Czech Technical University in Prague

Faculty of Electrical Engineering

DIPLOMA THESIS

Bc. Jiří Valtr

Mass Flow Estimation and Control in Pump Driven Hydronic Systems

Department of Cybernetics

Supervisor: **Ing. Jiří Dostál**

Prague, January 2017

Author Statement

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, dated ....

signature

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student:	Bc. Jiří V a l t r
Study programme:	Cybernetics and Robotics
Specialisation:	Robotics
Title of Diploma Thesis:	Mass Flow Estimation and Control in Pump Driven Hydronic Systems

Guidelines:

1. Study theory regarding pump modelling, nonlinear data fusion, optimal feedback control.
2. Develop a simulation model of a pump-heat exchanger setup. Calibrate the model, if data from the real test-bed are available.
3. Develop a mass flow estimator using all available information.
4. Develop a mass flow controller using optimal control techniques.
5. Validate the proposed estimation and control methods on the simulation model and the real test-bed, if it is available.

Bibliography/Sources:

[1] Jitendra R. Raol - Multi-Sensor Data Fusion with MATLAB® - CRC Press, 2009.
[2] Mohinder S. Grewal, Angus P. Andrews - Kalman Filtering: Theory and Practice with MATLAB - John Wiley & Sons, 2014.
[3] Kirk E. Donald – Optimal Control Theory: An Introduction - Courier Corporation, 2004.
[4] Grundfos - The Centrifugal Pump - Grundfos Research and Technology, 2008.

Diploma Thesis Supervisor: Ing. Jiří Dostál

Valid until: the end of the winter semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, September 29, 2016

Abstract

This thesis deals with mass flow in hydronic systems. It studies the theory of hydronic system modelling and the theory of data fusion with focus on state estimation techniques. A simulation model of a hydronic system is developed and implemented in Simulink. Data recorded on a real hydronic test-bench are used to analyse the pump characteristic and to calibrate the simulation model. A discrete mathematical model of the hydronic system is derived and used in development of an extended Kalman filter mass flow estimator, which uses the information coming from measurements of electric power supplied to the pump and from identification of the hydraulic resistance of the system. A mass flow controller for the hydronic system is further developed, which uses the estimated mass flow as state observer to track a mass flow reference. Two different approaches to the controller design are presented. One uses a simple PI controller, the other is with an LQR state-feedback controller.

Keywords: data fusion, extended Kalman filter, hydronic system, pump control

Abstrakt

Tato práce se zabývá hmotnostním průtokem v otopných systémech. Rozebírá teorii modelování otopných systémů a teorii fúze dat se zaměřením na metody odhadování stavu. Simulační model otopného systému je vyvinut a implementován v Simulinku. Data naměřená na fyzickém modelu otopného systému jsou využita k analýze charakteristiky pumpy a ke kalibraci simulačního modelu. Je odvozen a diskretizován matematický model otopného systému, který je následně použit při vývoji rozšířeného Kalmanova filtru. Ten odhaduje hmotnostní průtok s využitím informací z měření elektrického příkonu pumpy a z identifikovaného hydraulického odporu systému. Dále je vyvinut regulátor hmotnostního průtoku, jež používá odhad průtoku jako pozorovatele stavu ke sledování referenční hodnoty hmotnostního průtoku. Jsou představeny dva rozdílné přístupy návrhu regulátoru. První využívá jednoduchý PI regulátor a druhý je regulátor s LQR stavovou zpětnou vazbou.

Klíčová slova: fúze dat, rozšířený Kalmanův filtr, otopný systém, řízení pumpy

Acknowledgements

I wish to thank everyone who has in any way helped me with this work. First and foremost, I would like to thank my supervisor, Ing. Jiří Dostál, whose continuous support, inspiring suggestions and patient guidance were instrumental for its completion. I also thank my teammates and fellow students Ondřej, Tomáš, Jan and Andrej for their help with the programming and in the lab. Finally, my deepest thanks go to my family, whose never-ending support throughout my studies made this work possible.

Contents

List of Figures

List of Symbols

State-space Model

$\mathbf{A}$	state matrix of a linear system
$\mathbf{B}$	input matrix of a linear system
$\mathbf{C}$	output matrix of a linear system
f	non-linear dynamic model function
$\mathbf{F}$	Jacobian of the dynamic model function
h	non-linear measurement function
$\mathbf{H}$	Jacobian of the measurement function
j	dimension of the measurement vector
k	time step index
l	dimension of the input vector
m	dimension of the state vector
$\mathbf{u}$	input vector
$\mathbf{x}$	state vector
$\mathbf{z}$	measurement vector

Kalman Filter

$\mathbf{K}$	Kalman gain
$\mathbf{P}_k^-$	*a priori* error covariance at time step k
$\mathbf{P}_k$	*a posteriori* error covariance at time step k
$\mathbf{v}$	random process noise

$\mathbf{V}$	process noise covariance matrix	
$\mathbf{w}$	random measurement noise	
$\mathbf{W}$	measurement noise covariance matrix	
$\hat{\mathbf{x}}_k^-$	*a priori* state estimate at time step k	
$\hat{\mathbf{x}}_k$	*a posteriori* state estimate at time step k	

Hydronic System Model

I	inertance	$[\mathrm{Pa\,h^2\,kg^{-1}}]$
n	angular velocity of the pump impleller	$[\mathrm{revolutions\,s^{-1}}]$
n_0	nominal angular velocity of the pump impleller	$[\mathrm{revolutions\,s^{-1}}]$
p	pressure	[Pa]
P_e	electric power	[W]
q	mass flow	$[\mathrm{kg\,h^{-1}}]$
R	hydraulic resistance	$[\mathrm{Pa\,h^2\,kg^{-2}}]$
s	pump speed (relative to nominal speed)	[–]

Other Symbols

$\mathbf{I}$	identity matrix

Chapter 1

Introduction

Most hydronic systems of today carry a fundamental paradox in their design. On one hand, there is a circulator (usually a pump) in the system driving the liquid within in order to convey thermal energy from the heat source to all parts of the heated space. On the other hand, to control the distribution of the conveyed heat as necessary, throttling valves are positioned around the hydronic network. These in essence hinder the operation of the circulator, forcing it to overcome greater resistance.

One way to avoid this paradox lies in completely removing the throttling valves from the system and using small pumps in their place. The pumps have better ability to influence the amount of conveyed heat through the control of mass flow passing through them and instead of imposing greater resistance on the system, they aid the main circulator effort. Both of these advantages create potential for substantial energy savings. However, to be able to exploit this potential as much as possible, it is necessary to have precise knowledge about the amount of heat being distributed as well as the ability to control it well. Thus, there is need for a good mass flow estimator and a good mass flow controller.

When estimating the mass flow of a hydronic liquid through a pump in a hydronic system, without actually directly measuring it, there are several sources of information one can turn to. First and foremost, there is the electric power driving the pump. It is directly connected to the mass flow produced by the pump and since it is easily measurable, it serves as the main source of information. Further useful information about mass flow comes from the knowledge of the hydraulic assembly of the hydronic network. Such knowledge combined with a known pressure characteristic of the pump can be also used to infer the mass flow.

Having several sources of information for a single quantity entails utilization of data fusion techniques to produce an overall result of the estimation. In this thesis, I make use of Kalman filtering to merge the individual estimates together. Moreover, Kalman filters are also used to process the noisy measurements for the individual sources of information. I explore non-standard implementations of the Kalman filter to reduce the risk of numerical difficulties during estimation.

To facilitate the development of a mass flow estimator which would use the combined information, a good model of a hydronic system is needed. I thus first formulate a mathematical model of the hydronic system at hand. To verify its correctness and to allow for experimenting, I then use a hydronic system Simulink model created by my colleagues at the Department of Control Engineering of the Czech Technical University, adjusted to fit the structure of the mathematical model. To calibrate the parameters of the model, I study the physical model of a hydronic system, which was also available to me at the department. Statistical analysis of the measurements made on the physical model was also used to tune parameters of the Kalman filter.

Once the best possible estimate of the mass flow is attained, I turn to the development of a mass flow controller. I develop a controller, which utilizes the estimated mass flow as a state observer output to optimally control the actual mass flow. Two design options are presented, one using a simple PI feedback controller, the other using a more complex approach with Jacobian linearization and a LQR regulator.

1.1 Organization of the Thesis

The thesis is organized in the following way. Below this introduction, Chapter 2 introduces the reader to the theory of data fusion and especially Kalman filtering. Chapter 3 describes the way the hydronic system is modelled mathematically. Chapter 4 presents both the physical and simulational testing facilities used throughout this thesis. Chapter 5 provides an in depth description of the actual mass flow estimation process and deals with actual MATLAB and Simulink implementations of the estimator. Chapter 6 addresses the design of a controller which utilizes the estimate of the mass flow provided by the estimator. Finally, Chapter 7 presents some experiments performed on the developed estimator and controller, to showcase their functionality.

Chapter 2

Data Fusion

In this chapter, I study data fusion. First, I define what data fusion is from a very general point of view. Second, I overview possible classifications of the data fusion processes that are being used across different fields of research, listing some examples in the process. Later, I provide an introduction to the techniques aimed specifically at state estimation. Last, I focus more in detail on Kalman filtering, which is the technique I used in my work.

2.1 Data Fusion Definition

Humans accept inputs from five different senses. The human brain then by some process, not yet fully understood by us, combines and transforms these inputs into a sensation of one reality, giving us a much more complete picture of our surroundings than would have been possible using any of the senses separately [1]. The process in the human brain can be understood as being a data fusion process.

In general, any task involving estimation can benefit from the use of data fusion [2]. In this thesis, I understand the term *data fusion* as being synonymous to the term *information fusion*. In other words, instead of only considering fusion of raw data obtained from sensors, I am interested in combining any information available whatsoever. Aside from sensor data, this could mean for example taking advantage of any knowledge coming from system identification or associated physical constraints. Another example can be the human depth perception, which aside from the obvious information provided by binocular vision also heavily relies on information coming from familiar size of objects recognized in the scene, changes to the scene due to the motion of the observer himself and several other cues [3].

A generally accepted [2] definition of data fusion comes from the Joint Directors of Laboratories workshop [4]: *"A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance."* In short, data fusion can be defined as combination of multiple sources to obtained improved information. The improvement of the information can be in the sense of smaller cost, higher quality or greater relevance. In [1], the related term *sensor fusion* is defined as the combination of sensory data, such that the resulting information is better than it would be if these sensors were used individually.

2.2 Data Fusion Classification

Data fusion being a multidisciplinary area, it is hard to define a clear classification of the employed methods and techniques. For the purposes of this overview however, I have chosen several possible classifications, which are proposed in literature.

The first criteria, based on which it is possible to classify, is the relations between the data sources [1, 2]:

(i) *Complementary*: The data sources are not dependent on each other directly. Each of the data sources views a different region of the observed system and the data they provide is merged to give a more complete picture. A good example of complementary data fusion is the use of multiple cameras, each observing different parts of a scene.

(ii) *Redundant* (or *competitive*): Two or more data sources provide information about the same target. Fusion of data from a single sensor at different instants is possible. Data fusion in this case provides increased precision, confidence and robustness. Several cameras viewing the same scene would be an example of redundant data fusion.

(iii) *Cooperative*: When the data provided by independent data sources are combined to infer typically more complex information, that would not be available from a single data source. A three-dimensional reconstruction of a scene by combining the data from several two-dimensional images of it is an example of cooperative data fusion.

The problem of mass flow estimation, which I am dealing with in this thesis, falls into the second category—the redundant data fusion, as I am trying to estimate one particular quantity, based on all the information available.

Another possible classification criteria for data fusion types is the nature of the input/output data [2]:

(i) *Data in-data out*: The most basic data fusion method, where raw data are both input and output. It typically makes the data more reliable or accurate.

(ii) *Data in-feature out*: Features or characteristics describing the target are extracted from the data sources.

(iii) *Feature in-feature out*: Also called *feature fusion*, it addresses a set of features with aim to refine them or obtain new features.

(iv) *Feature in-decision out*: This kind of data fusion obtains a set of features and provides a decision based on them as output.

(v) *Decision in-decision out*: Also known as *decision fusion*, it fuses input decisions to obtain better or new ones.

As far as this thesis is concerned, the focus will be on *data in-data out* data fusion, as the required output of a mass flow estimator is a data stream to be used in control of the system.

Available data fusion techniques can be classified into three non-exclusive categories [2]: (i) *data association*, i.e. determining the target corresponding to the measurements, (ii) *state estimation*, i.e. determining the state of the target, and (iii) *decision fusion*, which aims to make high-level inference about events produced by the target. Of particular interest to me is the *state estimation*, as my goal is to estimate a state of a hydronic system.

2.3 State Estimation Techniques

As stated before, state estimation techniques have the objective of determining the state of a system. This is done based on observation or measurements. However, there is no guarantee that the measurements are relevant, as they can be influenced by noise. Finding values of the state that fit the measured data as much as possible is the general problem which state estimation is trying to solve. Following now is an overview of the most common state estimation techniques [2].

2.3.1 Maximum Likelihood Estimation

Maximum likelihood (ML) estimation is a technique based on probabilistic theory. Probabilistic estimation is in general appropriate, when the state variables follow an unknown probability distribution. The base of ML is the likelihood function $\lambda(\mathbf{x})$ defined as

$$\lambda(\mathbf{x}) = p(\mathbf{z}|\mathbf{x}) \tag{2.1}$$

i.e. the probability density function of the sequence of observations $\mathbf{z}$ given the value of the state $\mathbf{x}$. The working principle of a ML estimator is maximizing the likelihood function over possible values of $\mathbf{x}$.

The likelihood function can be derived from a model of the sensors used, which can be either analytical or empirical. Finding this model, which is necessary in order to be able to calculate the prior distribution, is the main problem of applying this estimation technique.

