Aalborg University

Model Predictive Control of Batch Production in Livestock Stables

Department of
Control & Automation

Group:
CA10-936

Master thesis

June 14, 2018

Second year of M.Sc. study
Electronic and IT
Fredrik Bajers Vej 7
DK-9220 Aalborg East, Denmark
http://www.es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Topic:

Model Predictive Control of Batch Production in Livestock Stables

Project:

Master thesis

Project time:

September 2017 - June 2018

Project group:

CA10-936

Participants:

Daniel Bähner Andersen
Nicolaj Vinkel Christensen

Supervisors:

Jan Bendtsen

Synopsis:

The work conducted during this thesis is focused on the development of a model describing broiler growth based on climate parameters and trajectory tracking control. The modeling incorporates five different structures where multiple parameter estimations are conducted in a Monte Carlo fashion. The system identification result in a model deemed acceptable for control use.
The reference tracking model predictive control structure is specified for a batch environment and incorporates a Kalman filter for state estimation. Elements from iterative learning control are included thus memory is introduced in the controller. The control structure is implemented and tested in a simulation environment and additionally applied on a broiler house where a live test is conducted. The live test is however inconclusive due to unusual weather conditions.
In the end, a discussion reflects on the results of the simulation and live test combined with a conclusion that sums up the thesis.

Number of pages: 78
Appendix: 35
Completed June 14, 2018

Preface

This work covers a Master thesis in Control and Automation at Aalborg University, Department of Electronic Systems. The project is produced by group CA10-936.
The goal of this project is to do model predictive control of batch production in livestock stables

The report starts with a small introduction to the problem, how it is modeled and how parameters are estimated. This is followed by the derivation of the controller and implementation. On top of this is the implementation tested both during simulation and a live test. A discussion about ideas for improvement leads up to the final conclusion.

The reader can find a nomenclature at the beginning of the report which includes acronyms, a symbolic list, and a notation list. In the appendix, detailed derivation, in-depth descriptions and relevant measurements used for this project can be found.

Aalborg University, June 13, 2018

Daniel Bähner Andersen
dban13@student.aau.dk

Nicolaj Vinkel Christensen
nvch13@student.aau.dk

Nomenclature

Acronyms

API	Application Programming Interface
ARX	Autoregressive Exogenous
BMPC	Batch Model Predictive Control
FCR	Feed Conversion Ration
IID	Independent and Identically Distributed
ILC	Iterative Learning Control
MET	Meterologisk Institutt
MLP	Multilayer Perceptron
MPC	Model Predictive Control
NRMSE	Normalized Root Mean Square Error
OECD	Organisation for Economic Co-operation and Development
RNN	Recurrent Neural Network
RTC	Ready To Cook

Symbols

Symbol	Description	Unit
$\mathbf{A}$	System matrix of the broiler model	$[\cdot]$
$\mathbf{A}_N$	Extended system matrix	$[\cdot]$
$\mathbf{A}^e$	System matrix of the error model	$[\cdot]$
$\mathbf{b}$	Linear part of the minimization problem	$[\cdot]$
$\mathbf{B}$	Input matrix of the broiler model	$[\cdot]$
$\mathbf{B_d}$	Input disturbance matrix of the broiler model	$[\cdot]$
$\mathbf{B}^e$	Input matrix of the error model	$[\cdot]$
c	Constant part of the minimization problem	$[\cdot]$
$\mathbf{c}$	A row vector of centroid positions	$[\cdot]$
$\mathbf{C}$	Output matrix of the broiler model	$[\cdot]$
$\mathbf{C}^e$	Output matrix of the error model	$[\cdot]$
$\mathbf{D}$	Feedforward matrix of the broiler model	$[\cdot]$
$\mathbf{D_d}$	Feedforward disturbance matrix of the broiler model	$[\cdot]$
$\mathbf{d}_k$	Disturbance vector of the controller model	$[\cdot]$
$\mathbf{d}'$	Unknown dynamics in batch error model	$[\cdot]$
$\mathbf{D}_k$	Disturbance sequence	$[\cdot]$
$\Delta\mathbf{D}_k$	Change in the disturbance sequence	$[\cdot]$
$\Delta\hat{\mathbf{D}}_k$	Predicted change in disturbance sequence	$[\cdot]$
$\mathbf{e}_k$	Error sequence	$[\cdot]$
$\tilde{\mathbf{e}}_k$	Error contribution from the applied input because of the dynamics	$[g]$
$\mathbf{e}_k^d$	Error contribution from $\bar{\mathbf{u}} - \hat{\bar{\mathbf{u}}}$, disturbances etc.	$[g]$
$\bar{\mathbf{e}}_k$	The part of $\mathbf{e}_k^d$ that repeats itself from batch to batch	$[g]$
$\mathbf{e'}_k$	Output of time-wise error model	$[g]$
$\hat{\mathbf{e}}_k$	Estimated error sequence	$[g]$
H_y	Humidity	[%]
$\mathbf{H}$	Output matrix of time-wise error model	$[\cdot]$
k	Batch index	$[\cdot]$
$\mathbf{K}$	Kalman gain	$[\cdot]$
m	Prediction horizon	$[\cdot]$
N	Batch length	$[\cdot]$
$\mathbf{Q}_1$	Weighting matrix for the state	$[\cdot]$
$\mathbf{Q}_2$	Weighting matrix for change in input	$[\cdot]$
$\mathbf{P}_k$	Covariance matrix	$[\cdot]$
$\mathbf{R}$	Quadratic part of the minimization problem	$[\cdot]$
R_m	Measurement variance	$[g^2]$
$\mathbf{R}_s$	State variance	$[g^2]$
$\mathbf{R}_w$	Biased batch variance of the covariance matrix	$[g^2]$
$\mathbf{R}_v$	Batch specific variance of the covariance matrix	$[g^2]$
t	Time index	$[\cdot]$
T_{out}	Outside temperature	[°C]
u_k	Applied temperature	[°C]
$\bar{\mathbf{u}}_k$	Reference temperature sequence	[°C]
$\hat{\bar{\mathbf{u}}}_k$	Best estimate of $\bar{u}$	[°C]
$\mathbf{U}_k$	Temperature trajectory	[°C]
$\hat{\bar{\mathbf{U}}}_k$	Input sequence	[°C]
$\Delta\mathbf{U}_k$	Change in temperature trajectory	[°C]
$\Delta\hat{\mathbf{U}}_k$	Predicted change in temperature trajectory	[°C]
$\Delta\mathbf{U}_{lb}$	Lower bound of the temperature trajectory	[°C]
$\Delta\mathbf{U}_{ub}$	Upper bound of the temperature trajectory	[°C]

Symbol	Description	Unit
v_k	Batch specific input disturbance	$[\cdot]$
w	Input disturbance in broiler model	$[\cdot]$
w_k	Batch repetitive disturbance	$[\cdot]$
$\mathbf{W}^{\circ}$	RNN model output weights	$[\cdot]$
$\mathbf{W}^{h}_{y,a}$	RNN model delayed output weights	$[\cdot]$
$\mathbf{W}^{h}_{b}$	RNN model input weights	$[\cdot]$
x	State in state-space model	$[\cdot]$
$\hat{x}$	Estimated state	$[\cdot]$
$\mathbf{y}_k$	Output trajectory	$[g]$
$\hat{\mathbf{y}}_k$	Estimated output trajectory	$[g]$
$\bar{\mathbf{y}}$	Reference output trajectory	$[g]$
$\bar{y}$	Mean of the weight measurement	$[g]$
$\mathbf{z}$	A row vector of $\mathbf{Z}$	$[\cdot]$
$\mathbf{Z}$	Data matrix for the $k-$means algorithm	$[\cdot]$
γ	Initial scaling parameter of covariance matrix	$[\cdot]$
$\mathbf{\Gamma}$	Extended input matrix	$[\cdot]$
$\mathbf{\Psi}$	Extended disturbance matrix	$[\cdot]$
θ	PParameters in system identification	$[\cdot]$
θ^{h}	RNN model hidden layer bias	$[\cdot]$
θ°	RNN model output bias	$[\cdot]$
$\Upsilon(\cdot)$	Cost function of the MPC	$[\cdot]$

Glossary of Mathematical Notation

This section sums up the mathematical notation and terminology used in this report.

Upper and lower bounds of a variable

$$\underline{x} < x < \overline{x} \tag{1}$$

Where $x \in \mathbb{R}$ and $\overline{x}$ and $\underline{x}$ are the upper and lower bounds, respectively.

Intervals

$$[a, b] = \{x \in \mathbb{R} | a \leq x \leq b|\} \tag{2}$$

Where a and b are the start and end points of the interval, respectively.

Vectors and matrices

Vectors and matrices are noted with bold fonts, such that $\boldsymbol{v}$ is a vector:

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^{(n \times 1)} \tag{3}$$

and $\boldsymbol{M}$ is a matrix:

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1p} \\ m_{21} & m_{22} & \dots & m_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \dots & m_{np} \end{bmatrix} \in \mathbb{R}^{(n \times p)} \tag{4}$$

Continuous vector variables are noted with $\boldsymbol{v(t)}$ such that:

$$\boldsymbol{v(t)} = \begin{bmatrix} v_1(t) \\ v_2(t) \\ \vdots \\ v_n(t) \end{bmatrix} \in \mathbb{R}^{(n \times 1)} \tag{5}$$

While discrete vector variables are referred to as sequences and noted with $\boldsymbol{v[t]}$, such that:

$$\boldsymbol{v[t]} = \begin{bmatrix} v_1[t] \\ v_2[t] \\ \vdots \\ v_n[t] \end{bmatrix} \in \mathbb{R}^{(n \times 1)} \tag{6}$$

is a sequence, where t is the time step between two entries.

Contents

Introduction 1

A main source to fulfilling the world's demand for meat is the poultry industry, in which the flesh of mainly chickens and other domestic fowl are processed as food. The average yearly production of poultry meat during 2014-16 was 113.8 billion kg of ready to cook (RTC). According to the 2017 OECD 10 year outlook, poultry meat production is predicted to reach 131.6 billion kg RTC in 2026. Corresponding to an expansion of roughly 13% compared to the levels of 2017. Based on the outlook, poultry meat production is predicted to exceed pig meat during 2017 and thus become the worlds main meat production type, compared to global output levels [1].

The global consumption of meat is by the same outlook predicted to increase accordingly. The annual growth rate of meat consumption is predicted to be 1,09% as a global average. The main contribution to the annual consumption growth arises from an expanding population, which for the outlook period is predicted to increase from 7.3 billion to 8.2 billion people. This amounts to an annual increase of 0.83%, where the remaining 0.26% is attributable to an increase in per capita consumption [1].

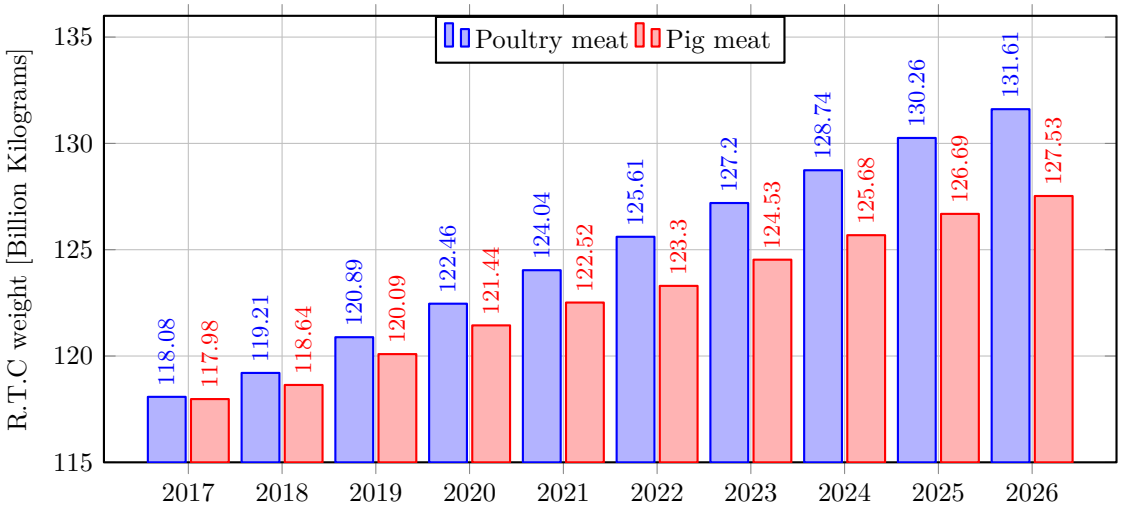

Figure 1.1: Projection of global poultry and pig meat productions according to the OECD 10 year outlook [1].

The focus of this thesis is concentrated upon meat production from chickens which are known as broilers, as these are estimated to make up 83% of poultry meat consumption per capita in 2017 [2]. Modern broilers are raised in large open houses with flock sizes ranging from 30 to 40 thousand broilers. The broilers arrive at the farm shortly after being hatched where each broiler at arrival weighs in around 40 grams. When placed in their house, the broilers are introduced to a preheated house with switching lines of feed and water across the house width, spanning the length of the house floor. The hatched broilers are incapable of regulating their own body temperature during the first 10 to 14

days, thus the house is heated to approximately 34 degrees ensuring a body temperature of 40-41 degrees. As the broilers grow and depending on their weight gain is the temperature lowered to 20 degrees. The feed and water supply are unlimited and provided as needed, thus the consumption is only dependent on the broiler behavior. The broilers are on average staying in their house 34 days and reach a weight of approximately 2050 grams, achieved with approximately 3100 grams of feed [3].

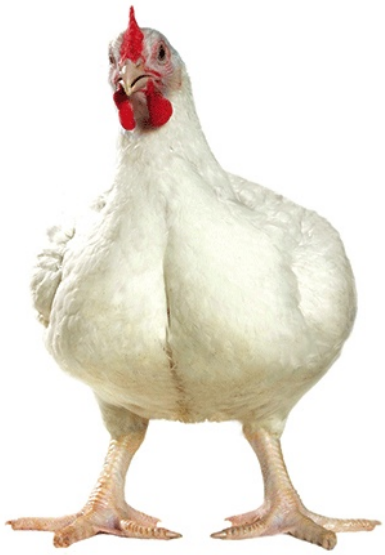

Figure 1.2: The Ross 308 broiler, present at all the farms that provide data to this thesis.

The broiler behavior is in modern broiler stables controlled by the climate and environment. The broilers are highly sensitive to temperature which is a key component to both regulate the behavior and monitor the well being of the broilers. Further climate factors that influence the broiler behavior are air humidity, CO2 level, and wind chill effects if the ventilation is working aggressively. For the remaining environment, the light control is an important factor. Broiler houses do not incorporate windows, thus the day and night cycle is determined fully by the indoor lighting control scheme. These climate factors are described in [4].

The evolution of broiler houses has followed the general modernization of all industries and thus are modern broiler houses closely monitored and feature redundant measurements of a large number of environmental parameters. This data is stored and thus production data for a large number of houses, spanning across multiple years is available for further analysis. The existence of such data creates the basis for this project and allows for a data-driven modeling of real-world broiler behavior. It is thus possible to analyze the effect of environmental parameters on the broiler behavior and performance metrics such as growth and food consumption. As of today's standard, modern broiler houses feature measurements of all vital parameters, but the use of this is not utilized beyond day to day monitoring purposes of the individual house. Environmental parameters such as lighting schemes and climate control set points e.g heat and humidity, are all adjusted by the farmer manually.

Figure 1.3: Inside of a broiler house with young broilers. The alternating food and water lines throughout the house are visible on the floor.

Considering the extent and growth prediction for modern broiler farming combined with the evolution and integration of modern data acquisition technology, it is desirable to investigate the use of new approaches within the fields of broiler managing and optimization of vital performance metrics. Solutions can range from aiding the farmer in selecting appropriate set points for optimal climate control or complete control structures optimizing specific performance metrics. It is believed that this can be used to aid the farmer, leading to optimized production as advanced control techniques such as model predictive control (MPC) can be applied.

Most literature focuses elsewhere as e.g optimal feeding by changing the broiler feed composition or control of the broilers heat generation thus conflicts between environmental, financial, and welfare aspects can become compatible [5, 6]. Only recent studies by Simon V. Johansen focus on the dynamic interconnection between broiler weight and broiler house environmental conditions. These studies focus on obtaining a nonlinear model with the purpose of forecasting the broiler weight by the use of neural network models [7]. As this field of study is new and no additional prior studies exist it is desirable to investigate the results of an approach using only linear models can achieve.

The thesis is written in collaboration with the Danish company SKOV A/S which specializes in climate and farm management solutions for poultry and pig farming. The company was founded in Denmark and provides monitoring components, management platforms, ventilation components and complete solutions for poultry houses. Due to their specialization, they are able to provide insight into practices and challenges of poultry farming and knowledge of modern solutions for climate control and management for poultry farms.

Part I

Model Development

System Identification 2

This chapter addresses the modeling conducted during the project. Thereby including a presentation of the provided broiler house data and the chosen model structure. This is followed by a presentation of the data-driven model estimation method used and a model verification. Leading to a final model, suitable for control purposes.

2.1 Data Description

The data provided for this project originates from ten broiler houses placed at three locations. At the beginning of the project, data from four houses were made available and later additional data from six houses. For each house and its corresponding dataset, production data covering between 24 and 33 batches is found. A batch represents between 30 to 40 thousand broilers, depending on the size of the house. Each of these batches found in a dataset contains measurement data and system references from the corresponding house during the batch, which stretches across the broilers growth period of approximately 34 days. The available data is in most cases recorded both prior to the stocking of new broilers and subsequent to them being displaced from the house, thus most data sets contain data corresponding closer to 36 days.

Prior to making the data accessible for this project the raw data has been processed. As the raw data is extracted from each house it is corrected such that per bird units are adjusted according to mortality of the broilers. Furthermore, sensor readings are processed and an average value across the specific sample interval is provided. This is necessary as the raw data is created by a sampling method where a new sample is only taken if the value has changed significantly compared to the last sampled value. There is due to this method no specific sample time only an upper limit. In order to convert this raw data into suitable data for modeling and control purposes, is the raw data processed prior to being accessible for this project. During this process, is a linear interpolation between all sample points made. The desired samples are generated by performing integrals where the span of the limits corresponds to the desired sample interval, this is conducted throughout the entire dataset.

The data is available with a sample interval of either 1, 3, 6, 12 or 24 hours. During the system identification, it is chosen to use the sample interval of 1 hour, in order to utilize as much information as possible and as this will make faster control possible.

Each dataset of a batch contains a large amount of different and redundant measurements, containing information about all vital climate parameters, status and set points of the ventilation system, light intensity measurements, broiler weight, food and water consumption, etc. Only a limited number of parameters will be used in the model and the selection of these will be explained throughout the following section. The desired measurements are extracted from the dataset and divided into their respective categories, either input, output or disturbance. An example of the available data for a few variables and the chosen sample interval during a batch is shown in *Figure 2.1.*

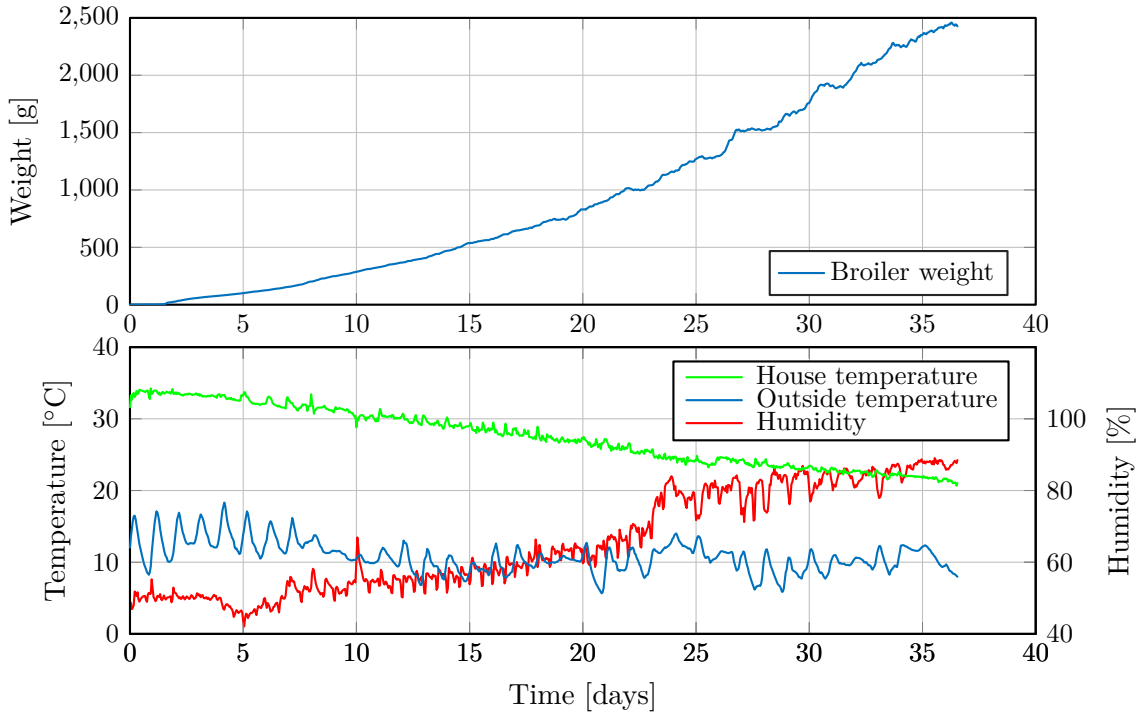

Figure 2.1: Specific example of the available broiler house batch data originating from house six, batch nine.

The data in *Figure 2.1* presents the system behavior during a typical batch. The weight of the broiler rises as desired and the daily weight gain increases slightly as a bigger broiler is capable of a greater weight gain. The broiler house temperature is closely monitored and regulated by the farmer, thus a clear descending trend is visible. The temperature is approximately set at 34 degrees for the newly hatched broilers, depending on the specific conditions. This allows their body to focus the energy use on growth instead of maintaining body temperature as they still have an insufficient feather coat [3]. The temperature is constantly adjusted to the bird behavior but will in general during the batch period be lowered towards 20 degrees at day 34, according to standard farming practice for the specific broiler type [8]. The humidity in the broiler house increases throughout the batch period. This is predominantly caused by an increasing biomass in the broiler house. The outside temperature varies according to day, night and season thus reflecting the climate in the region where the broiler house is located.

Regarding the presented data, a sample mean is created for each measurement, a special case is the measured broiler weight where not only the sample mean is calculated but also a time depended factor is added. This is due to the measured broiler weight becoming increasingly negative biased as the broilers grow [9]. The broiler weight is measured by a scale the broilers jump onto, thus as they grow, heavier broilers will tend to stay off the scale where lighter broilers have a higher tendency to visit the scale. Further explanation of this problem can be seen in [4].

For some batches, the desired measurement data is unavailable or incomplete during the batch period. A possible reason being that data logging in some of these houses is in an experimental phase. Due to this fact, batches which do not provide the desired data measurement or quality are discarded. Furthermore, the batch data is shortened if it extends beyond the period of broilers being in the house. This data sorting results in 161 batches which contain the desired measurements and thus are suitable for further use.

2.2 Model Structure

Modeling and control of broiler batch production by the use of climate control variables is a new area for control design. Previous work is very limited and what exists, is based on a nonlinear modeling approach. It is thus relevant to explore the capabilities of a linear approach. The system is modeled with a linear model, as these are well known and there exists a wide variety of suitable control techniques.

The goal of the model design in this project is to create a model, suitable for control purposes, which has the growth behavior of a broiler, based on climate conditions in a broiler house. By achieving a model of this type, it is possible to develop a controller which can aid the farmer, by finding the optimal reference trajectory, leading to optimized production as advanced control techniques such as MPC can be applied.

The selection of inputs, outputs, and disturbances used in the model is based on the work done by Simon V. Johansen where the effect of different environmental conditions in broiler houses with respect to broiler growth are investigated [4]. From the set of significant conditions found, the inputs, outputs, and disturbances used in the model are chosen.

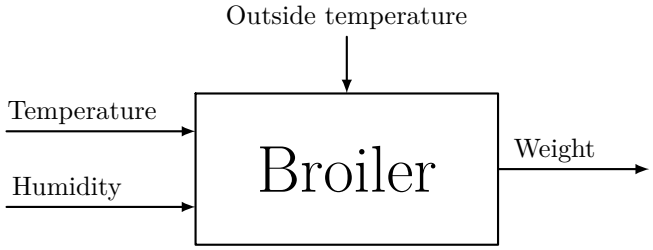

Figure 2.2: Block diagram showing the modeling approach.

The model structure shown in *Figure 2.2* is centered around the growth of a single broiler, based on knowledge and measurements from broiler batches. From a control perspective, the broiler growth behavior is represented as the system. Inputs to the system are variables that affect the growth and behavior of the broiler directly, it is desirable that these are independent and can be controlled. The temperature and humidity in the broiler house are variables that come close to this definition and are thus selected as inputs to the system.

The outputs of the system are defined as entities that the broiler can effect. In the studied broiler production, feed and water are unlimited and supplied as needed, thus the consumption is determined by the broiler. Feed and water consumption are outputs of the system but it is chosen not to use these, instead, the broiler weight will be the used output of the system. This is chosen as the broiler weight is strongly correlated with the feed consumption, naturally caused by the growth behavior of the broiler, requiring more energy as it grows. Likewise is the water consumption also strongly correlated with both growth and feed consumption. Therefore to keep the model simple as a starting point, only the broiler weight is chosen as the output of the system.

