implemented in MATLAB Simulink. In MATLAB, parameters will be tuned and the models validated. This step with software implementation and simulation is of great value in order to avoid damaging the equipment later on when implementing the solutions on hardware. The hardware implementation will be done using C-code, either written by hand or generated via Simulink.
When designing a control strategy for a system, a valid model can be valuable and even necessary when software simulations are required. Since the controller designed in this thesis is implemented on a real hardware system, it is important that the control signal is physically reasonable in order to avoid harming the hardware. The control signal is saturated in the hardware set up but an unreasonable control signal might still have a negative effect on the system. To validate that the control signal is reasonable, a decent model of the system and its components is required.

In most cases, determining a valid model of a system requires much effort, especially for systems with nonlinearities. In order to find a model the system is divided into smaller subsystems. Since the speed of the electric motor can be measured, the motor is modeled as a separate subsystem. Since no more signals except for the piston position are available, the gear pump, hydraulic hoses and cylinder will be modeled as a combined subsystem. In order to find approximate models for these subsystems, identification experiments were performed. These identification experiments consisted of various step responses by setting a target reference speed for the electric motor. The actual motor speed as well as the piston movements were then recorded and the models configured to fit the data.

### 2.1 Hardware Limitations

When designing the model of the system, some hardware limitations may need to be taken into consideration.

- $\omega_{\text{max}}$ is the maximum speed of the electric motor, which can also be interpreted as the maximum speed of the pump and thereby a maximum flow
rate through the cylinder.

- $M_{\text{max}}$ is the maximum torque of the electric motor, in other words the limited strength of the motor.

- $P_{\text{max}}$ is the maximum load pressure of the system, this puts a limit on the possible forces applicable to the piston which in turn causes a limit on maximum acceleration of the piston.

### 2.2 Electric Motor

The electric motor is pre-equipped with a controller that makes the motor track a requested speed. The input to the internal controller is the reference angular velocity and the actual velocity as output, instead of a voltage input resulting in an angular velocity. Ideally the implemented controller would make modelling of the motor redundant since the output would be a replica of the input. This is however not the case. From Figure 2.1 it is clear that the step response of the electric motor has a time delay of 1 time unit and a steady-state error of approximately 5%. The rise time of the motor is about 3 time units which is significant for this application. The figure also illustrates that there are nonlinearities to consider since the slope of the step response seems constant.

![Step response, electric motor](image)

**Figure 2.1:** Plot showing a step response for the speed of the electric motor. The plot also illustrates that the motor speed has a small steady-state error and the constant slope shows that it has a maximum acceleration rate.
2.2 Electric Motor

Figure 2.2: Plot showing a step response for the speed of the electric motor. The step response with the model gives a decent approximation of the measured data.

The motor is modeled using a three parameter model

\[ G(s) = \frac{K_p}{sT + 1} e^{-sT_L}, \quad (2.1) \]

where \( K_p \) is the gain, \( T \) is the time constant and \( T_L \) is the delay. The constants in (2.1) can be determined from the step response in Figure 2.1 as is done in Enqvist et al. [2010]. The parameters were chosen as \( K_p = 0.95 \), \( T = 2.5T_s \) and \( T_L = T_s \), where \( T_s \) is the sample time. For low rotational speeds the model (2.1), although fairly simple, accurately describes the relation between commanded and measured rotational speed. This relation can be seen in Figure 2.2.

When large changes in rotational speeds are commanded, the model predicts higher speeds than measured, see Figure 2.3. In order to describe the behavior of the electric motor also for these cases, a rate limit is added to (2.1). The need of a rate limit is probably caused by the maximum torque \( M_{max} \) mentioned in Section 2.1. This torque limit would cause a constant slope in the step response and is the same for all loads. However, when a big load is applied on the piston, the torque demand on the electric motor is higher. With the constant limit and higher load this would result in a step response with a flatter slope. This behavior could then be modeled by assuming that the rate limit is dependent on the load. The rate limit was tuned to correspond to the load disturbances expected in this work. The parameters in model (2.1) had to be reconfigured after the rate limit was added to better fit the majority of the data and were chosen as \( K_p = 0.98 \), \( T = T_s \) and \( T_L = T_s \). In the hardware setup up the rotational speed of the electric...
Figure 2.3: Plot showing a step response for the speed of the electric motor. The model estimates a much faster increase in rotational speed than measured. The model also estimates a bigger steady-state error than measured.

The model is saturated to within $\pm \omega_{max}$. With the added saturation and rate limit the modeled rotational speed can now be described by

\[
\omega(t) = f_s(G(p)h_r(\omega_{ref}(t)))
\]

\[
h_r(\omega(t)) = \begin{cases} 
\omega(t - T_s) - \sigma, & \dot{\omega}(t) < -\sigma, \\
\omega(t - T_s) + \sigma, & \dot{\omega}(t) > \sigma, \\
\omega(t), & \text{otherwise}
\end{cases}
\]

\[
f_s(\omega(t)) = \begin{cases} 
-\omega_{max}, & \omega(t) < -\omega_{max}, \\
\omega_{max}, & \omega(t) > \omega_{max}, \\
\omega(t), & \text{otherwise}
\end{cases}
\]

where $G(p)$ is the three parameter model (2.1), $\omega_{ref}(s)$ is the requested rotational speed, $\omega_{max}$ is the maximum rotational speed, $T_s$ is the sample time and $\sigma$ is the rate limit. $h_r$ here defines the rate limit function and $f_s$ defines the saturation. Combined with the rate limit the model accurately describes the input-output relation of the modeled electric motor, and can be used for controller synthesis.

In Figure 2.4 a validation of the model is presented. The output of the model is plotted together with the measured output for a step change in the reference signal. From the figure it is seen that the proposed model accurately describes the relation between the input and output of the motor. The steady-state error used for the new model fits most of the step sizes larger than one, which are the most
common step sizes. For steps smaller than one the model estimates a slightly smaller steady-state error than measured. Before implementation in Simulink the model was transformed to a discrete time model, with sample time $T_s$, using the MATLAB command `c2d` with the zero order hold option.

![Step response, electric motor model with rate limit](image)

**Figure 2.4:** Plot showing a step response for the speed of the electric motor. It illustrates the resemblance between the model estimations and the recorded data. Here a rate limiter was used to better model the behavior of the electric motor.

### 2.3 Gear Pump and Hydraulic Cylinder

This subsystem will be based on the measured speed of the motor (input) and the piston position (output). It will be harder to find an appropriate model for this system since it is impossible to get a clean step response with the measured motor speed. Since a step response will not be an option another approach has to be considered. This identification could for instance be done by using white noise as input and fit the model to the frequency response Glad and Ljung [2004]. In this thesis a more analytical way of modeling is attempted.

The flow through a gear pump, like the one used in this system, can be calculated as done in Chiang et al. [2014]:

$$
\Phi(t) = D_p \omega(t) - C_{lp} P_L(t) \tag{2.3}
$$

where $\Phi$ is the flow through the cylinder, $D_p$ is the displacement constant for the pump, $\omega$ is the rotational speed of the motor, $C_{lp}$ is a leakage factor and $P_L$ is
the load pressure. To calculate a simple approximation of the flow one can make
the assumption that there is no leakage, which in turn makes the flow linearly
dependent on the motor speed. This approximation is in most cases reasonable
since \( C_{lp} \rho_L(t) \) usually is much smaller than \( D_p \omega_p \).

The flow \( \Phi \) has the unit volume per second and can hence be related to the piston
movements as

\[
\Phi(t) \approx A \dot{x}(t)
\]

where \( A \) is the area of the piston and \( \dot{x} \) its movement. By rearranging (2.4) the
piston position \( x(t) \) can be described as

\[
x(t) \approx \int_0^t \frac{\Phi(\tau)}{A} d\tau.
\]

The model then takes the shape of a Ziegler Nichols model and look as follows:

\[
Y(s) = b e^{-sT_L} \Phi(s)
\]

where \( Y(s) \) is the piston position, \( b \) represents the unknown gain, \( T_L \) is the delay
and \( \Phi(s) \) is the flow. By combining (2.3), (2.6) and the approximation \( C_{lp} = 0 \), the
model is:

\[
Y(s) = \frac{C_{pos}}{s} e^{-sT_L} \Omega(s)
\]

where \( C_{pos} \) has to be tuned to fit validation data.