2.3.2 Particle Filter

Particle filter is a technique belonging to a class of algorithms referred to as Monte Carlo methods. Specifically, particle filter is sometimes called the sequential Monte Carlo method. The basis of the method is using many random samples called particles to build the posterior density function. The random particles are propagated over time within two phases: the prediction phase and the update phase. In the prediction phase, each particle is modified according to a model with some added noise. After that, in the update phase, the weight each particle has been assigned is reevaluated using the sensor observation. Each iteration is concluded by the resampling step, which discards particles with the low weights, multiplies particles with high weights and often also generates a certain number of new particles for increased robustness.

While being more flexible than other estimation method in coping with non-linear dependencies and non-Gaussian probability densities, the particle filter has some disadvantages. A very large number of particles is required to obtain small uncertainty in the estimation. The optimal number of particles itself is also hard to establish in advance, often making empirical tuning a necessity.

The most popular state estimation method—the Kalman filter—which I use in my thesis, is studied more in depth in the following section.

2.4 Kalman Filtering

One of the most well-known and often-used tools that can be used for stochastic estimation and data fusion is the Kalman filter [5]. It is named after Rudolph E. Kalman, who in 1960 published his seminal paper describing a recursive solution to the discrete-data linear filtering problem [6].

The Kalman filter provides a mathematical framework for inferring the unmeasured variables from indirect and noisy measurements [7]. It is essentially a set of mathematical equations that implement a predictor-corrector type estimator, that is optimal in the sense that it minimizes the estimated error covariance—when some presumed conditions are met [5]. However, even though the conditions necessary for optimality are rarely met, the filter works well in spite of that. It has become a universal tool for integrating different sensor and data collection systems into an overall optimal solution.

2.4.1 Kalman Filter

First, I look at the linear and discrete Kalman filter, as it was originally formulated by Kalman, where the model of the process to be estimated is linear and where the measurements occur and the state is estimated at discrete points in time. It addresses a general problem of trying to estimate a state $\mathbf{x} \in \mathbb{R}^m$ of a discrete-time process with a measurement $\mathbf{z} \in \mathbb{R}^j$ given the input $\mathbf{u} \in \mathbb{R}^l$. The process is governed by the following linear stochastic difference equations

$$\begin{aligned}\mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{v}_{k-1} \\ \mathbf{z}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{w}_k\end{aligned}$$

where $\mathbf{A}$ is a $m \times m$ matrix relating the state at the previous time step $k-1$ to the state at the current time step k, $\mathbf{B}$ is a $m \times l$ matrix relating the input $\mathbf{u}$ to the state $\mathbf{x}$ and $\mathbf{C}$ is a $j \times m$ matrix relating the state $\mathbf{x}$ to the measurement $\mathbf{z}$. The random variables $\mathbf{v}$ and $\mathbf{w}$ represent the process and measurement noise respectively. They are assumed to be independent, white and to follow normal probability distributions as in

$$\begin{aligned}p(\mathbf{v}) &\sim \mathcal{N}(0, \mathbf{V}) \\ p(\mathbf{w}) &\sim \mathcal{N}(0, \mathbf{W})\end{aligned}$$

i.e. with zero means and covariances $\mathbf{V}$ and $\mathbf{W}$ respectively, where $\mathbf{V}$ and $\mathbf{W}$ are the process noise and measurement noise covariance matrices respectively.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at the next time step and then obtains feedback in the form of measurements [5]. These two actions are often called the *temporal update* (or *prediction*) and the *measurement update* (or *correction*). To be able to correctly describe them, I need to define $\hat{\mathbf{x}}_k^- \in \mathbb{R}^m$ to be the *a priori* state estimate at step k given the knowledge of the process prior to step k, and $\hat{\mathbf{x}}_k \in \mathbb{R}^m$ to be the *a posteriori* state estimate at step k given measurement $\mathbf{z}_k$. Analogously, let $\mathbf{P}_k^-$ be the *a priori* error covariance and $\mathbf{P}_k$ be the *a posteriori* error covariance.

Now I have everything ready to write down the temporal update and the measurement update as two sets of equations. First, the *temporal update* equations are responsible for projecting the current state and error covariance estimates in time to obtain the *a priori* estimates for the next time step.

Temporal Update

1. Project the state ahead
$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}u_k$$

2. Project the error covariance ahead
$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{V}$$

Second, the *measurement update* equations incorporate a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

Measurement Update

1. Compute the Kalman gain
$$\mathbf{K}_k = \mathbf{P^-}_k\mathbf{C}^T\left(\mathbf{C}\mathbf{P}_k^-\mathbf{C}^T + \mathbf{W}\right)^{-1}$$

2. Update estimate with measurement z_k
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(z_k - \mathbf{C}\hat{\mathbf{x}}_k^-\right)$$

3. Update the error covariance
$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{C}\right)\mathbf{P}_k^-$$

The $m \times j$ matrix $\mathbf{K}$ is chosen to be the gain that minimizes the *a posteriori* error covariance and is generally called the Kalman gain. The $\mathbf{I}$ is the identity matrix.

The details of derivation of these equations are beyond the scope of this text and can be found in literature. A good overview is given in [5], while a more in depth explanation can be found in [7].

Describing the Kalman filter operation in words rather than with equations, it can be said that the filter performs the *temporal* and *measurement updates* recursively on its earlier results. In the *temporal update* phase, it predicts the next estimated value, based on the previous estimate and the knowledge of the system model. This prediction involves not only the update of the estimated mean, but also of the covariance, which is a measure of uncertainty of the estimate. The result of the prediction are the so called *a priori* estimates of state mean and error covariance.

The first task during the measurement update is to minimize the *a posteriori* error covariance by choosing the Kalman gain value. After actually taking the measurement of the process, this Kalman gain is used to correct the prediction of the estimated mean, creating the *a posteriori* state mean estimate. Lastly, the error covariance is also updated according to the Kalman gain, yielding the *a posteriori* error covariance estimate.

At the next time step, the two updates are repeated on the *a posteriori* results of the previous step, giving a recursive nature to the Kalman filter operation. This is a very appealing feature of the filter, as it eases its practical implementation.

2.4.2 Extended Kalman Filter

The standard Kalman filter is only applicable to systems, which can be described linearly. However, in real world there are many system of highly non-linear nature to which it would be convenient to be able to use it. This problem arose as early as in the fall of 1960 when Kalman presented the filter to Stanley F. Schmidt of the Ames Research Center of NASA [7]. Recognizing its applicability the trajectory estimation and control problem for the Apollo project, Schmidt and his colleagues at Ames developed what is now known as the extended Kalman filter, which has been used ever since for many real-time non-linear applications of Kalman filtering.

The key working principle of the extended Kalman filter is linearization about the current mean and covariance [5]. Partial derivatives of the process and measurement functions are used to compute the estimate of covariance. Keeping the same notation as with the linear Kalman filter, the equations governing the process are now

$$\mathbf{x}_k = f\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_{k-1}\right)$$
$$\mathbf{z}_k = h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)$$

where f is a non-linear dynamic model function relating the previous state, process noise and current input to the state at the current time step and h is a non-linear measurement function relating the current state, input and measurement noise to the current measurement. Note, that in practice the individual values of the process and measurement noises are not known at each time step so the state and measurement are are approximated without them.

Another important thing to note is that there is a fundamental flaw in this approach, since the distributions of the various random variables are no longer normal after undergoing the non-linear transformations. Thus, the extended Kalman filter only approximates the optimality of the estimation by linearization [5]. Although there are other non-linear versions of the Kalman filter solving this problem, such as the unscented Kalman filter, the extend version is used throughout this thesis.

The equations of the *temporal update* as well as the equations of the *measurement update* in the extended Kalman filter are very much like those in the standard Kalman filter. The difference stems from not having explicit state matrix $\mathbf{A}$, input matrix $\mathbf{B}$ and measurement matrix $\mathbf{C}$. When updating the state mean estimate these are replaced with the non-linear function f in the case of matrices $\mathbf{A}$ and $\mathbf{B}$ and with the function h in the case of matrix $\mathbf{C}$. To update the estimate of covariance, partial derivatives of the non-linear functions f and h are needed to replace the matrices. In practice, the partial derivatives of f and h are gathered in the Jacobian matrices $\mathbf{F}$ and $\mathbf{H}$ respectively.

After the above mentioned changes, the equations for the *temporal update* have the following form.

Temporal Update

1. Project the state ahead
$$\hat{\mathbf{x}}_k^- = f\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0\right)$$

2. Project the error covariance ahead
$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{V}$$

The equations for performing the *measurement update* in the extended Kalman filter are the following.

Measurement Update

1. Compute the Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T \left(\mathbf{H} \mathbf{P}_k^- \mathbf{H}^T + \mathbf{W} \right)^{-1}$$

2. Update estimate with measurement z_k

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - h \left(\hat{\mathbf{x}}_k^-, \mathbf{0} \right) \right)$$

3. Update the error covariance

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H} \right) \mathbf{P}_k^-$$

As far as the actual operation of the extended Kalman filter is concerned, it is virtually identical as the operation of the regular Kalman filter. The recursive nature of repeating the same updates on the previous results remains the same, as does the structure of these updates.

Writing down the complete set of equations of the extended Kalman filter concludes my study of data fusion techniques. At this point, I shift focus to the study of hydronic systems, which is the topic of the following chapter.

Chapter 3

Hydronic System Model

In this chapter, I aim to create a model of a hydronic system. Having such a model will facilitate the development and testing of a mass flow estimator as well as the controller. To create the model, I first form a mathematical model, describing a hydronic systems dynamics based on first principles. This model is fundamental for the mass flow estimator, as it needs to be able to predict the internal state of the system based on few observations.

3.1 System Model

When modelling a hydronic system, one must consider several variables describing its state. The most important ones are mass flow q, and pressure p. Mass flow (or just flow) is the amount of fluid passing through an area per unit of time. Pressure is the force exerted by a fluid per unit area. These two variables have a nice parallel in electrical systems, commonly referred to as the electronic-hydraulic analogy, where instead of flow there is electric current and instead of pressure there is electric potential [8]. Just as in electrical systems, one is typically only interested in electric potential differences (i.e. voltages), when describing fluid systems, one is typically interested in the pressure differences.

Taking the analogy even further, it is possible to model a hydronic system similarly to how electrical circuits are modelled. Therefore, I can start the description of a hydronic system by drawing a circuit analogous to an electrical circuit and describing its behaviour with a differential equation in a similar way to how I would describe the electrical one. Here, I sketch such a circuit, describe its elements and finally put together a differential equation for the motion of a fluid within.

Dynamics of a hydronic system with many pumps affecting each other may be challenging to describe, due to the pumps influencing mass flow through each other. A way around this problem lies in designing the hydronic system with many pumps in such a way, that the influence between the individual pumps is negligible and can thus be ignored. One such design of the hydronic system may be that there is a primary loop which contains the main circulator and the heat source. Each of the other pumps is connected to this primary loop in parallel, forming several secondary loops. Moreover, the input and output pipes of the secondary loops are connected very close together, making the pressure difference between the connections in the primary circuit negligible. In such configuration, there is almost no pressure interaction not only between the individual secondary circuits, but also between each of the secondary circuits and the primary circuit itself, as putting the input and output pipes close together creates something analogous to a short circuit.

The biggest advantage of this design is that the secondary circuits containing the pumps, which are of interest to me, can be modelled as separate systems altogether. Therefore, the hydronic system I am interested in modelling is very simple. Most importantly, it contains a pump or more generally speaking a non-ideal pressure source. An electrical analogy for the pressure source would be a voltage source. Additionally, the hydronic system I am describing contains resistance and inertance. Resistance describes the difficulty with which a fluid moves through the system, while inertance arises as the result of the lumped momentum of a flowing fluid.

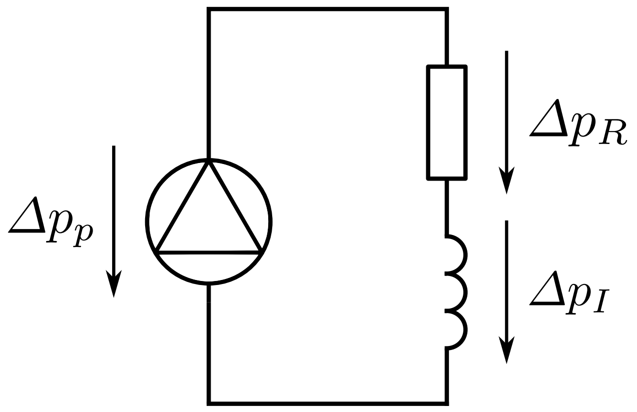

Figure 3.1: Hydronic system with a pump modelled as an electric circuit. Hydraulic resistance and inertance represented by symbols for resistor and inductor respectively. Arrows indicate the direction in which the pressure decreases.

A circuit representing a hydronic system with a pump is shown on Figure 3.1.

Note that the resistance and inertance, although they are spread in the system as a whole, are represented as a single resistance element and a single inertance element. The symbols of resistor and inductor are used for the resistance and inertance elements respectively.

To construct a differential equation for the motion, or mass flow, of a liquid in the system, I need to look at the pressure. The key thing here is to remember, that since there is a closed loop, the pressure of interest is the pressure difference not the absolute value of the pressure. Thus, the pressures $\varDelta p_p$, $\varDelta p_R$ and $\varDelta p_I$ marked in the Figure 3.1 refer to the pressure difference across each individual element rather than to the absolute value of pressure in the element. The arrows denote the direction in which the pressure decreases, i.e. they point from a region of higher absolute pressure towards a region of lower absolute pressure.

Just as in electrical circuits, where the Kirchhoff's voltage law tells us that the directed sum of voltages around a closed network is zero, I can assume here, that the directed sum of pressure differences around the sketched hydronic network is zero. In other words, I assume that the pressure $\varDelta p_p$ generated by the pump is equal to the total pressure loss around the system, which is partly due to resistance ($\varDelta p_R$) and partly due to inertance ($\varDelta p_I$). Expressing this in an equation yields

$$\varDelta p_p - \varDelta p_R - \varDelta p_l = 0 \tag{3.1}$$

Looking at the individual elements, I can express the pressures in more detail and substitute into (3.1).