A disturbance that affects the broiler house and the broilers is the outside temperature. Depending on how the houses are isolated and the amount of air renewal, the outside temperature will affect the temperature in the broiler house, especially during cold winters or warm summers. The humidity in the broiler house will furthermore always be affected by the changing seasons, weather type and, the outside temperature. The outside temperature is therefore added as an input disturbance to the model.

When considering the growth curve of a broiler, the weight at the next time step is strongly dependent on the weight at the current time step. An approach to modeling this behavior is with the system matrix being a pure integrator. In order to realize such a system a discrete time, first order, state-space representation is used. The general model structure is shown in *Equation: (2.1)*.

$$\begin{aligned} x[t+1] &= \boldsymbol{A}x[t] + \boldsymbol{B}\boldsymbol{u}[t] + \boldsymbol{B_d}w[t] \\ y[t] &= \boldsymbol{C}x[t] + \boldsymbol{D}\boldsymbol{u}[t] + \boldsymbol{D_d}w[t] \end{aligned} \tag{2.1}$$

Where

$x[t]$	$\in \mathbb{R}^{(1\times1)}$	is the state at time t,	[g]
$\boldsymbol{u}[t]$	$\in \mathbb{R}^{(2\times1)}$	is input vector at time t,	[°C, %]
$w[t]$	$\in \mathbb{R}^{(1\times1)}$	is input disturbance at time t,	[°C]
$y[t]$	$\in \mathbb{R}^{(1\times1)}$	is the output at time t,	[g]
$\boldsymbol{A}$	$\in \mathbb{R}^{(1\times1)}$	is the system matrix,	[·]
$\boldsymbol{B}$	$\in \mathbb{R}^{(1\times2)}$	is the input matrix,	[·]
$\boldsymbol{B_d}$	$\in \mathbb{R}^{(1\times1)}$	is the input desturbance matrix,	[·]
$\boldsymbol{C}$	$\in \mathbb{R}^{(1\times1)}$	is the output matrix,	[·]
$\boldsymbol{D}$	$\in \mathbb{R}^{(1\times2)}$	is the input feedforward matrix,	[·]
and $\boldsymbol{D_d}$	$\in \mathbb{R}^{(1\times1)}$	is the input disturbance feedforward matrix.	[·]

To determine if this is the best approach three estimations will be made. This allows a comparison and the best model structure can be chosen. The three models structures:

1. Estimation of only $\boldsymbol{B}$ and $\boldsymbol{B}_d$, without feedforward and leaving $\boldsymbol{A}$ and $\boldsymbol{C}$ as unity.
2. Estimation of $\boldsymbol{B}$, $\boldsymbol{B}_d$, $\boldsymbol{D}$ and $\boldsymbol{D}_d$, thus leaving $\boldsymbol{A}$ and $\boldsymbol{C}$ as unity.
3. Estimation of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{B}_d$, $\boldsymbol{D}$ and $\boldsymbol{D}_d$, with $\boldsymbol{C}$ as unity, to evaluate on the initial choice of the system being a pure integrator.

These model structures will all be used throughout the model design and based on these the final model structure and parameters are chosen.

2.3 Individual Estimations

In this section, the estimation procedure for the unknown elements in the state-space matrices is explained. The method and results are based on the model, described in Section *2.2: Model Structure*, and the data, described in Section *2.1: Data Description*. All results shown in this section are based on data from the first four houses, consisting of 55 batches, as only they were available at the time. The presented results in this section originate from model structure two, defining the system matrix as an integrator and including feedforward. Estimation of the two other model structures, mentioned in Section *2.2: Model Structure*, is carried out in the same way. A block diagram of the general parameter identification method is shown in *Figure 2.3*.

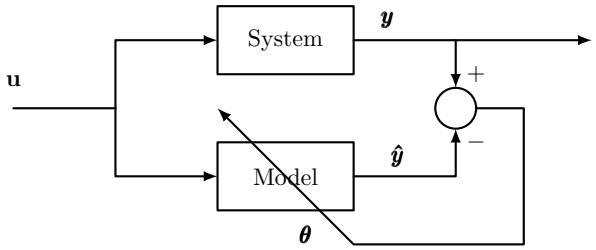

Figure 2.3: The measurements from the system, $\boldsymbol{y}$, are compared to the output of the model simulation, $\hat{\boldsymbol{y}}$, while introducing the same input for both systems. $\boldsymbol{\theta}$ denotes the model parameters, which is adjudge to minimized the error.

The estimation is done using three different strategies of splitting up the data. For each strategy, the individual datasets, corresponding to a batch, are lumped in sets differently.

1. Gather datasets from the same house in the same set.
2. Gather datasets of approximately equal length in the same sets.
3. Handle all datasets as one set.

The first two steps in the estimation procedure are the same for all three strategies and will, therefore, be explained only once.

The first step is to define the datasets as being either a training set or a test set. The training set is used to estimate the unknown elements in the matrices and the test set is used to check how well the estimated model performs. The datasets are for each strategy split up by randomly selecting 70% of the batches, using these as the training set, leaving the remaining 30% as the test set. An important note is that, for the first two strategies, the performance of each model is checked by a test set comprised of 30% randomly selected batches from each house or sets according to batch length. This is done to ensure that the estimated model is not biased towards one of the houses or batch length. An overview of the full data splitting process can be found in Appendix *F: Estimation Flow Graphs Figure F.2*.

The second step is to set initial guesses on the elements in the input and feedforward matrices and initial conditions of the system state. To clarifying, the state-space model is stated below:

$$\begin{aligned} x[t+1] &= x[t] + [B_{11}B_{12}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] + [D_{11}D_{12}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [D_d]w[t] \end{aligned} \tag{2.2}$$

The unknown parameters, $B_{11}, B_{12}, B_d, D_{11}, D_{12}$ and D_d are initialized by defining them as uniform random variables in the interval $[-0.1, 0.1]$, and the state, $x[t]$ as a normally distributed random variable with a mean of 40 grams and variance of 10 grams. The initial condition of the state is chosen positive as this results in a positive weight increase of the broiler, which cannot from a physical perspective be negative. Selecting the initial conditions according to a random variable allows an insight into the sensitivity of the estimation procedure and performance of the used solver. If different estimations arrive at roughly the same results, this suggests that the chosen solver is adequate and the estimation problem is no more sensitive to initial condition than the solver can overcome.

As written earlier the performance of all models is checked using the test set. This is done by simulating the growth of a broiler using the estimated model parameters and the inputs

of the test sets, which is then compared with the output of the test set as shown in *Figure 2.3*. The measured output and simulated output are compared using Normalized Root Mean Square Error (NRMSE) according to *Equation: (2.3)*, which gives a fit percentage for each batch in the test set. The mean fit percentages of the test set are considered the fit percentages for the estimated model.

$$fit = 100\left(1 - \frac{||\boldsymbol{y} - \hat{\boldsymbol{y}}||}{||\boldsymbol{y} - \bar{y}||}\right) \tag{2.3}$$

Where

	$\boldsymbol{y}$	is the vector of weight measurements,	[g]
	$\bar{y}$	is the mean of the weight measurements,	[g]
and	$\hat{\boldsymbol{y}}$	is the vector of outputs from the model.	[g]

The use of NRMSE as performance function results in fit percentages within the interval $[-\infty, 100]$. A fit of 0% implies that the compared model output has the correct average value but the trend is incorrect. If the fit is negative both average value and trend are incorrect.

2.3.1 Estimation Based on Houses

The first strategy is based on estimating a model for each house and then measure the performance, of all created models, on test data from all houses. This results in four training sets and one test set. As mentioned earlier, the test set is created by gathering the remaining 30% of batches from each house where 70% are selected randomly as training data. The procedure of splitting the data is illustrated in *Figure 2.4* where the number of training and test sets are displayed in green and red respectively.

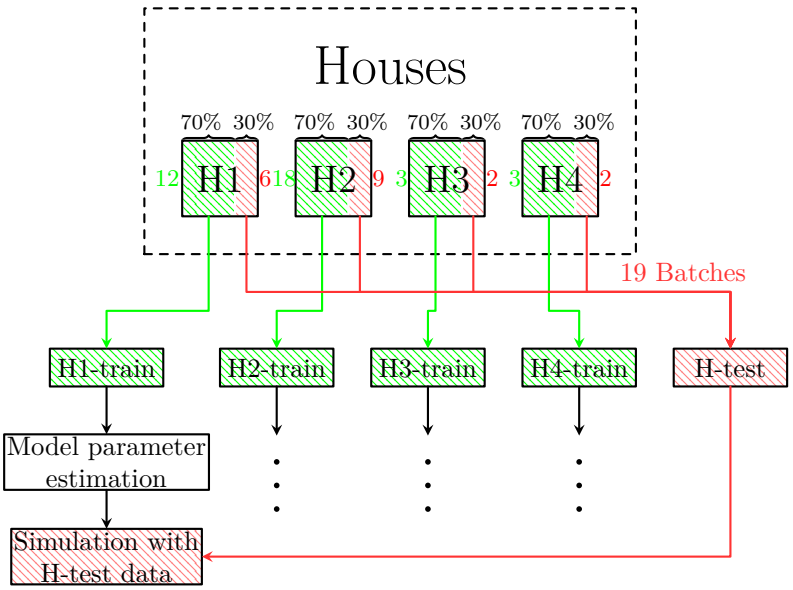

Figure 2.4: An overview of the model estimation process where the data is separated according to the associated house. In green, the number of training batches for each house and in red the number of test batches.

The four training sets, each comprised of data from the same house, are shown as *H1-train, H2-train, H3-train* and *H4-train* respectively. The test set used to evaluate the performance of each of the four models is shown as *H-test*. The complete parameter

estimation and simulation procedure is conducted for each training set, thus four models are obtained.

The estimation is done using the Matlab grey box estimator `greyest`. The grey box estimator is a gradient-based estimator that uses a linear `idgrey model`, an initial model that defines the model structure, and a dataset as input. The output is the model with the estimated parameters. For all three estimations, the initial model is the linear state-space model described in Section *2.2: Model Structure* and for this strategy, the data is the training set based on houses as mentioned earlier. The output is the model with the estimated elements of the input and feedforward matrices that provide the best performance using the training set. Further explanation of the `greyest` estimator is found in Appendix *B: Matlab Estimation Toolbox*. The result of the `greyest` estimator is compared to the test set, this is done by simulation. In *Figure 2.5* the two best and two worst results from one estimation is presented.

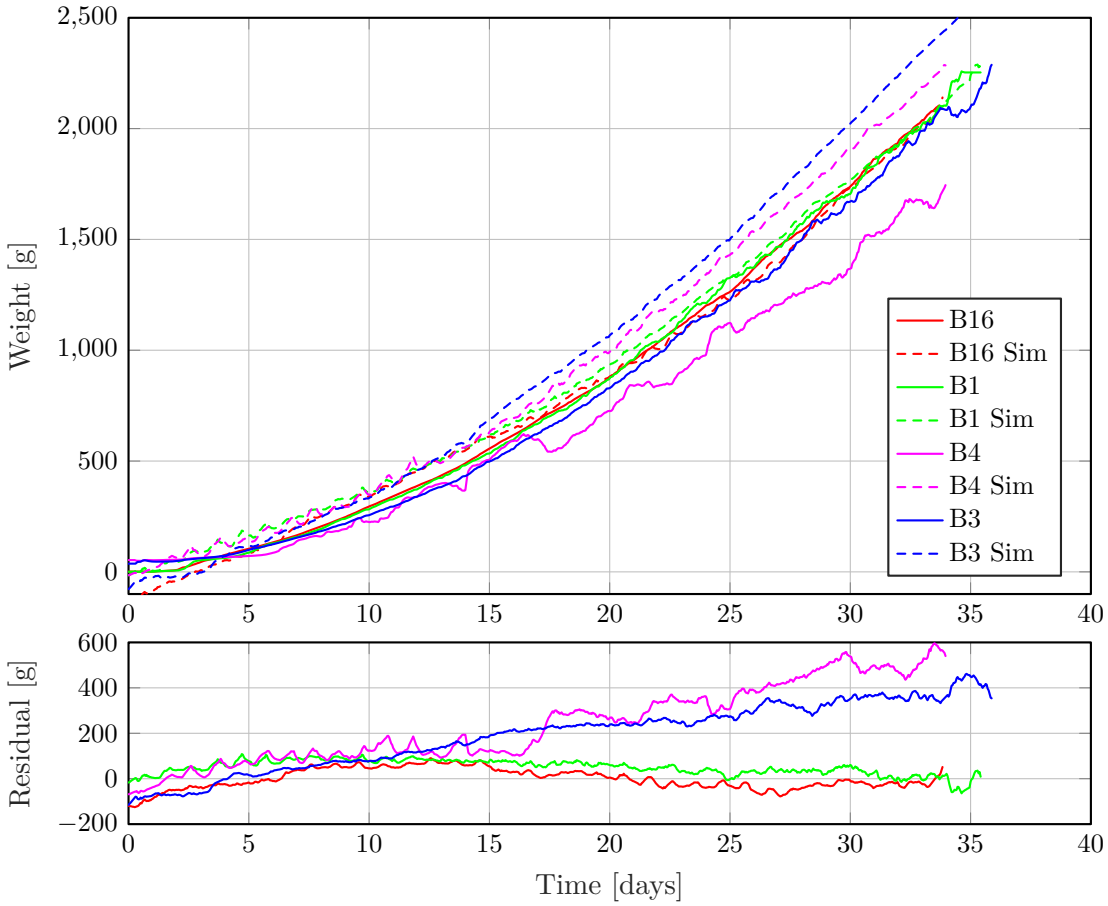

Figure 2.5: A simulation based on the estimated model parameters is conducted and the model output, shown as dashed lines, is compared with the measurements, shown as solid lines. From the complete test set the two best and two worst simulation results are presented.

The performance of the four models is evaluated with the use of the 30% selected test set. The test data consist of data from all four houses, this adds up to 19 batches corresponding to 19 test sets and simulations.

The presented results in *Figure 2.5* are based on house four. The training set of this estimation consists of three batches and the test set consists of 19 batches, where four of

these and their corresponding simulations are presented. The fit percentages for B16, B1, B3 and B4 are 92.93%, 91.39%, 64.59% and 43.53% respectively.

The presented results show that the model for batch B1 and B16 are capable of reproducing the correct behavior with the residual contained below a margin of 15 and 75 grams respectively. For the two remaining batches, B4 and B3, the residual indicates that the model is incapable of reproducing the correct behavior within a reasonable margin of error. It is, furthermore, seen that all models, even the good ones, start with a negative initial broiler weight even though the initial state is set positive during the simulation. It is also seen that the residual of B1 and B16 are negative at the start, then becomes positive and towards the end converges to the correct broiler weight. However, from a single estimation, it is not possible to determine if this is caused by an odd behaving test set or if some dynamics of the broiler are simply not described by this simple model structure.

2.3.2 Estimation Based on Batch Length

The second strategy is considered because the batches are of different length. This could be caused by measurements being stopped too early of running too long. This results in the last samples of the longer batches being cut off in order to make them the same length, which Matlab requires to allow estimation using the `greyest` estimator. This results in lost data which is not included in the estimation process and information are lost. Therefore this strategy is to separate the data into clusters according to the batch length, or sample size. In this process, the origin of the dataset, in other words, which house the batches are from, is not considered.

To avoid cutting off long batches, first, the question of how to split up the data must be answered. The task is to maximize the number of samples included in the estimation. This should be done such batches of approximately same length are placed in the same cluster. To achieve this a cluster search algorithm is used. A cluster search algorithm finds a number of center points and separates the data points according to the closest center point, where each data point is representing a batch. The more clusters that are used in this process the more data will be used, but for each cluster, an estimation must be made, and therefore each cluster result in a model with a new set of estimated model parameters. Furthermore if eg. a cluster is made for each dataset, the estimations are based on very little data, which would likely result in a model which is bad at reflecting other behavior than that of the dataset used to create the model. These argument implies that an optimal number of clusters can be found. This problem is considered in Appendix *A: Cluster Search*, where it is concluded that four clusters are appropriate.

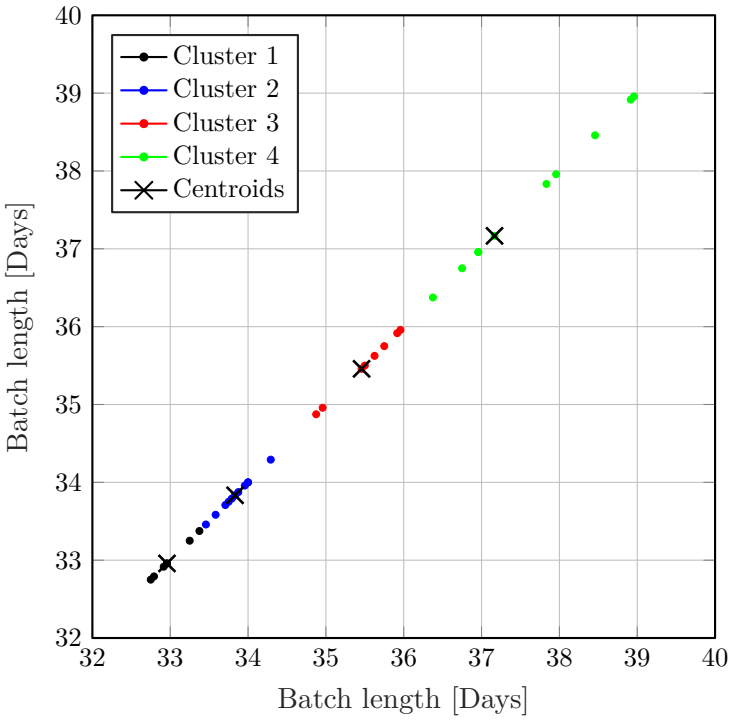

Figure 2.6: A plot of the length of batches to illustrate how they are distributed. The cluster search algorithm has split the data into four clusters which gives the maximum number of samples in the estimations.

The result shown in *Figure 2.6* is for the 55 batches which were first available, in Appendix *A: Cluster Search*, the result for all 161 batches is shown. As described in the start of Section *2.3: Individual Estimations* and as done for the first strategy, the data is now split into training and test sets. In this case, it results in four training sets and one test set. An overview of the process is presented in *Figure 2.7*.

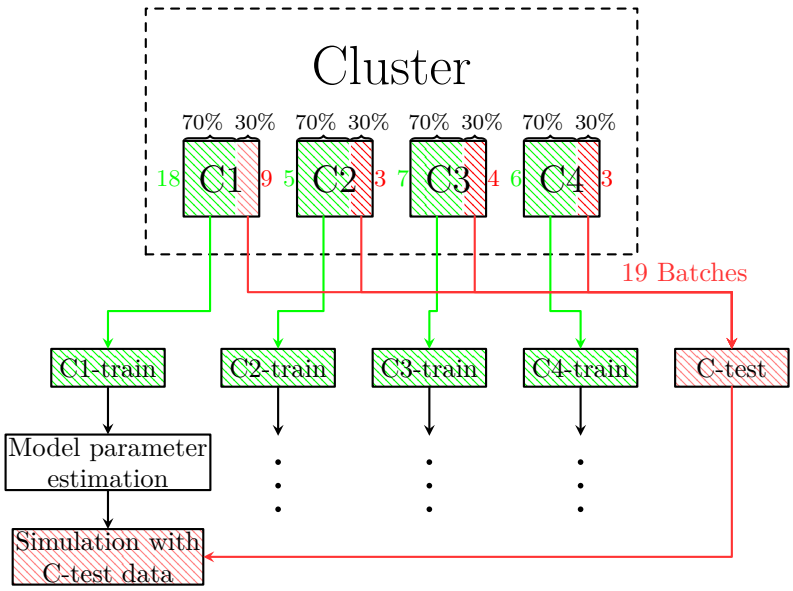

Figure 2.7: An overview of the model estimation process where the data is separated according to the associated cluster. In green, the number of training batches for each cluster and in red the number of test batches.

The estimation is again done using the Matlab grey box estimator `greyest`, explained in Appendix *B: Matlab Estimation Toolbox*. The four state-space models are used in a simulation, using the inputs and outputs from the test set, which results in a simulation output that can be compared to the measured output. A result of the estimation for one of the models related to one cluster can be seen in *Figure 2.8* where the two best and two worst results are presented.

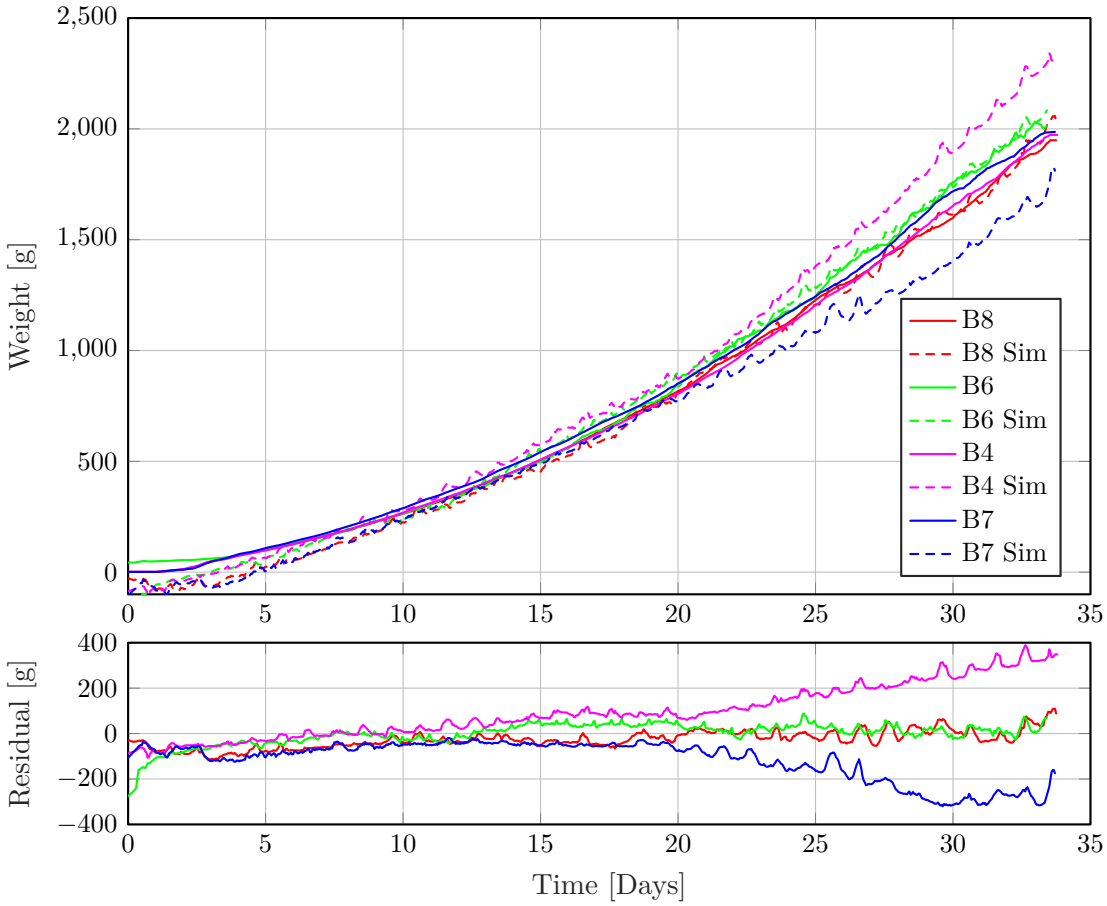

Figure 2.8: A simulation based on the estimated model parameters is conducted and the model output, shown as dashed lines, is compared with the measurement, shown as solid lines. From the complete test set the two best and two worst simulations results are presented.

The result seen in *Figure 2.8* is based on the second largest batches, seen as the red dots in *Figure 2.6* and cluster three in *Figure 2.7*. The training set of this estimation consists of seven batches and the test set consists of 19 batches, where four of these and their corresponding simulations are presented. The fit percentages for B8, B6, B7 and B4 are 92.21%, 92.02%, 75.07%, and 76.25% respectively. The residual of the two best fits is 71 and 109 gram, respectively. As for the estimation based on houses, it is seen that the initial values of all models are negative even though the initial states are selected positive.

2.3.3 Estimation with all Data at Once

The concept of the third strategy is to perform the estimation using all available data at once. This is done based on the idea that, the more data used in the estimation, the better will the model reflect a wide variety of scenarios. Furthermore training on a set

with data from all four houses could help to ensure a generic model which would fit the weight gain in all four houses, where different house dependent climate conditions could have an effect on the growth. An overview of the data splitting procedure is presented in *Figure 2.9*.

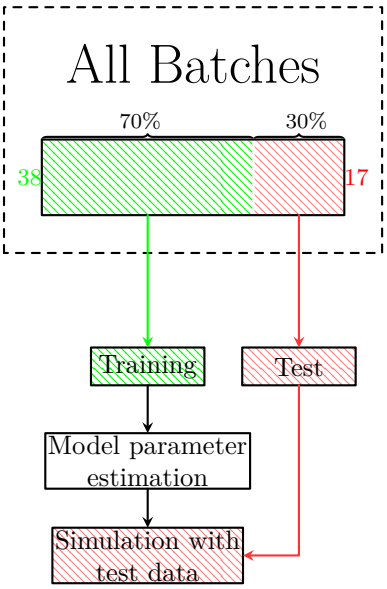

Figure 2.9: An overview of the model estimation process where the data is not separated. In green, the number of training batches and in red the number of test batches.