The fairly simple model (2.7) appears to describe the system remarkably well. A
plot displaying the resemblance to the real system can be seen in Figure 2.5. By
looking at the figure it is clear that the hydraulics can not be fully described by
this linear system since the gain \( C_{pos} \) seems dependent on the direction of which
the piston is moving. The gains dependency on the direction comes from the fact
that the cylinder used in the system is asymmetrical, whereas the different areas
for each side of the piston affects the flow in and out of the cylinder differently.
\( C_{pos} \) has been tuned for movement in positive direction and Figure 2.5 implies
that the gain when moving in the opposite negative direction, should be about
25% larger, which coincides quite well with the ratio \( \frac{A_1}{A_2} \approx 1.25 \).
2.3 Gear Pump and Hydraulic Cylinder

Figure 2.5: Plot showing the relation between modeled and measured piston position where the measured motor speed is the input for the model. It clearly shows that the model estimates positive slopes very well and seems to lack the capability to give a good estimation of negative slopes.

The model was modified by multiplying the gain with the increased factor whenever the rotational speed of the motor was negative. The increase gain was then tuned in Simulink to give a good approximation. This simple change resulted in a well behaving model of the hydraulics, which can be seen by looking at Figure 2.6. It is clear that for the validation data the model describes the actual system well. For data stretching over a longer time period the estimation slightly deviates but the principal behavior of the system is still accurate. The behavior for a longer simulation can be seen in Figure 2.7. The reason for the deviations at time 1100 is unclear. Possible explanations are that the pure integration approximation is too naive and that there are other factors that has to be accounted for, like for instance friction, leakage, inertia, dead bands and fluid transport delays. These factors have not been further investigated. They were however forwarded to the parallel master thesis since that thesis’ main purpose is to find a high performing model of the system. Another possible explanation is that at time 1100 the used reference motor speed takes the shape of a stair, which can be seen in Figure 2.8. This means that multiple steps are made subsequently which results in multiple approximation errors being added together, hence the model estimations deviates from the measured signal.
**Figure 2.6:** Plot showing the relation between modeled and measured piston position where the measured motor speed is the input to the model. For this model the gain is different depending on the movement direction making the model able to estimate both positive and negative slopes.

**Figure 2.7:** Plot showing the relation between modeled and measured piston position when doing a long simulation. For the first half of the simulation the model estimates are remarkably similar to the measured positions. For the other half of the simulation deviations between estimates and measurements start to appear.
2.4 Combined Model

To validate the combined model the two models were connected in series and simulated. The model gave a good estimation of the piston position and a comparison for a long simulation can be seen in Figure 2.9. From the figure it is clear that the model is not perfect, but still provides good results considering the simplicity of the model. Some differences can be seen when comparing Figure 2.7 and 2.9. The main differences between these simulations are that the model estimations are less accurate in Figure 2.9 and the deviations around time 1100 are bigger. The reason for these differences is that the model of the pump and cylinder is now driven by the simulated motor speed instead of the real measured speed. A figure showing the differences between the modeled motor speed and the measured for the same data as in Figure 2.9 can be seen in Figure 2.10. This figure illustrates that the modeled motor speed is not perfect and is a probable cause of the differences in Figure 2.9. For instance, the measured overshoots around zero would have the effect of slowing down the piston movement speed faster than what is simulated, whereas the parts where the simulated speed is higher than the actual speed will cause a result where the simulated piston movement is faster than the measured. Figure 2.11 shows a close up on the control signal before the increased deviations around the time 1100. From this figure it is clear that the modeled motor speed differs from the measured speed and is a probable cause of the increased deviations. Another difference between the modeled speed and the measured, seen in both Figure 2.10 and Figure 2.11, is that the measured speed appears to take longer time to reach the reference after a zero crossing. This is probably caused by unmodeled inertia that would counteract the changes that require a reversed rotational speed. These model errors increases the deviations from the actual system but are considered small enough to use the model for controller synthesis and validation. This combined model will therefore later
be used when tuning and comparing the different control strategies.

Figure 2.9: Plot showing the combined model estimation and the measured piston position when doing a long simulation. The estimations for the first half of the simulation are very accurate. For the other half of the simulation some clearly noticeable deviations appear.

Figure 2.10: Plot illustrating the simulated and measured motor speeds when doing a long simulation. It also makes it clear that the model is not perfect since there are obvious differences between the estimations and measurements. The stairs around 1000 are a probable cause of the deviation shown in Figure 2.9 around the time 1100.
2.5 State-Space Model

Some model based controllers require a state-space model. In this section, an approximate linear state-space model on the form

\[
\dot{x}(t) = Ax(t) + Bu(t), \quad (2.8a)
\]
\[
y(t) =Cx(t), \quad (2.8b)
\]

is determined. Since the combined process model presented in the previous section achieved good results it will form the basis from which the state-space model is formed. A block diagram showing the time continuous representation of the combined process model can be seen in Figure 2.12. In this block diagram the gains \(K_1\) and \(K_2\) represents the gains \(C_{pos}\) and \(C_{pos} C_{factor}\) used in Section 2.3.

The rate limit is hard to approximate using linear methods and is therefore neglected. This approximation will cause the model to estimate a faster rise time than that of the real system. One of the primary demands on the controller is that there must only be a very limited overshoot. If a controller predicts a slower movement than that of a real system, it will apply a large control signal to compensate for the slow response. The large control signal applied to a faster system will then cause an overshoot. Hence it is preferable to estimate a faster response than a slower since this would counteract an overshoot. This means that the state-space model will differ a bit in estimation compared to the combined process model. The saturation is moved before the three parameter model instead of after which causes the saturation to affect the control signal instead of an in-

![Figure 2.11: Plot showing a close up around the time 1000 on the control signal used during a long simulation. It is clear that there are a few differences between the model estimations and the measurements.](image)
ternal state. The moved saturation will however have a slight effect on the results. The previous location of the saturation would abruptly limit $\omega(t)$ to within $\pm \omega_{max}$. The new location of the saturation will limit $\omega_{ref}(t)$ instead and hence the dynamics of the filter will be noticeable even for the reference values that previously exceeded the limits. Since the time constant of the filter is very small this will however not have a significant impact on the results. The three parameter model is then separated into a filter and a time delay. With this separation the time delay can be handled separately. The new block diagram of the continuous time process model now looks as in Figure 2.13.

$\text{Figure 2.12: Block diagram showing a representation of the combined model in continuous time. The three blocks to the left represent the model of the electric motor while the other blocks represent the hydraulics model.}$

$\text{Figure 2.13: Block diagram showing a representation of the combined model in continuous time after the rate limit has been removed and the saturation moved.}$

$\text{Figure 2.14: Block diagram showing the assigned state variables in the simplified continuous time process model.}$
By assigning state variables to signals between the various blocks in Figure 2.13 a state-space model can be achieved. A new block diagram with the inserted state variables and the control signal can be seen in Figure 2.14. From the block diagram it is now possible to define the equations for the model

\[ X_1(s) = \frac{K_m}{sT + 1} U(s), \quad (2.9a) \]
\[ X_2(s) = e^{-sT_L} X_1(s), \quad (2.9b) \]
\[ X_3(s) = \begin{cases} \frac{K_1}{s} X_2(s), & X_2(s) \geq 0, \\ \frac{K_2}{s} X_2(s), & X_2(s) < 0 \end{cases}, \quad (2.9c) \]
\[ Y(s) = x_3(s). \quad (2.9d) \]

The interpretation of these dependencies is that \( x_1 \) is the control signal \( u \) after a low pass filter, \( x_2 \) is the same as \( x_1 \) but with a time delay of \( T_L \) seconds and \( x_3 \) is the integration of \( x_2 \) multiplied by a constant. The two states \( x_2 \) and \( x_3 \) are of special interest since these states have a clear connection to the actual speed of the electric motor (\( x_2 \approx \omega \)) and the measured piston position (\( x_3 = y \)).