3.2 Pipe Model

The main feature of a pipe in a hydronic system is the resistance it imposes on the passing fluid. Fluid friction resistance in hydronic networks with real-world elements behaves nonlinearly and is generally modelled as quadratic [8], following a law such as

$$\varDelta p_R = Rq^2 \tag{3.2}$$

where $\varDelta p_R$ is the pressure difference across the resistance, R is the hydraulic resistance, also called hydraulic resistance and q is the mass flow rate through the resistance.

3.3 Inertance Model

Inertance arises as the result of the momentum of a flowing fluid. If I consider incompressible fluid flowing down a pipe with constant cross sectional area A, I can treat it as a mass element subjected to two forces due to the pressure from each side. Figure 3.2 depicts such a situation, where a fluid mass element of length L flows down a pipe with velocity v, while being subject to forces F_1 and F_2.

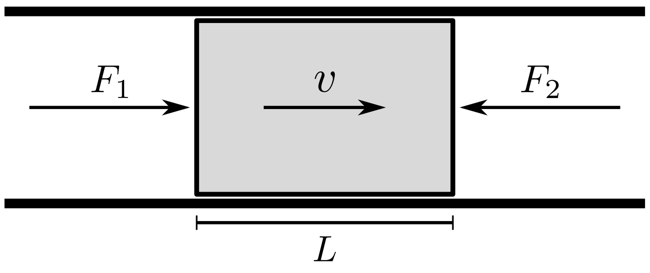

Figure 3.2: Fluid mass element (grey) of length L moving through a pipe at speed v, while being subject to forces F_1 and F_2.

Applying Newton's second law of motion to the element, I can write

$$m\dot{v} = F_1 - F_2 \tag{3.3}$$

where m is the mass of the fluid element and $\dot{v}$ is the rate of change of its velocity. Since the two forces F_1 and F_2 are due to pressure, the well known formula $F = pA$ can be used to rewrite the equation (3.3) into

$$m\dot{v} = A(p_1 - p_2) = A\Delta p_I \tag{3.4}$$

where A is the cross sectional area of the pipe, p_1 and p_2 are the pressures on the left and right side of the fluid element respectively and Δp_I is their difference. Considering the fact that the volume of the fluid element is $V = LA$ and that mass can be expressed as $m = \rho V$, where ρ is the fluid density, equation (3.4) can be modified to

$$\rho LA\dot{v} = A\Delta p_I \tag{3.5}$$

Noting that mass flow can be calculated as $q = \rho vA$ and therefore that rate of change of mass flow can be calculated as $\dot{q} = \rho\dot{v}A$, equation (3.5) can be further modified to yield

$$\dot{q}L = A\Delta p_I \tag{3.6}$$

Now, the pressure difference can be expressed, giving

$$\Delta p_I = \frac{L}{A}\dot{q} \tag{3.7}$$

This equation is very much like the equation for an inductor, which leads to the definition of inertance as $I = \frac{L}{A}$. Given that, the pressure loss due to the inertance can be finally written as

$$\Delta p_I = I\dot{q} \tag{3.8}$$

3.4 Pump Model

When the pump operates, energy is added to the shaft in the form of mechanical energy. It is then converted by the impeller to internal energy, i.e. static pressure, and kinetic energy, i.e. velocity. This process is described by the most important equation connected to pumps—the Euler's pump equation [9]. As shown in (3.9), the equation can be expressed as the sum of three contributions: (i) the static head due to the centrifugal force, (ii) the static head due to the velocity change through the impeller and (iii) the dynamic head.

$$H = \underbrace{\frac{U_2^2 - U_1^2}{2g}}_{\substack{\text{Static head} \\ \text{(centrifugal force)}}} + \underbrace{\frac{W_1^2 - W_2^2}{2g}}_{\substack{\text{Static head} \\ \text{(velocity change)}}} + \underbrace{\frac{C_2^2 - C_1^2}{2g}}_{\text{Dynamic head}} \tag{3.9}$$

where H is the total pump head, U is the tangential velocity of the impeller, W is the velocity of the fluid relative to the impeller, C is the absolute velocity of the fluid flowing through the impeller and g is the gravitational acceleration. The lower indices 1 and 2 of the velocities refer to their localization at the impeller inlet and outlet respectively. The total pump head is defined as

$$H = \frac{\Delta p_p}{\rho g} \tag{3.10}$$

where Δp_p is the pressure difference across the pump, ρ is the fluid density and g is the gravitational acceleration.

The above described Euler's pump equation describes the operation of an ideal pump. In reality, the pump performance is lower then predicted by the equation, because of a number of mechanical and hydraulic losses in impeller and pump casing. While the mechanical losses are simply caused by mechanical friction, the hydraulic losses have multiple different causes including flow friction, mixing, recirculation, leakage and incidence. Figure 3.3 shows the effect the various losses have on the pump characteristic. A very nice overview of all the loss types is given in [9].

Due to the fact, that there is no straightforward way to analytically express the real pressure generated by the pump, its characteristic is usually approximated by

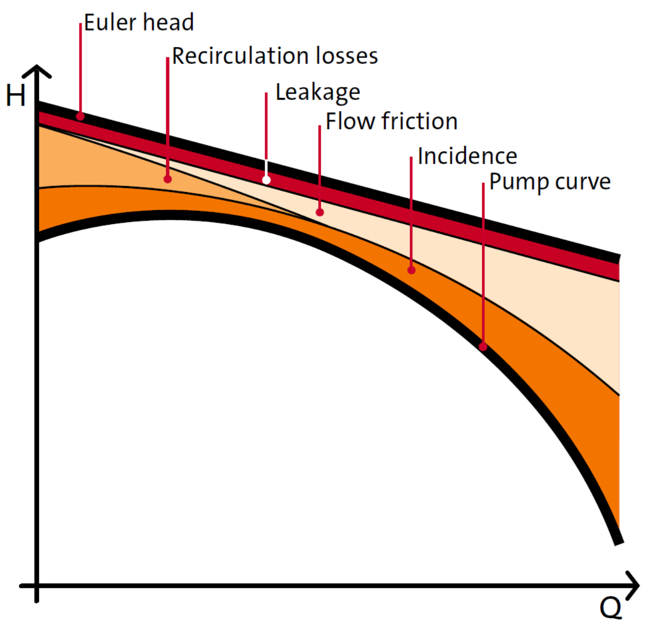

Figure 3.3: Reduction of theoretical Euler head characteristic of the pump due to losses. The real characteristic after taking the various losses into account is denoted as Pump curve in the plot. The horizontal axis shows volumetric flow (Q), the vertical axis shows head (H). Taken from [9].

a polynomial [10]. For the purposes of this model, I use the following polynomial of the second degree.

$$\varDelta p_p = a_2 q^2 + a_1 s q + a_0 s^2 \tag{3.11}$$

where p_p is the pressure generated by the pump, q is the mass flow through the pump, s is the pump speed and the coefficients a_2, a_1 and a_0 are called the pump coefficients. The pump speed is related to the angular velocity of the pump impeller in the following way

$$s = \frac{n}{n_0} \tag{3.12}$$

where n is the angular velocity of the pump impeller and n_0 is the nominal angular velocity of the pump impeller, i.e. the angular velocity at the maximum speed of the pump. Substituting for the speed in equation (3.11) yields

$$\varDelta p_p = a_2 q^2 + a_1 \frac{n}{n_0} q + a_0 \frac{n^2}{n_0^2} \tag{3.13}$$

3.5 Hydronic System Model

Having found the expression for each of the pressure differences, I can now go back to the equation (3.1) and modify it by substituting for $\mathit{\Delta} p_R$, $\mathit{\Delta} p_I$ and $\mathit{\Delta} p_p$ using equations (3.2), (3.8) and (3.13).

$$a_2 q^2 + a_1 \frac{n}{n_0} q + a_0 \frac{n^2}{n_0^2} - R q^2 - I \dot{q} = 0 \tag{3.14}$$

Expressing the rate of change of mass flow and grouping the terms together where possible yields

$$\dot{q} = \frac{a_2 - R}{I} q^2 + \frac{a_1}{I n_0} n q + \frac{a_0}{I n_0^2} n^2 \tag{3.15}$$

The equation (3.15) is the differential equation describing the motion of the fluid flowing in the hydronic system, which I set out to find at the beginning of this chapter. Next chapter follows up by discussing testing of the hydronic system model created here.

Chapter 4

Test-bench

This chapter deals with test-benches. First, I take a closer look into the study of physical properties of pumps, i.e. the most important part of the hydronic system for the estimation. In this study, I utilize a physical model of a hydronic system with a real pump installed. After that, I also set up a model in Simulink, which enables me to run simulations of the hydronic system's operation and to verify theoretical results.

4.1 Pump Electric Power Characteristic

The most important feature of the pump is its power characteristic, i.e. the relation between the electric power the pump consumes and the mass flow it generates. To study the power characteristic of a pump, I was able to use measurements from a real pump installed in a hydronic network model. In this section, I study the measurements from a statistical point of view and extrapolate a general polynomial representation of the measured characteristic.

4.1.1 Test-bench Set-up

The data with which I work in this section have been gathered using a physical hydronic network model built at the Department of Control Engineering of the Czech Technical University (see Figure 4.1). The test-bench included a boiler, a heat exchanger, two pumps and a throttling valve, with one of the pumps, and the boiler in a primary circuit and the second pump, the heat exchanger and the throttle valve in a secondary circuit. The main measured quantities of the test-bench were the pump's input electric current and voltage along with the revolutions of the pump impeller and the mass flow through the secondary circuit. There was also a thermometer measuring the temperature of the fluid circulating in the system.

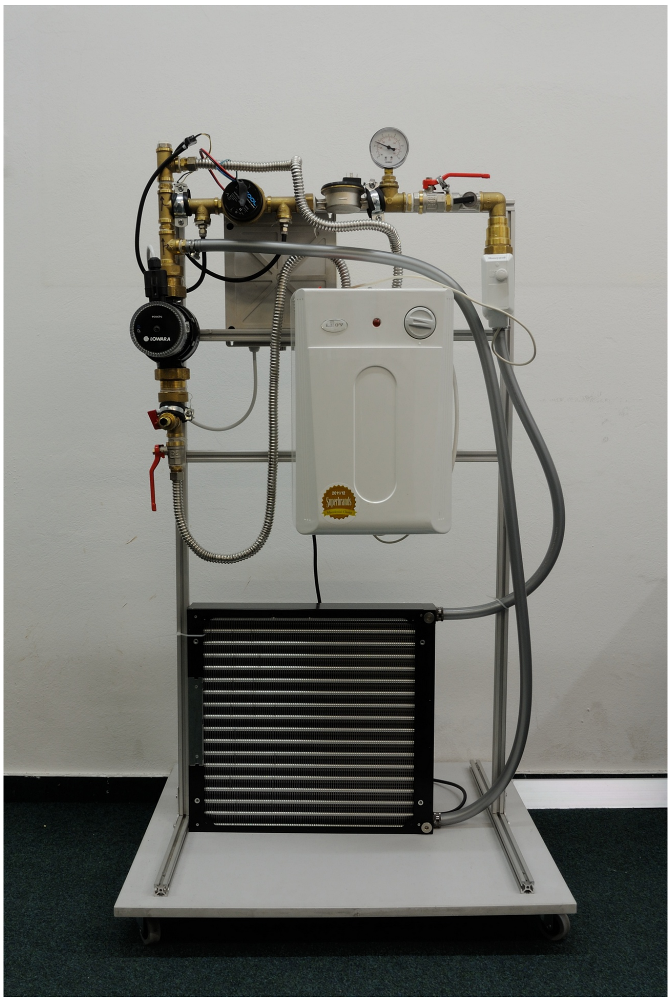
Figure 4.1: Physical hydronic network model.

The control of the whole system was done by a Nucleo board. The same board also took care of collecting the data and sending them to a connected computer running either a simple serial terminal programme or a complex Simulink model. In case that the Simulink model was used, the collected data were saved to the computer's hard drive.

4.1.2 First and Second Moment

Since electric power consumed by the pump is not measured directly, it needs to be calculated. When calculating the electric power, I need to consider the statistical properties of the measured quantities, which are in this case electric current and voltage, and the effects they will have on the resulting value of power. I am particularly interested in the standard deviation.

If I had the exact current and voltage values, I could calculate the power by simply using the well known formula

$$P = UI,$$

where P is power, U is voltage and I is current. However, since I only have sets of measured values affected by measurement errors, I need to take a statistical approach when calculating power. Specifically, I am interested in calculating mean and variance, i.e. the first and second moment.

I am assuming that voltage and current are not independent variables, so to calculate the mean of their product I use the formula for the product of two correlated random variables

$$\mathrm{E}(X_1 X_2) = \mathrm{E}(X_1)\,\mathrm{E}(X_2) + \mathrm{Cov}(X_1, X_2)$$

as given for example in [11]. In this case, I therefore get

$$\mathrm{E}(P) = \mathrm{E}(U)\,\mathrm{E}(I) + \mathrm{Cov}(U, I) \tag{4.1}$$

The formula given in [11] for calculating the second moment, the variance, of a product of two dependent variables is a bit more elaborate, but still rather straightforward, as nothing more than the cross covariance of the two variables is needed in the calculation.

$$\begin{aligned}\mathrm{Var}(X_1 X_2) = {} & \mathrm{Var}(X_1)\,\mathrm{Var}(X_2) + \mathrm{E}(X_1)^2\,\mathrm{Var}(X_2) + \mathrm{E}(X_2)^2\,\mathrm{Var}(X_1) \\ & + \mathrm{Cov}(X_1^2, X_2^2) - \mathrm{Cov}(X_1, X_2)^2 - 2\,\mathrm{Cov}(X_1, X_2)\,\mathrm{E}(X_1)\,\mathrm{E}(X_2)\end{aligned}$$

In this case, I therefore get

$$\begin{aligned}\mathrm{Var}(P) = {} & \mathrm{Var}(U)\,\mathrm{Var}(I) + \mathrm{E}(U)^2\,\mathrm{Var}(I) + \mathrm{E}(I)^2\,\mathrm{Var}(U) \\ & + \mathrm{Cov}(U^2, I^2) - \mathrm{Cov}(U, I)^2 - 2\,\mathrm{Cov}(U, I)\,\mathrm{E}(U)\,\mathrm{E}(I)\end{aligned} \tag{4.2}$$

Figures 4.2 and 4.3 show the resulting means and standard deviations (which are the square root of the calculated variances) at the measured points respectively.