As for the first two strategies, the batches are split into a training and a test set by the same method. Afterward, the training set is used in the estimation, a model with a set of estimated parameters is again achieved. The model and the test set is used to perform the simulation which is again used to measure the performance of the model. These steps are all done in the same way as explained for the first two strategies.

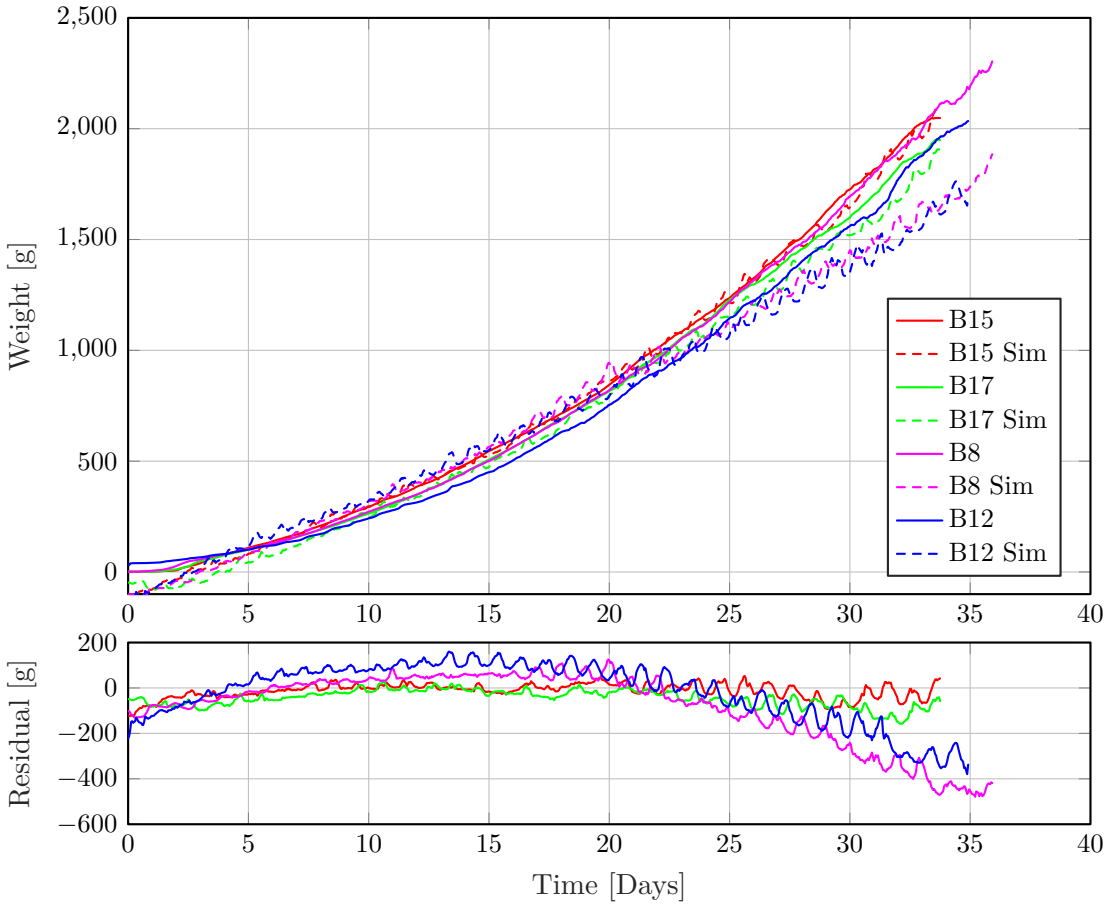

Figure 2.10: A simulation based on the estimated model parameters is conducted and the model output, shown as dashed lines, is compared with the measurements, shown as solid lines. From the complete test set are the two best and two worst results presented.

The results shown in *Figure 2.10* indicate some of the same behavior as seen in the two previous estimation strategies. All initial states are negative and even B15 and B17, the two best batches, are behaving a bit too linear to catch all dynamics of the broiler growth. The fit percentages for B15, B17, B12 and B8 are 94.37%, 90.18%, 78.08%, and 74.03% respectively. Because of the good fit percentages for the best batches, the model structure is deemed good enough to continue with.

As this estimation is based on all the data at once, the training set consists of 38 batches and the test set consists of 17 batches. The number of batches in the test set is smaller in this strategy due to rounding.

2.3.4 Estimation Considerations

A successful estimation has now been made for each of the four houses, each of the four clusters and all the data at once. This results in 9 different models that can be chosen between.

All results shown in Section *2.3.1 to 2.3.3* are results obtained with random bounded initial conditions and random selection of training and test sets. Therefore the results vary a lot and one estimation is not sufficient to determine which model is the best. In other words, the fit percentage of one estimation does not tell how well the model fits all data in general. A solution to this could be to run the estimation multiple times to

investigate which model gives the best result during multiple trails as the test set will be different at each iteration. This dilemma will be covered in the next section, Section *2.4: Monte Carlo Estimations*.

A concern about the different estimation strategies is the number of batches included in each estimation. The first strategy, estimation for each house, results in four models. However, dividing the batches in this way results in large size variation of the number of batches in each training set. The training sets consist of 12, 18, 3 and 3 batches for house 1-4 respectively. For the second strategy the, estimating according to batch length, the number of batches in the training sets are 5, 13, 7 and 11 batches.

It is desirable to obtain a model that fits the behavior of a wide variety of scenarios. To ensure this is the case, parameter estimation should be conducted using several batches. The data splitting strategy based on batch length naturally ensures this if the number of clusters is selected reasonably. However, this is not the case for the data splitting strategy base on houses, where the number of batches used for estimation depends on the provided data. As described in Section *2.1: Data Description* data was made available in two stages, for the additional houses added at a later stage the number of batches provided in each dataset is strongly varying. The added datasets would for the data splitting strategy based on houses result in an additional number of models corresponding to the number of added houses. As some of the added houses only contain very few batches, it is no longer possible to ensure that estimations are based on a wide variety of batches. Due to this fact is the strategy abandoned and will not be used in the further research.

The presented model simulations show a mix of well fitted and poorly fitted results. This indicates that the model and estimation procedure in some instances is incapable of catching the true system dynamics. In order to ensure that the full potential of the currently selected model structure is explored, the structure is kept and will be evaluated further as multiple estimations are conducted.

2.4 Monte Carlo Estimations

The models described in Section *2.3: Individual Estimations* are obtained with initial conditions of the model parameters selected from random distributions and random selection of test and training sets. Due to stochastic behavior included in the estimation method, further research has to be made in order to determine which model performs the best. As mentions in Section *2.3.4: Estimation Considerations* only the models based on batch length and the model using all data at once, estimation strategy two and three from Section *2.3: Individual Estimations*, will be considered. As described in Section *2.1: Data Description* extra data became available during the modeling process. Therefore all results shown in this section are based on 161 batches.

In Section *2.3.4: Estimation Considerations* it is concluded that only estimating once does not provide a clear picture of which model performs the best. Therefore an estimation, that estimates multiple times in order to clarify which model converges to the best fit percentage, is performed. Each iteration gives a fit percentage for all five models and as more and more estimations are performed the average fit percentages across all models will converge.

This Monte Carlo simulation iterates the single estimation presented earlier, 500 times as this is deemed enough for the fit percentages to converge. At every iteration of the Monte Carlo approach, the single estimations are initialized at new, thus the random 70/30 splitting of training and test set as well as the random initialization of parameter

values is conducted. As for the individual estimations, the performance of every generated set of model parameters is tested on a test set which originates from batches across all clusters.

This Monte Carlo estimation procedure is carried out for all three model structures described in Section *2.2: Model Structure*. An overview of the results is presented in *Table: 2.4* where an average fit of the 500 iterations for all combinations of model structures and estimation methods is shown. The three model structures are:

Structure 1:

$$\begin{aligned} x[t+1] &= x[t] + [B_{11}B_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] \end{aligned} \tag{2.4}$$

Structure 2:

$$\begin{aligned} x[t+1] &= x[t] + [B_{11}B_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] + [D_{11}D_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [D_d]w[t] \end{aligned} \tag{2.5}$$

Structure 3:

$$\begin{aligned} x[t+1] &= [A_1]x[t] + [B_{11}B_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] + [D_{11}D_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [D_d]w[t] \end{aligned} \tag{2.6}$$

Model:	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**All Data**
Structure 1	77.62%	78.68%	78.86%	75.24%	78.68%
Structure 2	79.95%	80.78%	80.81%	77.63%	80.74%
Structure 3	63.74%	64.96%	63.80%	57.26%	64.74%

The results indicate that model structure two yields the best results as the highest average fit percentages are obtained with this model, further explanation of the complete results is found in Appendix *C: Choosing Integrator Model Structure*. As model structure two is deemed the best structure only it is further reviewed in the main report, the converging fit percentages for the five models of structure two are shown in *Figure 2.11*.

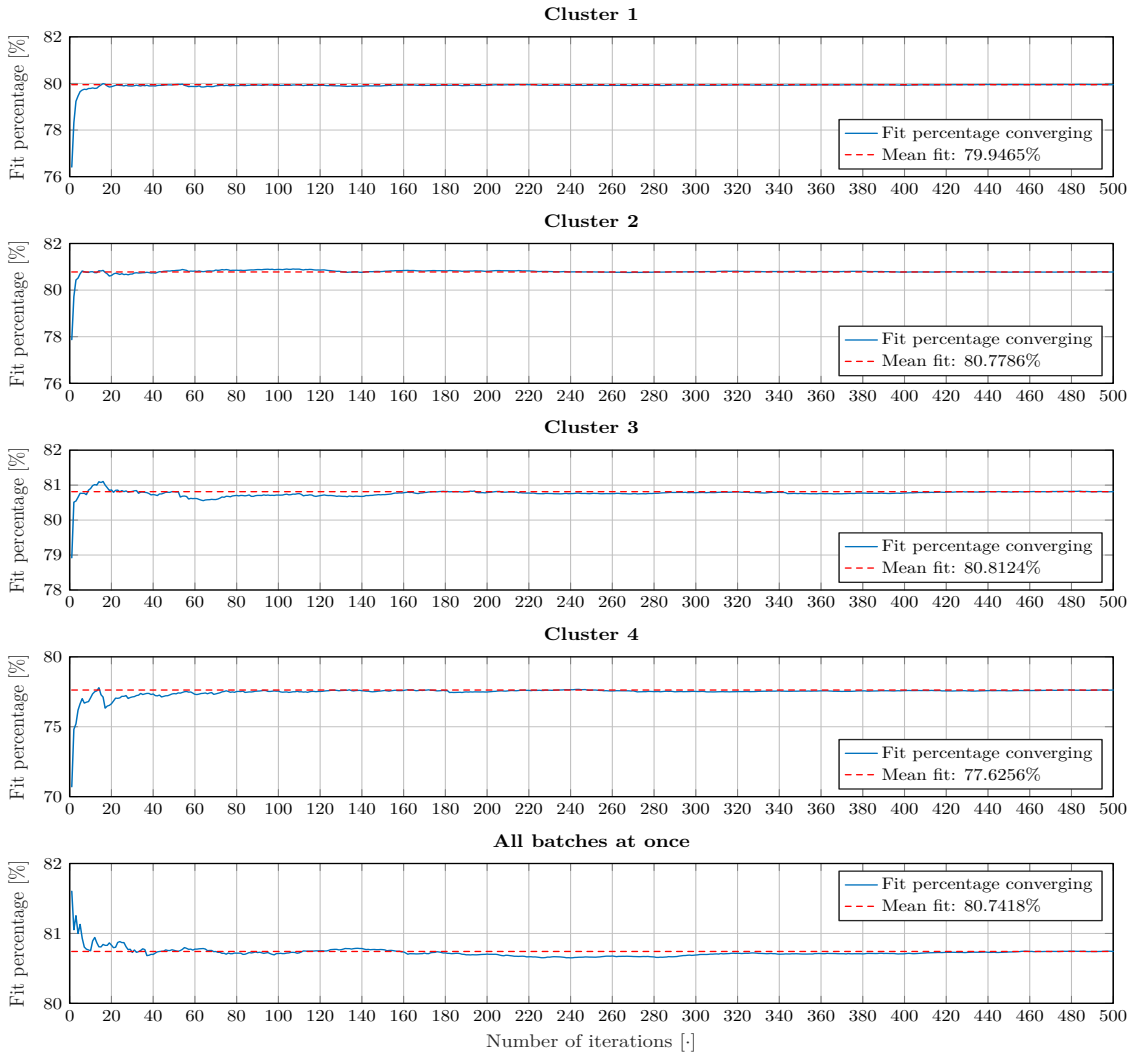

Figure 2.11: The fit percentages of the five models as they converge.

From this, it can be seen that "Cluster 3", shown as the red cluster in *Figure 2.6* converges to the highest fit percentage, 80.81%. However, as the estimation is a numerical process with random initial condition some estimations are, as seen in Section *2.3: Individual Estimations*, sentenced to go wrong. This is not desirable as an incorrect model will drag the overall fit percentage in the wrong direction. *Figure 2.12* shows the first 100 iterations for the estimation, "Cluster 3", and it shows how some of the estimations are diverging significantly from the mean.

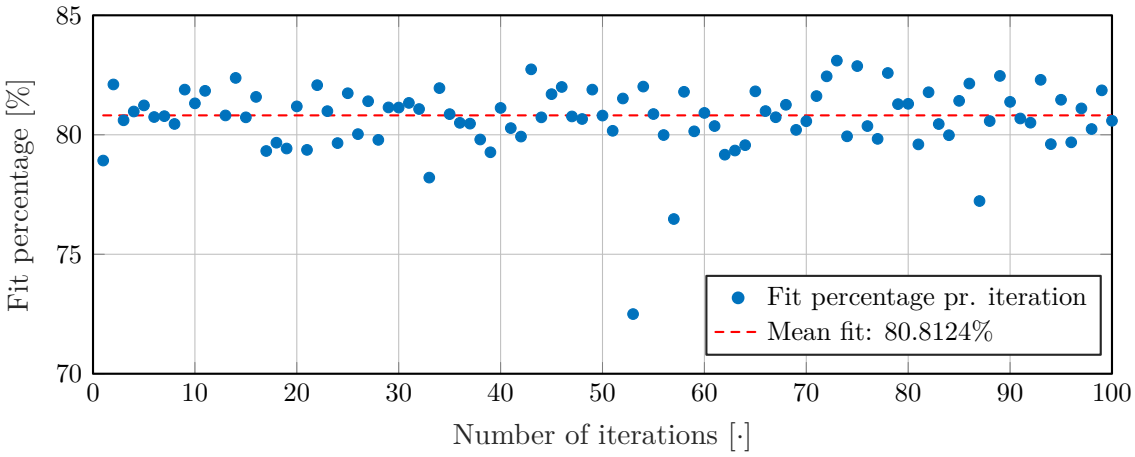

Figure 2.12: The fit percentages for each of the first 100 iterations.

It would be desirable to remove models with low fit percentages, keeping only the best model with the highest fit. This will, however, due to the design of the Monte Carlo simulation not ensure that the selected model is best at reflecting the variety of scenarios as originally desired. The model could originate from a lucky match between training and test set and thus have an artificially high fit percentage. In order to explain this issue fictional batches are described and shown only with the purpose of explaining this issue. The issue originates from the random division between the training and test data, some combinations will naturally cause better results than others. In *Figure 2.13* and *2.14*, six fictional batches are displayed in two scenarios where they are randomly divided. The batches are colored in pairs that behave the same way. In other words, pairs of the same color have the same biased behavior, which could be caused by a bias toward the houses, the season etc. It should be stressed that this is only a fictional example and the difference between batches, e.g. spikes or deviation in weight at the end of a batch, is only an expression of an anomaly.

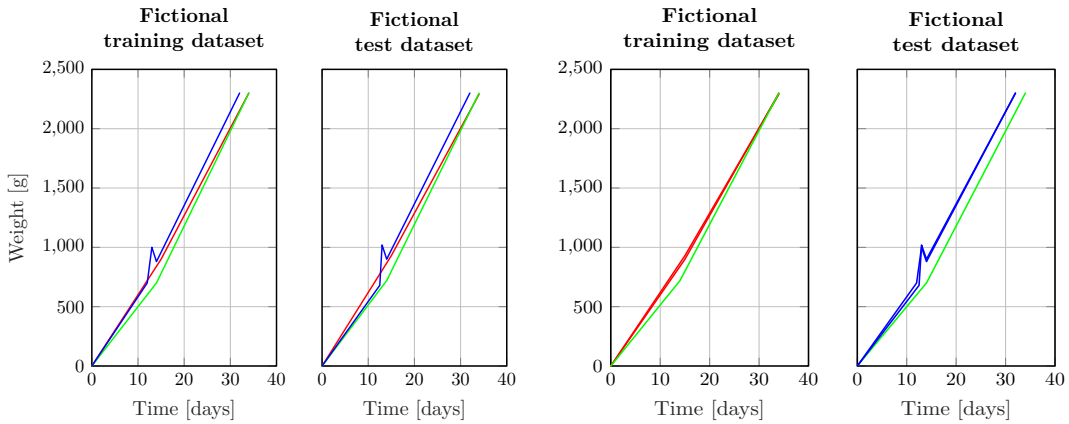

Figure 2.13: **Case 1:** The test data and training data are almost identical which would result in a good fit.

Figure 2.14: **Case 2:** The test and training data are split up so they to some extent are different.

For case 1 test and training sets are almost identical which would result in a good estimation. For case 2, the faster behavior of the blue batches will not be included in the estimation and therefore the model would be slower than desired which will affect the fit percentages. The opposite can be said if the data is to be divided such that

the batches in the training and test data are swapped. These considerations lead to the question of which estimation to keep and which to discard. This problem is solved by using Grubbs test for outliers, which from a statistic perspective determines whether a result is an outlier or not. If an estimation is categorized as an outlier, it is removed. Removing the outliers changes the average fit percentages slightly, however, the conclusion is the same as previously written. The number of outliers that are removed from each estimation process of 500 iterations ranges from zero to four. This clearly indicates that the estimation is successful in the vast majority of the iterations. The Grubbs test and the complete result for all three model structures after outliers are removed are shown in Appendix *C: Choosing Integrator Model Structure*.

2.5 Integrator Model Estimation

In Section *2.4: Monte Carlo Estimations* it is concluded that model structure two, see *Equation: (2.2)*, yields the best results out of the three different structures. Structure two has the system matrix as an integrator and the input and feedforward matrices are estimated. For this model structure five estimations with 500 iterations were performed, resulting in a total of 2500 models. The focus of this section is to determine which of the 2500 sets of parameters to use.

Based on the five Monte Carlo estimations, using the four clusters and the total dataset, three models are selected from each estimation. This results in 15 models, which in order to evaluate their performance are used in 15 simulations that are carried out on all 161 batches. The three different models, or rather the parameters contained in the models for the simulations, are selected between the results from all 500 iterations of each estimation, as follows:

1. The parameter set, of all 500, that yields the highest fit percentage, during estimation.
2. The average of all 500 parameter sets.
3. The parameter set, of all 500, that yields the lowest fit percentage, during estimation.

The following simulations are conducted with all results, including potential outliers.

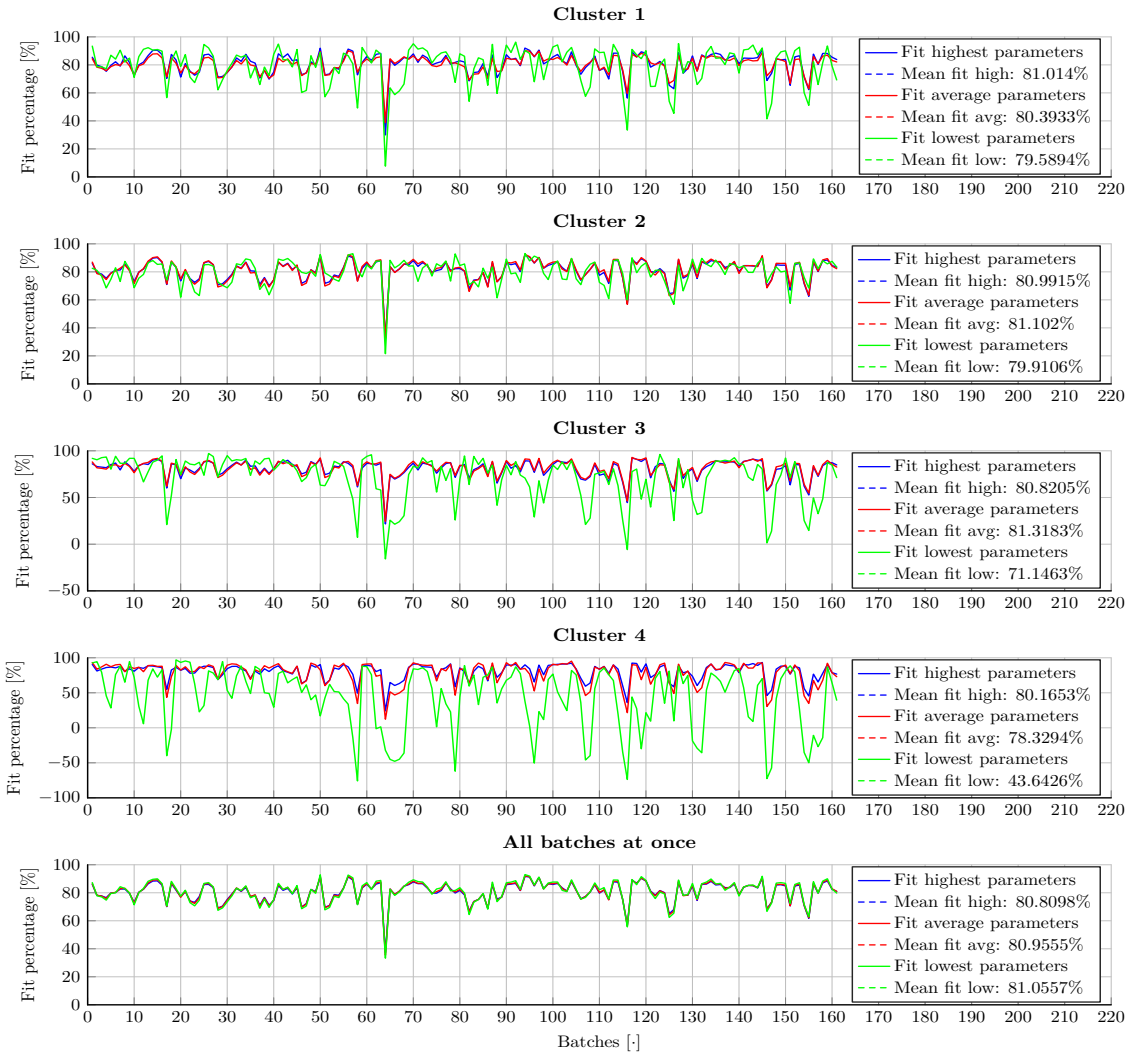

Figure 2.15: The fit percentages of the three different models for all five estimations.

In *Figure 2.15* the results of all 15 simulations are plotted. For cluster 1 and 4, it is seen that the parameters from the model with the highest fit percentage in the estimation also was the highest fit percentage when used on all batches. This observation is not in line with the statement concerning unfavorable separation of test and training set, from Section *2.4: Monte Carlo Estimations*. The statement suggests that the highest or lowest fit percentages for the test set might not necessarily be the highest or lowest overall fit percentage due to the scrambling of training and test set. However, for cluster 2 and 3, it is seen that the models based on the average of all parameters have the highest fit percentage. This is to be expected as the Monte Carlo estimation of the parameters is cycling through various training and test set combinations, thus the average based models are exploiting more information across all batches. It is, furthermore, seen that the model based on averaged parameters from cluster 3 has the overall highest fit percentage. The same behavior was seen during the Monte Carlo simulation, see *Figure 2.11* where cluster 3 likewise yielded the highest results.

For the last estimation, based on all data at once, it is seen that the model based on parameters from the iteration with the lowest fit percentage is actually the best performing when used on all batches. This is probably due to an unfavorable combination of training

and test data which gives the low initial fit percentage. However the estimation still managed to reproduce the dynamics of most of the batches. This supports the statement at the end of Section *2.4: Monte Carlo Estimations*, that an artificial low fit can occur and the Monte Carlo estimation is necessary.

However, as stated earlier these simulations are using all results including the outliers detected by the Grubbs test, see Appendix *C: Choosing Integrator Model Structure*. As the estimation process is numerical, it can not be guaranteed that all iterations find an optimal solution.

The same procedure is now tried for the results of the Monte Carlo estimation, but without outliers, as this might change some of the unexpected results.