The time delay presented in (2.9b) is somewhat problematic since it is hard to express in a linear state-space model. In order to get a linear state-space model this delay is approximated with

\[ e^{-sT_L} \approx \frac{1 - sT_L/2}{1 + sT_L/2}. \quad (2.10) \]

This approximation as well as the comprehension that the only difference between the two cases in equation (2.9c) is whether the constant \( K_1 \) or \( K_2 \) is used, allows for two separate linear state space models to be formed. The model used will then be dependent on the sign of \( x_2 \). This now results in the following representation:

\[ X_1(s) = \frac{K_m}{sT + 1} U(s), \quad (2.11a) \]
\[ X_2(s) = \frac{1 - sT_L/2}{1 + sT_L/2} X_1(s), \quad (2.11b) \]
\[ X_3(s) = \frac{K}{s} X_2(s), \quad (2.11c) \]
\[ Y(s) = X_3(s). \quad (2.11d) \]
\[ K = \begin{cases} K_1, & X_2(s) \geq 0 \\ K_2, & X_2(s) < 0 \end{cases} \]  \hspace{1cm} (2.12)

where the constants \( K_1 \) and \( K_2 \) can here be related to the factors \( C_{\text{pos}} \) and \( C_{\text{pos}} C_{\text{factor}} \) used in the combined process model in Section 2.4. Transforming (2.11) to the time domain results in a state-space model:

\[
\dot{x}(t) = \begin{bmatrix} -\frac{1}{T} & 0 & 0 \\ -\frac{2}{T} (1 + \frac{T_L}{2T}) & 0 & 0 \\ 0 & K & 0 \end{bmatrix} x(t) + \begin{bmatrix} \frac{K_m}{T} \\ -\frac{K_m}{T} \\ 0 \end{bmatrix} u(t) \hspace{1cm} (2.13a)
\]

\[
y(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(t) \hspace{1cm} (2.13b)
\]

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^T \hspace{1cm} (2.13c)
\]

which is now on the form presented in (2.8). The constant \( K \) is defined by (2.12). This state-space model was transformed to a discrete time model using the MATLAB command \texttt{c2d} with the zero order hold option. A plot from a simulation of the presented model, using the same data as when validating the process model, can be seen in Figure 2.15. This figure shows that the performance is slightly inferior than that of the process model. The same deviations that appeared in the combined process model can be seen here and on top of that, because of the neglected rate limit, the flanks are steeper. This results in a state-space model with slightly lower simulation performance compared to that of the combined process model presented in Section 2.4.

Due to the deviations and the fact that only one state can be observed, an observer is necessary to keep track of the states. This observer will be based on (2.13). The generalized equations representing an observer looks as follows, Glad and Ljung [2006],

\[
\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + K(y(t) - C\hat{x}(t)) = (A - KC)\hat{x}(t) + Bu(t) + Ky(t) \hspace{1cm} (2.14)
\]

where \( \hat{x} \) are the estimated states and \( K \) is the observer gain matrix. This observer was chosen as a Kalman filter and hence the observer gain was determined by using the MATLAB function \texttt{kalman}, with the discrete time version of model (2.13) as argument as well as the weights \( Q_{\text{kalman}} \) and \( R_{\text{kalman}} \). These weights are used to determine the Kalman filter and can be used as tuning constants were \( Q_{\text{kalman}} \) is applied to the model and \( R_{\text{kalman}} \) is applied to the measurement. The ratio of the weights determine whether the filter will primarily use the model
estimations or the measurements to estimate the next states. For instance, a high value on $Q_{\text{kalman}}$ and a low value on $R_{\text{kalman}}$ means that the measurements are more trusted than the model estimations. The weights were for this application defined as

$$Q_{\text{kalman}} = 10^2, \quad R_{\text{kalman}} = 10^{-2}. \quad (2.15)$$

With these weights the observer relies a great deal on measurements and only a small part on the model. The close relation between the observed states and the measurements was necessary in order to be able to fulfil the requirement of a limited overshoot. The state-space model with the observer is now able to track the states more accurately than without an observer. The results from a simulation using the same data as for the previous validations can be seen in Figure 2.16. In Figure 2.17 the observer estimations of the two remaining states can be seen. This figure makes it clear that the two states are very similar to the control signal used.
Figure 2.16: Plot showing the tracking performance of the state-space model using an observer. The model estimations are very similar to the measurements.

Figure 2.17: Plot showing the observer estimations of the two states not measured. Also in the figure is the control signal used. It is clear that both of the observed states are very similar to the control signal.
3

Control Strategies

This chapter will describe the control strategies that were investigated and why they were chosen. Some explanations on the theory behind the controllers will be provided as well as a procedure to follow for evaluation.

3.1 Deciding on Control Strategies

Since the system has previously been controlled using a basic PID-controller, this will be used as a reference controller. The other controllers will be compared to the old PID and hence the advantages and disadvantages of other controllers can be compared and discussed.

3.1.1 Retuned PID Controller

The pre-installed PID controller generates a quite oscillatory piston position. It has however proven to be able to handle large quantities of situations and various loads. A retuned PID will be implemented to see if the performance of the controller can be enhanced. A block diagram showing the implementation of the PID controller can be seen in Figure 3.1.

As can be seen in the block diagram the PID controller uses the control error, which is the difference between reference and measurement, as input and gives a reference value for the electric motor as output. The PID basically consists of three parts, the proportional, the integral and the derivative parts. The proportional part is a gain multiplied with the control error, the integral part is a gain times the integrated value of the control error and the derivative part is a gain times the derivative of the control error. These parts are then added together which then forms the control signal for the electric motor. The equation of the
controller is:

\[ u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{d}{dt}e(t) \]  

(3.1)

where \( u(t) \) in this case is \( \omega_{ref}(t) \), \( e(t) = y_{ref}(t) - y(t) \) and the parameters \( K_P, K_I, K_D \) are the gains for the respective parts of the controller.

These parameters will initially be tuned by performing a self pulsation experiment using a relay and thereafter fine tuned by hand with the goal of finding a good compromise between speed and limited overshoot. The control error \( e(t) \) might contain measurement noise, hence it is passed through a low-pass filter with time constant \( T_D \) before derivation to avoid differentiating a high frequency signal. In order to avoid controller wind-up conditional integration is used Enqvist et al. [2010]. Hence the integral contribution is not updated when \( |u| > \omega_{max} \).

### 3.1.2 Feed Forward from Reference

Since one of the most important requirements for the controller is high performance reference tracking, a PID controller with an added feed-forward from reference signal will be used. This customization should improve the tracking results while still keeping the advantages of handling a variety of disturbances with a PID controller. The feed forward link requires a decent model of the system for this strategy to be able to improve the tracking performance. A block diagram showing the implementation of the feed forward controller can be seen in Fig-

![Block diagram showing the implementation of the PID controller. The block marked F is the controller while G\(_\text{motor}\) and G\(_\text{cyl}\) are the two process models presented in Chapter 2.](image-url)
3.1 Deciding on Control Strategies

Figure 3.2: Block diagram showing the implementation of the PID controller with a feed forward from reference. The block marked $F_1$ is the feed forward from reference and the block marked $F_2$ is a PID controller. $G_{motor}$ and $G_{cyl}$ are the two process models presented in Chapter 2. $G_m$ is a process model with similar behavior to that of the combination of $G_{motor}$ and $G_{cyl}$.