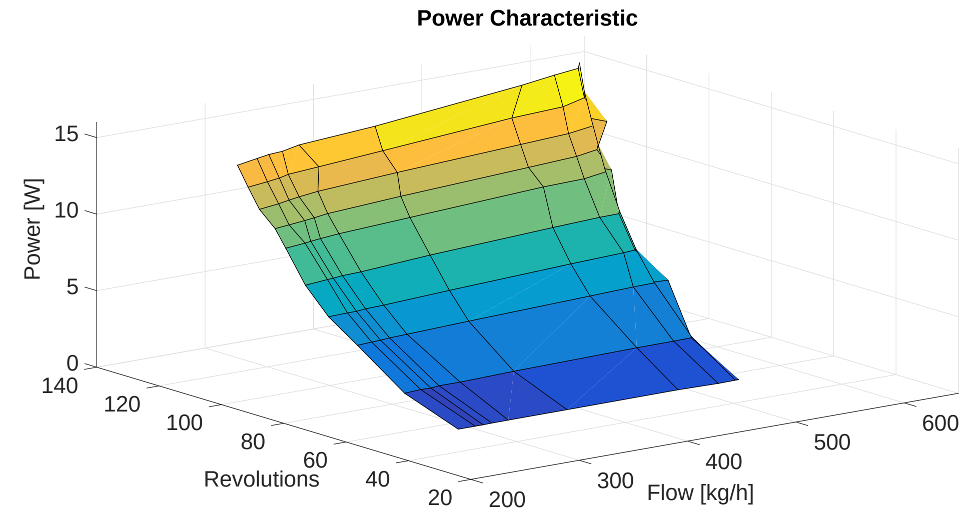

Figure 4.2: Power mean at the measured points of the pump power characteristic.

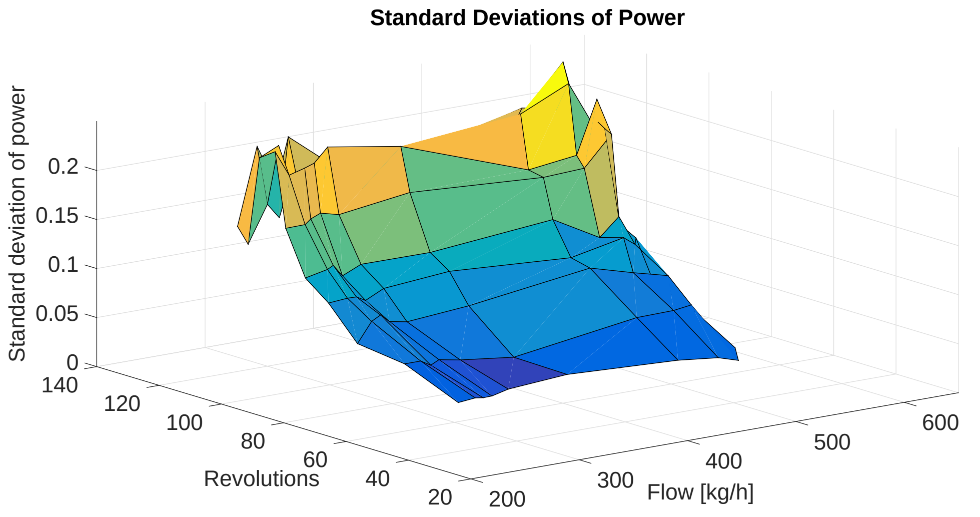

Figure 4.3: Standard deviations of power at the measured points of the pump power characteristic.

In Figure 4.2, it can be seen, that the power data show a linear growth trend with increasing flow and a quadratic growth trend with increasing revolutions. Observations, which as the reader will be able to see later, are true for Figure 4.3 as well.

4.1.3 Data Properties

Throughout this section, I assume, that the measured data for voltage and current as well as the derived power all follow the normal Gaussian distribution. To verify this assumption, I made several tests on the data. First, I plotted a histogram for each measured data point and tried to fit it with a Gaussian bell curve. From these plots, I was able to empirically say, that the data roughly follow normal distribution. Figure 4.4 shows a sample of the data histograms with fitted Gaussians. (Only a sample has been included here for better clarity in print.) The black arrows on the bottom and on the left show the general trends in measured revolutions and flow respectively.

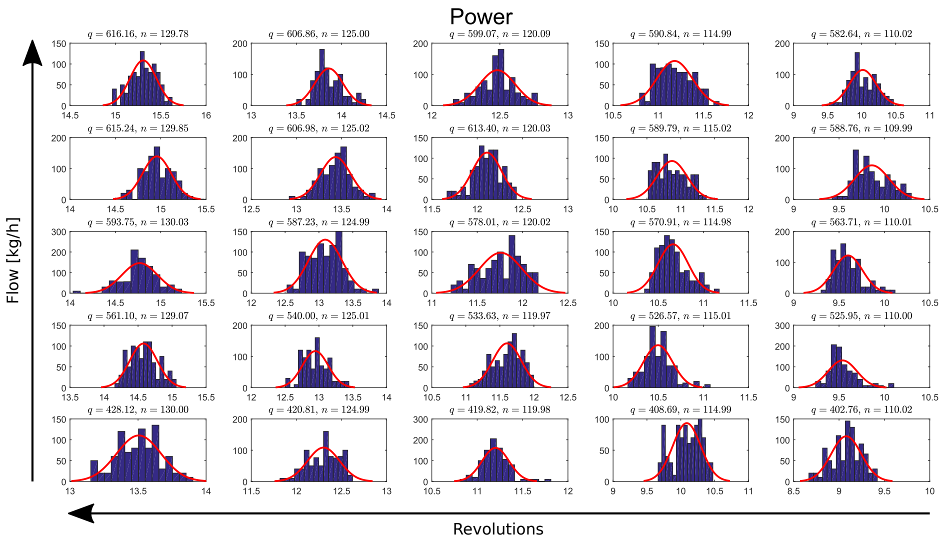

Figure 4.4: Power measurement histograms with fitted Gaussians. Revolutions per second (n) of the pump impeller increase from right to left, while mass flow (q) increases from bottom to top as indicated by the arrows.

To verify the normality assumption on the data further, I also used several statistical normality tests, conveniently implemented in MATLAB out of the box. To be specific, I ran the Chi-square goodness-of-fit test [12], the Lilliefors test [13] and the One-sample t-test [14].

The Chi-square goodness-of-fit test, also known as the Pearson's chi-squared test, determines whether a data sample comes from a probability distribution (in our case the normal distribution), with parameters estimated from the data. The test groups the data into bins, calculates the observed and expected counts for those bins and computes the Chi-square test statistic

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i}$$

where N is the total number of observations, O_i is the number of observations of type i and E_i is the expected observation count of type i based on the hypothesized distribution [15].

The Lilliefors test is a modification of the Kolmogorov-Smirnov test, which is based on the largest vertical difference between the hypothesized and empirical distribution. The Lilliefors test uses parameters based on the data, rather then manually specified ones [16].

The One-sample t-test tests the null hypothesis that the data come from a normal distribution with a given estimated mean and unknown variance. The test statistic is

$$t = \frac{\sqrt{n}(\bar{x} - \mu)}{S_n}$$

where n is the sample size, $\bar{x}$ is the sample mean, μ is the hypothesized mean and S_n is the sample standard deviation [15]. The test decision comes from calculating the probability of observing the test statistic under the null hypothesis.

The results of the test at the standard 5 % significance level were mixed. The first two tests generally rejected the hypothesis, that the data come from a normal distribution, while the One-sample t-test unequivocally confirmed it. This is not a very definite assurance of the correctness of my assumption. However, taking into account the histogram test, I consider it to hold.

Now that the data have been analysed, I can proceed to showing the pump characteristic in which I am interested the most—the power to flow characteristic. As can be seen in Figure 4.5, the characteristic has quite nice properties for high power situations, being reasonably sloped and therefore well usable. In low power circumstances however, especially around 2 W, the characteristic is almost flat and any estimate of mass flow based on that region of the characteristic therefore suffers from great uncertainty.

4.1.4 Polynomial Fitting

So far, I have only worked with a sparse characteristic comprised of the measured data points. However, to get a more complete and universally usable characteristic, I have to somehow interpolate the missing values between the sparsely

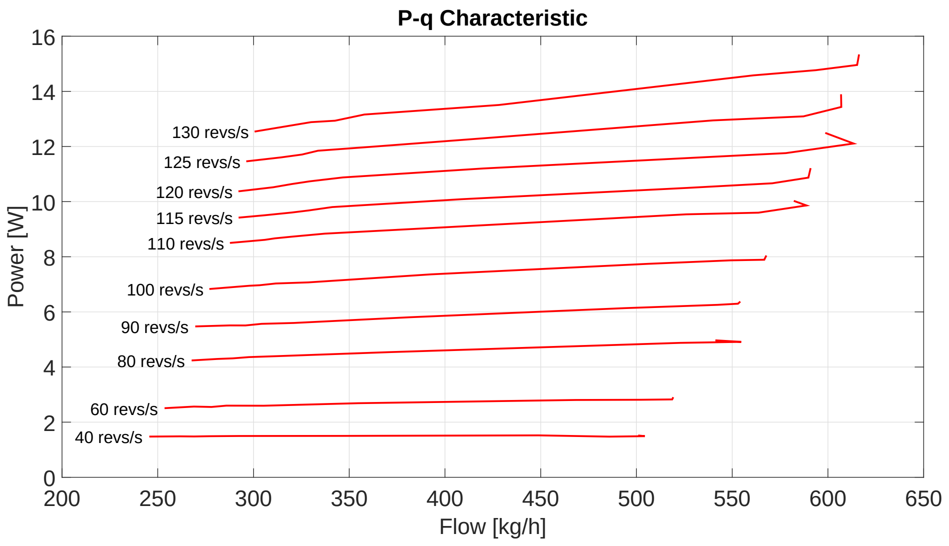

Figure 4.5: P-q (power-flow) characteristic. The individual lines each represent the characteristic at specific revolutions per second (revs/s) count of the impeller as marked on the plot to the left of each characteristic.

distributed points. A method I chose to accomplish this is multidimensional polynomial fitting.

As I observed before from the three dimensional pump characteristic (Figure 4.2), the data show a linear growth trend in one dimension and a quadratic growth trend in the other dimension. Therefore, the characteristic can be fitted with the following polynomial, consisting of monomials with two variables of degrees up to one and two respectively.

$$P_e(q,n) = b_0 + b_1 q + b_2 qn + b_3 n + b_4 n^2 \tag{4.3}$$

where P_e is the electric power, q is the mass flow, n is the angular velocity of the pump's impeller and b_0, b_1, b_2, b_3 and b_4 are the fitted polynomial's coefficients.

To perform the fitting itself, I used an extension of the standard MATLAB `polyfit` function, the `polyfitn`, published at [17]. The result of the polynomial fitting to the three dimensional power characteristic is shown on Figure 4.6. The fitting turned out to be very precise, with the root-mean-square error $(RMSE)$ not exceeding 0.15 W and the coefficient of determination (R^2) being 99.86 %.

Polynomial of the same structure (i.e. with two independent variables of degrees one and two respectively) proved to be the best fit for the standard deviation of power as well. Figure 4.7 shows the result of fitting a polynomial to the stan-

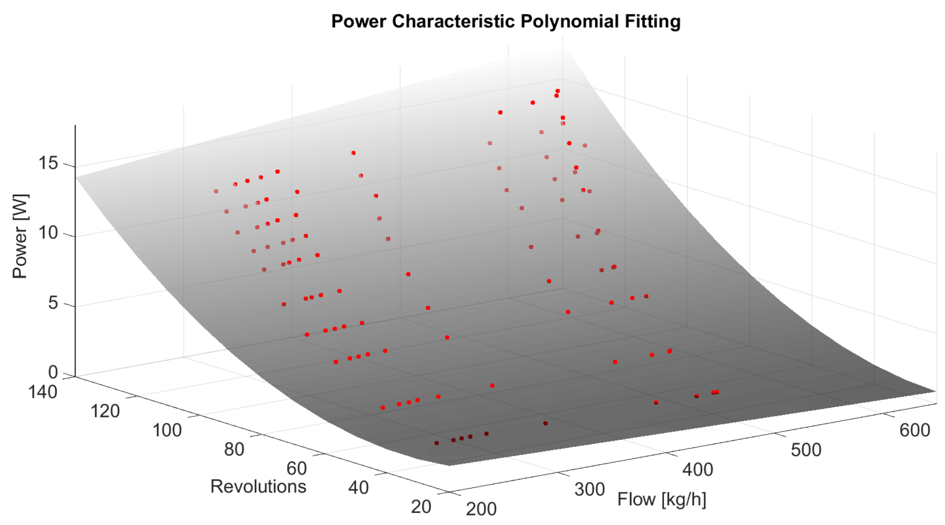

Figure 4.6: Polynomial fitting of the pump mean power characteristic; $RMSE = 0.1469\,\mathrm{W}$, $R^2 = 99.86\,\%$.