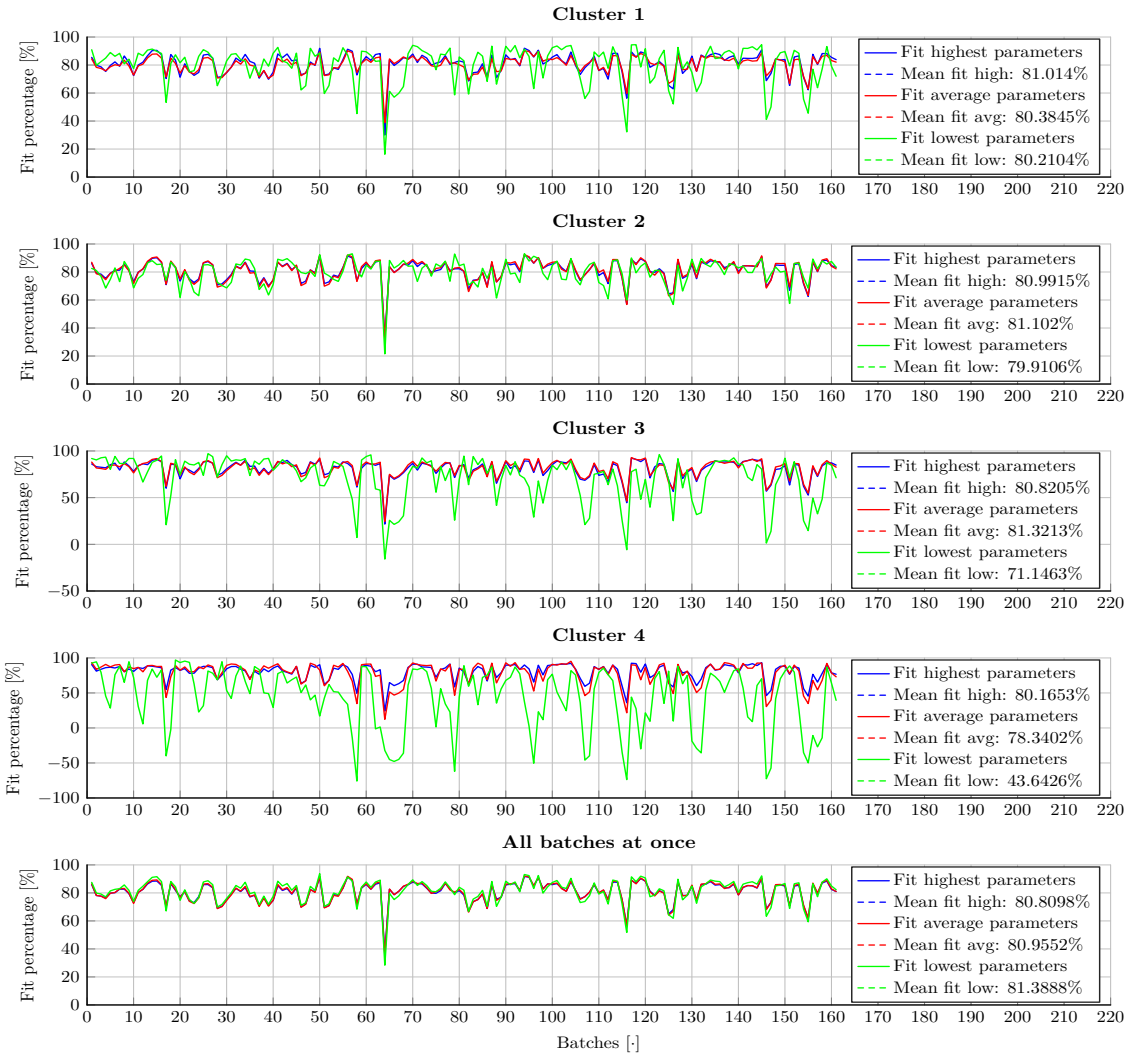

Figure 2.16: The fit percentages of the three different models for all five estimations without outliers.

The fit percentages of the 15 models when outliers are removed, *Figure 2.16*, are approximately the same as when outliers are included. This implies that the most iterations of the estimation are successful and few outliers exist. This is also seen in Appendix *C: Choosing Integrator Model Structure*, as very few results are categorized as outliers and removed. However, as the three best results in *Figure 2.16* are only separated

by 0.2%, it is hard to argue why one is better than the other. Therefore the parameters of the three models with the highest fit percentages are investigated further.

As the models perform almost the same, it could be caused simply by the models being approximately the same set of parameters. To recap, the model structure and the estimated parameters are stated here:

The estimated parameters are $B_{11}, B_{12}, B_d, D_{11}, D_{12}$, and D_d. To realize whether the parameters are actually the same, the six parameters are plotted for all iterations in *Figure 2.17*. The three sets of parameters extracted from the total of 15, named model A to C, are the *average parameters from cluster 3, cluster 2* and *all batches at once* respectively, as they are the three best performing average parameter sets.

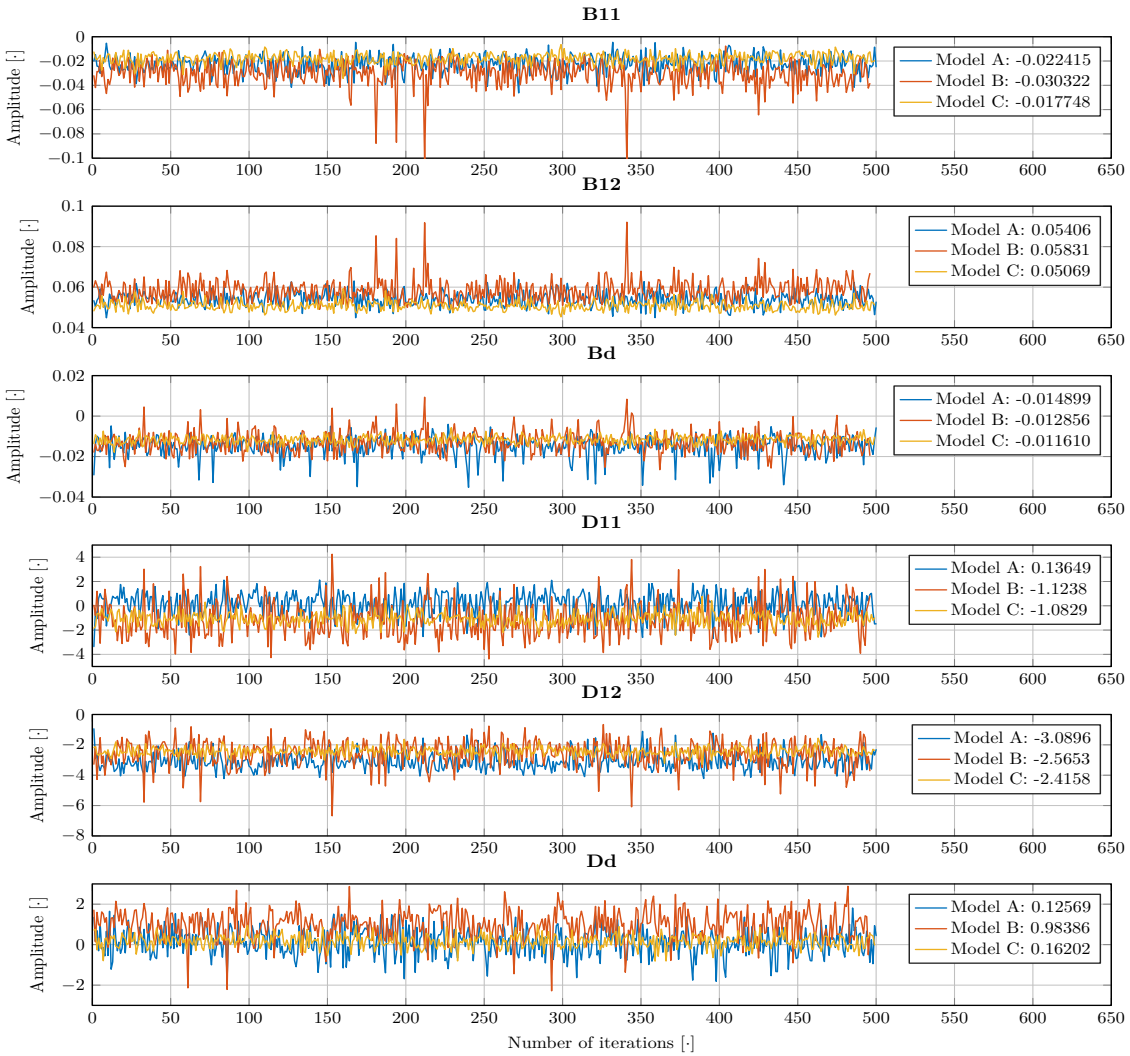

Figure 2.17: The parameters for all iterations, from the three best performing average models. The values shown in the legends are the mean of all iterations.

As it is seen in *Figure 2.17* the parameters in the input matrix are almost the same. The feedforward parameters are slightly different. Where D_{11}, the feedforward coefficient of the temperature, is positive for cluster 3, but negative for the two other models. However, the values of D_d are all positive but vary a lot, which seems to cancel out the difference in D_{11}, as all models perform similarly.

The best performing model from *Figure 2.16*, *"All batches, parameters from lowest fit"*, is not included as it is only one set of parameters. For the sake of convenience, this is named model D. However, this model might also be interesting to compare with the best performing set of average parameters. Therefore model D is stated in *Equation: (2.7)*.

$$\begin{aligned} x[t+1] &= x[t] + [-0.0273 \;\; 0.05637] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-0.014] w[t] \\ y[t] &= x[t] + [-0.1691 \;\; -2.838] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [0.6614] w[t] \end{aligned} \tag{2.7}$$

As it is seen the parameters are again almost the same, compared to model A, B, and C. The feedforward parameters are however more in line with the result from model B and C, as D_{11} is also negative. The fact that three out of the four selected models indicate the same parameters, implies that this is a more likely model structure.

The three models that indicate the same parameters are basically the same. However, to get an idea of the performance, besides the fit percentages, the three models are simulated and shown together with the mean and standard deviation of all available 161 batches.

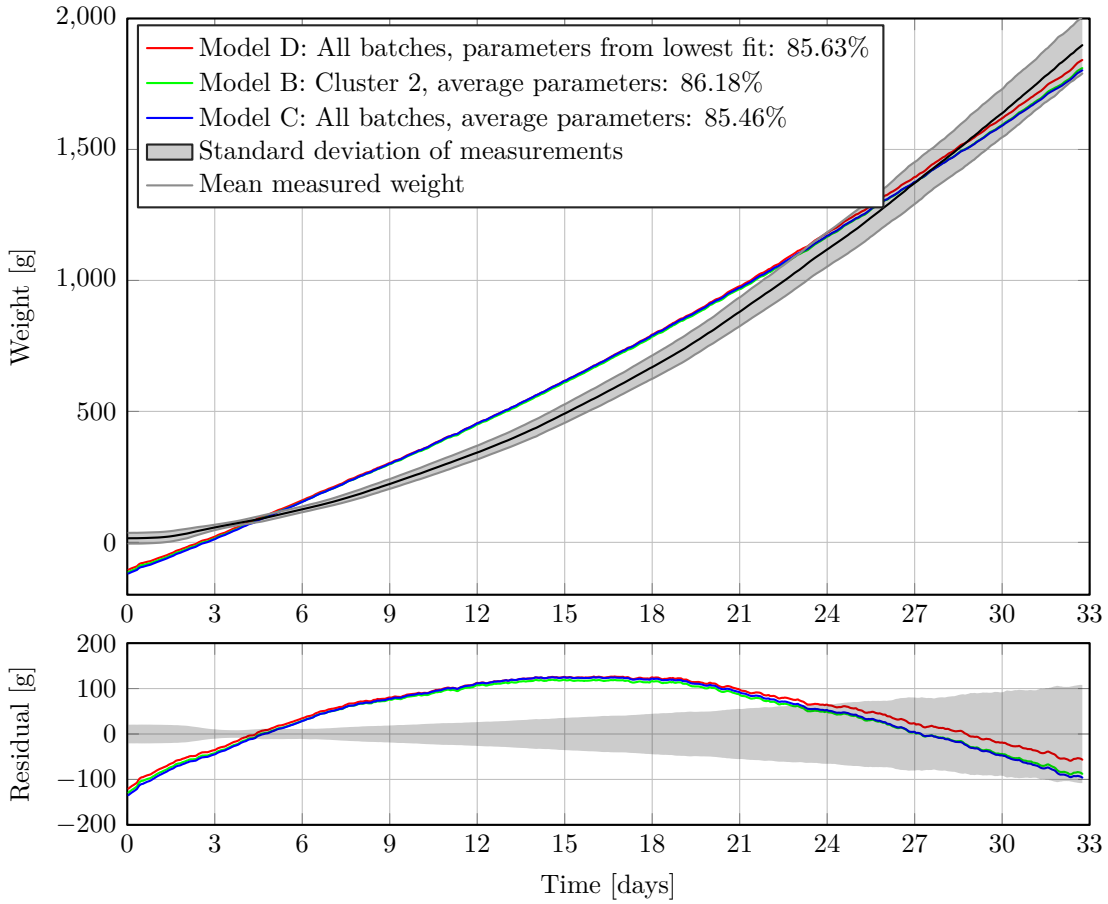

Figure 2.18: The mean of all batches, the standard deviation, and the three best models. A relative short batch period is represented, as a mean for all batches only exits for the length of the shortest batch.

As it is seen in *Figure 2.18* the three models behave very similar. However, the models do not truly reflect the dynamics of the data, as the models seem to follow a more linear trajectory. Furthermore, it is seen that the models start at approximately -160 grams

which is highly undesired as the weight of a broiler cannot be negative. These simulations are conducted with the initial state value set to 40 grams.

The linear trajectory indicates that the current model structure is not sufficient to describe the growth of a broiler. Signs of this where seen in Section *2.3: Individual Estimations*, however because of the separation of training and test data, as well as the different quality of the data it was not possible to fully determine if the model structure was sufficient at that time.

From a physical perspective, the almost linear trajectory, suggests that the broiler gains the same weight during the first days as the last days of the batch. Looking at the mean of all data this is not true. In other words, according to the provided data is the average broiler gaining around 140 grams in weight the first week and approximately 150 gram the last day during a batch. The dynamics of this clear difference in absolute weight gain, from day to day, is not caught by the model.

The models are not able to describe all dynamics of the broiler. However, these models are believed to be the best performing when considering a first-order model structure with a pure integrator and estimating the input and feedforward matrices. In order to determine if these results can be improved the models should be compared with models of other structures and optionally other estimations strategies.

2.6 Unstable Model Estimation

In this section, it is attempted to improve the previous modeling results, found in Section *2.5: Integrator Model Estimation*, which models a broilers weight during a batch, based on temperature and humidity as inputs and outside temperature as a disturbance. The improvement is attempted by utilizing a different design approach for the first order model originally presented in Section *2.2: Model Structure*. The parameters of the model are estimated by the same method as presented in Section *2.4: Monte Carlo Estimations*.

The best result from the previously conducted model design consists of parameter estimation for first order models with an integrator design. This model structure is proven incapable of sufficiently following the desired growth dynamics of a broiler with the current selected inputs, temperature and humidity. An approach that estimated the system matrix, instead of selecting it as an integrator was also tried. This provided results with a significantly lower fit and higher variation among the individual estimations.

In order to obtain an acceptable model, it is thus necessary to explore models with different characteristics. The previous models are all stable models, and with the selected inputs, they provide a dynamic behavior that is not as exponential growing as required. It is due to this finding, desirable to investigate unstable models as they, with the same inputs, will allow an exponential behavior due to the unstable nature of such a model. The instability of the model should be limited an thus only be slightly unstable as it otherwise could cause the system equations to blow up within a short time.

Finding the optimal pole placement of the system matrix is done with the help of the same estimation toolbox as previously used. Only a slightly unstable model is desired, thus the pole placement is constrained to the interval $[1.0001, 1.100]$ within the z-domain. Based on this constraint, parameters of two model structures will be found by estimation and compared by the exact same methods used in the previous model estimation.

The two model structures contain an estimation of:

4. A constraint to $[1.001, 1.100]$, B and B_d, thus leaving C as unity and D, D_d as zero.
5. A constraint to $[1.001, 1.100]$, B, B_d, D and D_d, thus leaving C as unity.

The estimation procedure is identical to the one presented in the Section *2.4: Monte Carlo Estimations*. The full in-depth results for both model structures are presented in Appendix *D: Choosing Unstable Model Structure*, the mean fit percentages obtained for both structures and all estimation methods are presented in *Table: 2.6* together with the results from the previous model structures.

Model:	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**All Data**
Structure 1	77.62%	78.68%	78.86%	75.24%	78.68%
Structure 2	79.95%	80.78%	80.81%	77.63%	80.74%
Structure 3	63.74%	64.96%	63.80%	57.26%	64.74%
Structure 4	83.19%	86.78%	86.79%	84.21%	85.92%
Structure 5	85.57%	88.68%	88.65%	86.39%	88.25%

It is based on the results, found that model structure five yields the highest fit percentages out of the two additional structures. This follows the results in the Section *2.4: Monte Carlo Estimations* where it is found that the model structure incorporating an estimation of the feedforward matrix D, yields the highest fit percentages. The previous best result, shown in Section *2.5: Integrator Model Estimation*, yielded a fit percentage of 80.81% the new design approach with a pole placed outside the unit circle is showing significantly higher results for both model structures. These results indicate that representing the system with an unstable model is more correct and reflects the system dynamics to a greater extent.

The best performing models for each of the additional structures are:

Structure 4:

$$\begin{aligned} x[t+1] &= [1.0024]x[t] + [0.019900 \;\; 2.6000 \cdot 10^{-3}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-2.9221 \cdot 10^{-4}]w[t] \\ y[t] &= x[t] \end{aligned} \tag{2.8}$$

Structure 5:

$$\begin{aligned} x[t+1] &= [1.0021]x[t] + [0.028800 \;\; 4.2423 \cdot 10^{-4}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-4.7552 \cdot 10^{-4}]w[t] \\ y[t] &= x[t] + [-3.0780 \;\; 0.32120] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [5.6000 \cdot 10^{-3}]w[t] \end{aligned} \tag{2.9}$$

A comparison of the two different model structures is presented in *Figure 2.19*. The presented models are the final models for each structure that are chosen according to Appendix *D: Choosing Unstable Model Structure* based on their performance and structure.

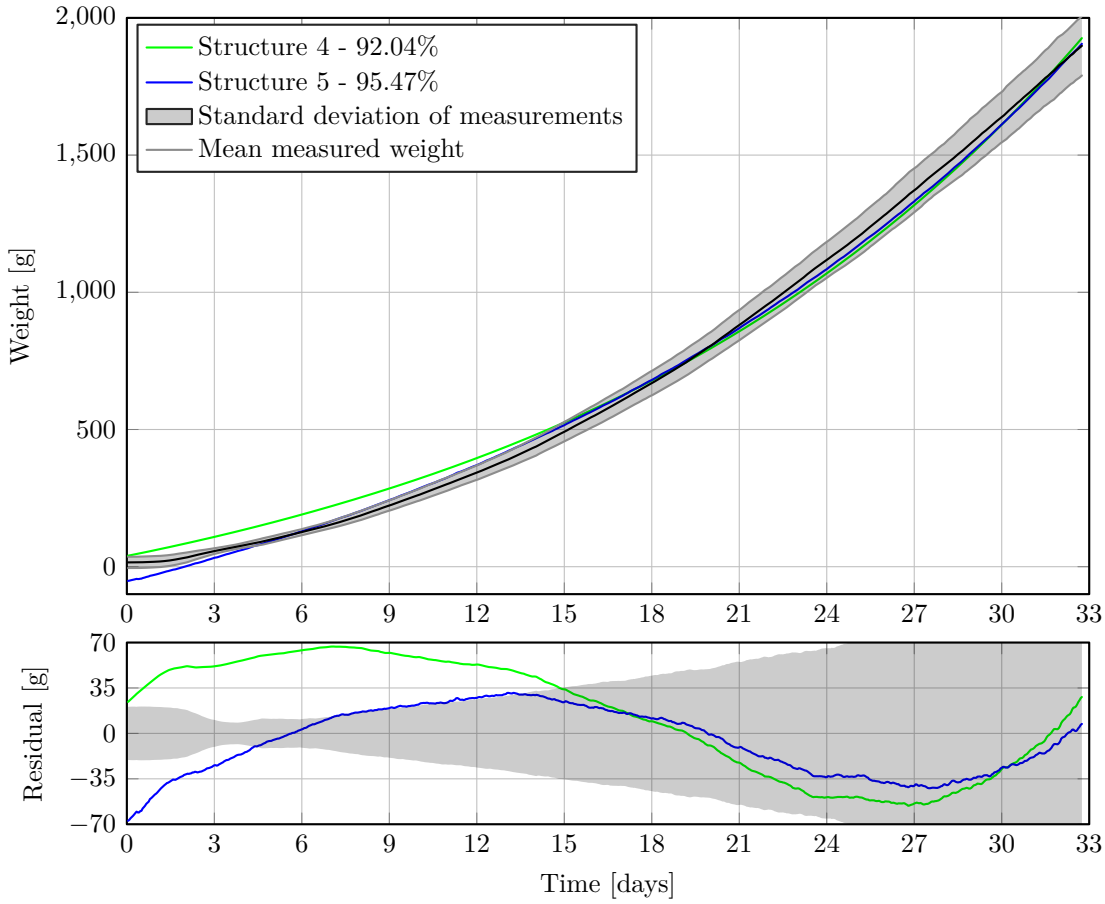

Figure 2.19: The mean of all batches, the standard deviation, and the two best models.

The presented simulation is conducted with an average of all 161 batches. Comparing the two model structures some significant differences are visible due to the inclusion/exclusion of the feedforward matrix. During the estimation of structure five, the overall effect of the feedforward matrix is found to be negative. The impact of this result is visible during simulation as structure five yields a significantly lower weight during the first days, compared to structure four. The higher fit percentage of structure four is thus also a result of the added feedforward matrix as the weight output is generally closer to the desired output compared to structure four.

Overall it can be concluded that the approach of mowing the system pole outside the unit circle has significantly improved the accuracy of the models. Leading to models that follow the desired trajectory closer and thus minimizing the residual and obtaining significantly higher fit percentages.

2.7 Model Validation

The purpose of this section is to validate the previous found broiler models with respect to both their performance and behavior. The validation is based on new data that has not been used throughout the system identification.

The models found in the previous sections are all based on data-driven parameter estimation. The estimation process is conducted in a Monte Carlo like method as described in Section *2.4: Monte Carlo Estimations*. During the estimation process, all available data is used. As a consequence, when analyzing and comparing model performance, the results

are based on data that previously has been used in the parameter estimation process.

The new data originates from the same broiler farm locations as previously used. As data from these farms are constantly gathered, new data is obtained since the modeling procedure started, thus data from these additional batches is now used to validate the models. The validation data is created as a mean of all the new available batches. Furthermore, the standard deviation is shown in the same way as in previous model performance comparisons. In total there are 30 new batches originating from 8 houses.

The comparison of the different model structures is presented in *Figure 2.20*, where also the residual of the broiler weight is shown.

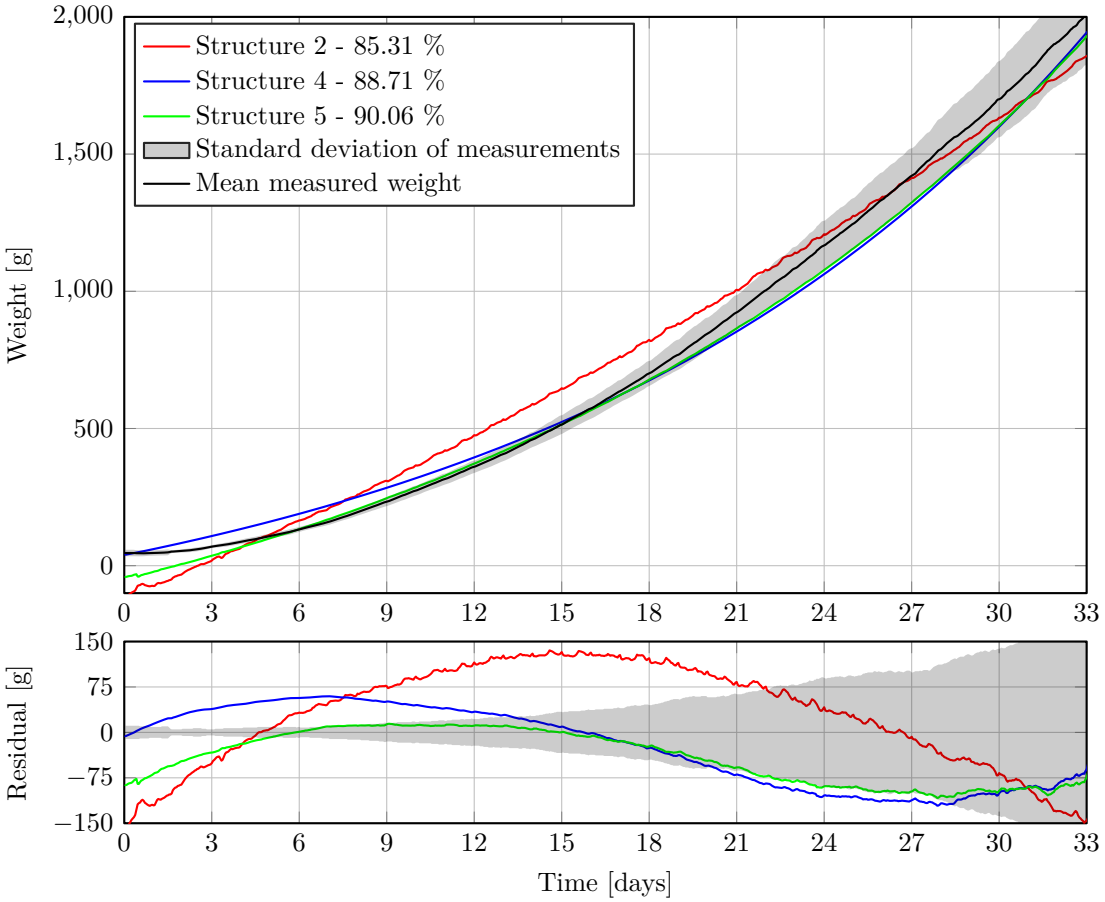

Figure 2.20: The mean weight of all 30 batches, the standard deviation, and the three best models.