The block marked $F_2$ in Figure 3.2 is a PID controller defined in the same way as (3.1). The purpose of the feed forward link $F_1$ is to give a smooth and accurate reference tracking. $G_{motor}$ and $G_{cyl}$ are the two process models presented in Chapter 2 whereas $G_m$ is a model with the desired characteristics of the closed loop system. In order to choose $F_2$ it is necessary to write down the closed loop transfer function $G_c$ for the control loop. Based on the block diagram in Figure 3.2 $G_c(s)$ is given by

$$G_c(s) = \frac{G_{motor}(s)G_{cyl}(s)(F_2(s)G_m(s) + F_1(s))}{1 + F_2(s)G_{motor}(s)G_{cyl}(s)} = G_m(s) + \frac{F_1(s)G_{motor}(s)G_{cyl}(s) - G_m(s)}{1 + F_2(s)G_{motor}(s)G_{cyl}(s)}.$$  \hspace{1cm} (3.2)

To achieve smooth reference tracking, $F_1(s)$ is chosen so that $G_c(s) \approx G_m(s)$. This can be done by defining $F_1(s)$ as

$$F_1(s) = \frac{G_m(s)}{G(s)}.$$  \hspace{1cm} (3.3)

where $\hat{G}(s)$ is an approximate process model based on the models $G_{motor}(s)$ and $G_{cyl}(s)$. The transfer function for $\hat{G}(s)$ is created by neglecting the rate limit and by using the mean value of the two gains for the different directions. These approximations lead to
\[
\hat{G}(s) = \frac{K_mC_{pos}(1 + C_{factor})/2}{s(sT + 1)}e^{-sT_L}.
\] (3.4)

\(G_m(s)\) is a process model that defines the desired behavior of the system. It is necessary for \(G_m(s)\) to have at least as many poles as \(\hat{G}(s)\) in order for the feed forward link to be proper, which in this case means that 2 poles are required. Since the goal of the controller is to achieve smooth reference tracking with no overshoot, \(G_m(s)\) is chosen as a second-order filter with real poles. The pole placement was chosen as a double pole at the same location as for the three parameter model of the electric motor in Section 2.2. Hence, \(G_m(s)\) is defined as

\[
G_m(s) = \frac{1}{(sT + 1)^2}.
\] (3.5)

With \(F_1(s)\) defined as in (3.3), a problem occurs with the time delay in (3.4) since this would require \(F_1(s)\) to use future reference values. To avoid this, \(G_m(s)\) is assumed to have the same delay and now becomes

\[
G_m(s) = \frac{1}{(sT + 1)^2}e^{-sT_L}.
\] (3.6)

With \(G_m(s)\) and \(\hat{G}(s)\) defined, it is now straight forward to define \(F_1\) as in (3.3).

### 3.1.3 Cascade PID

The system is based on two systems, an electric motor and the hydraulics. The speed of the motor can be measured which should allow for a cascade control to be implemented. The motor speed would then be controlled by the inner loop whereas the hydraulics can be controlled by the outer loop. Since the motor speed is already controlled by an internal controller the main purpose of the inner loop would be to eliminate the steady-state error of the motor speed and thereby make the system achieve higher performance. One demand on the system is that the inner loop needs to be significantly faster than the outer loop which would allow for the inner loop to be treated as a static gain by the outer loop. A block diagram showing how the controller is implemented can be seen in Figure 3.3. The controllers \(F_1\) and \(F_2\) are both PID controllers described in Section 3.1.1. The two controllers will be tuned separately. First, the inner loop is tuned on its own and thereafter the outer loop controller is tuned using the complete implementation, including the inner loop.
3.1 Deciding on Control Strategies

3.1.4 Linear Quadratic Controller

The LQ controller is a state-feedback controller, i.e., it calculates the control signal based on the reference value as well as the current states of the system. The control strategy requires a state-space model and hence the model presented in Section 2.5 will be used. The control signal is calculated as

$$ u(t) = L_r y_{ref}(t) - L x(t), \quad (3.7) $$

where $u(t)$ is $\omega_{ref}(t)$, $L_r$ is a scalar gain for the reference signal and $L$ is a feedback gain for the states. By inserting (3.7) into the general linear state-space model presented in (2.8) the closed loop state-space model is

\begin{align}
\dot{x}(t) &= A x(t) + B (L_r y_{ref}(t) - L x(t)) = (A - BL)x(t) + BL_r y_{ref}(t) \quad (3.8a) \\
y(t) &= C x(t) \quad (3.8b)
\end{align}

From this state-space representation, as well as the quadratic optimization problem presented in Section 1.5, it is possible to define the feedback gain $L$. The theory behind LQ does, however, not support a problem formulation with constraints. Hence the problem formulation was modified by removing the saturation on the control signal before determining the feedback gain. The control signal calculated by the controller is then saturated to provide an approximative solution to the original optimization problem. With no saturation the feedback gain can be

Figure 3.3: Block diagram showing the implementation of the cascade controller. The block marked $F_1$ is a PID controller used to make the piston follow the reference signal. The $F_2$ block is a PID controller used to enhance the tracking performance of the electric motor. $G_{motor}$ and $G_{cyl}$ are the two process models presented in Chapter 2.
determined using LQ theory as in Glad and Ljung [2003]. Basically, $L$ is chosen as

$$L = Q_2^{-1} B^T S,$$

(3.9)

where $S$ is the positive semi-definite solution to the algebraic Riccati equation

$$A^T S + S A + C^T Q_1 C - S B Q_2^{-1} B^T S = 0,$$

(3.10)

where $A$, $B$ and $C$ are the matrices defining the state-space presented in (3.8). $Q_1$ and $Q_2$ are the weights used in the linear quadratic problem formulation presented in Section 1.5. These calculations are not necessary to do by hand since the lqr command in MATLAB can be used. The state-space model and the weights are then used as input to the function. The theory behind defining $L_r$ can also be found in Glad and Ljung [2003]. The purpose of $L_r$ is to obtain a static gain equal to one for the closed control loop. For this application it is defined as

$$L_r = (C(BL - A)^{-1} B)^{-1}.$$

(3.11)

The LQ controller requires that all states are estimated. Since only one of the states are measured an observer like the one presented in Section 2.5 is required. Using this observer, the control signal is now defined as

$$u(t) = L_r y_{ref}(t) - L \hat{x}(t)$$

(3.12)

where $L_r$ and $L$ are defined as mentioned earlier and $\hat{x}$ are the estimated states. This control signal is the solution to the modified problem formulation without constraints. By saturating the control signal to stay within the limits $\pm \omega_{max}$, an approximative solution for the original optimization problem stated in Section 1.5 is obtained. A block diagram showing how the LQ controller is implemented can be seen in Figure 3.4.

### 3.1.5 Model Predictive Controller

The main idea with MPC is to compute the next control signal by solving an optimization problem with a finite time horizon at each time step. The control signal that defines the solution to the problem is then determined for each of the time instances within the finite time horizon. Although the control signals for all the next time instances during the finite horizon are determined only the control signal for the next time step is used. The problem is then solved again at the next time instance to compute the next control signal. The optimization problem that is solved for this application is similar to the problem presented in Section 1.5, but has a limited time horizon as well as an added term in the cost function for
3.1 Deciding on Control Strategies

Figure 3.4: Block diagram showing the implementation of the LQ controller. The block marked $L$ is the feedback gain while the block marked $L_r$ is the reference gain. The block marked observer is the observer presented in Section 2.5. $G_{\text{motor}}$ and $G_{\text{cyl}}$ are the two process models presented in Chapter 2.

The final time step. The problem formulation for the MPC to solve at each time step looks like:

\[
\begin{align*}
\text{minimize}_u & \quad \sum_{t=1}^{N-1} (Cx(t) - y_{\text{ref}}(t))^T Q_1 (Cx(t) - y_{\text{ref}}(t)) + \\
& \quad + \sum_{t=1}^{N-1} u(t)^T Q_2 u(t) + \\
& \quad + (x(N) - C^T y_{\text{ref}}(N))^T Q_3 (x(N) - C^T y_{\text{ref}}(N)) \\
\text{subject to} & \quad x(t+1) = f(x, u, t), \quad t = 0, ..., N - 1 \\
& \quad u(t) \in [-\omega_{\text{max}}, \omega_{\text{max}}]
\end{align*}
\]

where $N$ is the limited time horizon and $f(x, u, t)$ is the state-space model presented in Section 2.5. $Q_1$, $Q_2$ and $Q_3$ are the weights that determine the cost of each term in the optimization problem. $Q_1$ is here chosen significantly higher than $Q_2$ since the importance of fast reference tracking is regarded much higher than fulfilling the secondary goal by reducing oscillations in the control signal. $Q_3$ is here chosen as the solution $S$ to the Riccati equation mentioned in Section 3.1.4. By choosing $Q_3$ equal to $S$, stability of the closed loop system can be assured, Enqvist et al. [2010].