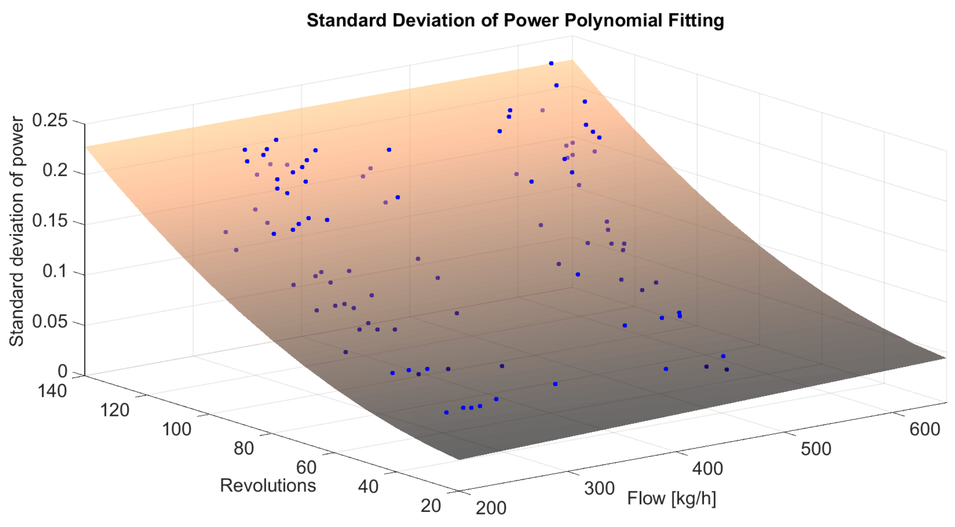

Figure 4.7: Polynomial fitting of the pump standard deviations of power characteristic; $RMSE = 0.0256\,\mathrm{W}$, $R^2 = 78.55\,\%$.

dard deviation of power graph. In this case, the fit was not quite as precise, but still provided a good approximation with the $RMSE$ being $0.02\,\mathrm{W}$ and the R^2 reaching $78.55\,\%$.

4.2 Simulink Model

To be able to run simulations of the hydronic system, I used a Simulink model. However, instead of creating a model from scratch, I worked with an existing Simulink test-bench previously created by my colleagues at the Department of Control Engineering, which I adjusted extensively to fit my needs.

The Simulink test-bench is constituted by a primary and a secondary hydronic circuit, with only the primary being relevant to my needs. The primary circuit consists of two parts: the hydraulic part and the thermal part. The thermal part which mainly includes the boiler is not very important for my goals either. The crucial part is in fact the hydraulic part, as it encompasses everything important for the mass flow. There is a model of a pump, a resistance and it also includes the dynamics of inertance.

4.2.1 Pump Electric Power Model

The first important addition to the hydronic model was the electric power model of the pump. Since the most important information for the mass flow estimator comes from the measurement of the electric power supplied to the pump, the model needs to somehow emulate it. To do so is the purpose of the pump electric power model.

The result of the pump analysis described in the previous section was a polynomial fitted to the electric power characteristic of the pump. Because there is no actual electric power which would drive the pump in the Simulink model and which could be observed, it has to be estimated based on this characteristic. Figure 4.8 illustrates the procedure. The current simulated mass flow, which is a known quantity throughout the simulation, and the current pump speed, which is also known during simulation and easily precisely measured in a real system, serve as the input information. A two dimensional flow-power characteristic appropriate at the given pump speed is first selected from the polynomial. The mass flow value is then transformed using this characteristic to the expected mean value of the electric power.

To imitate the statistical properties of the real power measurement, the knowledge of the standard deviation is then used. The appropriate value of standard deviation is first picked out by evaluating the polynomial fitted to the standard deviation of power in a similar fashion in which it has been done with the power mean. The transformed mean value is then obfuscated by adding a random noise which has this appropriate standard deviation to it.

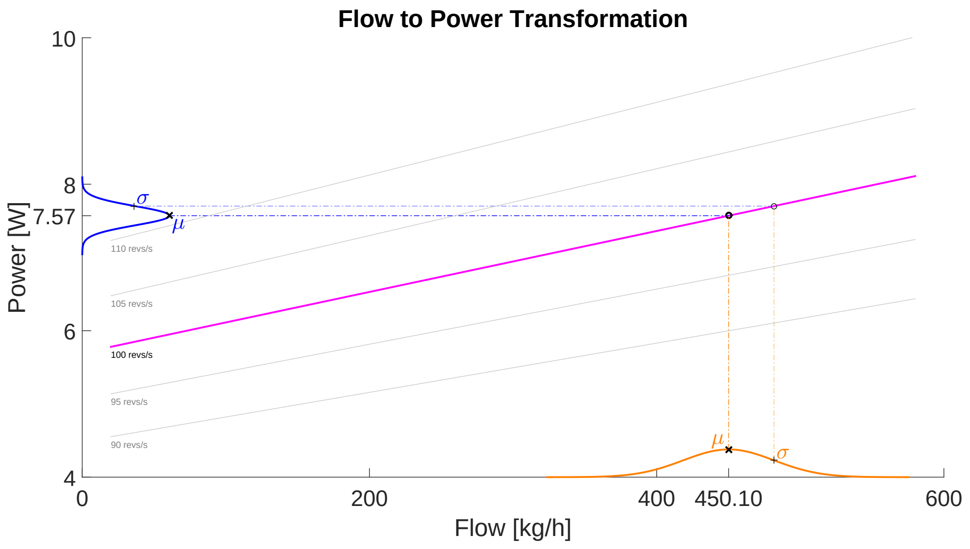

Figure 4.8: Flow to power transformation. Transformed mean value is marked by μ; transformed value at the distance of one standard deviation from the mean value is marked by σ.

Going back to the Simulink model, as shown on Figure 4.9, it takes in the mass flow and the revolutions values as inputs and evaluates the two fitted polynomials based on these inputs. The results of the evaluations are then combined, with the use of a generated normally distributed random number.

With the ability to produce measurements of electric power now added to the simulator, it is a good representation of a real hydronic system test-bench. It contains everything that I need to test a mass flow estimator. In the following chapter, I deal with designing such an estimator.

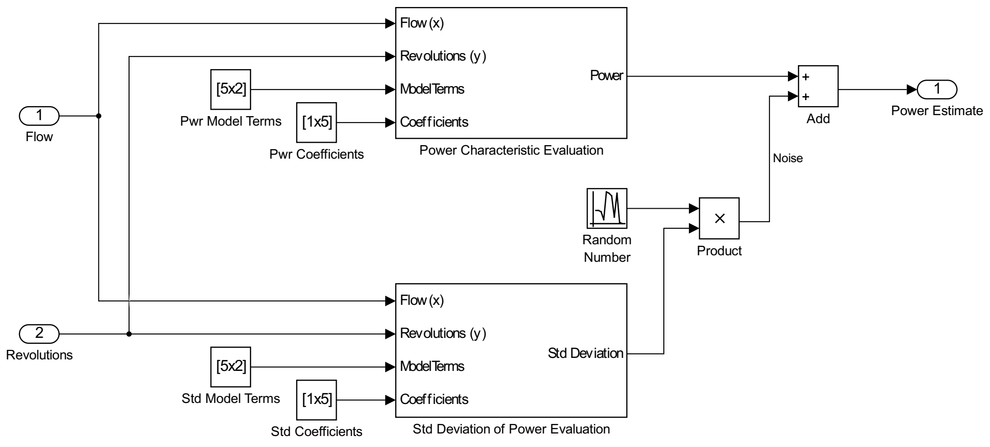

Figure 4.9: Pump electric power model schema. The model evaluates the polynomial fitted to the power characteristic giving an estimate of the electric power measurement.

Chapter 5

Mass Flow Estimator

In this chapter, I discuss my implementation of the mass flow estimator. I start of with a simple measurement based Kalman filter estimator, which has no knowledge of the system dynamics. Later, I build on it to create the final estimator, which is essentially an extended Kalman filter (EKF), which utilizes a state-space model. In the process, I overview the resources I used when implementing the EKF in MATLAB and later in Simulink. I also deal with the numerical robustness of my implementation of the EKF, taking advantage of alternative implementation methods.

5.1 MATLAB Toolbox

The first step in the creation of the mass flow estimator is implementing the Kalman filter itself. Thankfully, this has already been done by many others, so I can build on the fruits of their work. A nice toolbox for Kalman filtering, which I have initially chosen to work with, has been created by Simo Särkä, et al. at the Aalto University School of Science [18]. It is called the EKF/UKF Toolbox for MATLAB and it focuses mainly on the extended and unscented versions of Kalman filters and Kalman smoothers. It however also considers regular Kalman filters and provides many miscellaneous functions for discretization of linear systems, probability calculations, etc. Several demo problem solutions demonstrating the functionality of the toolbox are also included—a detail which was of great importance and help in my work.

The main functionality of the toolbox is provided in the functions calculating the temporal update and the measurement update of a Kalman filter. The toolbox includes several versions of these functions, one for each version of the Kalman filter it considers. As an example, for the linear Kalman filter, there is a function `kf_predict` which performs the temporal update and a function `kf_update` which performs the measurement update. Analogously, for the extended Kalman filter using linear approximation, there are the `ekf_predict1` and the `ekf_update1` functions. Input parameters of these functions are used to specify the different variances as well as model of the system to which the Kalman filter is applied. An in depth description of all the toolbox capabilities is available in its documentation [18].

5.2 Estimator with Simplified Linear Model

Perhaps the simplest and easiest way to implement an estimator lies in using a regular linear Kalman filter, which would operate without any knowledge of the non-linear model of the system. The only information such a Kalman filter uses is a noisy measurement of the observed variable, which is in this case the electric power. However, because the observed output equation for electric power (**??**) is not linear and I want to use a linear filter at this stage, I instead transform the noisy measurement of the electric power driving the pump to mass flow using the P-q characteristic of the pump (see Figure 4.5). This is essentially the reverse of the process used to model the electric power in the Simulink model (see Figure 4.8). In the Simulink test-bench, this transformation of the measurement of electric power to the mass flow is done at run-time by the `flowEstimator` block.

The temporal update step of this Kalman filter is based on the assumption, that the update of the state variables is entirely stochastic, following the equation

$$\hat{x}_{k+1}^{-} = x_k + v_k \tag{5.1}$$

where v is the random process noise. As is clear from equation (5.1) the state matrix $\mathbf{A}$ is in this case in fact an identity matrix, meaning that the a priori estimate of the state in mean value remains unchanged, only its uncertainty increases. This leaves the bulk of the correction of the value to the measurement update step of the Kalman filter, making the estimate thoroughly dependent on the measurements them self.

An estimator like this generally does not perform very well compared to an estimator with knowledge of the model. However, in case that the estimated states do not change rapidly and the measurements are not biased it performs sufficiently well. In steady state situations especially, it can perform almost as good as the more advanced estimators, which I discuss further on.

5.3 Extended Kalman Filter Estimator

The logical improvement over the simple Kalman filter estimator described in the previous section is taking advantage of the knowledge of the non-linear system model, which is discussed in detail in Appendix B. With the complete state-space model of the system from the appendix, including the calculated Jacobians, I have all the theory I need to create the improved mass flow estimator. Since this model is a non-linear one, the use of an extended Kalman filter is necessary.

When using the EKF/UKF toolbox, the transition from a linear Kalman filter to the extended Kalman filter is very simple, the substitution of different temporal and measurement update functions being essentially the only thing required to do so. However, the inclusion of the system model presents more work, as both the function f relating the state at the previous time step to the state at the current time step and the function h relating the current state to the current measurement need to be implemented as a MATLAB functions together with functions evaluating their respective Jacobian matrices. Handles to these MATLAB functions are passed to the toolbox functions during estimation for it to be able to calculate the temporal and measurement updates.

After the above has been done, the variance matrices describing the process and measurement noise needed to be set. The variance of the measurement of the electric power driving the pump have been examined previously in Chapter 4, where I have fitted the standard deviation characteristic with a polynomial. The estimator evaluates this polynomial at each time step based on the current input and last estimated state values, so that the variance matrix can be adjusted accordingly. The variance for the process noise has been set empirically with the help of the Simulink model, which is a common practice when implementing Kalman filters. Having done that, the extended Kalman filter estimator which uses all the information available to it is ready to be used.

5.3.1 Numerically Robust Implementation

During development and testing of the non-linear estimator, I often ran into trouble when the calculation of the covariance matrix diverged for various reasons, such as inaccuracies in the model or very noisy input data. Having suspected that this was due to numerical inaccuracies caused by round-off errors, I looked for ways to reduce the sensitivity of the calculations to such errors, reducing the likelihood of such issues in the process. I found the solution in employing alternative implementation methods for the functions performing the temporal update and the measurement update of the extended Kalman filter.

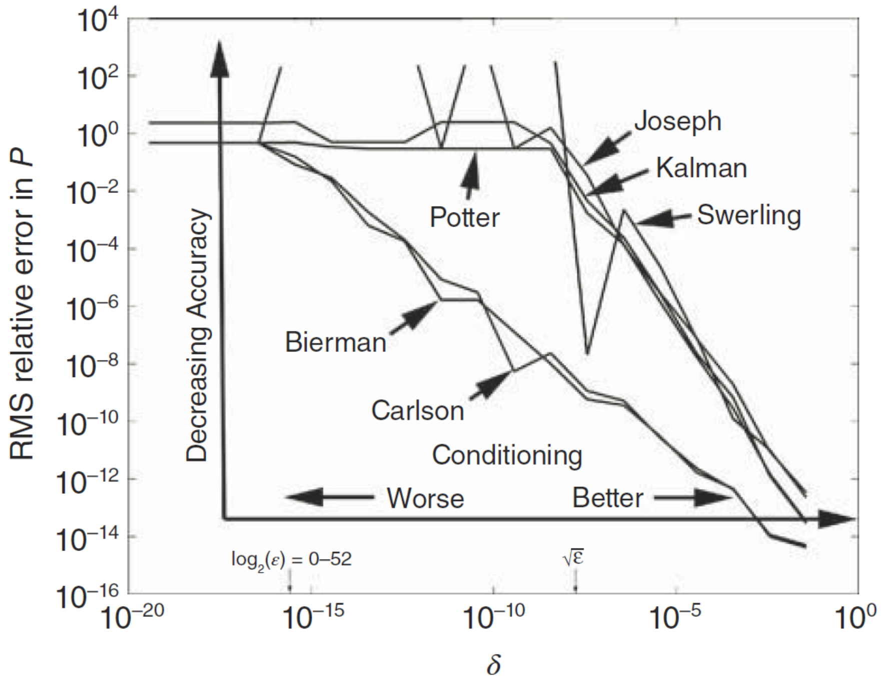

Figure 5.1: Comparison of performance of several methods of implementation of the Kalman filter on a variably ill-conditioned problem. The graph shows the degradation of the Riccati equation observational updates with varying problem conditioning. Taken from [7].