The comparison includes only the best model found in Section *2.5: Integrator Model Estimation* as the remaining behave and perform almost identically. The different modeling approach taken in Section *2.6: Unstable Model Estimation* is clearly visible as the two models, structure four and five behave significantly different than the pure integrator model, represented as structure 2. The integrator model does not reflect the true broiler dynamics as the model behavior is close to a true ramp. The challenges of this model structure seem persistent as the same behavior was experienced when the averaged test data is used to validate the model structures performance.

The behavior shown by the two unstable model structures is very similar to the results found during the usage of the averaged test data, as shown in *Figure 2.19*. Apart from the higher fit percentages compared to structure two, is the behavior of the two new

structures closer to reflecting the true broiler dynamics. Comparing the results found during the usage of test data and the new results obtained by the validation data reveals that the fit percentages across all three model structures have increased significantly. The best-obtained fit with the test data and the new results, based on the validation data, are shown in *Table: 2.7*

Model:	**Validation data**	**Test data**
Structure 2	85.31%	80.81%
Structure 4	88.71%	86.79%
Structure 5	90.06%	88.68%

The increased fit percentages could be caused by the smaller amount of batches used to generate the averaged validation data. Since only 30 new batches are available compared to the 161 in the previous performance evaluation, this implies that the new batches represent a much smaller spread in time. As the spread is smaller, it is likely that the batches are more identical as long time changes in eg. broiler genetics, food composition or similar factors that slowly evolve over time, cannot yet make an impact on the batches. Some dynamics might have been diluted by the averaging procedure of the 161 batches thus not reflecting all of the true input and output dynamics seen during a batch, thus artificially lowering the fit percentages.

The results show a significant performance gain for the pure integrator, structure two, whereas the remaining structures also gaining fit percentages but within a more reasonable range. It is furthermore noticeable that the performance gap between model structure four and five is close to maintained.

The simulation based on the newly obtained validation data shows for all three model structures that the previously reported performance is valid and even shows a performance gain for all structures. Furthermore are the displayed model structure dynamics and behaviors similar to the previously found, which indicates that the previously discovered model structure properties are valid.

For further use during the development of a controller, one model needs to be selected to center the control structure around. Based on both the results found during validation and as described in Section *2.5: Integrator Model Estimation*, does the pure integrator structure not perform as desired and will thus not be used for control purpose. Model structure four and five show similar performance, though structure five has a slightly higher fit percentage as this model structure incorporated a feedforward term.

The controller development will be based on model structure four, as this structure yields good fit percentages and does not incorporate a feedforward term and thus can be implemented in more advanced control structures without additional considerations. The explicit model structure that will be used originates from *Equation: (2.8)* and is repeated here:

Structure 4:

$$x[t+1] = [1.0024]x[t] + [0.019900 \;\; 2.6000 \cdot 10^{-3}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-2.9221 \cdot 10^{-4}]w[t]$$

$$y[t] = x[t]$$

It is deemed important that the model structure is rather simple, such that the focus can be shifted towards more advanced control options. Model structure four is a good match between simplicity and performance and is thus chosen to be used for control purpose.

Part II

Control Design

Controller 3

In this chapter, the design of the controller structure is explained. The chosen control approach includes a batch-wise error model, a time-wise error model, a Kalman filter and a model predictive controller.

3.1 Control Problem

A general introduction to the control strategy will be given throughout this section. Thus covering the main idea behind the chosen approach and explaining the general control structure.

As earlier described in Section *2.2: Model Structure* is the goal of the controller development to design an MPC structure that aids the farmer in selecting the optimal climate conditions in order to achieve the desired broiler weight. The models found throughout the previous chapters are models describing the weight of a broiler based on temperature and humidity as inputs and the outside temperature as a disturbance. In a real-world practical situation will it not be possible to control both the temperature and humidity fully independently as relative humidity is affected by the specific temperature.

When considering the model from a control perspective it is chosen only to use the temperature as the input, where both humidity and the outside temperature are regarded as disturbances. Considering the model from this aspect represent a scenario in line with the farmer's main climate focus which is the house temperature. The two disturbances are controlled internally by the ventilation system, the humidity is kept within a specified bound and the outside temperature is actively accounted for by the temperature control. Rearranging the model structure to a form with one input and two disturbances implies that the MPC structure will consist of one manipulated variable and two measured disturbances. A block diagram presenting an overview of the general structure is shown in *Figure 3.1.*

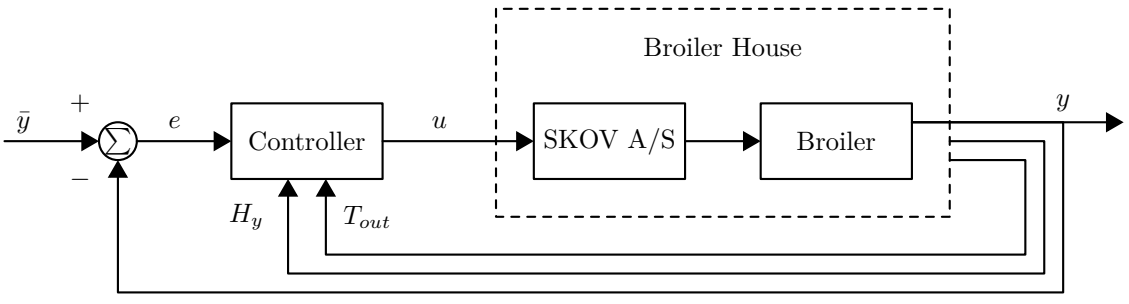

Figure 3.1: Block diagram of the general control structure.

The general control structure is centered around the broiler house which represents the existing system. The broiler block inside the broiler house represents the rearranged model with temperature as the input, weight as output and humidity and outside temperature as disturbances. The block named SKOV A/S inside the broiler house represents the existing

broiler management platform, including all process control parameters as climate control, food supply, light scheduling etc. The MPC developed throughout the following sections will according to the block diagram act as a high-level controller. The controller applies a desired temperature reference for the existing climate control system in order to achieve a reference broiler weight. Additionally, are the two disturbances actively handled through the internal model of the controller.

The described controller design is based on regular MPC design which is applicable for a wide range of systems. The current system can be viewed as a batch production process due to the characteristics described in Section *1: Introduction*. This property can be utilized to design a controller that is tuned towards a batch-specific application. An MPC designed towards batch operation will be the focus of the controller development. The design incorporates shrinking the control and prediction horizons to the batch length such that the horizons are always equal to the remaining batch length. This approach is called shrinking horizon MPC or Batch-MPC (BMPC) and takes advantage of the batch structure of the broiler production [10].

Further specialization of the BMPC is possible by incorporating element from iterative learning control (ILC), where information from previous batches is stored and used to improve future batches [11]. The introduction of elements from ILC also imposes the introduction of two different time measures. These consists of a short- and long-term time scale where the short-term scale is defined as the sample time applied in a batch and the long-term as the batch count. An illustration of these properties is presented in *Figure 3.2*.

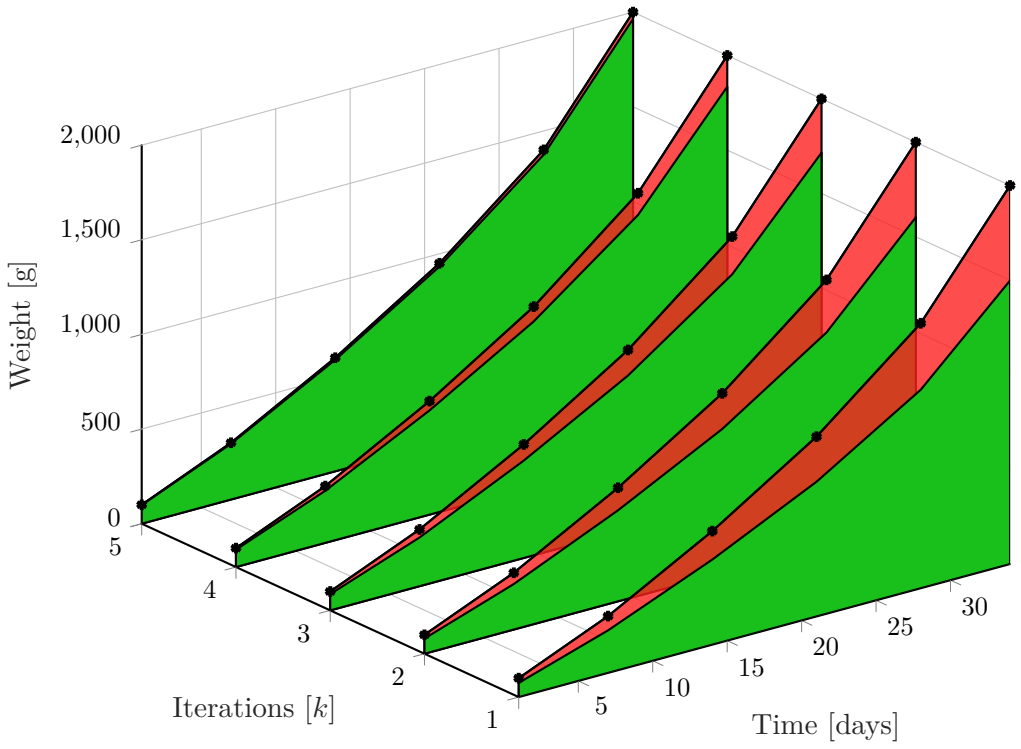

Figure 3.2: A 3 dimensional plot of the short- and long-term time scales used under ILC. The green area is the archived broiler growth and the red is the residual from the reference trajectory. As batches are completed the residual gets smaller.

As two-time scales are defined due to the introduction of ILC, two control objectives are considered. The short-term objective is handled by the BMPC as the focus is reference

tracking and thus minimizing the tracking error across the batch horizon is defined as a cost function inside the BMPC.

The long-term objective is to drive the error towards zero as the batch count increases towards infinity. This long-term objective can be achieved by the use of ILC as this control type is capable of monotonically decreasing the error towards zero [12].

Due to the repeating characteristics of the broiler growth process the available broiler model and the trajectory tracking objective it is obvious to combine both ILC and BMPC into one control strategy. Different combinations are successfully used for a wide variety of systems and use cases. These are spanning from temperature tracking in an experimental batch reactor [13] to control of a boiler-turbine power plant [14] and more [15] [16].

Combining the two control methods allows compensating some of the undesirable properties each method has. A classic implementation of pure ILC will not be capable of canceling real-time disturbances as the tracking error is only decreasing across complete batch iterations. Trough the use of BMPC it will be possible to handle real-time disturbances. Including elements of ILC in the BMPC design benefits the BMPC operation as it will enable a decrease in the tracking error across both time scales. A pure BMPC will conduct sequential reference tracking, thus not being able to guarantee that long-term disturbances are handled desirably.

As the goal of both the short and long-term objective function is to minimize the residual from the reference trajectory, a model describing the error is needed. In the next section, this will be derived.

3.2 Batch-wise Error Model

Considering the system as a batch process, for which a mathematical description of the residual from the reference trajectory is needed. A generic model describing the broiler growth can be stated as:

$$\mathbf{y} = \mathfrak{N}(\mathbf{u}, \mathbf{d}') \tag{3.1}$$

Where

	$\mathbf{y} \in \mathbb{R}^{(N\times 1)}$	is the broiler weight,	[g]
	$\mathfrak{N}$	is the function describing the real system,	$[\cdot]$
	$\mathbf{u} \in \mathbb{R}^{(N\times 1)}$	is the applied temperature,	[°C]
	$\mathbf{d}' \in \mathbb{R}^{(N\times 1)}$	is the unknown disturbances,	$[\cdot]$
and	$N \in \mathbb{R}^{(1\times 1)}$	is the length of the batch in samples.	$[\cdot]$

The nominal reference is then defined as:

$$\bar{\mathbf{y}} = \mathfrak{N}(\bar{\mathbf{u}}, 0) \tag{3.2}$$

Where

	$\bar{\mathbf{y}} \in \mathbb{R}^{(N\times 1)}$	is the reference weight trajectory,	[g]
and	$\bar{\mathbf{u}} \in \mathbb{R}^{(N\times 1)}$	is the input required to obtain the reference weight trajectory.	[°C]

From *Equation: (3.1)* and *(3.2)* an expression for the error trajectory can be defined:

$$\mathbf{e} \triangleq \bar{\mathbf{y}} - \mathbf{y} = \mathfrak{N}(\bar{\mathbf{u}}, 0) - \mathfrak{N}(\mathbf{u}, \mathbf{d}') \tag{3.3}$$

Where

$\mathbf{e} \in \mathbb{R}^{(N\times 1)}$ is the error sequence across the batch horizon. [g]

The theoretical optimal input trajectory $\bar{\mathbf{u}}$, is unknown. However $\hat{\bar{\mathbf{u}}}$, the best estimate of $\bar{\mathbf{u}}$, can be determined using data from previous batches, the handbook[17] or simply by experience. By defining both $\bar{\mathbf{u}}$ and $\hat{\bar{\mathbf{u}}}$ it is possible to formulate an expression describing the error contribution from both the applied input because of the dynamics and the error due to $\bar{\mathbf{u}} - \hat{\bar{\mathbf{u}}}$, disturbances and model errors:

$$\mathbf{e} = \mathbf{G}(\mathbf{u}, 0) + \mathbf{G}(\bar{\mathbf{u}} - \hat{\bar{\mathbf{u}}}, \mathbf{d}) = \tilde{\mathbf{e}} + \mathbf{e}^{\mathbf{d}} \tag{3.4}$$

Where

$\mathbf{G}$	is a function describing the error in broiler weight,	$[\cdot]$
$\tilde{\mathbf{e}} \in \mathbb{R}^{(N\times 1)}$	is the error contribution from the applied input because of the dynamics,	[g]
and $\mathbf{e}^d \in \mathbb{R}^{(N\times 1)}$	is the error contribution from $\bar{\mathbf{u}} - \hat{\bar{\mathbf{u}}}$, disturbances etc.	[g]

Both $\tilde{\mathbf{e}}$ and $\mathbf{e}^{\mathbf{d}}$ are error sequences for a specific batch, therefore an index is needed to separate error sequence batch-wise. As shown *Figure 3.2* are batches indexed as k resulting in the error sequences being indexed as $\tilde{\mathbf{e}}_{\mathbf{k}}$ and $\mathbf{e}_k^d$.

A mathematical description of both $\tilde{\mathbf{e}}_{\mathbf{k}}$ and $\mathbf{e}_k^d$ should be found, this is conducted in the following. The error contribution $\tilde{\mathbf{e}}_k$ is described first, this contribution will be based on the linear model found in Section *2.7: Model Validation*. The additional error part, $\mathbf{e}_k^d$ is unknown for now, though it will be modeled subsequently to $\tilde{\mathbf{e}}_k$.

3.2.1 Modeling of $\tilde{\mathbf{e}}_k$

The first term from *Equation: (3.4)*, $\tilde{\mathbf{e}}_k$, contains the model of the known system. Assuming a perfect model, $\bar{\mathbf{u}} - \hat{\bar{\mathbf{u}}}_{\mathbf{k}} = 0$, and no disturbances the error contribution is defined as:

$$\tilde{\mathbf{e}}_k = \bar{\mathbf{y}} - \mathbf{y}_k \tag{3.5}$$

The broiler growth $\mathbf{y}_k$ is described by model structure four, see *Equation: (2.8)*. As discussed in Section *3.1: Control Problem* the model is rearranged such that temperature is the only input and thus both humidity and outside temperature are disturbance entries of the model. The final model used for control purpose is stated below for convenience:

$$\begin{aligned} x_k[t+1] &= \underbrace{[1.0024]}_{\mathbf{A}} x_k[t] + \underbrace{[0.019900]}_{\mathbf{B}} \underbrace{u_1[t]}_{u_k} + \underbrace{[-2.9221 \cdot 10^{-4} \;\; 2.6000 \cdot 10^{-3}]}_{\mathbf{E}} \underbrace{\begin{bmatrix} T_{out}[t] \\ H_y[t] \end{bmatrix}}_{\mathbf{d}_k} \\ y_k[t] &= x_k[t] \end{aligned} \tag{3.6}$$

Equation: (3.6) is then substituted into *Equation: (3.5)* for time t.

$$\tilde{e}_k[t+1] = \bar{y}[t+1] - (\mathbf{A}x_k[t] + \mathbf{B}u_k[t] + \mathbf{E}\mathbf{d}_k[t]) \tag{3.7}$$

The same expression is now set up for $[t+1]$.

$$\tilde{e}_k[t+2] = \bar{y}[t+2] - \mathbf{A}(\mathbf{A}x_k[t] + \mathbf{B}u_k[t] + \mathbf{E}\mathbf{d}_k[t]) + \mathbf{B}u_k[t+1] + \mathbf{E}\mathbf{d}_k[t+1] \tag{3.8}$$

This is repeated for the batch length N to obtain an expression describing the error sequence of an entire batch.

$$\underbrace{\begin{bmatrix} \tilde{e}_k[t] \\ \tilde{e}_k[t+1|t] \\ \tilde{e}_k[t+2|t] \\ \vdots \\ \tilde{e}_k[t+N-1|t] \end{bmatrix}}_{\tilde{\mathbf{e}}_k} = \underbrace{\begin{bmatrix} \bar{y}[t] \\ \bar{y}[t+1] \\ \bar{y}[t+2] \\ \vdots \\ \bar{y}[t+N-1] \end{bmatrix}}_{\bar{\mathbf{y}}} - \underbrace{\begin{bmatrix} 1 \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{(N-1)} \end{bmatrix}}_{\mathbf{A}_N} x_k[t]$$
$$+ \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ \mathbf{B} & 0 & \dots & 0 \\ \mathbf{AB} & \mathbf{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{(N-1)}\mathbf{B} & \mathbf{A}^{(N-2)}\mathbf{B} & \dots & 0 \end{bmatrix}}_{\mathbf{\Gamma}} \underbrace{\begin{bmatrix} \hat{\tilde{\mathbf{u}}}_k[t|t] \\ \hat{\tilde{\mathbf{u}}}_k[t+1|t] \\ \hat{\tilde{\mathbf{u}}}_k[t+2|t] \\ \vdots \\ \hat{\tilde{\mathbf{u}}}_k[t+N-1|t] \end{bmatrix}}_{\hat{\tilde{\mathbf{U}}}_k} \tag{3.9}$$
$$+ \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ \mathbf{E} & 0 & \dots & 0 \\ \mathbf{AE} & \mathbf{E} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{(N-1)}\mathbf{E} & \mathbf{A}^{(N-2)}\mathbf{E} & \dots & 0 \end{bmatrix}}_{\mathbf{\Psi}} \underbrace{\begin{bmatrix} \mathbf{d}_k[t|t] \\ \mathbf{d}_k[t+1|t] \\ \mathbf{d}_k[t+2|t] \\ \vdots \\ \mathbf{d}[t+N-1|t] \end{bmatrix}}_{\mathbf{D}_k}$$

The extended system is denoted:

$$\tilde{\mathbf{e}}_k = \bar{\mathbf{y}} - \mathbf{A}_N x_k[t] + \mathbf{\Gamma}\hat{\tilde{\mathbf{U}}}_k + \mathbf{\Psi}\mathbf{D}_k \tag{3.10}$$

Where

$\tilde{\mathbf{e}}_k$	$\in \mathbb{R}^{(N\times 1)}$	is the error sequence calculated for the whole batch,
$\bar{\mathbf{y}}$	$\in \mathbb{R}^{(N\times 1)}$	is the reference trajectory of the broiler weight,
$\hat{\tilde{\mathbf{U}}}_k$	$\in \mathbb{R}^{(N\times 1)}$	is the input sequence,
$\mathbf{D}_k$	$\in \mathbb{R}^{(2N\times 1)}$	is the disturbance sequence,
$\mathbf{A}_N$	$\in \mathbb{R}^{(N\times 1)}$	is the extended state matrix,
$\mathbf{\Gamma}$	$\in \mathbb{R}^{(N\times N)}$	is the extended input matrix,
and $\mathbf{\Psi}$	$\in \mathbb{R}^{(N\times 2N)}$	is the extended disturbance matrix.

A model for $\tilde{\mathbf{e}}_k$ is now described and as the model is based on the lifted system dynamics it is describing the error contribution along the entire batch length.

3.2.2 Modeling of $\mathbf{e}_k^d$

Returning to *Equation: (3.4)*, it was found that a model for $\mathbf{e}_k^d$ is needed. As $\mathbf{e}_k^d$ is unknown and affected by disturbances the error contribution from this part could be modeled as a state-space system with random inputs. The output of the system will have the property that it is specific for the k^{th} batch, thus being batch-wise varying. A state-space model describing this behavior is defined, with random inputs below:

$$\begin{aligned} \mathbf{x}_{k+1}^e &= \mathbf{A}^e\mathbf{x}_k^e + \mathbf{B}^e\mathbf{w}_k \\ \mathbf{e}_k^d &= \mathbf{C}^e\mathbf{x}_k^e + \mathbf{v}_k \end{aligned} \tag{3.11}$$

Where

$\mathbf{w_k}$ and $\mathbf{v}_k$ are batch index wise zero mean and independent identically distributed (i.i.d.) random vectors. $[\cdot]$

It is chosen to model the random inputs as an i.i.d. processes, however this might leave some room for improvement as a higher deviation in weight is to be expected as the broiler grows. Because the error $\bar{\mathbf{e}}_k^d$ does not evolve from batch to batch if no input or disturbances are changed, it is chosen that $\mathbf{A}^e = \mathbf{I}$. Furthermore it is chosen that $\mathbf{B}^e = \mathbf{I}$ and $\mathbf{C}^e = \mathbf{I}$. This choice is made both to simplify the model and as no information regarding the state-space matrices is known. The result simplifies to the state-space model below:

$$\begin{aligned} \bar{\mathbf{e}}_{k+1}^d &= \bar{\mathbf{e}}_k^d + \mathbf{w}_k \\ \mathbf{e}_k^d &= \bar{\mathbf{e}}_k^d + \mathbf{v}_k \end{aligned} \tag{3.12}$$

The state $\bar{\mathbf{e}}_k^d$ can be interpreted as the part of $\mathbf{e}_k^d$ that repeats itself from batch to batch, whereas the contribution from the noise $\mathbf{v}_k$ is specific to the k^{th} batch and is thus only expected to appear during the specific batch. The input noise $\mathbf{w}_k$ is considered as the part of the noise that repeats itself from batch to batch and is thus contributing to the repeating part of the error for the batch $\mathbf{e}_{k+1}^d$.

As for $\mathbf{e}_k^d$ the part of $\mathbf{e}_k$ that repeats itself is defined as $\bar{\mathbf{e}}_k$. Assuming that the input $\hat{\bar{\mathbf{u}}}_k$ and $\hat{\bar{\mathbf{u}}}_{k+1}$ is the same, then $\bar{\mathbf{e}}_k$ and $\bar{\mathbf{e}}_{k+1}$ is defined as:

$$\bar{\mathbf{e}}_k = \tilde{\mathbf{e}}_k + \bar{\mathbf{e}}_k^d \tag{3.13}$$

$$\bar{\mathbf{e}}_{k+1} = \tilde{\mathbf{e}}_{k+1} + \bar{\mathbf{e}}_{k+1}^{\mathbf{d}} \tag{3.14}$$

and:

Subtracting *Equation: (3.13)* from *Equation: (3.14)* and using the definitions of *Equation: (3.10)* yields an expression describing how a change in input affects the error.

$$\begin{aligned} \bar{\mathbf{e}}_{k+1} - \bar{\mathbf{e}}_k = \bar{\mathbf{y}} - \mathbf{A}_N x_{k+1}[t] + \mathbf{\Gamma}\hat{\bar{\mathbf{U}}}_{k+1} + \mathbf{\Psi}\mathbf{D}_{k+1} - (\bar{\mathbf{y}} - \mathbf{A}_N x_k[t] + \mathbf{\Gamma}\hat{\bar{\mathbf{U}}}_k + \mathbf{\Psi}\mathbf{D}_k) \\ + \bar{\mathbf{e}}_{k+1}^d - \bar{\mathbf{e}}_k^d \end{aligned}$$

This can be rewritten as:

$$\begin{aligned} \bar{\mathbf{e}}_{k+1} &= \bar{\mathbf{e}}_k - \mathbf{\Gamma}(\hat{\bar{\mathbf{U}}}_{k+1} - \hat{\bar{\mathbf{U}}}_k) - \mathbf{\Psi}(\mathbf{D}_{k+1} - \mathbf{D}_k) + \mathbf{w}_k \\ &= \bar{\mathbf{e}}_k - \mathbf{\Gamma}\mathbf{\Delta}\mathbf{U}_{k+1} - \mathbf{\Psi}\mathbf{\Delta}\mathbf{D}_{k+1} + \mathbf{w}_k \end{aligned} \tag{3.15}$$

The output equation can be stated in the same way as for *Equation: (3.12)*

$$\mathbf{e}_k = \bar{\mathbf{e}}_k + \mathbf{v}_k \tag{3.16}$$

If no previous batches exist, a vector of zeros, $\bar{\mathbf{e}}_0$ is used as the initial state and the best available estimate of $\mathbf{U}_0$ used as temperature curve.