The quadratic problem is solved using a solver for quadratic programming problems. Since the computational capacity of the controller is limited, it is of great importance to find a solver that does not require much computational power.
A promising candidate for this task is CVXGEN, Mattingley and Boyd [2013], which can be implemented directly using C-code. The CVXGEN solver also provides compatibility for use with MATLAB as well as a user friendly interface in which to define the problem and generate code for the solver. Using the CVXGEN in MATLAB is very straightforward since it provides tools to create a MATLAB function based on the generated code. The solver predicts all future states within the time horizon based on the current states. In order for the future state estimations to be accurate, the current states has to be estimated. Since only one of the states can be measured the observer presented in Section 2.5 is used to estimate the remaining states. A block diagram showing the implementation of the MPC can be seen in Figure 3.5.

### 3.2 Controller Evaluation

The controller types listed in Section 3.1 are the candidates to evaluate in this thesis. In order to compare the strategies with each other they all have to be evaluated in the same way. These tests must be designed to cover a variety of cases and thereby provide a basis for the final evaluation of the controllers. The following tests will be performed:

- Step responses
- Chirp as input
- Disturbances
3.2 Controller Evaluation

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → α</td>
<td>α → 0</td>
</tr>
<tr>
<td>0 → 3α</td>
<td>3α → 0</td>
</tr>
<tr>
<td>0 → 5α</td>
<td>5α → 0</td>
</tr>
<tr>
<td>0 → 7α</td>
<td>7α → 0</td>
</tr>
<tr>
<td>−α → 0</td>
<td>0 → −α</td>
</tr>
<tr>
<td>−3α → 0</td>
<td>0 → −3α</td>
</tr>
<tr>
<td>−5α → 0</td>
<td>0 → −5α</td>
</tr>
<tr>
<td>−7α → 0</td>
<td>0 → −7α</td>
</tr>
<tr>
<td>−α → α</td>
<td>α → −α</td>
</tr>
<tr>
<td>−3α → 3α</td>
<td>3α → −3α</td>
</tr>
<tr>
<td>−5α → 5α</td>
<td>5α → −5α</td>
</tr>
<tr>
<td>−7α → 7α</td>
<td>7α → −7α</td>
</tr>
</tbody>
</table>

Table 3.1: Values for step responses to perform by setting the target piston position value, both for software simulation in MATLAB and on the hardware set up. α is a constant chosen to provide good coverage of the main working area.

All controller strategies will be tested using the same inputs and starting points. The measured piston position and motor speed will then provide a clear picture of how the controllers handle different situations as well as their performances while doing so.

3.2.1 Step Responses

The purpose of the step responses is to see whether the overshoot and steady-state errors are within the limits. These tests should also give a clear comparison of the speed of the different controllers. The step responses will be performed by setting the target piston position to fixed positions, wait for the piston to settle, and then set to a new target value. The values to test will all be a factor of α, which corresponds to approximately 0.15 in the rescaled axes. The constant α was chosen to provide good coverage of the main working area during the simulations. The values chosen can be seen in Table 3.1.

3.2.2 Chirp as Input

By applying a sine as input to the system, the reference tracking for inputs of various frequencies and amplitude can be tested. In other words the tracking capability of the system can be evaluated, it will also become clear whether the system is fast enough as well as what happens if it is not. Also the system’s ability to handle high frequency changes can be tested. To cover as wide range as possible, a chirp signal will be used with which it will be possible to determine if or when the system deviates from the reference, and at what frequency that occurs. The chosen amplitudes and the frequency increase for the chirp signal can be seen in Table 3.2. The amplitudes have all been derived as a factor of α whereas the frequency function is chosen dependent on time and hence forward
Table 3.2: Amplitude and frequencies chosen for testing, both for software simulation in MATLAB and on the hardware set up.

<table>
<thead>
<tr>
<th>Amplitude</th>
<th>Frequency</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2\alpha$</td>
<td>$\psi(t)$</td>
<td>Amplitude $\cdot \sin(\psi(t) \cdot t)$</td>
</tr>
<tr>
<td>$4\alpha$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$6\alpha$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

referred to as $\psi(t)$,

$$\psi(t) = \frac{2\pi}{\psi_{factor}(t)}, \quad \psi_{factor}(t) = t_0 - \frac{t_0 - t_1}{t_{sweep}} t$$  

where $t_0$ is the time period in the beginning of the chirp, $t_1$ is the time period at the end of the chirp and $t_{sweep}$ is the time it takes for the chirp to reach the final frequency. Note that the chirp signal might be risky to apply to the hardware, hence some handling for shutting down the system will have to be implemented. This limit will be set to stop the electric motor if the piston position diverges from the reference and reaches a value of $\pm 8\alpha$.

### 3.2.3 Disturbances

Simulations on how the controllers handle different types of disturbances is an important part of the evaluation. The possibilities to manipulate the hardware are limited, hence simulations will be different for software and hardware. In MATLAB, software simulations affected by step disturbances will be performed. This disturbance is meant to simulate the impact on the piston position from sudden forces applied to the load. The steps will be applied on the measured piston position and be of various height and starting points. A table showing the planned disturbances can be seen in Table 3.3. The reference signal will be kept constant during these simulations to give a clear view of how the step disturbances are handled. Software simulations with a disturbance in the form of low pass filtered white noise will also be performed. These simulations should provide knowledge of how randomized changes in both directions affect the system. This noisy signal will be applied during a simulation where the reference is the step signal presented in Table 3.1.

When investigating the disturbances effect on the hardware system it is only possible to generate one type of disturbance. The hardware set up for testing the controllers is connected to the same type of cylinder used in the system. It is possible to control the back pressure in this cylinder and thereby make the piston in the controlled system harder to move. The increased back pressure is meant to simulate the effect of a high load applied to the system and can be related to increasing the weight of the load. The regular hardware simulations will be performed using a minimum load with no extra back pressure. By doing simulations using the step signal presented in Table 3.1, with very high back pressure, a clear
3.2 Controller Evaluation

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \rightarrow \zeta$</td>
<td>$\zeta \rightarrow 0$</td>
</tr>
<tr>
<td>$0 \rightarrow 3\zeta$</td>
<td>$3\zeta \rightarrow 0$</td>
</tr>
<tr>
<td>$0 \rightarrow 5\zeta$</td>
<td>$5\zeta \rightarrow 0$</td>
</tr>
<tr>
<td>$-\zeta \rightarrow 0$</td>
<td>$0 \rightarrow -\zeta$</td>
</tr>
<tr>
<td>$-3\zeta \rightarrow 0$</td>
<td>$0 \rightarrow -3\zeta$</td>
</tr>
<tr>
<td>$-5\zeta \rightarrow 0$</td>
<td>$0 \rightarrow -5\zeta$</td>
</tr>
<tr>
<td>$-\zeta \rightarrow \zeta$</td>
<td>$\zeta \rightarrow -\zeta$</td>
</tr>
<tr>
<td>$-3\zeta \rightarrow 3\zeta$</td>
<td>$3\zeta \rightarrow -3\zeta$</td>
</tr>
<tr>
<td>$-5\zeta \rightarrow 5\zeta$</td>
<td>$5\zeta \rightarrow -5\zeta$</td>
</tr>
</tbody>
</table>

Table 3.3: Step limits and the direction of the external forces to be tested using software simulation in MATLAB. The step base $\zeta$ is chosen to be half the value of the constant $\alpha$, which was used to describe the step signal described in Table 3.1.

view of how the weight of the load affects the performance of the controllers can be presented. Simulations with these two pressure levels should then provide the extreme values of the loads impact on the system. Other loads are then expected to cause a performance somewhere in between that of the two extreme values.