Several different methods for decreasing the sensitivity of the Kalman filter to round-off errors are discussed in detail in [7, Chapter 7]. Figure 5.1 illustrates how the standard Kalman filter and some of the alternative implementation methods perform on a variably ill-conditioned problem. As is noted in [7], for this particular example, the accuracies of the methods labelled "Bierman" and "Carlson" appear to degrade more gracefully than the others as the conditioning worsens.

For my extended Kalman filter estimator, I have chosen to utilize an implementation of the Bierman-Thornton UD filtering, which uses modified Cholesky factors of the covariance matrix P of the state estimation uncertainty for the Kalman filter. The fundamental part of this procedure is the factorization of P to the modified Cholesky factors, which have the form

$$P = UDU^T$$

where U is a unit triangular matrix (i.e. with ones on the diagonal) and D is a diagonal matrix with positive diagonal entries. The Bierman algorithm for the measurement update and the Thornton algorithm for the temporal update in the Riccati equation are then used on the modified Cholesky factors U and D.

Incorporating these methods into the workflow of the estimator effectively means replacing the functions performing temporal and measurement update from the EKF/UKF Toolbox (i.e. the functions `ekf_predict1` and `ekf_update1`) by functions performing the pair of algorithms described above and prepending them by a function performing the initial UD factorization of the covariance matrix. After this has been done, I had an alternative implementation of the extended Kalman filter estimator, which is more likely to produce accurate results even if the conditioning of the estimation problem becomes bad, although it is theoretically not any better than the standard implementation, if it were to run on a computer with infinite precision.

5.4 Simulink S-Function

To be able to use the estimator at simulation runtime so that it is be able to perform the estimation on-line, I needed to implement it as a Simulink S-Function. S-Functions provide a way to embed complex code into a Simulink model. Since all the previously written code was in MATLAB, I used the Level-2 MATLAB S-Functions, which allow for creation of custom blocks with multiple input and output ports and capable of handling any type of signal produced by a Simulink model using the MATLAB language.

The working principle of the implemented S-Function is such, that it stores the mean values and covariance matrix UD factors of the estimated states. These stored variables are then updated at each simulation time step using the current input values according to the MATLAB functions implementing the extended Kalman filter described in the previous section. The estimated means of the states as well as the corresponding estimated output value are then channelled to the output of the Simulink block. These output values are ready to be used for evaluation of the estimator performance and, more importantly, for the purpose of control of mass flow in the hydronic system.

Performance of the implemented mass flow estimator is examined in Chapter 7. Before that however, the next chapter deals with the control of mass flow in the hydronic system.

Chapter 6

Control

The next goal of this thesis, now that the mass flow estimator has been developed, is to utilize the estimation in an optimal mass flow controller which would follow some input (reference) mass flow. To control the mass flow, the pump driving the liquid in the hydronic system needs to be controlled, so in this chapter, I develop a controller for the pump. First, I use a simple PI feedback controller. After that, I try to develop a more complex controller utilizing optimal control techniques.

6.1 PI Controller

The proportional-integral (PI) controller is a special case of the proportional-integral-derivative (PID) controller without the derivative term. The working principle of the controller is very simple. It continuously calculates the difference between a desired reference setpoint and a measured process variable. This difference is called the error. The controller attempts to minimize this error over time by adjusting its control output, which is calculated as a sum of the error and its integral weighted by the proportional and the integral gains respectively. Expressed in an equation, the control output of the PI controller is calculated as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau \tag{6.1}$$

where t is the present time, $e(t)$ is the error, K_p is the proportional gain and K_i is the integral gain.

In the hydronic system in question, there is no direct measurement of the controlled variable—the mass flow—so it is not possible to calculate the error in the usual way. Instead of the measurement, I use the estimate of mass flow provided by the mass flow estimator developed earlier in this thesis. A schema of the hydronic system being controlled by a PI controller is shown on Figure 6.1. The tuning of the PI controller parameters was done by hand. The reason for it being that there is not a systematic way for tuning it for non-linear systems.

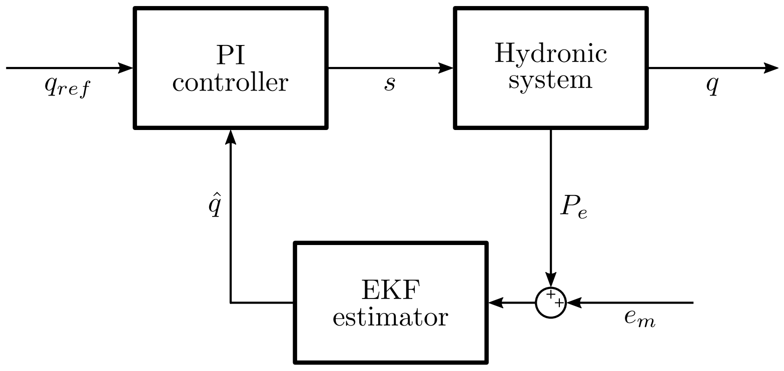

Figure 6.1: Schema of the hydronic system with PI Controller. The desired mass flow setpoint is denoted q_{ref}, while $\hat{q}$ and q denote the mass flow estimate and the actual mass flow in the system respectively. The measured electric power P_e is obscured by a measurement error e_m. The controller output is the pump speed s.

The disadvantage of using the PI controller in this scenario is that it does not use the knowledge of the hydronic system model in any way. Even though it works reasonably well, as will be shown in the next chapter, I now look for a way of utilizing this knowledge in the creation of a better controller.

6.2 Jacobian Linearization Based Controller

When attempting to control a non-linear system, it is possible to use an approach called *Jacobian linearization of a non-linear system about the trajectory* [19]. In this approach I use a feedforward controller calculating the steady state input of the system based on the reference mass flow. The trajectory of the feedforward controller is the used for the linearization. Once the linearization is done, it is possible to use the result to design an optimal linear-quadratic regulator (LQR), which is a form of state-feedback controller.

6.2.1 Jacobian Linearization

Considering a non-linear system governed by the equation

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{6.2}$$

a point $\bar{\mathbf{x}}$ is called an equilibrium point if there is a specific equilibrium input $\bar{\mathbf{u}}$, such that

$$f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = 0 \tag{6.3}$$

Since $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is an equilibrium point and input, it is guaranteed, that if the system starts at $\mathbf{x}_0 = \bar{\mathbf{x}}$ and the constant input $\mathbf{u}_k = \bar{\mathbf{u}}$, then the state of the system will remain fixed at $\mathbf{x}_k = \bar{\mathbf{x}}$ for all time steps k.

Defining deviation variables, allows for describing what happens in case the system deviates from the equilibrium a little.

$$\begin{aligned} \delta_{xk} &= \mathbf{x}_k - \bar{\mathbf{x}} \\ \delta_{uk} &= \mathbf{u}_k - \bar{\mathbf{u}} \end{aligned}$$

Substituting the deviation variables into the differential equation (6.2) yields

$$\delta_{xk+1} = f(\delta_{xk}, \delta_{uk}) \tag{6.4}$$

Doing first order Taylor expansion of of the right hand side of equation (6.4) results in

$$\delta_{xk+1} \approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \delta_{xk} + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \delta_{uk} \tag{6.5}$$

Remembering, that $f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = 0$ leaves

$$\delta_{xk+1} \approx \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \delta_{xk} + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \delta_{uk} \tag{6.6}$$

The difference equation (6.6) approximately governs the deviation variables $\delta_{x\,k}$ and $\delta_{u\,k}$ as long as they remain small.

Looking closely at the partial derivatives in equation (6.6), one can notice, that when considering the hydronic system at hand the first one is actually the state equation Jacobian $\mathbf{F}$ introduced and derived in Appendix B (or more precisely its first element, since I am only controlling the mass flow) evaluated at the equilibrium. Defining the second partial derivative in equation (6.6) to be another Jacobian $\mathbf{G} = \frac{\partial f}{\partial u}\big|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}}$ one can write

$$\delta_{xk+1} = \mathbf{F}\delta_{xk} + \mathbf{G}\delta_{uk} \tag{6.7}$$

Equation (6.7) is called the Jacobian linearization of the original non-linear system about the equilibrium point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ which for small deviations approximately governs the relationship between the deviation variables δ_x and δ_u.

The above works well for simple cases. However, it is not satisfactory for the control of the hydronic system, where the deviations are relatively large. Thankfully a generalization of this approach exists. It lies in redefining the Jacobian matrices as being *time-varying*

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}_k\,=\,\bar{\mathbf{x}}_k \\ \mathbf{u}_k\,=\,\bar{\mathbf{u}}_k}} \qquad \mathbf{G}_k = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}_k\,=\,\bar{\mathbf{x}}_k \\ \mathbf{u}_k\,=\,\bar{\mathbf{u}}_k}}$$

Having done that, the governing difference equation changes to

$$\delta_{xk+1} = \mathbf{F}_k \delta_{xk} + \mathbf{G}_k \delta_{uk} \tag{6.8}$$

This new equation (6.8) is called the Jacobian linearization of the system about the trajectory and acts as a generalization of the linearizations about the equilibrium points [19].

6.2.2 Feedforward Controller

To bring the system close to the desired reference fast and to create a trajectory for the Jacobian linearization, I design a feedforward controller calculating the steady state control input necessary for the reference state. This can be done simply by solving for s in the equation

$$0 = \frac{a_2 - R}{I} q^2 + \frac{a_1}{I} sq + \frac{a_0}{I} s^2 \tag{6.9}$$

which is the equation (3.15) with the rate of change of mass flow $\dot{q}$ equal to zero. I denote the solution to this steady state equation s_{ff}.

6.2.3 Linear-quadratic Regulator

Utilizing the result of Jacobian linearization, I now design a linear-quadratic regulator. In essence, the LQR algorithm is a way of designing an optimal state-feedback controller to drive the deviation variables to zero. Generally, given a linear system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

the algorithm calculates an optimal gain matrix K such that the state-feedback law

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k \tag{6.10}$$

minimizes the quadratic performance measure

$$J = \sum_{k=0}^{\infty} \left(\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k \right) \tag{6.11}$$

where $\mathbf{Q}$ is a real symmetric positive semi-definite matrix and $\mathbf{R}$ is a real symmetric positive definite matrix [20].

In my case, I only have a linear equation for the deviation variables

$$\delta_{xk+1} = \mathbf{F}_k \delta_{xk} + \mathbf{G}_k \delta_{uk}$$

so the state-feedback law takes the form of

$$\delta_{uk} = -\mathbf{K}\delta_{xk} \tag{6.12}$$

and it minimizes the performance measure

$$J = \sum_{k=0}^{\infty} \left(\delta_{xk}^{\ T} \mathbf{Q} \delta_{xk} + \delta_{uk}^{\ T} \mathbf{R} \delta_{uk} \right) \tag{6.13}$$

Thus, this state-feedback clearly only takes care of the deviations from the trajectory, i.e. deviations from the reference.

Adding together the feedforward control s_{ff} and the LQR state-feedback control $s_{LQR} = \delta_{uk}$ evaluated at the reference mass flow and the corresponding input provided by the feedforward controller forms the entire control input regulating the non-linear system. The schema shown on Figure 6.2 shows the layout of the entire Jacobian linearization based control complete with the EKF mass flow estimator, the feedforward controller calculating the steady state input and the LQR state-feedback accounting for any deviations.

6.2.4 Gain Scheduling

The first implementation of the controller described above has been written as using Level-2 MATLAB S-Functions, which evaluated of all the necessary calculations directly at runtime. Due to the relative complexity of these calculation, especially those necessary for the LQR algorithm, the simulations ran quite slowly.

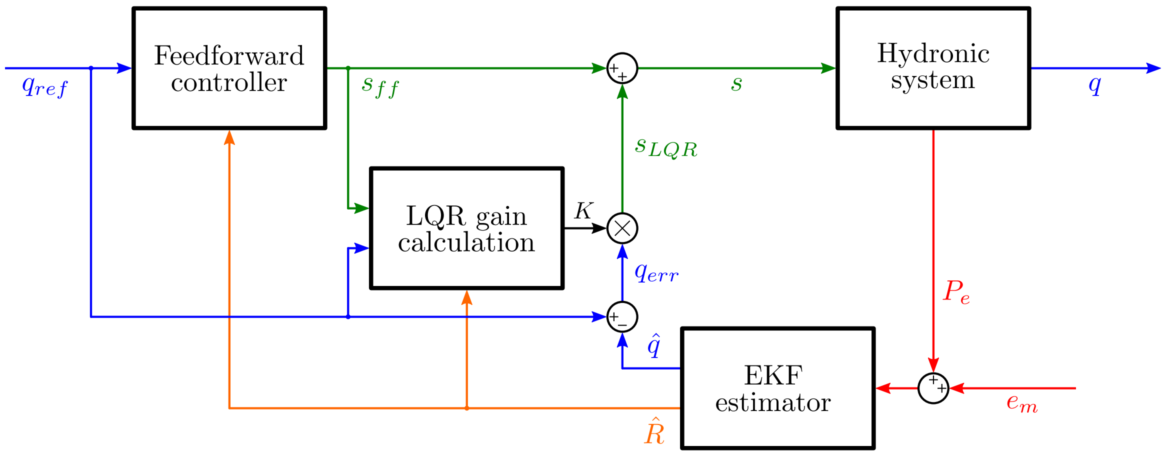

Figure 6.2: Schema of the hydronic system with Feedforward and LQR Controller. EKF estimator provides estimates of mass flow $\hat{q}$ and hydraulic resistance $\hat{R}$ based on the measurements of electric power P_e. These are utilized in the feedforward controller and the LQR state-feedback controller to provide control input back to the hydronic system based on reference mass flow q_{ref}.