A state-space system describing the change in error from batch to batch is now obtained. This model is however not suitable for control purposes within the individual batches. Therefore adapting the model to include time dependency is necessary, this will be conducted in the next section.

3.3 Time-wise Error Model

For control purposes a time depending model is necessary. In this section, a time-wise error model, that can be used for real-time prediction, is formulated using the batch-wise model.

The batch-wise model found in Section *3.2: Batch-wise Error Model* describes the long-term system evolution, that is along iterations of the batch count, thus complete sequences of each property, input, output, error etc. are described. This long-term time scale is depicted in *Figure 3.2* as the axis labeled "Iterations". In order to obtain a model suitable for ordinary time-wise control it is necessary to reformulate the model such that it represents both the long-term and short-term time scales, thus reflecting both axes labeled "Iterations" and "Time" according to *Figure 3.2*.

First the matrices describing the dynamics along an entire batch, $\mathbf{\Gamma}$ and $\mathbf{\Psi}$, are partitioned according to time:

$$\mathbf{\Gamma} \triangleq \Big[\mathbf{\Gamma}[0]\,\mathbf{\Gamma}[1]\cdots\mathbf{\Gamma}[N-1]\Big] \in \mathbb{R}^{(N\times N)} \tag{3.17}$$

$$\mathbf{\Psi} \triangleq \Big[\mathbf{\Psi}[0]\,\mathbf{\Psi}[1]\cdots\mathbf{\Psi}[N-1]\Big] \in \mathbb{R}^{(N\times N)} \tag{3.18}$$

Where

$\mathbf{\Gamma}[t]$, $\mathbf{\Psi}[t]$	are the columns of $\mathbf{\Gamma}$ and $\mathbf{\Psi}$ describing the impulse response for the remaining batch at time t.	$[\cdot]$

A time dependent error sequence, starting at time t, is defined as:

$$\mathbf{e}_k[t] \triangleq \mathbf{e}_k \qquad \text{where } \mathbf{\Delta U}_k[t] = \cdots = \mathbf{\Delta U}_k[N-1] = 0 \tag{3.19}$$

Using the system equation from the error model, *Equation: (3.15)* and *(3.16)*, and the new definitions the error sequence can be written as:

$$\begin{aligned}\mathbf{e}_k[t] = \bar{\mathbf{e}}_{k-1}[t] - \mathbf{\Gamma}(0)\mathbf{\Delta U}_k[0] - \mathbf{\Psi}[0]\mathbf{\Delta D}_k[0] - \cdots - \mathbf{\Gamma}[t-1]\mathbf{\Delta U}_k[t-1] \\ -\mathbf{\Psi}[t-1]\mathbf{\Delta D}_k[t-1] + \mathbf{w}_{k-1} + \mathbf{v}_k\end{aligned} \tag{3.20}$$

Similar an expression for $\mathbf{e}_k[t+1]$ can be formulated. Subtracting these, exactly like done in *Equation: (3.15)*, an expression describing the dynamics from $\mathbf{e}_k[t]$ to $\mathbf{e}_k[t+1]$ is obtained.

$$\mathbf{e}_k[t+1] = \mathbf{e}_k[t] - \mathbf{\Gamma}[t]\mathbf{\Delta U}_k[t] - \mathbf{\Psi}[t]\mathbf{\Delta D}_k[t] \qquad ,t = 1,\cdots,N \tag{3.21}$$

Furthermore the initial error sequence at $t = 0$ is defined. As information from last batch is available after the first batch this is utilized and $\mathbf{e}_k[0]$ is therefore defined as:

$$\mathbf{e}_k[0] = \bar{\mathbf{e}}_{k-1}[N] + \mathbf{w}_{k-1} + \mathbf{v}_k \tag{3.22}$$

As the initial condition $\mathbf{e}_k[0]$ depends on $\bar{\mathbf{e}}_{k-1}[N]$, an expression describing the dynamics from $\bar{\mathbf{e}}_k[t]$ to $\bar{\mathbf{e}}_k[t+1]$ is also needed. This is obtained in the same way as for $\mathbf{e}_k$.

$$\bar{\mathbf{e}}_k[t+1] = \bar{\mathbf{e}}_k[t] - \mathbf{\Gamma}[t]\mathbf{\Delta U}_k[t] - \mathbf{\Psi}[t]\mathbf{\Delta D}_k[t] \qquad ,t = 1,\cdots,N \tag{3.23}$$

$$\bar{\mathbf{e}}_k[0] = \bar{\mathbf{e}}_{k-1}[N] + \mathbf{w}_{k-1} + \mathbf{v}_k \tag{3.24}$$

Combining *Equation: (3.21)* and *Equation: (3.23)* a state-space system model suitable for control design is obtained.

$$\begin{bmatrix}\bar{\mathbf{e}}_k[t+1]\\ \mathbf{e}_k[t+1]\end{bmatrix} = \begin{bmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{e}}_k[t]\\ \mathbf{e}_k[t]\end{bmatrix} - \begin{bmatrix}\mathbf{\Gamma}[t]\\ \mathbf{\Gamma}[t]\end{bmatrix}\mathbf{\Delta U}_k[t] - \begin{bmatrix}\mathbf{\Psi}[t]\\ \mathbf{\Psi}[t]\end{bmatrix}\mathbf{\Delta D}_k[t] \tag{3.25}$$

$$\mathbf{e}'_k[t] = \begin{bmatrix}0 & \mathbf{H}[t]\end{bmatrix}\begin{bmatrix}\bar{\mathbf{e}}_k[t]\\ \mathbf{e}_k[t]\end{bmatrix} \qquad ,t = 1,\cdots,N \tag{3.26}$$

For real time control the output $\mathbf{e}'_k[t]$ should be minimized. The state $\bar{\mathbf{e}}_k[t]$ seems to be redundant, however, $\bar{\mathbf{e}}_k[t]$ which repeats itself batch-wise is set as the initial condition of the states:

$$\begin{bmatrix}\bar{\mathbf{e}}_k[0]\\ \mathbf{e}_k[0]\end{bmatrix} = \begin{bmatrix}\mathbf{I} & 0\\ \mathbf{I} & 0\end{bmatrix}\begin{bmatrix}\bar{\mathbf{e}}_{k-1}[N]\\ \mathbf{e}_{k-1}[N]\end{bmatrix} - \begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}\mathbf{w}_{k-1} - \begin{bmatrix}0\\ \mathbf{I}\end{bmatrix}\mathbf{v}_k \tag{3.27}$$

$\mathbf{H}[t]$ defines which errors to use as outputs. The goal is to minimize the error at all times in the batch, but it is of course not possible to change the error at time $t-1$ when that time has passed, due to causality. This results in time a varying structure and therefore $\mathbf{H}[t]$ is defined as:

$$\mathbf{H}[t] = [\underbrace{0}_{\mathbb{R}^{(1\times t-1)}} \quad \underbrace{\mathbf{I}}_{\mathbb{R}^{(1\times N-t)}}] \in \mathbb{R}^{(1\times N)} \tag{3.28}$$

For real-time control $\bar{\mathbf{e}}_k[t]$ is stored for the next batch. In reality, only the element of $e_k[t]$ representing the current time step can be measured. Therefore a state estimator is needed to estimate the complete error sequence. In the next section design of a Kalman filter is explained.

3.4 Kalman Filter

In Section *3.3: Time-wise Error Model* a state-space system describing the dynamics of two error sequences, $\bar{\mathbf{e}}_k[t]$ and $\mathbf{e}_k[t]$, is defined. It is, however, only possible to measure the error at the current time step and therefore a state observer is required. It is chosen to use an optimal state observer, called a Kalman filter. A Kalman filter estimates a state using a model of the system, an initial state, measurement variance and a measurement. According to a weighting factor, the Kalman gain, the Kalman filter determines how much to trust the model versus the measurement. As more measurements become available the estimate will converge towards the real state value, assuming all noises are Gaussian. The Kalman filter is a recursive filter and it consists of two steps, a prediction step which calculates a prior estimate and an update step which updates the estimate using the measurement, the posterior estimate.

3.4.1 Prediction Step

The dynamic of the error sequences, is used to predict the error. This is called the prior estimate and is defined as:

$$\hat{\mathbf{x}}_k[t|t-1] = \mathbf{A}\hat{\mathbf{x}}_k[t-1|t-1] + \mathbf{B}\mathbf{u}_k[t-1] \tag{3.29}$$

Where
$\hat{\mathbf{x}}_k[t|t-1]$ is the prior state estimate, [g]
and $\hat{\mathbf{x}}_k[t-1|t-1]$ is the posterior estimates from last sample. [g]

Using the system model, defined as *Equation: (3.26)*, the prior state estimate can be written as:

$$\begin{bmatrix}\hat{\bar{\mathbf{e}}}_k[t|t-1]\\ \hat{\mathbf{e}}_k[t|t-1]\end{bmatrix} = \begin{bmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\hat{\bar{\mathbf{e}}}_k[t-1|t-1]\\ \hat{\mathbf{e}}_k[t-1|t-1]\end{bmatrix} - \begin{bmatrix}\mathbf{\Gamma}[t-1]\\ \mathbf{\Gamma}[t-1]\end{bmatrix}\mathbf{\Delta U_k}[t-1] - \begin{bmatrix}\mathbf{\Psi}[t-1]\\ \mathbf{\Psi}[t-1]\end{bmatrix}\mathbf{\Delta D}_k[t-1] \quad (3.30)$$

Where:

$$\begin{bmatrix}\hat{\bar{\mathbf{e}}}_k[0|0]\\ \hat{\mathbf{e}}_k[0|0]\end{bmatrix} = \begin{bmatrix}\hat{\bar{\mathbf{e}}}_{k-1}[N|N]\\ \hat{\bar{\mathbf{e}}}_{k-1}[N|N]\end{bmatrix}$$

As the system matrix is the identity matrix the error will remain the same as long as the input and disturbance are not changed. Furthermore, the initial error sequence is defined as the final estimated error sequence from the last batch. This is done to utilize the information from the last batch in order to improve the long-term performance of the controller, thus allowing the controller to iteratively learn.

The variance of the states are defined as a covariance matrix. The prior covariance matrix is updated each time step and defined as:

$$\mathbf{P}_k[t|t-1] = \mathbf{A}^T\mathbf{P}_k[t-1|t-1]\mathbf{A} + \mathbf{R}_s \quad (3.31)$$

Where

	$\mathbf{P}_k[t\|t-1]$	is the prior estimate of the covariance matrix,	$[g^2]$
	$\mathbf{P}_k[t-1\|t-1]$	is the posterior estimates from last sample,	$[g^2]$
and	$\mathbf{R}_s$	is the state noise.	$[g^2]$

The definition and *Equation: (3.26)* can be used to describe the covariance matrix:

$$\begin{bmatrix}\mathbf{P}_k^{11}[t|t-1] & \mathbf{P}_k^{12}[t|t-1]\\ \mathbf{P}_k^{21}[t|t-1] & \mathbf{P}_k^{22}[t|t-1]\end{bmatrix} = \begin{bmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{P}_k^{11}[t-1|t-1] & \mathbf{P}_k^{12}[t-1|t-1]\\ \mathbf{P}_k^{21}[t-1|t-1] & \mathbf{P}_k^{22}[t-1|t-1]\end{bmatrix}\begin{bmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{bmatrix} + \mathbf{R_s} \quad (3.32)$$

Again the system matrix is the identity matrix, so the covariance matrix is only changed by the state noise, $\mathbf{R_s}$. As for the prior state estimate, it is desirable to utilize information from last batch. Therefore the initial condition of the covariance matrix for the current batch is defined as the variance of $\bar{\mathbf{e}}_{k-1}$ at the final time N:

$$\begin{bmatrix}\mathbf{P}_k^{11}[0|0] & \mathbf{P}_k^{12}[0|0]\\ \mathbf{P}_k^{21}[0|0] & \mathbf{P}_k^{22}[0|0]\end{bmatrix} = \begin{bmatrix}\mathbf{P}_{k-1}^{11}[N|N] + \mathbf{R}_w & \mathbf{P}_{k-1}^{11}[N|N] + \mathbf{R}_w\\ \mathbf{P}_{k-1}^{11}[N|N] + \mathbf{R}_w & \mathbf{P}_{k-1}^{11}[N|N] + \mathbf{R}_w + \mathbf{R}_v\end{bmatrix} \quad (3.33)$$

Where

$\mathbf{R}_w$ and $\mathbf{R}_v$	are random variables describing the batch-wise variance of the covariance matrix.	$[g^2]$

$\mathbf{R}_v$ is only added to $\mathbf{P}^{22}$ as this noise is specific for the current batch. If there are no previous batches, $k = 1$, the covariance matrix is defined as:

$$\begin{bmatrix}\mathbf{P}_1^{11}[0|0] & \mathbf{P}_1^{12}[0|0]\\ \mathbf{P}_1^{21}[0|0] & \mathbf{P}_1^{22}[0|0]\end{bmatrix} = \begin{bmatrix}\gamma\mathbf{I} & \gamma\mathbf{I}\\ \gamma\mathbf{I} & \gamma\mathbf{I}\end{bmatrix} \quad (3.34)$$

Where

γ	is a positive scalar and fairly large.	$[g^2]$

By choosing γ large, the initial variance is big and therefore the model is not trusted as much. A prior estimate of both the state and the covariance matrix is now obtained. In the next section, the information from the measurement is included.

3.4.2 Update Step

The update step uses the prior estimates and the measurement of the current error to update both the estimated states and the covariance matrix. The updated estimates are called the posterior estimates. How much the prior estimate and measurement can be trusted is determined by a weighting factor, called the Kalman gain. The Kalman gain is defined as:

$$\mathbf{K}_k[t] = \mathbf{P}_k[t|t-1]\mathbf{C}^T[t] \cdot \Big[\mathbf{C}[t]\mathbf{P}_k[t|t-1]\mathbf{C}^T[t] + R_m[t]\Big]^{-1} \tag{3.35}$$

Where

	$\mathbf{K}_k[t]$	$\in \mathbb{R}^{(2N\times 1)}$	is the Kalman gain,	$[\cdot]$
	$\mathbf{C}[t]$	$\in \mathbb{R}^{(2N\times 1)}$	is the output equation for the system,	$[\cdot]$
and	$R_m[t]$	$\in \mathbb{R}^{(1\times 1)}$	is the variance of the measurement at time t.	$[\mathrm{g}^2]$

By substituting the covariance and output matrix with their definitions from *Equation: (3.26)* and *Equation: (3.32)* the Kalman gain is defined by:

$$\mathbf{K}_k[t] = \begin{bmatrix}\mathbf{P}_k^{12}[t|t-1]\\ \mathbf{P}_k^{22}[t|t-1]\end{bmatrix} \mathbf{H}^T[t] \cdot \Big[\mathbf{H}[t]\mathbf{P}_k^{22}[t|t-1]\mathbf{H}^T[t] + R_m[t]\Big]^{-1} \tag{3.36}$$

The Kalman gain is now used to update the state estimate according to:

$$\hat{\mathbf{x}}_k[t|t] = \hat{\mathbf{x}}_k[t|t-1] + \mathbf{K}_k[t] \cdot \big[y_k[t] - \mathbf{C}[t]\hat{\mathbf{x}}_k[t|t-1]\big] \tag{3.37}$$

Which result in:

$$\begin{bmatrix}\hat{\tilde{\mathbf{e}}}_k[t|t]\\ \hat{\mathbf{e}}_k[t|t]\end{bmatrix} = \begin{bmatrix}\hat{\tilde{\mathbf{e}}}_k[t|t-1]\\ \hat{\mathbf{e}}_k[t|t-1]\end{bmatrix} + \mathbf{K}_k[t] \cdot \left[e_k[t] - [0 \;\; \mathbf{H}[t]] \begin{bmatrix}\hat{\tilde{\mathbf{e}}}_k[t|t-1]\\ \hat{\mathbf{e}}_k[t|t-1]\end{bmatrix} \right] \qquad , t = 1, \cdots, N \tag{3.38}$$

The posterior state estimate is the output of the Kalman filter and this is used for real time control. However the covariance matrix is needed for the next iteration and therefore it is updated according to:

$$\mathbf{P}_k[t|t] = \big(\mathbf{I} - \mathbf{K}_k[t]\mathbf{C}[t]\big) \cdot \mathbf{P}_k[t|t-1] \tag{3.39}$$

Again this can be setup for the dynamics of *Equation: (3.26)*:

$$\begin{aligned}\begin{bmatrix}\mathbf{P}_k^{11}[t|t] & \mathbf{P}_k^{12}[t|t]\\ \mathbf{P}_k^{21}[t|t] & \mathbf{P}_k^{22}[t|t]\end{bmatrix} &= \Big(\mathbf{I} - \mathbf{K}_k[t] \cdot [0 \;\; \mathbf{H}[t]\Big) \begin{bmatrix}\mathbf{P}_k^{11}[t|t-1] & \mathbf{P}_k^{12}[t|t-1]\\ \mathbf{P}_k^{21}[t|t-1] & \mathbf{P}_k^{22}[t|t-1]\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{P}_k^{11}[t|t-1] & \mathbf{P}_k^{12}[t|t-1]\\ \mathbf{P}_k^{21}[t|t-1] & \mathbf{P}_k^{22}[t|t-1]\end{bmatrix} - \mathbf{K}_k[t]\mathbf{H}[t] \begin{bmatrix}\mathbf{P}_k^{21}[t|t-1] & \mathbf{P}_k^{22}[t|t-1]\end{bmatrix}\end{aligned} \tag{3.40}$$

Implementing *Equation: 3.30, 3.32* to *3.34* and *Equation: 3.36, 3.38* and *3.40* recursively will result in the optimal estimate of the states, assuming all disturbances are Gaussian distributed.

An estimate of both error sequences is now obtained for the entire batch length. In the next section, this is used to set up an optimizations problem for which an input sequence that minimizes the error is found.

3.5 Model Predictive Control

The previous Section *3.4: Kalman Filter* defines an optimal observer that at time t predicts both the batch-wise repeating error sequence and the non-repeating error sequence up to time N. This can, however, be adjusted such that the error sequence at time t is calculated until time $t+m$, where m is a prediction horizon. Furthermore, a control horizon is defined, this horizon determines how far into the future the optimal control input is calculated. The length of the control horizon is defined to be equal to the length of the prediction horizon.

Extracting a part of the Kalman filters output such that the length matches the prediction horizon m, permits the use of MPC, which is an advanced control method that uses an internal model to predict the systems future behavior and thus the future inputs. The structure of the MPC algorithm can be seen in *Figure 3.3*:

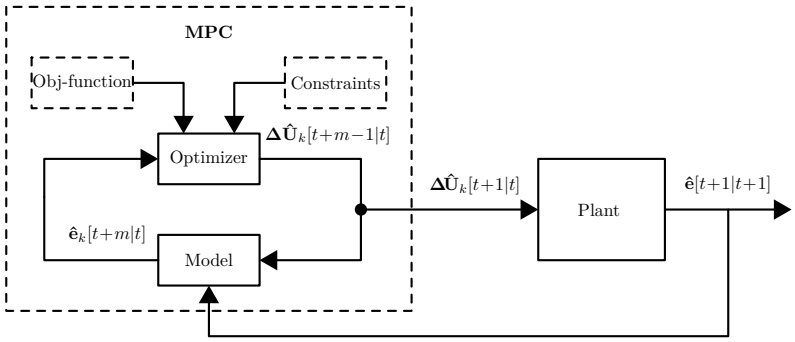

Figure 3.3: A block diagram of MPC algorithm [18].

Due to the time-varying nature of the system, the horizon m must be shrunk as t approaches N.

Allowing the MPC to predict the error sequences at time $t+m$ requires that the system is extended. In Section *3.2: Batch-wise Error Model* the system was extended for the entire batch length and this will be reused for the MPC resulting in:

$$\hat{\mathbf{e}}_k[t+m|t] = \hat{\mathbf{e}}_k[t|t] - \mathbf{\Gamma}[t+m-1|t]\Delta\hat{\mathbf{U}}_k[t+m-1|t] - \mathbf{\Psi}[t+m-1|t]\Delta\hat{\mathbf{D}}_k[t+m-1|t] \quad (3.41)$$

where:

$$\mathbf{\Gamma}[t+m-1|t] = [\mathbf{\Gamma}[t], \cdots, \mathbf{\Gamma}[t+m-1]] \in \mathbb{R}^{(N\times m)}$$

$$\mathbf{\Psi}[t+m-1|t] = [\mathbf{\Psi}[t], \cdots, \mathbf{\Psi}[t+m-1]] \in \mathbb{R}^{(N\times 2m)}$$

$$\Delta\hat{\mathbf{U}}_k[t+m-1|t] = \begin{bmatrix} \Delta U_k[t] \\ \Delta\hat{U}_k[t+1] \\ \vdots \\ \Delta\hat{U}_k[t+m-1] \end{bmatrix} \in \mathbb{R}^{(m\times 1)}$$

$$\Delta\hat{\mathbf{D}}_k[t+m-1|t] = \begin{bmatrix} \Delta\mathbf{D}_k[t] \\ \Delta\hat{\mathbf{D}}_k[t+1] \\ \vdots \\ \Delta\hat{\mathbf{D}}_k[t+m-1] \end{bmatrix} \in \mathbb{R}^{(2m\times 1)}$$

To keep the notation easy to read the extended signals are redefined while setting up the MPC:

$$\begin{aligned}\boldsymbol{\Gamma}[t+m-1|t] &= \boldsymbol{\Gamma}^m[t]\\ \boldsymbol{\Psi}[t+m-1|t] &= \boldsymbol{\Psi}^m[t]\\ \boldsymbol{\Delta}\hat{\mathbf{U}}_k[t+m-1|t] &= \boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]\\ \boldsymbol{\Delta}\hat{\mathbf{D}}_k[t+m-1|t] &= \boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]\end{aligned}$$

The dynamics of the error sequence, described by *Equation: (3.41)*, from time t to $t+m-1$ is very similar to the approach in Section *3.2: Batch-wise Error Model*. The columns of both $\boldsymbol{\Gamma}^m[t]$ and $\boldsymbol{\Psi}^m[t]$ are defined as in *Equation: (3.9)*.

To minimized both positive and negative values of $\hat{\mathbf{e}}_k[t+m|t]$ a quadratic expression is set up. Furthermore, a term that punishes $\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]$ is added, to punish changes in the control signal separately.

$$\underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\ \Upsilon(\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]) = \underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\frac{1}{2}\{\hat{\mathbf{e}}_k^T[t+m|t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t+m|t] + \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\mathbf{Q}_2\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]\} \quad (3.42)$$

Where

	$\mathbf{Q}_1$	is the weighting matrix for the state,	$[\cdot]$
and	$\mathbf{Q}_2$	is the weighting matrix for change in input.	$[\cdot]$

The first term, punishing $\hat{\mathbf{e}}_k[t+m|t]$, does not depend on $\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]$ and can therefore not be minimized straight away. To enable minimization *Equation: (3.42)* must be rewritten to a form that depends on $\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]$. This is done by applying the definition from *Equation: (3.41)*.

$$\begin{aligned}\underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\ \Upsilon(\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t], \boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]) = \underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\frac{1}{2}\Big\{&\Big(\hat{\mathbf{e}}_k[t|t] - \boldsymbol{\Gamma}^m[t]\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] - \boldsymbol{\Psi}^m[t]\boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]\Big)^T\\ &\mathbf{Q}_1\Big(\hat{\mathbf{e}}_k[t|t] - \boldsymbol{\Gamma}^m[t]\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] - \boldsymbol{\Psi}^m[t]\boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]\Big) + \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\mathbf{Q}_2\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]\Big\}\end{aligned} \quad (3.43)$$

Multiplying the terms of *Equation: (3.43)* together result in:

$$\begin{aligned}\underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\ \Upsilon(\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t], \boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]) = \underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\frac{1}{2}\Big\{&\hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t|t] - \hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\boldsymbol{\Gamma}^m[t]\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]\\ &- \hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\boldsymbol{\Psi}^m[t]\boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t] - \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\boldsymbol{\Gamma}^{mT}[t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t|t]\\ &+ \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\boldsymbol{\Gamma}^{mT}[t]\mathbf{Q}_1\boldsymbol{\Gamma}^m[t]\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] + \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\boldsymbol{\Gamma}^{mT}[t]\mathbf{Q}_1\boldsymbol{\Psi}^m[t]\boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]\\ &- \boldsymbol{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\boldsymbol{\Psi}^{mT}[t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t|t] + \boldsymbol{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\boldsymbol{\Psi}^{mT}[t]\mathbf{Q}_1\boldsymbol{\Gamma}^m[t]\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]\\ &+ \boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\mathbf{Q}_2\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] + \boldsymbol{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\boldsymbol{\Psi}^{mT}[t]\mathbf{Q}_1\boldsymbol{\Psi}^m[t]\boldsymbol{\Delta}\hat{\mathbf{D}}_k^m[t]\Big\}\end{aligned} \quad (3.44)$$

Rearranging from *Equation: (3.42)* to *Equation: (3.44)* result in an objective function that consists of a quadratic, linear and constant part. A function of this structure can be put on the from:

$$\underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\ \Upsilon(\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]) = \underset{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t]}{min}\frac{1}{2}\Big\{\boldsymbol{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\mathbf{R}\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] + \mathbf{b}\boldsymbol{\Delta}\hat{\mathbf{U}}_k^m[t] + c\Big\} \quad (3.45)$$

Where

$\boldsymbol{R} \in \mathbb{R}^{(m\times m)}$

$\boldsymbol{b} \in \mathbb{R}^{(1\times m)}$

$c \in \mathbb{R}^{(1\times 1)}$

Separating *Equation: (3.44)* into a quadratic, a linear and a constant part result in $\mathbf{R}, \mathbf{b}$ and c parameters defined as:

$$\mathbf{R} = \mathbf{\Gamma}^{mT}[t]\mathbf{Q}_1\mathbf{\Gamma}^{m}[t] + \mathbf{Q}_2 \tag{3.46}$$

$$\begin{aligned}\mathbf{b} = &- \hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\mathbf{\Gamma}^{m}[t] - \hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1^T\mathbf{\Gamma}^{m}[t] + \mathbf{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\mathbf{\Psi}^{mT}[t]\mathbf{Q}_1^T\mathbf{\Gamma}^{m}[t] \\ &+ \mathbf{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\mathbf{\Psi}^{mT}[t]\mathbf{Q}_1\mathbf{\Gamma}^{m}[t]\end{aligned} \tag{3.47}$$

$$\begin{aligned}c = &-\hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\mathbf{\Psi}^{m}[t]\mathbf{\Delta}\hat{\mathbf{D}}_k^{m}[t] - \mathbf{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\mathbf{\Psi}^{mT}[t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t|t] \\ &+\mathbf{\Delta}\hat{\mathbf{D}}_k^{mT}[t]\mathbf{\Psi}^{mT}[t]\mathbf{Q}_1\mathbf{\Psi}^{m}[t]\mathbf{\Delta}\hat{\mathbf{D}}_k^{m}[t] + \hat{\mathbf{e}}_k^T[t|t]\mathbf{Q}_1\hat{\mathbf{e}}_k[t|t]\end{aligned} \tag{3.48}$$

The above minimization problem is however subject to constraints. To ensure that the change in temperature from batch to batch is not too big, $\mathbf{\Delta}\hat{\mathbf{U}}_k^{m}[t]$ should be bounded. An upper and lower bound for $\mathbf{\Delta}\hat{\mathbf{U}}_k^{m}[t]$ can be written as the following affine inequality:

$$\begin{bmatrix}\mathbf{I} \\ -\mathbf{I}\end{bmatrix}\mathbf{\Delta}\hat{\mathbf{U}}_k^{m}[t] \geq \begin{bmatrix}\mathbf{\Delta U}_{ub} \\ \mathbf{\Delta U}_{lb}\end{bmatrix} \tag{3.49}$$

Where

	$\mathbf{\Delta U}_{ub}$	is the allowed positive change in the temperature trajectory from batch to batch,	[°C]
and	$\mathbf{\Delta U}_{lb}$	is the allowed negative change in the temperature trajectory from batch to batch.	[°C]

Equation: (3.45) and *(3.49)* is a quadratic problem which is easily solved if it is convex. To guarantee this the next section will examine the convexity of the problem.