### 3.2.4 Robustness Analysis

Apart from the evaluations based on step responses, chirp signals and disturbances, a basic robustness analysis will be performed. These tests are meant to illustrate the impact of relative model errors and are performed by multiplying the gain of the electric motor with an uncertainty constant. The true system $G^{0}(s)$ can then be described as

$$G^{0}(s) = (1 + \Delta G)G(s) \quad (3.15)$$

where $\Delta G$ is the relative model error, assumed to be a real value, and $G(s)$ is the model used for controller synthesis. Here the uncertain constant is equal to the term $(1 + \Delta G)$. A relative model error larger than zero corresponds to the system being faster than modeled. These tests will provide valuable information of the control strategies performance when the real system is either faster or slower than modeled. The analysis will be performed based on software simulations in MATLAB Simulink.

An analysis will also be performed where the rate limit is either increased or decreased. The manipulated rate limit corresponds to a true model were changes applied to the system take effect either faster or slower than modeled, i.e. an increased rate limit means that changes to the system are applied faster.
In this chapter the results will be presented. Based on these results the control strategies will be compared and evaluated. When possible, the simulations from both software and hardware will be compared to illustrate their differences. The proposed control strategies were all implementable in software. All controllers were also implementable on hardware except for the MPC due to some unknown compatibility error. Hence, there are no simulations of the MPC on hardware. The software simulations with the MPC are however still used to provide a basis for comparing the different strategies.

4.1 Controller Performance

This section presents the results from the tests presented in Section 3.2. Plots showing an overview of the tests will be presented as well as close up illustrations to highlight the differences in performance for the proposed controllers.

4.1.1 Step responses

When applying a step response signal as reference in a software simulation, it immediately became clear that the cascade PID controller could not be used for this application. The proposed control strategy would initially track the reference but as soon as a big step was applied, the simulated piston position quickly diverged. This behavior is probably a result of the inner loop not being significantly faster than the outer loop. If the inner loop is not much faster than the outer loop it can not be regarded as a static gain by the outer loop. Since the controller appeared unstable, the cascade PID controller was eliminated as a possible control strategy for this application.
With the rescaled axes $\gamma \approx 0.003$, which means that the overshoot limit for the simulations with step responses is approximately 0.030 and the acceptable steady-state error is approximately 0.009.

**Software Simulation**

The remaining four control strategies all produced satisfactory performances. An overview showing the behavior of the four control strategies can be seen in Figure 4.1. At a quick glance it looks as if they all gave identical performances. With a closer look on the results some differences are however seen. A close up of a positive step, seen in Figure 4.2, shows that the MPC reaches the target reference first with the LQ controller following tightly behind. The regular PID as well as the PID with feed forward from reference are the slowest and gave identical results. The reason the PID based controllers get a steady-state error is that the integral gain was chosen very small and close to zero to fulfil the overshoot demand for all possible situations. Hence the steady-state error is not handled within the time frame of the step. This steady-state error is however well within the required limit.

![Software simulation with step signal as reference](image)

**Figure 4.1:** Overview of the results for the different control strategies from a software simulation using a step response signal as reference.
4.1 Controller Performance

**Figure 4.2:** Close up of the results from a software simulation with a step response in the positive direction for the different control strategies. The step simulated in this figure went from zero to one. The PID controllers gave identical results. The PID based controllers also got a small steady-state error of approximately \(\frac{1}{20}\) of the accepted limit.

**Figure 4.3:** Close up of the results from a software simulation with a step response in the negative direction for the different control strategies. The step simulated in this figure went from one to zero. The PID controllers gave identical results.
When zooming in on the results for a step in the negative direction, another difference is presented and can be seen in Figure 4.3. Here, the controllers are almost equal in terms of speed although the PID based controllers both produce some overshoots. The overshoots are not that big and stay within the required limits, but they are still noticeable. The MPC and LQ both reach the target without overshoots, with the MPC being slightly faster. The two close ups presented were chosen since they both provide clear illustrations on steps where the difference in performance is more noticeable. For smaller steps the behavior of the controllers are roughly the same as for these steps, although the differences between them are harder to distinguish since they are much smaller.

**Hardware simulation**

Since the MPC was never successfully implemented, no hardware simulation of the MPC was recorded. The remaining three controllers were however simulated on hardware. A hardware simulation using the old PID controller was also performed to provide a comparison between the old controller and the new strategies. A plot showing the results from the hardware simulations using the step signal as reference can be seen in Figure 4.4. This figure shows that all controllers succeed in tracking the reference and that there are a few differences. A close up on an illustrative positive step response can be seen in Figure 4.5. This plot clearly shows that the response of the old controller is both slower than the others and produces an overshoot. The new controllers all have smooth responses without causing the piston position to overshoot. It also clearly shows that the LQ controller is slightly faster than the PID based controllers and that all the proposed controllers get a small steady-state error.

The reason that the PID based controllers get a steady-state error is the same here as for the software simulation, the integral gain is very small. A possible explanation for the steady-state error of the LQ controller is that the real electric motor requires a significant input to start spinning, meaning that it is effected by a dead band. Hence it might be the case that the computed control signal is too small to make the motor start spinning within the given time frame. The steady-state error could be handled by tuning the parameters for a faster controller. This would however result in an increased overshoot and hence will not be done. When looking at a close up on a step in the negative direction, seen in Figure 4.6, the results are quite different. In this direction the three new controllers all perform similarly in terms of speed but reach the target with different signs on the steady-state error. The old PID controller is significantly slower than the others but does however not generate a steady-state error. A note of consideration from these simulations is that the results differ a bit from those of the software simulations. These differences show that the model is not perfect whereas the similarities show that the model is still able to give a decent representation of the system.
4.1 Controller Performance

**Figure 4.4:** Overview of the results for the different control strategies from a hardware simulation using a step response signal as referencce.

**Figure 4.5:** Close up of the results from a hardware simulation with a step response in the positive direction for the different control strategies. The PID and PID with feed forward controllers gave practically identical results.
4.1.2 Chirp as Input

The purpose of the tests with a chirp signal as reference was to see how the controllers handle reference signals with high frequency and illustrate their performance while doing so. These tests were first performed in a software simulation. None of the simulated controllers diverged enough to reach the safety limit and therefore all controllers qualified for testing using hardware. Three chirp signals were used with the same frequency function but with different amplitudes, stated in Section 3.2.3. The time constants in (3.14) were for these simulations defined as $t_0 = 200$, $t_1 = 20$ and $t_{\text{sweep}} = 1200$.

Software Simulation

For the software simulations using the chirp signal as input the same type of results were gathered for all tests. The different amplitudes used made the results from the simulations differ from one another. In Figure 4.7 the control errors when applying the chirp signal with the biggest amplitude are seen. This figure illustrates that the tracking capability of the proposed controllers changes around time 900. Equation (3.14) and Table 3.2 implies that this time corresponds to a frequency of approximately 14 Hz, which is a very high frequency for this application. Although the signals overlap, the simulation results show that the LQ and MPC both handle the increasing frequency in the same way. Both control strategies result in a control error with increasing amplitude for inputs of increasing frequency. The control errors of the PID based controllers both deviate from the center and tend to produce a control error with a positive sign. The deviation is

**Figure 4.6:** Close up of the results from a hardware simulation with a step response in the negative direction for the different control strategies. The PID and PID with feed forward controllers gave practically identical results.
however very small for the PID controller and similar to the model based control strategies. The deviation for the PID controller with feed forward from reference is much bigger. The PID with feed forward from reference does however appear to produce a much smaller control error than the other control strategies up until time 900. The smaller control error interprets as the PID with feed forward from reference being better at tracking the reference for these frequencies.

![Software simulation with chirp signal as reference](image)

**Figure 4.7:** Overview of the control error for the different control strategies from a software simulation using a chirp signal with amplitude $6\alpha$ as reference.

A close up on the overview simulation can be seen in Figure 4.8. From this figure it is clear that the PID controller with feed forward from reference is superior when it comes to tracking a reference signal with low frequency since the control error is very small. The three other control strategies all produce similar performance with the PID giving a slightly bigger control error compared to the other two. The overviews of the software simulations with the remaining two chirp signals, with different amplitudes, can be seen in Figure 4.9 and Figure 4.10. They both provide results similar to those illustrated in Figure 4.7, although the characteristic control error produced by the controllers tend to shift to the right in the time domain with decreasing amplitude.
Figure 4.8: Close up of the control error from a software simulation with a chirp signal as reference for the different control strategies.