To circumvent this issue, I chose to take advantage of the gain scheduling technique.

The basis of this technique lies in replacing the computationally demanding linearization and optimal gain calculations by the use of pre-computed values of the gain. The gain values are first calculated off-line for a number of possible state and input values, which are uniformly distributed within the operating space of the system. These values are then interpolated and used as a "schedule" or lookup table during simulation.

The schedule of K for the LQR state-feedback controller used in my implementation is shown on Figure 6.3. Even though the use of such schedule means losing some precision compared to the direct calculation approach, with reasonable density of the pre-calculated schedule, it is easily outweighed by the benefit of radically increased simulation speed.

With controller design finished, I can finally proceed to the evaluation of the performance of the estimator as well as the performance of the controllers themselves.

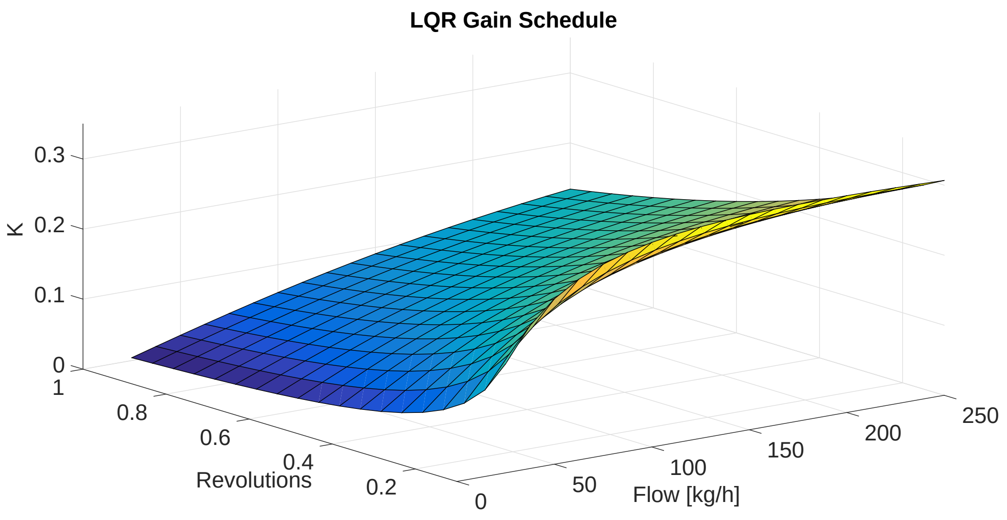

Figure 6.3: LQR gain schedule. A graph showing the pre-calculated values of the LQR gain K for different values of mass flow and pump revolutions uniformly distributed around the operating space of the system.

Chapter 7

Experiments

In this chapter, I present the experimental results of the designed mass flow estimator and the developed mass flow controller. All the presented experiments were performed on the simulation model of the hydronic system implemented in Simulink. A physical model of the hydronic system has unfortunately not been available for comparison. I begin with the review of performance of the mass flow estimator, after which I move on to the mass flow controller.

7.1 Mass Flow Estimation

For the evaluation of mass flow estimation, I did not use any form of a controller in the simulation model. Instead, I directly set up the pump speed input with a series of predefined setpoints. Since the hydraulic resistance in the system is constant, this means that after a each setpoint had been set, the mass flow were constant as well once the transients had died out until a new setpoint was introduced.

7.1.1 Estimator with Simplified Linear Model

The first experiment I ran, was with the Simplified Linear Model Estimator. This estimator has no knowledge of the non-linear model and uses measurements of electric power transformed to mass flow by the P-q characteristic as a base for its estimate. As can be seen on Figure 7.1, such estimator works reasonably well for high flow situations. However, its performance when the mass flow becomes lower is completely unsatisfactory.

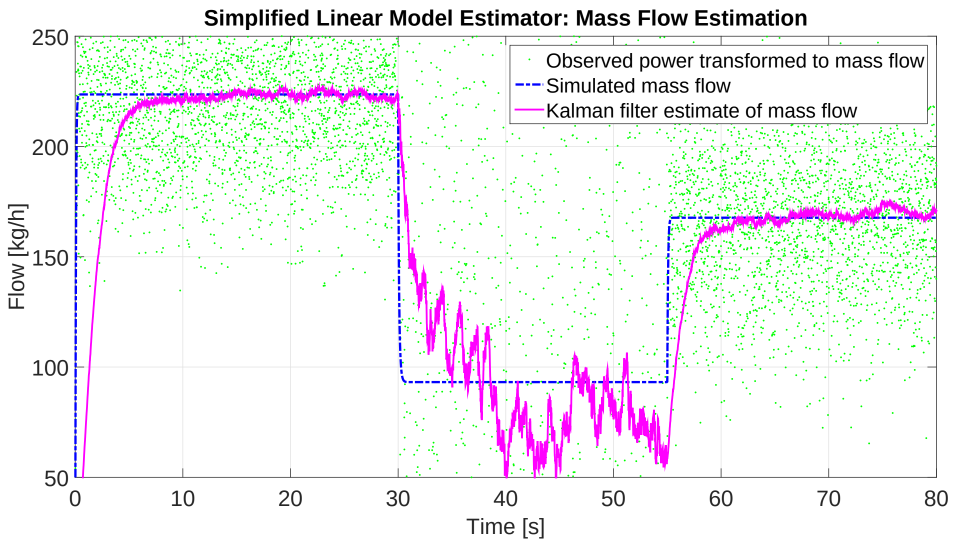

Figure 7.1: Mass flow estimation using the simplified linear model estimator. The estimate eventually gets close to the real value of mass flow when it is high, but fails to do so very precisely, when it is low.

Part of the reason for the poor performance is due to the estimator not taking into account the changes of variance in the data for different values of mass flow. Adding this feature could somewhat lower the fluctuation of the estimate, even though it would not cure it completely. However, even when the flow is higher, the time it takes the estimator to converge to the correct value of mass flow is relatively high.

7.1.2 Extended Kalman Filter Estimator

The more complex extended Kalman filter estimator with knowledge of the non-linear system performs much better. Obtaining information directly from the measurement of power, the estimator is able to estimate the true mass flow very precisely, as can be seen on Figure 7.2. The estimated mass flow is within 1 kg/h of the true simulated value.

The EKF estimator does not suffer from the imprecision in low mass flow situations as the simple estimator did. Thanks to the fact, that the mathematical model in the estimator fits closely to the model used in the simulation, the estimate also follows any changes in mass flow due to changes in input pump speed very precisely.

Figure 7.3 show, that the extended Kalman filter estimator also does a good

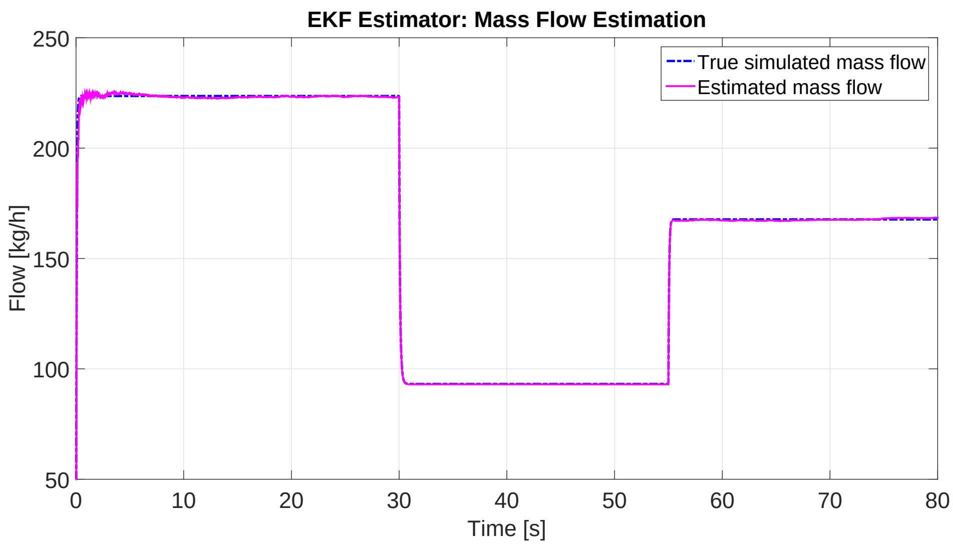

Figure 7.2: Mass flow estimation using the extended Kalman filter estimator.

job filtering the noisy electric power measurement. The estimate of electric power is very smooth even when the variance of noise in the measurements is relatively high.

As opposed to the simple estimator, the EKF estimator is also able to identify and estimate the hydraulic resistance of the hydronic system quite precisely without any initial knowledge about it. Figure 7.4 shows that the estimate of resistance is very close to the true value. Nevertheless, the resistance is not estimated absolutely precisely and some of the disparity between the estimated and the actual true mass flow is in fact due to this imprecision.

The certainty the estimator has about its estimation is reflected in the mass flow standard deviation calculated from the internal Kalman filter covariance matrix and shown in Figure 7.5. The plot clearly shows, that after some initial uncertainty, the standard deviation stays approximately below 2 kg/h. The root-mean-square error of the estimation was in fact in this case 1.12 kg/h. This makes the estimator very precise with the estimation error two orders of magnitude smaller than the actual mass flow value.

Overall the results of the extended Kalman filter estimator are very satisfactory and mean a significant improvement over the simple estimator used initially. As such, they also serve as good base for the next step—experiments with the mass flow control.

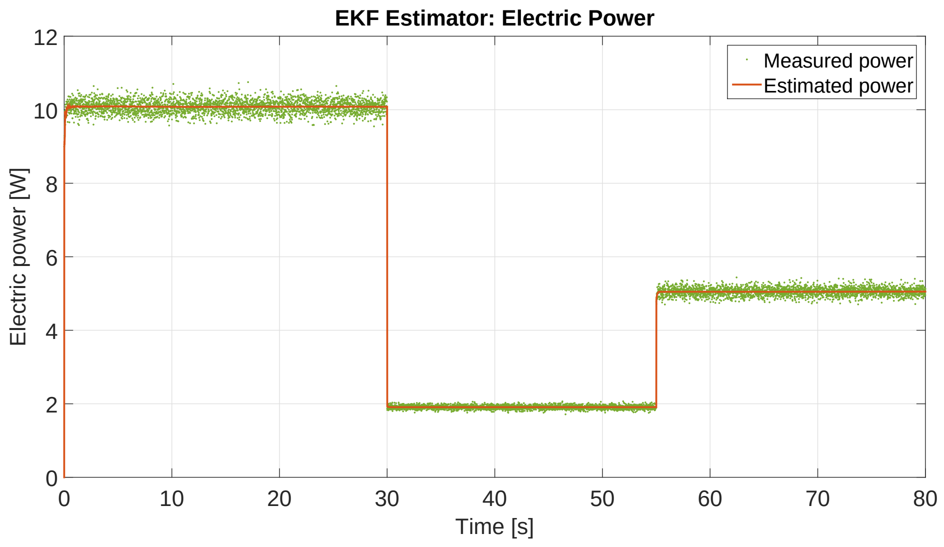

Figure 7.3: Electric power measurements acting as the main source of information for the extended Kalman filter estimator.

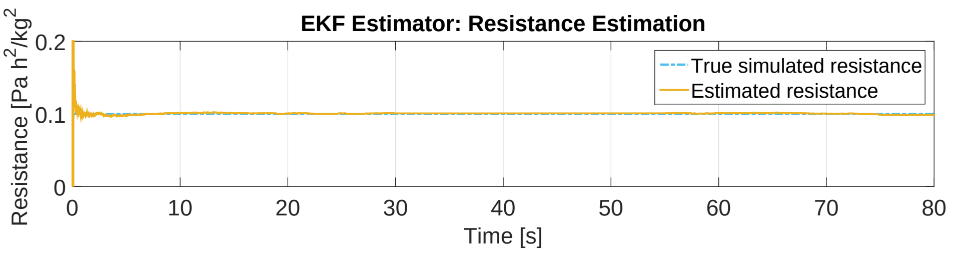

Figure 7.4: Hydraulic resistance estimated by the extended Kalman filter estimator compared to its true value.

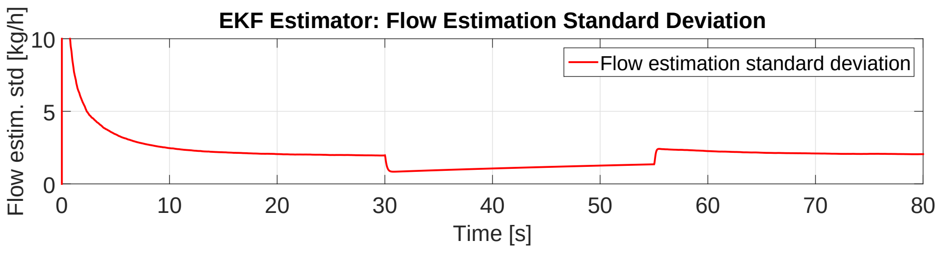

Figure 7.5: Standard deviation of mass flow calculated from the internal covariance matrix of the extended Kalman filter estimator expresses how certain the estimator is about its estimate.

7.2 Mass Flow Control

In this section, I show experimental results of the designed mass flow controllers. Each of the controllers in this section has been given a series of predefined setpoints for the mass flow and tasked to control the pump speed in such a way, that the true simulated mass flow would follow this reference as closely as possible. Since the true value of mass flow is not known, the controllers rely on its estimate provided by the extended Kalman filter estimator, which has been reviewed above. I start with the simpler of the two different approaches to the control used—the simple PI controller—to conclude this section with the review of the more complicated one—the Jacobian linearization based controller.