3.5.1 Convexity

A mathematical description of the deviations in broiler weight has now been set up on a quadratic form. Solving *Equation: (3.45)* will, however, become significantly easier if the problem is convex. Furthermore, the solution is guaranteed to be the optimal solution if *Equation: (3.45)* is a convex problem. A convex surface is illustrated in *Figure 3.4*, as it is shown a global minimum exists. In *Figure 3.5* a non-convex surface is illustrated, which does not have a unique global minimum.

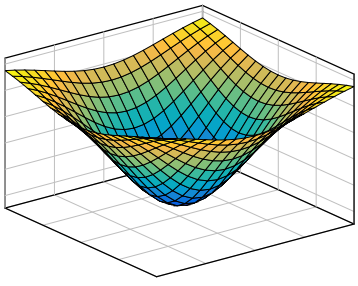

Figure 3.4: Quadratic convex surface.

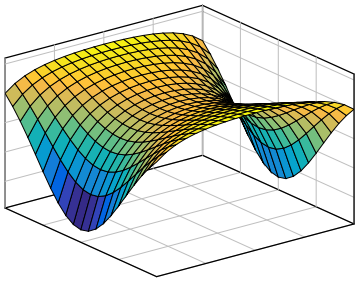

Figure 3.5: Quadratic non-convex surface.

If the minimization problem is convex, the global minimum can be found by taking the derivative of the function. The minimization problem is only convex if the objective function is convex and a feasible solution exists within the set formed by the constraints.

To ensure convexity the objective function must have a positive semi-definite Hessian matrix. Therefore the Hessian of *Equation: (3.45)* is found:

$$\Upsilon(\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t]) = \frac{1}{2}(\mathbf{\Delta}\hat{\mathbf{U}}_k^{mT}[t]\mathbf{R}\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t] + \mathbf{b}\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t] + c) \tag{3.50}$$

$$\frac{\partial\Upsilon(\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t])}{\partial\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t]} = \frac{1}{2}(\mathbf{R}\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t] + \mathbf{b}) \tag{3.51}$$

$$\frac{\partial^2\Upsilon(\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t])}{\partial(\mathbf{\Delta}\hat{\mathbf{U}}_k^m[t])^2} = \frac{1}{2}\mathbf{R} \tag{3.52}$$

Equation: (3.52) shows that convexity is ensured if:

$$\mathbf{R} \geq 0 \tag{3.53}$$

$\mathbf{R}$ is defined in *Equation: (3.46)* as $\mathbf{R} = \mathbf{\Gamma}^{mT}[t]\mathbf{Q}_1\mathbf{\Gamma}^m[t] + \mathbf{Q}_2$. As $\mathbf{\Gamma}^m[t]$ is a quadratic term $\mathbf{R}$ is positive semi-definite if $\mathbf{Q}_1 \geq 0$ and $\mathbf{Q}_2 \geq 0$. $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are control parameters of the MPC, meaning that they can simply be chosen to fit this criteria, hence the problem is convex.

The constraint *Equation: (3.49)* is an affine inequality which defines a half-space of the quadratic problem. Any half-space of the quadratic problem will be convex as the quadratic problem is convex. Furthermore, the intersection of convex sets is convex, hence the constraints are also convex.

Control System Implementation 4

This chapter explains how the controller designed in Chapter *3: Controller* is implemented in a simulation environment.

4.1 Controller Implementation

The control algorithm developed in Section *3: Controller* consists of multiple components that work together. In addition, the controller should be implemented along with a plant as close to the real-world behavior as possible. It is thus chosen to utilize a plant model that is based on a recurrent neural network (RNN) model made by Simon V. Johansen [7].

The dynamics of the broiler are slow, and as the control strategy is furthermore partitioned in both a long and short-term objective the long-term objective allows an implementation method where multiple simulations of batches in a row are conducted. Thus simulating the long-term objective beforehand and thereby preparing the controller for real-world implementation.

In both a simulation environment and a genuine stable, the controller only encounters disturbances present at the current time step. However the quadric problem from Section *3.5: Model Predictive Control* depends on $\mathbf{\Delta}\hat{\mathbf{D}}_k[t+m-1|t]$, the predicted disturbance sequence for the whole batch length. Therefore a disturbance estimator, predicting the outside temperature and humidity along the entire batch length is needed.

Recalling *Figure 3.1* and including the components just mentioned, the implemented structure can be visualized as *Figure 4.1*.

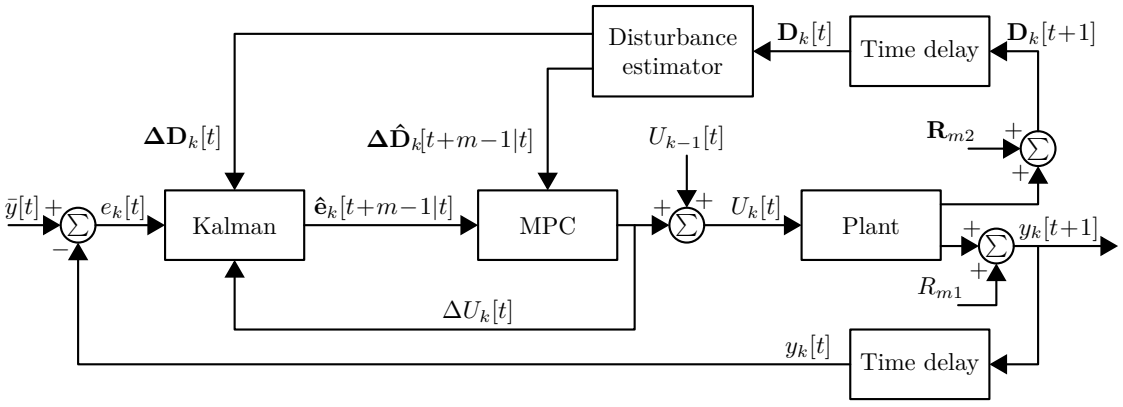

Figure 4.1: Block diagram of the algorithm implemented.

The implemented control structure consists of a Kalman filter, disturbance estimator and, MPC. As this structure is implemented in simulation additional noises are added in order to represent real-world behavior. These noises are measurement noise, R_{m1},

and disturbance noise, $\mathbf{R}_{m2}$, where only the measurement noise is nonzero for now as a functional, rather basic, implementation is desirable at first to ensure correct working principle.

In addition to the noises is the input from the previous batch added to the small signal output from the MPC. This is in line with the definition of the error model used in the MPC, originating from *Equation: (3.15)*, where it is stated that the current error depends on the error in previous batch and the change in input and disturbance. Due to this model structure, it is necessary to store the complete input and disturbance sequence from the previous batch. The previous disturbance sequence is applied internally in the disturbance estimator.

The RNN model utilized in the simulation environment as plant model is originally specified with a sample time of 24 hours. The model chosen as a base for the control structure is formulated with a sample time of one hour. In order to secure compatibility between the two models is a change in sample time for at least one necessary. The RNN model can be trained to fit a specific sample time, and this affects the computational demand. It is trough test of different combinations found that a sample time of three hours is optimal to ensure the validity of both models and keep the computational demand reasonable. The final model found in Section *2.7: Model Validation* is resampled to an equivalent three-hour model using zero order hold on the inputs. The resampled model is applied in the internal model of the MPC. The RNN plant model is adjusted and trained such that it fits the three hour sample time. Due to these changes are all time steps from this point on given as three-hour samples.

The block diagram presented in *Figure 4.1* provides an overview of the implemented control structure. However to fully visualize the ILC components included in the control structure the following extended block diagram is introduced.

In addition to the previous block diagram, the extended diagram includes batch delays and the initial conditions for the Kalman filter.

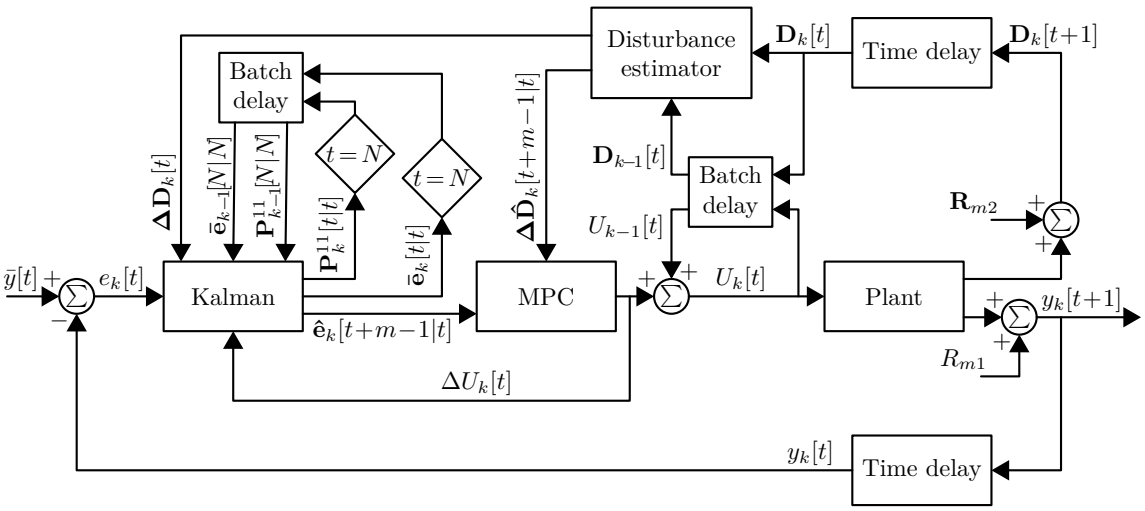

Figure 4.2: Extended block diagram of the implementation structure.

The extended block diagram presented in *Figure 4.2* reflects the same working principle as the previous diagram, *Figure 4.1*. The batch delay affects both the control signal and the disturbance, these are stored and the delayed sequences are utilized to restore the full

signal plant input and create the small signal disturbance for the MPC.

The internal components of the Kalman filter are also stored, thus effectively exposed to a batch delay as indicated in the block diagram. This particular batch delay only occurs if $t = N$, that is, if the process is at terminal time N, which is indicated by the diamond in the diagram. The stored matrices at the terminal time are the upper left corner of the covariance matrix, $\mathbf{P}_k^{11}$, and the repeating part of the error sequence, $\bar{\mathbf{e}}_k$. These are used as the prior estimate of the covariance matrix and the error sequence respectively. Utilizing this iterative process yields an ongoing convergence of the Kalman filter as the progress obtained through a specific batch is stored and further improvement can be made throughout the following batch.

Based on these additional components are both the general MPC structure and the ILC inspired concepts visualized by the extended block diagram. Trough the next sections further explanation of the plant model and the disturbance estimator is given.

4.1.1 Plant Model

The purpose of the plant model is to reflect a broiler house during simulation as the control structure is tuned and verified. It is desirable to utilize a plant model that is as close to real-world behavior as possible. Using such a model minimizes the risk of performance mismatch between simulation and real-world behavior. As MPC is used, will using a proper plant model also reveal any mismatches between the internal model and the plant model. Besides the desire to utilize a close to real-world behaving model, it is necessary that the model incorporates the in- and outputs used throughout the control structure.

The selected plant model originates from the work by Simon V. Johansen and consists of an RNN model structure [7]. A specific RNN model is trained with the use of data from 12 batches originating from the same house and identified by the batch count of the house at the time of training. The model is based on a widely used multilayer perceptron (MLP) model, where the specific type is described as a nonlinear ARX model. The model is structured with one hidden layer and a hyperbolic tangent activation function for this layer. Furthermore, is a linear function used for the output layer. The model structure is represented by the following equation:

$$\begin{aligned} \hat{\mathbf{y}}[t+1|\mathcal{W}] &= \mathbf{W}^{\circ} tanh(\mathcal{X}) + \theta^{\circ} \quad with \\ \mathcal{X} &= \sum_{a=1}^{j} \mathbf{W}_{y,a}^{h} \hat{\mathbf{y}}[t - m_a + 1|\mathcal{W}] + \sum_{b=1}^{i} \mathbf{W}_{u,b}^{h} \mathbf{u}[t - n_b + 1] + \theta^{h} \end{aligned} \tag{4.1}$$

Where

$\mathbf{W}^{\circ}$	$\in \mathbb{R}^{(N_y \times N_h)}$	is the output weights	$[\cdot]$
$\mathbf{W}_{y,a}^{h}$	$\in \mathbb{R}^{(N_h \times N_y)}$	is the delayed output weights,	$[\cdot]$
$\mathbf{W}_{b}^{h}$	$\in \mathbb{R}^{(N_h \times N_u)}$	is the input weights,	$[\cdot]$
θ^{h}	$\in \mathbb{R}^{(N_h)}$	is the hidden layer bias,	$[\cdot]$
and θ°	$\in \mathbb{R}^{(N_y)}$	is the output bias.	$[\cdot]$

N_u is the input dimension, N_h is the dimension of the hidden layer, and N_y is the output dimension. For the use case of the ongoing control design, is the RNN model specified such that it fits the three hour sample time and the input to the model is the broiler house temperature and the provided output is the broiler weight. Furthermore is the broiler weight output accompanied by sequences of the two disturbances, humidity and outside temperature for the whole batch. These disturbance sequences originate from the disturbance measured during the batch associated with the specific RNN model. As a

result are the disturbances batch-wise invariant and thus constant for the specific RNN model.

4.1.2 Disturbance Estimator

The disturbance affecting the system should be known along the entire batch in order to take it into account when minimization the error trajectory. The change in disturbance, $\mathbf{\Delta}\hat{\mathbf{D}}_k[t+m-1|t]$, is defined as $\hat{\mathbf{D}}_{k-1}[t+m-1|t] - \hat{\mathbf{D}}_k[t+m-1|t]$ and is a $\mathbb{R}^{(2m\times 1)}$ sequence, however only $\mathbf{\Delta D}_k[t]$, the first element, is known. In other words, the humidity and outside temperature can only be measured at the current time step. Therefore an estimation of both humidity and outside temperature during a batch should be developed.

Outside Temperature
Estimating the future outside temperature is basically a weather forecast, which is already available from various national meteorological institutes. As the batch is 34 days long, a corresponding length temperature forecast is desirable. It is, however, unrealistic to obtain a 34-day forecast that is fully trustworthy. The longest public available temperature forecast from a meteorological institute with an Application Programming Interface (API) originates from the Norwegian national meteorological institutes, Meteorologisk Institutt (MET), and it is nine to ten days long [19]. A nine-day prediction makes it possible to determine the change in outside temperature from the last batch and thus define the outside temperature the next nine days.

The API allows the user to enter a geographical position, by specifying latitude and longitude. A file, containing temperature, wind speed, humidity, cloudiness etc. is displayed on a web page. To read out the data needed for this project, a Matlab script has been made. The script reads the web page, sorts out the data and localizes the temperature forecast with a corresponding time stamp. The first approximately two and a half days of the forecast contains a sample time of one hour, however, any forecast dated further on is given in six-hour intervals. The simulation environment, however, uses a three hour sample time. This change in sample times is handled by applying a cubic interpolation [20]. The temperature forecast for the first nine days, with both the original data and the three-hours resampled version, is presented below:

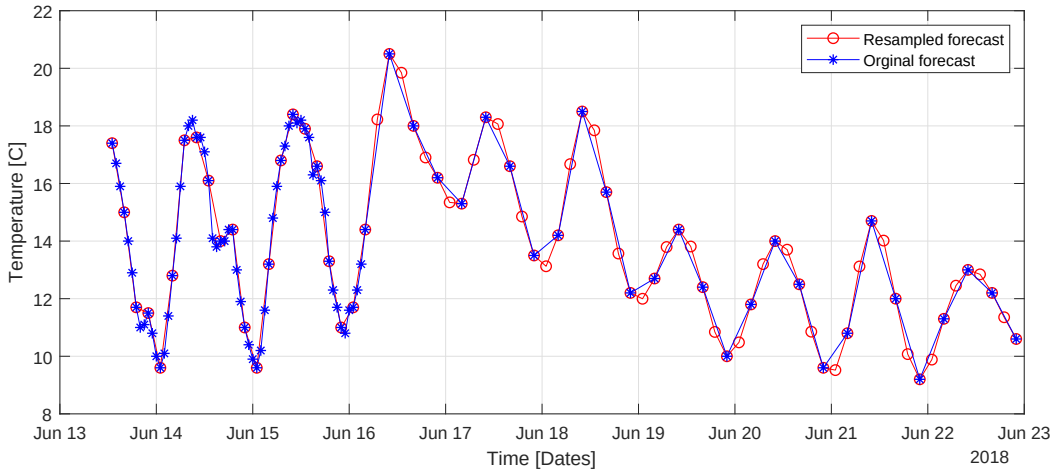

Figure 4.3: A nine-day temperature forecast. The blue forecast is the original forecast directly from MET and the red forecast is the three-hours resampled version.

The process stretches across 34 days and the farmer uses time in between batches to clean

the stable etc..

Because of this, is the outside temperature batch-wise affected by the changing seasons. Therefore the seasons affect the outside temperature from batch to batch.

The expected temperature from month to month can be estimated using historical data. *Table: 4.1* describes the mean temperature across a year, month by month [21].

	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sept**	**Oct**	**Nov**	**Dec**
Temp day	3.2	3.1	6.6	11.9	15.5	18.4	21.8	20.8	17.3	12.6	8.2	4.9
Temp mean	1.4	1.1	3.5	7.7	11.3	14.3	17.4	16.7	13.7	9.8	6.3	3.0
Temp night	-0.9	-1.2	0.4	3.7	7.2	10.1	13.0	12.9	10.3	6.8	4.0	0.7

Table 4.1: Day, mean and night temperatures for each month of the year in Denmark. The data is an average of the historical temperature from the years 2006-2015.

Using the same interpolation strategy as earlier a model, with three-hour sample intervals, describing the temperature of an average day in a given month can be made. This will, however, result in a model where the expected mean temperature of all days in eg. May are 11.3 $°C$ and then at the transition to June, the expected mean temperature will jump to 14.3 $°C$ in a discrete manner. To take this into account the temperature for a specific day is calculated by weighting the temperature curve from the current month and next month, depending on which day in the month it is. A code snippet below shows the algorithm as pseudocode.

```
day = 1;              % The current day of the year
for m = 1:12          % Counter that keep tracks on the month
q = 0;              % Counter to keep track on the weighting factor
for d = 1:lastday     % Number of days in the month
    temp_tmp(day)=temp_m(m)+(q/lastday)*(temp_m(m+1)-temp_m(m));
    day = day + 1;
    q = q + 1;
end
end
temp_y = shift_half_month(temp_tmp); %
```

`Temp_y()` is the created temperature curve describing an entire year. `temp_m()` is the data describing the average temperature curve for a given month. The variable `m` is the current month such that January = 1, February = 2 and so forth. The temperature of the current and next month are weighted with the factor `q/lastday` and then saved as `temp_tmp()`. However, because `temp_tmp()` result in the first day of the month having the average temperature of the specific month, it needs to be shifted forwards half a month, which is done by `shift_half_month(temp_tmp)`. Now a forecast for the first nine days and a model for historical data for every day of the year is available. To obtain an estimated temperature forecast for the next 34 days these should be gathered. However, in order to ensure that the transition phase between the two parts is smooth, an algorithm very similar to the code above is used again. This results in a forecast as presented below.

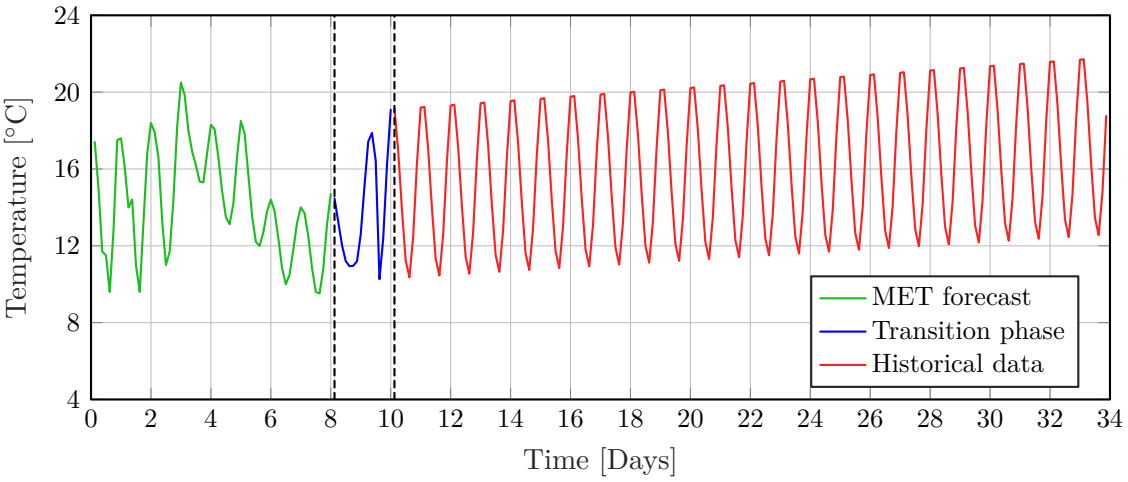

Figure 4.4: The final 34 day forecast of the outside temperature and the elements it is composed of.

Day one through eight in *Figure 4.4*, the green part, is based on the forecast from MET. Day nine, or as long as the MET forecast continues, is a weighted average of the forecast and historical data. This is the section between the two dotted black lines. From day 10 and forward the forecast is pure historical data.

With the temperature forecast it is now possible to find the outside temperature component of $\mathbf{\Delta\hat{D}}_k[t+m-1|t]$ and apply it in a real-time control scenario.

Humidity
The humidity is only measured at the current time step and thus future humidity values should be estimated. This is preferably conducted in a manner similar to the estimate of the future outside temperature, however, such an approach is not viable. A forecast for the humidity inside the broiler house is not existing and thus a different approach must be taken. It is possible to obtain a forecast of the outside humidity however this does not provide any valuable information of the humidity inside the broiler house.

The humidity inside the broiler house is mainly affected by the amount of biomass in the broiler house, furthermore are factors as the particular litter type and volume strongly affecting the humidity. Finally, it is possible to apply humidity set points through the ventilation system, similar to the temperature set points. Different farmers are thus able to run different humidity profiles trough batches, serving their specific needs. Due to these factors should a humidity model include both the amount of biomass, litter type, litter condition and finally the humidity bounds stated by the farmer.

Due to these factors, it is, for now, most reasonable to base the humidity forecast on historical data, that is the humidity data recorded in the previous batches. It is chosen to base the prediction on the humidity through the previous 10 completed batches.