Figure 4.9: Overview of the control error for the different control strategies from a software simulation using a chirp signal with amplitude $4\alpha$ as reference.
Hardware Simulation

The hardware simulations with the chirp signals as reference produced results very similar to those of the software simulations, which can be seen in Figure 4.11. The controllers all show signs of failing at around the time 900. The PID with feed forward from reference is still superior when it comes to tracking performance for the lower frequencies. The proposed PID based controllers no longer seem to give a control error that deviates to the positive sign. This behavior reveals the differences between the model and the real system. A possible explanation is that the increased frequency results in higher frictional forces being applied to the system. These frictional forces may impact the piston in the asymmetrical cylinder differently, depending on the movement direction, and hence balance out the positive deviations and keep the control error centred around zero. The previously installed PID controller produces a control error that deviates towards the positive side of the piston position and does not seem to be able to track the piston position at all after the 900 mark.

A close up on the overview simulation can be seen in Figure 4.12. The figure reveals a result that is very similar to those of the software simulations but more oscillatory. The PID with feed forward from reference still achieves the best tracking result for lower frequencies. The other controllers, including the old PID controller, tend to produce the same level of control error as for the software simulations, where the old PID controller produces the smallest control error of the three. An overview of the control error from simulations using a chirp signal with the two smaller amplitudes can be seen in Figure 4.13 and Figure 4.14. They both show similar results as in Figure 4.11 but with the characteristic control error behavior shifted slightly to the right in the time domain with decreasing amplitude.
Figure 4.11: Overview of the control error for the different control strategies from a hardware simulation using a chirp signal with amplitude $6\alpha$ as reference. All the proposed controllers tend to result in a control error that oscillates around zero whereas the old PID controller does not.

Figure 4.12: Close up of the control error from a hardware simulation with a chirp signal as reference for the different control strategies. The control error was far more oscillatory for the hardware simulations compared to the software simulations.
4.1 Controller Performance

Figure 4.13: Overview of the control error for the different control strategies from a hardware simulation using a chirp signal with amplitude $4\alpha$ as reference.

Figure 4.14: Overview of the control error for the different control strategies from a hardware simulation using a chirp signal with amplitude $2\alpha$ as reference.
4.1.3 Disturbances

This section provides results from simulations with various disturbances. The procedures for testing will follow those stated in Section 3.2.

Software Simulation - Step Disturbance

The step disturbances are meant to simulate the impact of sudden forces applied to the load which would result in a sudden movement of the piston. Since no model of the load exists the force disturbance was approximated by applying the step disturbance directly on the measured piston position. This disturbance then corresponds to a force applied to the load in the form of a short pulse. When simulating the impact of step signals as disturbance the reference signal was kept constantly equal to zero. The controllers are then expected to move the piston back to the center in a smooth and controlled manner. An overview of the step disturbance impact on the piston position as well as the step disturbance applied can be seen in Figure 4.15. The overview clearly states that all the simulated controllers are able to handle the disturbance and move the piston back to the center position without causing overshoots and oscillatory behavior.

![Software simulation with step signal as disturbance](image)

**Figure 4.15:** Overview of the results for the different control strategies from a software simulation using a step response signal as disturbance on the piston position.

A close up on how the controllers handle a disturbance step in positive direction can be seen in Figure 4.16. The lines representing the PID based controllers overlap in the figure which means that they produce equal results. The LQ and MPC also overlap and are slightly slower than the PID based controllers. A close up on a step disturbance in the negative direction can be seen in Figure 4.17. The PID controllers overlap in this figure as well and the LQ overlaps the MPC. The figure clearly states that for the tuning used here, the PID based controllers handle steps
in this direction slower than the model based control strategies. In both close ups it is shown that all the proposed controllers counteracts the step disturbances in less than 10 time units without overshoots or oscillatory behavior.

![Software simulation with step signal as disturbance](image)

**Figure 4.16:** Close up of the results from a software simulation with a step response as disturbance in the positive direction for the different control strategies.

![Software simulation with step signal as disturbance](image)

**Figure 4.17:** Close up of the results from a software simulation with a step response as disturbance in the negative direction for the different control strategies. The PID controllers give identical results.
Software Simulation - Measurement Noise

This section will investigate the impact of filtered white noise as disturbance during software simulation, which corresponds to the position sensor being affected by measurement noise. The noise was added to the measured piston position while using the step response signal as reference. A close up on a positive step can be seen in Figure 4.18. This figure clearly shows that the noise has the same impact on all controllers and seem to be handled in the same way. Also seen in the figure is that the MPC gets a slight overshoot as a result of the added noise. All of the controllers do however remain close to the reference value.

![Software simulation with filtered white noise disturbance](image)

**Figure 4.18:** Close up on a software simulation of a positive step response with added low-pass filtered white noise as disturbance.

A close up on a step in the negative direction, seen in Figure 4.19, reveals the same type of result as for the step in the positive direction. All of the proposed controllers seem to be affected by the noise in the same way and no clear differences in how they handle the disturbances can be seen. All in all the disturbance using filtered white noise is handled in the same way by all controllers. The similarities between these steps and the simulations without the noise in Figure 4.2 and Figure 4.3 reveals that the added noise does not cause unexpected behavior from the proposed control strategies.
Figure 4.19: Close up on a software simulation of a negative step response with added low-pass filtered white noise as disturbance.

Hardware Simulation - High Back Pressure

This section will present the results from the hardware simulations where a disturbance in the form of high back pressure was used. These simulations correspond to a load applied to the system with significantly higher weight than modeled. An overview of a step response simulation can be seen in Figure 4.20. This figure reveals that all controllers reach the target reference but at a slower rate compared to when no added back pressure was used seen in Figure 4.4.

The close up on a positive step seen in Figure 4.21 reveals that all of the proposed control strategies achieve very similar performances, with the LQ being slightly faster than the other two. The old PID controller reaches the target reference first but does so with a small overshoot. All the proposed controllers reach the target within the steady-state error zone about 7 time units faster compared to the old PID controller. The same type of results goes for steps in the negative direction but here all controllers reach the zone of the accepted steady-state error in approximately the same time. This can be seen in Figure 4.22.

A clear difference between the proposed controllers and the previously implemented PID controller is the produced motor speeds from these simulations, which can be seen in Figure 4.23. This figure illustrates the measured motor speeds during a step in the positive direction. All the proposed control strategies produce similar and rather smooth motor speeds whereas the old PID controller produce a far more oscillatory motor speed. This result implies that all the proposed controllers achieves the secondary goal where oscillations in the control signal is to be avoided, since an oscillatory control signal results in an oscillatory measured motor speed.
Figure 4.20: Overview of the results for the different control strategies from a hardware simulation using a step response signal as reference. For this simulation a high level of back pressure was applied to the system.

Figure 4.21: Close up on the results for the different control strategies from a hardware simulation with a positive step response as reference. The proposed controllers all behave similarly and reach the target reference slower than the previously implemented PID controller.
4.1 Controller Performance

Figure 4.22: Close up on the results for the different control strategies from a hardware simulation with a negative step response as reference. The proposed controllers all behave similarly and reach the target reference slower than the previously implemented PID controller.

Figure 4.23: Close up on the resulting motor speeds for the different control strategies from a hardware simulation with a positive step response as reference. The proposed controllers all behave similarly and produce a much smoother motor speed than the previously implemented PID controller.
4.1.4 Robustness Analysis

In this section the results of the robustness analysis are presented. This analysis consists of software simulations were either the gain of the system is different than modeled or has a modified rate limit. The impact of a relative model error of $+20\%$, which corresponds to $\Delta G = 0.20$, can be seen in Figure 4.24 and the definition of $\Delta G$ can be seen in Section 3.2.4. From this figure it is clear that a positive relative model error of this size would cause clear overshoots that exceed the demands put on the system. Simulations reveal that the LQ and MPC strategies are able to handle a slightly higher model error compared to the PID based controllers. This increase is however small and the limit for the maximum relative model error is approximately 0.09 for all control strategies. With a higher model error the control strategies fail to fulfil the requirements put on the system.