7.2.1 PI Controller

As shown on Figure 7.6 even a simple empirically tuned PI controller is able to control the mass flow to follow the reference well, provided that the estimate of the actual true mass flow used for the tracking is good enough. The control input necessary for this control may be seen on Figure 7.7.

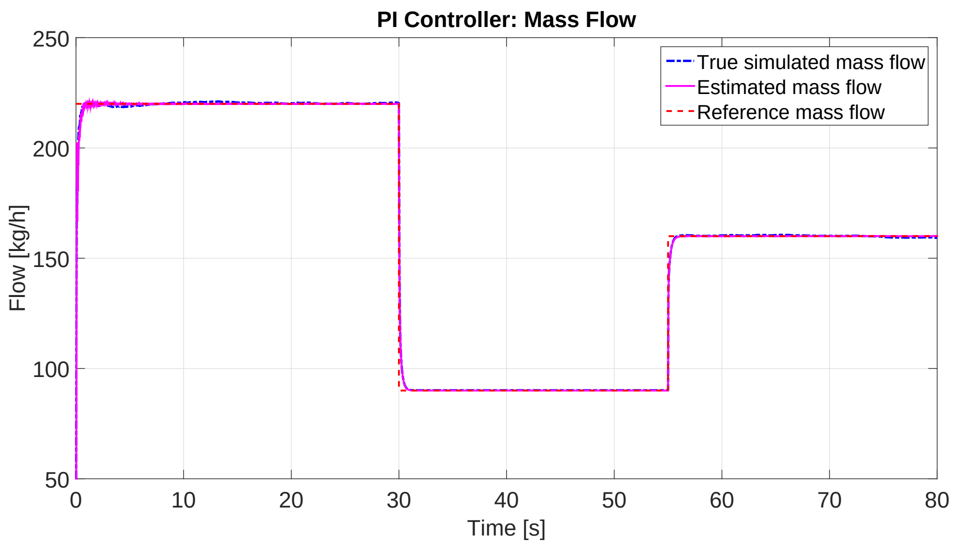

Figure 7.6: Mass flow control with a simple PI controller.

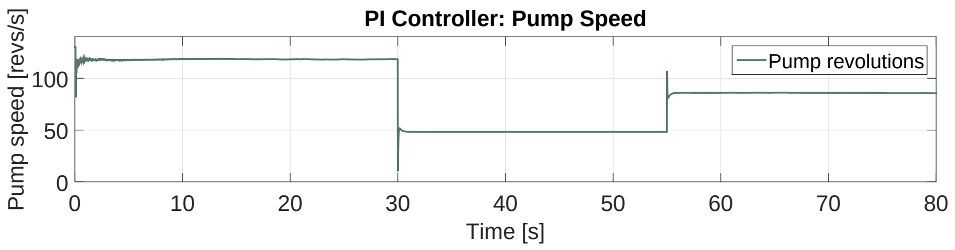

Figure 7.7: Pump speed control input determined by the PI controller.

7.2.2 Jacobian Linearization Based Controller

The results of the much more complex control based on Jacobian linearization and incorporating a feedforward controller and an LQR state-feedback controller are shown on Figures 7.8 and 7.9.

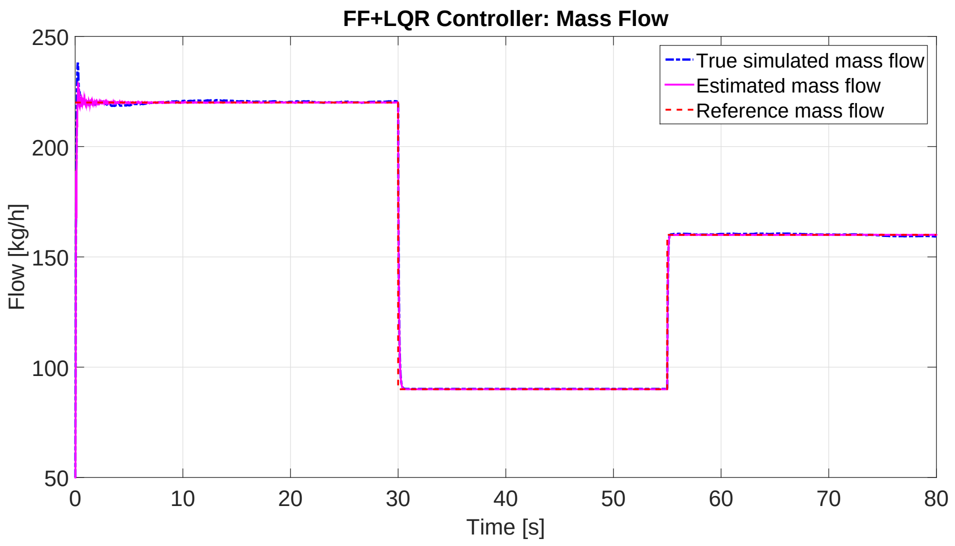

Figure 7.8: Mass Flow control with the feedforward and LQR controller.

While it certainly does show an improvement over the simple PI controller, it is not a very substantial one. The complex controller does achieve the setpoint faster and does so using somewhat smoother control input (i.e. with smaller spike at the transition between setpoints). However, its towering complexity when compared to the PI controller, makes the latter preferable in situations, where fast transitions are not a vital requirement.

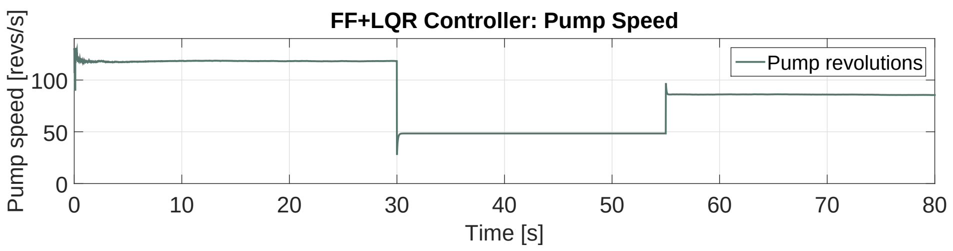

Figure 7.9: Pump speed control input determined by the feedforward and LQR controller.

Chapter 8

Conclusion

In this thesis, I have dealt with mass flow in hydronic systems. My goal has been to develop a simulation model of a hydronic system consisting of a pump and a heat exchanger and to develop a mass flow estimator for this setup. The mass flow estimator was supposed to fuse all available information to make estimates of mass flow through the pump. The next goal of this work was to develop a mass flow controller for the system using optimal control techniques.

I began with a theoretical study of data fusion and estimation theory, reviewing several methods of state estimation applicable to hydronic systems. The method which I chose to use for the design of the mass flow estimator—the Kalman filtering—was analysed in more detail.

The theory of hydronic system modelling was the next subject of my focus. I first derived a mathematical model of each of the components of the hydronic system separately, with extra focus on the model of the pump, before joining them together into a mathematical model of the complete hydronic system. This mathematical model served as a foundation for the setup of simulation model implemented in Simulink. I used data recorded from a physical model of a hydronic network to calibrate the simulation model of the pump. I also used the data for the study of pump electric power characteristic with respect to mass flow and pump impeller revolutions. I fitted the data with a multidimensional polynomial for later use in the estimation.

In the design of the mass flow estimator I first derived the discrete version of the mathematical model of the hydronic system. I then used this discretized model to formulate a state-space description of the non-linear system. This state-space description was then used in an extended Kalman filter which performed the actual estimation. After that, I augmented the state-space description with the hydraulic resistance parameter, for the Kalman filter to be able to estimate it together with the mass-flow. This not only increased the reliability of the estimation, but also removed the need for the resistance to be identified in some other way.

When implementing the mass flow estimator, I first created a version relying solely on the data and the electric power-to-mass flow characteristic of the pump. Then, I improved upon it by incorporating the non-linear model and by the use of the extended Kalman filter. In the final version of the estimator, I utilized an alternative implementation method of the Kalman algorithm robust to round-off errors, which uses modified Cholesky factors of the covariance matrix, to increase the overall precision.

Finally, I created a mass flow controller which uses the mass flow estimate to track a reference mass flow setpoint. I present two alternative versions of the controller, the first being a simple PI controller, the other a more complex Jacobian linearization based controller, which consists of a feedforward part and an optimal LQR state-feedback controller for control of the deviations from the trajectory.

Simulation experiments with the estimators presented towards the end of this thesis show that the extended Kalman filter mass flow estimator performs very well. Its precision stands out especially when compared with the simple linear estimator relying solely on the data. In my experiments, the estimation error was two orders of magnitude smaller than the estimated value. Thanks to the good performance of the estimator, it is possible to control the mass flow even with a very simple PI controller as is illustrated by the experiments. Comparing the complex Jacobian linearization based controller with the optimal LQR state-feedback to the PI controller shows, that little improvement of the control is actually achieved by its usage.

Unfortunately, I was unable to verify the performance of the mass flow estimator and the mass flow controller on a physical model of the hydronic system, as a suitable test-bench has not been made functional in time. However, other than that, I fulfilled every task given to me in the assignment.

Thus, the component on which future improvements should concentrate primarily is the mass flow estimator rather than the mass flow controller. Possible extensions of the estimator include the inclusion of information coming from the temperature drop across the heat exchanger in the estimation. Such information would make the estimator more robust towards imprecisions of the dynamic model of the mass flow and the possible discrepancy between the modelled pump and reality. As for the controller, the use of non-linear model predictive control might be feasible. However, as it has been shown that even a simple PI controller gives satisfactory results, it might not bring a significant improvement.

Bibliography

[1] J. R. Raol, *Multi-Sensor Data Fusion with MATLAB®*. CRC Press, 2009.

[2] F. Castanedo, "A review of data fusion techniques," *The Scientific World Journal*, vol. 2013, 2013.

[3] M. S. Landy, L. T. Maloney, E. B. Johnston, and M. Young, "Measurement and modeling of depth cue combination: in defense of weak fusion," *Vision research*, vol. 35, no. 3, pp. 389–412, 1995.

[4] F. E. White, "Data fusion lexicon," tech. rep., DTIC Document, 1991.

[5] G. Welch and G. Bishop, "An Introduction to the Kalman Filter: SIGGRAPH 2001 Course 8," in *Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques*, pp. 12–17, 2001.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[7] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, Inc., 2014.

[8] C. Sullivan, "Lumped fluid systems." `http://www.dartmouth.edu/~sullivan/22files/Fluid%20sys%20anal%20w%20chart.pdf`, 2004.

[9] Grundfos, *The Centrifugal Pump*. Grundfos Research and Technology, 2008.

[10] J. Dostál, "Decentralized control of hydronic building systems." An unpublished Doctoral Study Report, January 2015.

[11] F. Frishman, *On the Arithmetic Means and Variances of Products and Ratios of Random Variables*, pp. 401–406. Dordrecht: Springer Netherlands, 1975.

[12] MathWorks, "Chi-square goodness-of-fit test — MATLAB documentation." `http://www.mathworks.com/help/stats/chi2gof.html`, 2016. [Online; accessed 4 September 2016].

[13] MathWorks, “Lilliefors test — MATLAB documentation.” `http://www.mathworks.com/help/stats/lillietest.html`, 2016. [Online; accessed 4 September 2016].

[14] MathWorks, “One-sample t-test — MATLAB documentation.” `http://www.mathworks.com/help/stats/ttest.html`, 2016. [Online; accessed 4 September 2016].

[15] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference.* Springer Science & Business Media, 2013.

[16] N. M. Razali, Y. B. Wah, *et al.*, “Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests,” *Journal of Statistical Modelling and Analytics*, vol. 2, no. 1, pp. 21–33, 2011.

[17] J. R. D’Errico, “Polyfitn.” `https://www.mathworks.com/matlabcentral/fileexchange/34765-polyfitn`, 2016. [Online; accessed 4 September 2016].

[18] S. Särkkä, J. Hartikainen, and A. Solin, *Optimal Filtering with Kalman Filters and Smoothers: a Manual for the Matlab toolbox EKF/UKF.* Aalto University School of Science, 1.3 ed., 2011.

[19] A. Packard, K. Poolla, and R. Horowitz, “Dynamic systems and feedback.” `http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/pph02-ch19-23.pdf`, 2002.

[20] D. E. Kirk, *Optimal Control Theory: An Introduction.* Courier Corporation, 2004.

[21] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing, Volume 1 of Fortran Numerical Recipes.* Numerical Recipes in FORTRAN: The Art of Scientific Computing, Cambridge University Press, 1992.

[22] E. W. Weisstein, “Euler Forward Method. From MathWorld–A Wolfram Web Resource..” `http://mathworld.wolfram.com/EulerForwardMethod.html`, 2016. [Online; accessed 30 November 2016].

[23] P. J. Smith, *Joint state and parameter estimation using data assimilation with application to morphodynamic modelling.* PhD thesis, University of Reading, 2010.

[24] G. Plett, “Dual and joint EKF for simultaneous SOC and SOH estimation,” in *Proceedings of the 21st Electric Vehicle Symposium (EVS21), Monaco*, pp. 1–12, 2005.

Appendix A

CD Content

A CD is attached to the printed version of this work containing the text of the thesis in PDF format, the data measured on the physical test-bench and the simulation data for the experiments. The directory structure on the CD along with the description of its contents is shown below.

```
/
|__ data
|   |__ experiments_simulation............simulation data used in Chapter 7
|   |__ flow_simulations .............. miscellaneous data used in development
|   |__ pump_characteristic ................ data from the physical test-bench
|   |__ PumpData.m ....... MATLAB class easing interaction with test-bench data
|__ thesis.pdf ......................... diploma thesis report in PDF format
```

Appendix B

Non-linear System Model

Not included due to confidentiality restrictions.