**Figure 4.24:** Plot illustrating the impact of a relative model error of $\Delta G = 0.20$ while performing a software simulation using the step response signal as reference. The model error causes clear overshoots that exceed the demands put on the system.
When applying a relative model error smaller than zero, like for instance $\Delta G = -0.2$, no overshoots are visible. The step responses are however a bit slower but all control strategies reach the target reference in a smooth and controlled manner. This can be seen in Figure 4.25. This implies that a negative relative model error will always be manageable by the control strategies and fulfil the requirements put on the system. The tracking performance will however decrease in terms of speed for relative model errors closer to $-1$, which is assumed to be the lowest possible value for the model error.

![Robustness analysis, decreased gain](image)

**Figure 4.25:** Plot illustrating the impact of a relative model error of $\Delta G = -0.20$ while performing a software simulation using the step response signal as reference. The model error causes slower step responses compared to simulations without the model error.

The simulations where the rate limit of the model was modified proved to generate the same type of results as the relative model error with a few differences. For these simulations a rate limit chosen higher than modeled resulted in the fast responses seen in Figure 4.24 where the relative model error $\Delta G$ was positive, but without the overshoots. A lower rate limit than modeled caused the same slow re-
responses seen in Figure 4.25 where the relative model error was negative, but with the overshoots seen in Figure 4.24. Simulations revealed that the rate limit could be decreased by a factor of approximately 9% before the overshoots exceeded the requirements. The fact that an increased rate limit enhanced the performance is not surprising since this implies that the controllers can manipulate the system more easily, whereas a decreased rate limit hinders the controller from applying changes to the system and thereby makes it more difficult to control.

4.2 Discussion

All controllers achieve the primary goals presented in Section 1.4. They all end up with a steady-state error less than $3\gamma$ as well as an overshoot less than $10\gamma$. The proposed controllers also achieve the goal of keeping the piston position non-oscillatory for a constant reference value. Regarding the goal of having a fast response the differences between the control strategies start to appear. While performing software simulations with a step signal as reference the MPC and the LQ control strategies appear to be superior. They both reach the target reference faster than the other control strategies while maintaining a non-existent overshoot for almost all step responses of various sizes. There are however a few step sizes that cause a small overshoot even for these control strategies, hence they were not tuned to be more aggressive. The LQ also shows superiority for step responses simulated using hardware.

When it comes to the simulations using a chirp signal as references some major differences appear, at least for the software simulations. For the parts with a low frequency reference, the tracking performance of the PID controller with feed forward from reference is unmatched. None of the other strategies are close to the same level of tracking capability and all other strategies achieve almost equal levels of control error. As the frequency of the chirp signal increases the controllers start to behave differently. With the increasing frequency it is now the model based controllers LQ and MPC as well as the regular PID controller that gets the upper hand.

At around the frequency 14 Hz all of the proposed control strategies lose their tracking capability and a control error significantly higher than zero appear. The control error seems to deviate towards the positive side for the PID with feed forward from reference which is the reason the other control strategies get the upper hand. The deviating control error is caused by the simulated piston position deviating from oscillating around zero. This deviation from the centralized position is not ideal and the behavior of the other control strategies, where the control error remains around zero, is preferable. A possible explanation for the behavior of the PID with feed forward from reference, is that with the increased frequency the difference between simulated piston position and requested gets bigger. This strategy produces a control signal that relies on both the PID controller and the feed forward link and when the differences increase, the two parts of the control signal might counteract each other. This might occur if the control error is pos-
itive at the same time as the reference signal is increasing, then the PID would demand a negative control signal whereas the feed forward link would demand a positive control signal.

Hardware simulations using the chirp signal as reference does however reveal that the differences between the control errors for the control strategies are much smaller in reality for the higher frequencies. In hardware simulations the control error for the PID with feed forward from reference no longer deviates from the centre. That the control error no longer deviates might be caused by unmodeled features of the system such as dead-band for the electric motor and friction in the cylinder. With these unmodeled effects it is possible that the speed of the system was estimated too optimistic when constructing the model. These two features would both counteract changes of the piston movement and cause the piston to slow down around the centralized position. This means that by designing a new model, where these effects have been taken into consideration, the same results as for the hardware simulations could be achieved even during software simulations.

Since all control strategies handled the simulations with disturbances in a similar fashion, no noticeable difference can be pointed out. The only disturbance that had a major impact on the controllers was during the hardware simulation with increased back pressure. For this simulation all of the proposed controllers appeared slower than the previously installed controller, which is a probable result of the old controller having a significant integral gain and hence it is able to handle this type of disturbance better. All controllers did however reach the acceptable steady-state error vicinity in approximately the same time. This implies that the load has a major impact on the system. The high back pressure made the controllers slower at reaching the target reference. Since this load was very high, the performance of the control strategies is expected to increase when smaller, more common loads are applied.
In this thesis five control strategies for piston position control of an EH system were developed and evaluated. The strategies chosen for evaluation was a PID controller, a PID controller with feed forward from reference, a PID based cascade controller, an LQ controller and an MPC controller. The control strategies were first implemented in a software environment created with MATLAB Simulink, and later implemented on actual hardware by code generation of the Simulink models using Real-Time-Workshop. To synthesize the control strategies an approximate process model was formed. This model was tuned to fit recorded data and provided accurate estimations of the behavior of the real system. Based on this model, a linear state-space model was created to synthesize the model based control strategies LQ and MPC.

During software simulations the cascade controller proved to be unstable when large step responses were used as reference. The probable cause of this instability was that the inner control loop, due to the nonlinear rate limit in the electric motor, was not fast enough compared to the outer control loop. The remaining four other control strategies all showed promising results. Although very basic, the PID controller provided decent results during the simulations and performed equally to the PID with feed forward from reference during step responses, at least for larger steps. The PID controller with feed forward from reference provided a slight enhancement of step responses of low height compared to the PID controller. The main advantage with the added feed forward link was shown when chirp signals were applied as reference. For the lower frequencies the PID with feed forward from reference produced a control error almost equal to zero, which was significantly lower compared to the other control strategies. For the higher frequencies all control strategies failed to track the reference signal. These higher frequencies are however unreasonably high for this application.
The model based controllers LQ and MPC both showed significant improvements for step responses compared to the PID based control strategies. For large step responses the MPC provided a slight enhancement compared to the LQ. For simulations were a chirp signal was used as reference the model based control strategies produced result very similar to those of the PID controller. One major downside with the MPC control strategy was that the implementation on hardware proved to be unsuccessful. With the failed hardware implementation the MPC also failed to fulfil the requirement of finding a controller that could be implemented on the actual system.

With the cascade controller and the MPC discarded, three control strategies remained for evaluation. All three control strategies improved the performance of the closed loop system compared to simulations using the previously installed PID controller. All three control strategies succeeded in fulfilling the requirements put on the system. The LQ controller excelled during the simulations with step responses whereas the reference tracking of the PID with feed forward from reference during simulations with chirp signal as input was unmatched, at least for lower frequencies. The LQ controller produced a simulated piston position with a clear resemblance to the reference signal during the chirp simulations but had a slight time delay, hence the LQ strategy produced a more noticeable control error.

With the significant increase of performance for step responses as well as the tracking performance for chirp signals as reference with the right appearance and only a small time delay, the LQ is recommended for final implementation. The LQ strategy is easily reconfigured by tuning the weights for the terms regarding reference tracking and for the control signal. Since the system consists of clear nonlinearities, one downside with the LQ is the requirement of a linear state-space model since this implies that the controller will always be based on an approximation of the real system. The linear state-space model used for the synthesis of the LQ controller does however provide an accurate approximation of the real hardware system with the help of the observer. If the system is used for a different application than modeled, a new state-space model would have to be created before a well configured LQ controller can be synthesized. This should however be pretty straightforward by using the same reasoning as when finding the model used here. With an even better model, the performance of the LQ control strategy is expected to increase even more.


E Gnesi, J.C. Maré, and J.L. Bordet. Modeling of EHA module equipped with fixed-displacement vane pump. In The 13th Scandinavian International Con-


Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Simon André