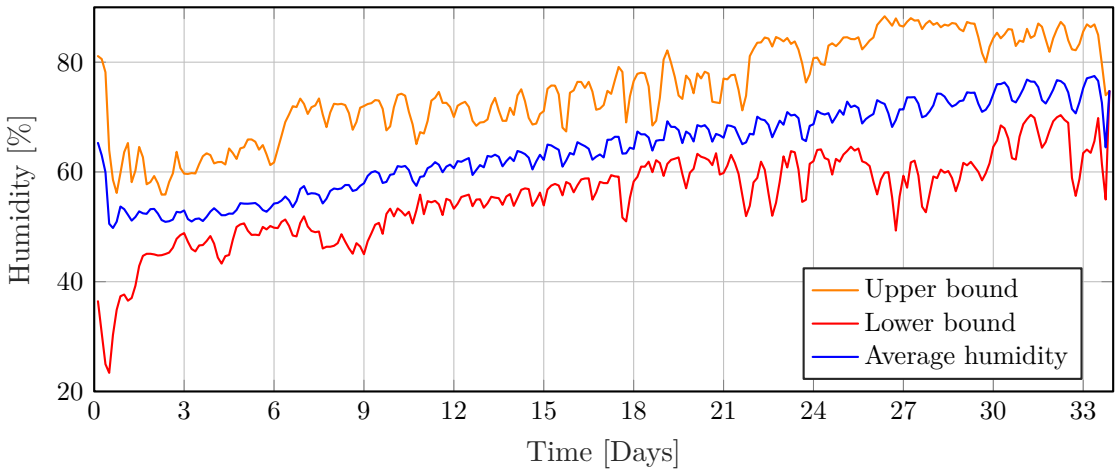

Figure 4.5: The predicted humidity, based on the previous 10 batches.

The sample vise average humidity across a batch, estimated by the use of the 10 previous batches is shown in *Figure 4.5*. The sample vise upper and lower bounds are also presented to indicate the possible deviation in the prediction. This averaged historical data based humidity is utilized as the humidity prediction for the next batch. This process is repeated when a new batch is started, thus always using the 10 most recent batches.

Based on this prediction is the humidity component of the disturbance estimator now determined. Both disturbances can now be predicted across the length of the batch, thus also enabling implementation of the control structure on a real broiler house.

Simulation and Results 5

During this chapter is the controller tuned, such that the optimal performance is achieved. Furthermore are results from both simulation and live test presented and explained.

5.1 Simulation

The implemented control structure is tested in a simulation environment such that the performance can be verified prior to any real-world testing. In order to obtain the best possible performance is the controller tuned by the use of the simulation environment.

The main focus of the controller tuning is centered around the performance of the MPC. The tuning parameters in the MPC originate from the weighting matrices, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ used in the cost function. Both matrices are chosen to be time-independent, thus consisting of the same amount of punishment for the specific term, regardless of the time. Due to this choice is the tuning procedure a matter of scaling the ratio between the two weighting matrices. For simplicity is the weighting matrix for the state, $\mathbf{Q}_1$, chosen as identity and only the weighting matrix for the change in input, $\mathbf{Q}_2$, scaled to achieve the best performance. The MPC tuning is conducted by increasing the diagonal elements of the weighting matrix for the change in input, from zero until 300 in steps of two. For each value of the weighting matrix are 15 batches simulated. This allows the algorithm to converge and it is thus possible to study how the long-term control objective is handled. The fit percentage is calculated for each batch and plotted according to the batch and the weighting factor. The sweep across the described weighting factor is presented in *Figure 5.1*, it is chosen only to present the results for a weighting factor between 100 and 300 as the fit drops significantly for lower values and thus blurs the significant findings.

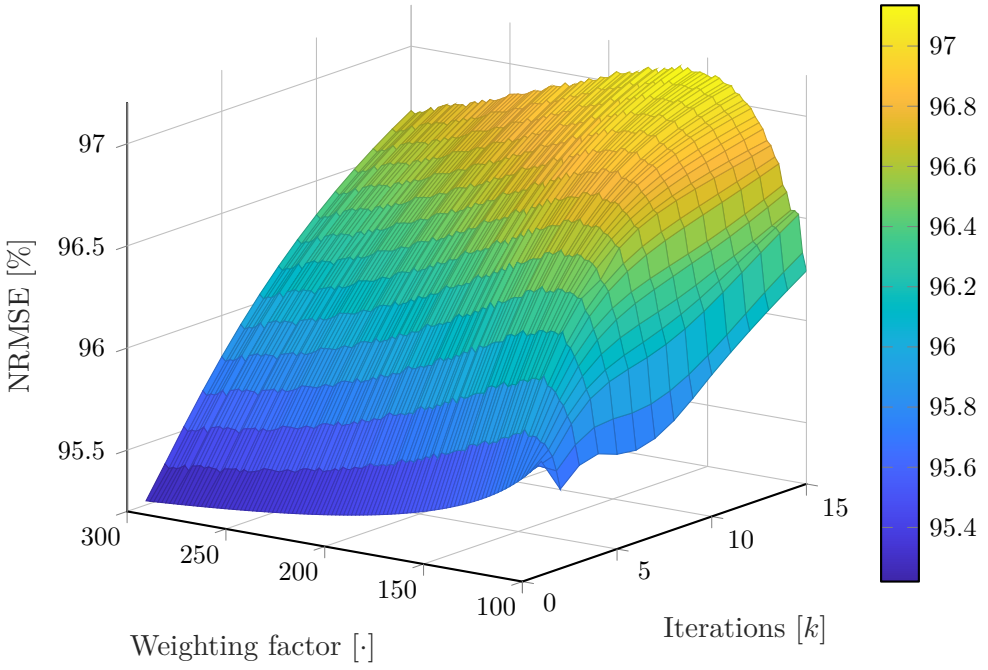

Figure 5.1: Performance based on weighting factor and batch count.

The sweep indicates that there exists an optimal weighting factor that ensures both higher fit percentages and thus also a steeper learning curve. Both of these characteristics are desirable as the higher fit percentages reflect the short-term performance goal and the steeper learning curve reflects the long-term performance goal. The optimal weighting factor is by closer inspection determined to 164 with an uncertainty of ± 2 as the sweep is conducted in steps of two.

The optimal weighting factor is now determined and further investigation of the long- and short-term performance is presented in *Figure 5.2*. The tracking error is presented with respect to both sample time and batch count.

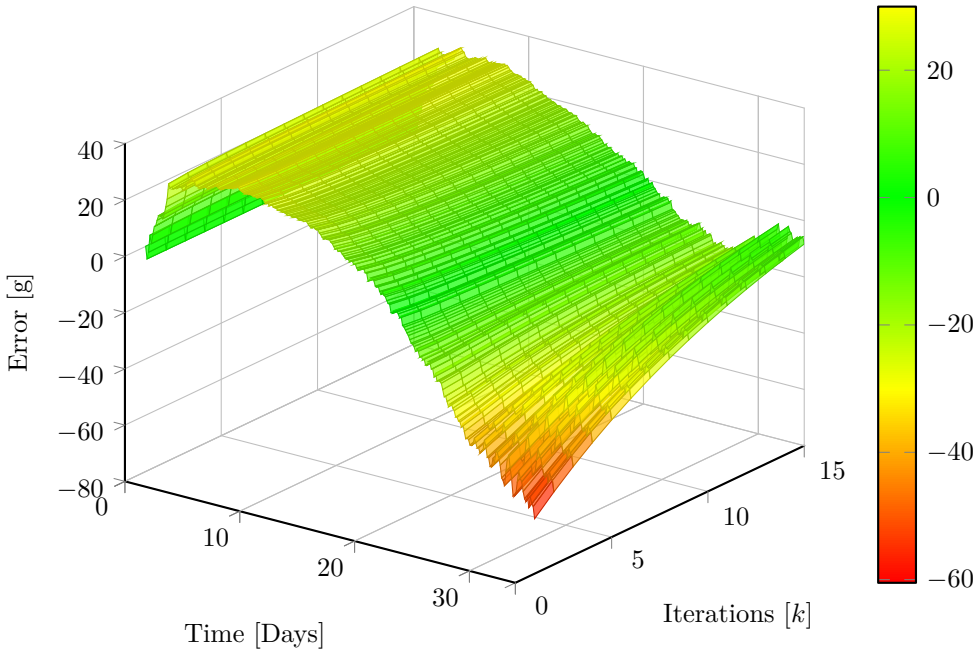

Figure 5.2: Performance across 15 batches as the weighting factor is chosen to 164.

The tracking error across batches shows a significant improvement. Looking at the error at the end of the first batch reveals a deviation of approximately -60 grams. Considering the error at the end of the last batch shows that it is reduced to approximately -8 grams.

As the optimal weighting parameters are determined is the algorithm simulated across 90 batches to ensure that it continues to learn and to investigate if the control input and fit percentages converge. During simulation, is the implementation described in Section *4.1: Controller Implementation* followed, however, the use of the disturbance estimator is omitted. Under simulation conditions are the control algorithm fully centered around the RNN plant model. As described earlier in Section *4.1.1: Plant Model* is the weight output from the RNN model accompanied by a batch-wise static disturbance sequence. This sequence corresponds to the disturbances experienced during the batch associated with the deployed RNN model.

The references concerning both weight and temperature are chosen based on recommendations provided by the broiler handbook of the utilized broiler type [17]. The weight reference is based on the handbook as it is important to apply a reference that is obtainable. The temperature is chosen as a simple line with constant slope based on the handbook, however, this is of less significance as the controller is able to adjust the temperature sequence iteratively.

The fit percentage across 90 batches is presented in *Figure 5.3*.

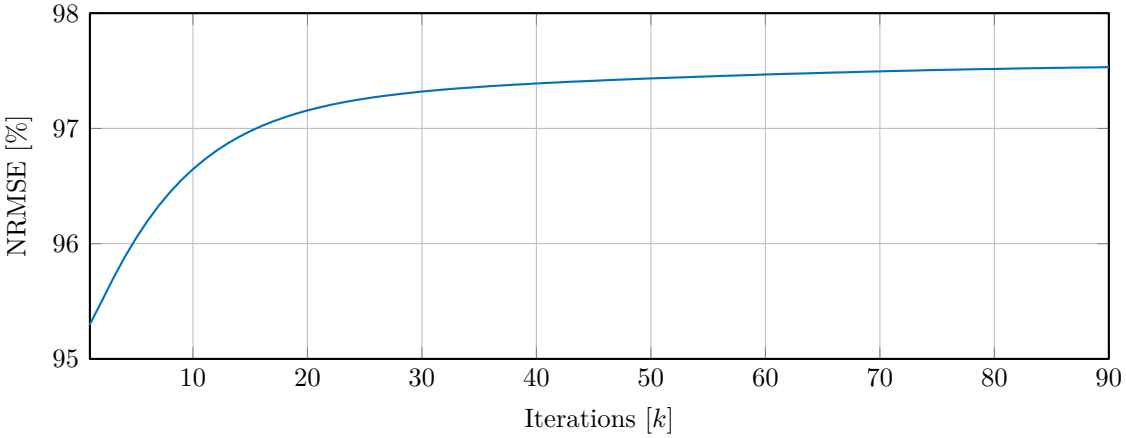

Figure 5.3: The fit percentage of the simulated output weight according to the reference.

The fit percentage obtained as a result of the 90 batches long simulation verifies that the algorithm constantly learns and improves across batches. This property is also visible in *Figure 5.4* as the simulated weight output is getting closing to the reference trajectory as the batch count increases. The results in *Figure 5.4* only presents the development during the first 30 batches in order to maintain the figure readable. The blue color is the first batch, and as more batches are simulated the color turns green. The change in weight across all 30 batches is most significant from day 25 and onwards. This is shown in the right part of *Figure 5.4* where an enlargement of the weight trajectory from day 30 to 34 is presented.

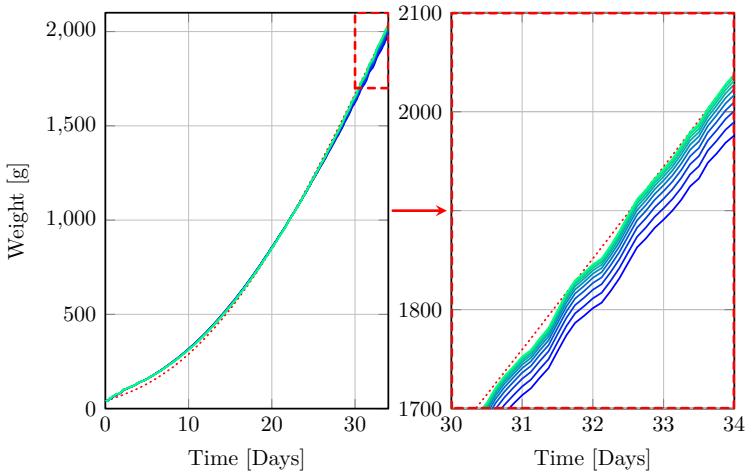

Figure 5.4: The blue color is the first batch, and as more batches are simulated the color turns green. **Left:** Reference weight trajectory and simulated weight across 30 batches. **Right:** A zoom of the growth curve the last four days.

The batch-wise increasing behavior generally corresponds to the changes in input temperature where the greatest changes likewise occur from day 25 and onwards. An exception is the change in temperature across batches for the first three days. These changes are not particularly visible in the output weight, it is, however, most likely just a numerical issue as a temperature change early in the batch only results in a minor numerical weight change compared to an equal change on a later stage in the batch. The

first 30 weight trajectories obtained by the simulation are presented in *Figure 5.5* and accompanied by enlargements of the initial and final sections.

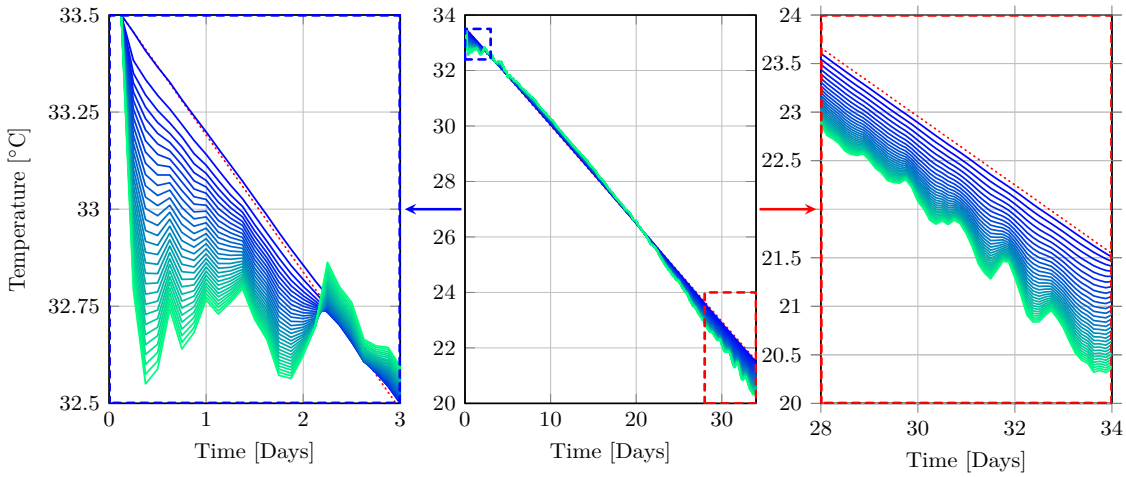

Figure 5.5: The blue color is the first batch, and as more batches are simulated the color runs green. **Left:** A zoom of the temperature curve the first three days. **Middle:** Initial temperature trajectory and the iterated trajectories across 30 batches. **Right:** A zoom of the temperature curve the last six days.

The iterated temperature curve in *Figure 5.5* reveals how the input sequence is changing as the algorithm learns and thus evolves along batches. Considering the temperature curve for the last couple of batches the behavior indicates a slight ripple tendency during the last three days. Compared to the fit percentage and output, it does not indicate any problem but this behavior would likely be considered undesirable in a real-world control scenario as the broilers are sensitive to rapid temperature changes. It is in a real-world scenario likely that the output weight might respond differently as changing disturbances are to be expected and thus will the input temperature behave differently. It is also significant to consider that a batch count of 30 batches represents approximately three and a half years of run time, therefore will a simulation under the same conditions for such a period not necessarily reflect the true real-world conditions as the disturbance, broiler genetics, and house specific conditions are likely to vary during such a period.

The simulation results show that the algorithm operates desirably, this is indicated by the output weight closing in on the reference and minimizing the error trajectory. The input temperature is generally within an acceptable range. Finally, the iterative fit percentages indicate a converging behavior as expected and thus the performance gain reduces, as the batch count increases.

5.2 Live Test Results

Throughout this section will the real-world controller test and the obtained results be presented. The control algorithm is tested on a real-world broiler house equipped with a state of the art SKOV A/S climate control system and located in northern Denmark.

It is prior to implementing the control algorithm on the broiler house desirable that the controller acquires knowledge about the specific house. The learning procedure of the control algorithm is presented in the previous Section *5.1: Simulation* and consists of batch-wise learning based on simulation. The RNN model is during simulation selected

such that it originates from the house that the control algorithm is to be implemented on. It is through the use of this procedure possible to prepare the controller for a specific house.

Conducting real-world tests on livestock stables requires a number of precautions as the health of animals are at stake. It is due to this fact not allowed to implement the controller directly on the livestock stable. The implementation sequence is initialized by the off-line calculation of a temperature reference determined with the simulation environment. Applying the reference requires a sampling process as the climate system only facilitates eight temperature set points throughout the 34 day batch period. This reference undergoes a sanity check and approval process from both a broiler application expert at SKOV A/S and the farmer. After approval is the reference applied to the broiler house climate control system.

It is in agreement with the farmer allowed that a weekly update of the temperature reference is possible. However, any changes must undergo the implementation process mentioned earlier. Due to this structure is the remaining of this section divided into stages where first an initial temperature reference is described followed by the weakly updates. Additional measurements not presented during the weekly updates as well as the complete weight and temperature measurements can be found in Appendix *E: Peripheral Live Test Results*.

5.2.1 Initial Adaption

Generating the initial temperature reference for the live test is conducted in the previously described simulation environment. The control structure is simulated across 90 batches, which corresponds to approximately 11 years of production time. The iterated temperature sequence obtained after 90 batches is chosen and the implementation process described earlier is commenced.

The new temperature sequence, compared to the original reference, suggest a lower initial temperature and lower final temperature as well. Furthermore is the temperature slightly higher between day four and day 20, compared to the original reference. This behavior minimizes the tracking error and thus increases the fit percentage as discussed in Section *5.1: Simulation*.

The sampling process determines eight points throughout the temperature curve. These points are placed by the use of an algorithm that utilizes free-knot splines to fit one-dimensional data [22] [23]. The algorithm is however only able to select floating point coordinates, therefore a rounding procedure is necessary as the climate system only allows integers. The adjusted and implementable temperature is presented in *Figure 5.6*.

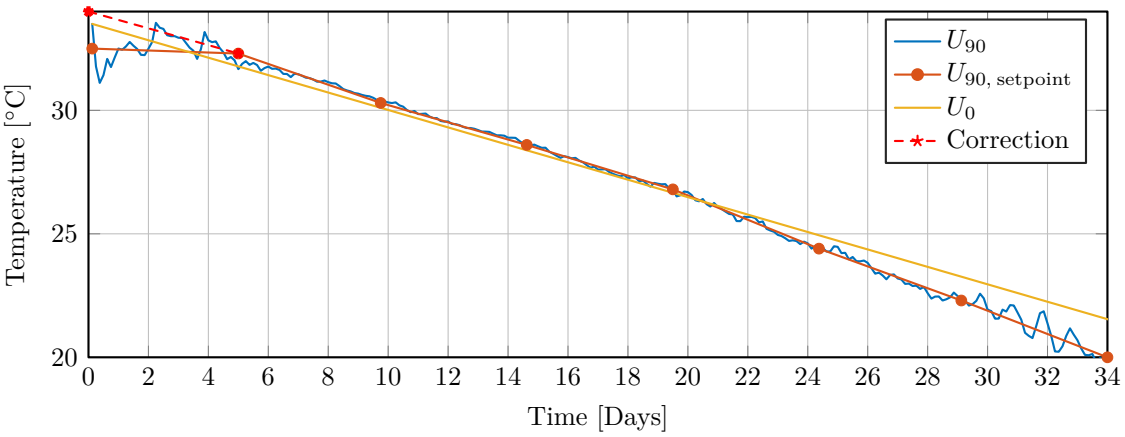

Figure 5.6: The temperature curve after 90 batches, with corresponding set points. The yellow line is the initial reference at $k = 0$ and the dashed red line from day 0 to 5 is the correction requested by the farmer.

An implementable temperature sequence is now determined, and the last step before actual implementation is the approval of the sequence. The suggested temperature sequence is assessed and the initial temperature is considered to low compared to the broilers that are inserted during the specific batch. It is, therefore, demanded that the initial temperature is raised from 32.5 °C to 34.0 °C, this change is shown in *Figure 5.6*. The final temperature sequence is determined and implemented on the broiler house. The internal climate control unit placed in the broiler house is now fully responsible for following the newly designed temperature reference.

5.2.2 Week One

It is possible to update the temperature sequence for the remaining runtime of the batch as one week of the live test has passed. Recorded data during the week from the broiler house is utilized to determine if any temperature changes should be made. The data is updated on a daily schedule and features all relevant measurements. The data is imported into the simulation environment that is used to recreate an online control scenario, and thus simulate the batch as if it was an online control task. The data is, in other words, feed to the simulation environment in order for the controller to learn. The recreating of the online control scenario in the simulation environment and the subsequent simulation is split up according to *Figure 5.7*.

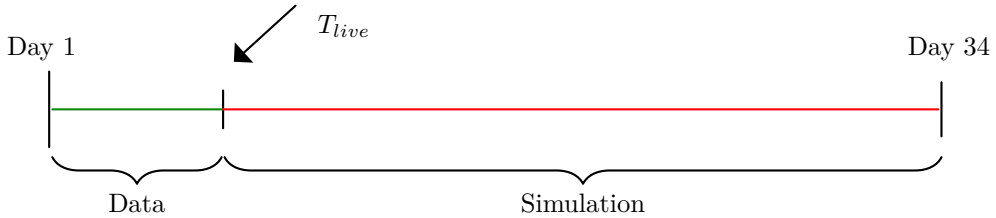

Figure 5.7: An illustration that shows when and where the measured data is utilized in the simulation, such that the controller learns.

Measured data is available until time T_{live} and up to this point are the signals, $y_k[t]$, $\Delta U_k[t]$ and $\Delta D_k[t]$ from the block diagram in *Figure 4.2* consisting of measured data as the simulation is conducted. From T_{live} and onwards is the output of the MPC again

applied to the RNN plant model and the simulation is conducted as originally described. It is therefore not possible to recreate a perfect scenario as the outputs of the MPC cannot be applied as long as measured data is available. The simulation is effectively conducted in an open-loop fashion until no more measurements are available as it is desired to utilize all weight measurements and thus obtain the correct error sequence. The algorithm is provided with the recorded measurements concerning temperature, weight, and disturbance. This procedure allows the Kalman filter to commence a converging behavior as recorded live measurements are provided. When no more live measurements are available is the loop closed and the simulation environment is utilized to determine the future temperature sequence based on the RNN plant model. The recorded temperature data, the implemented reference, and the previously described simulation are presented in *Figure 5.8*.

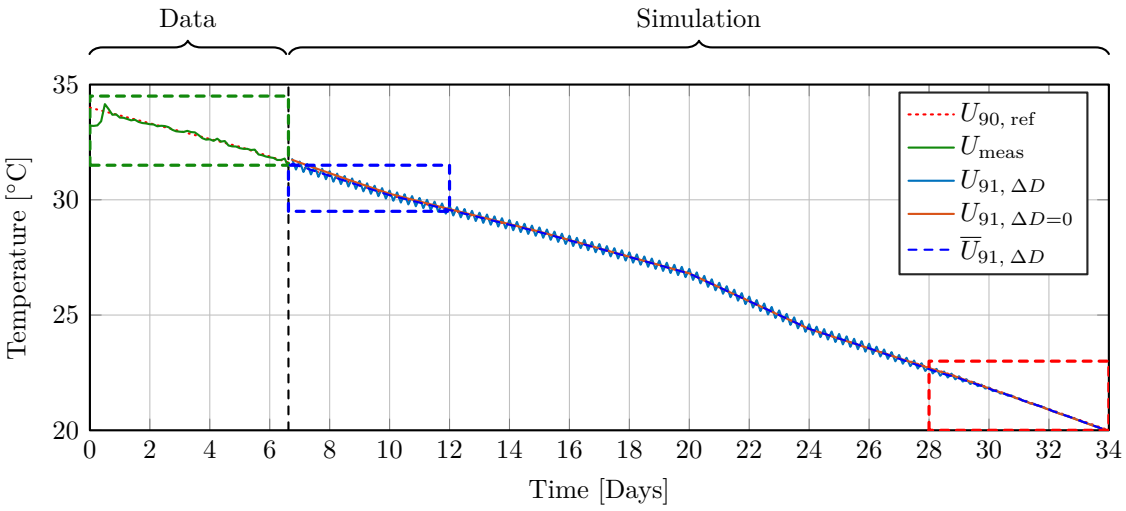

Figure 5.8: A gathered temperature curve that consists of the measured temperature, until the black dashed line, and the following simulation.

The split between measured data and simulation results is indicated by both a black vertical line and the brackets at the top of the figure. To investigate the details of *Figure 5.8* three magnified sections of the figure are presented in *Figure 5.9*. The labeling scheme will be kept during the live test and are therefore omitted in the following magnified figures.

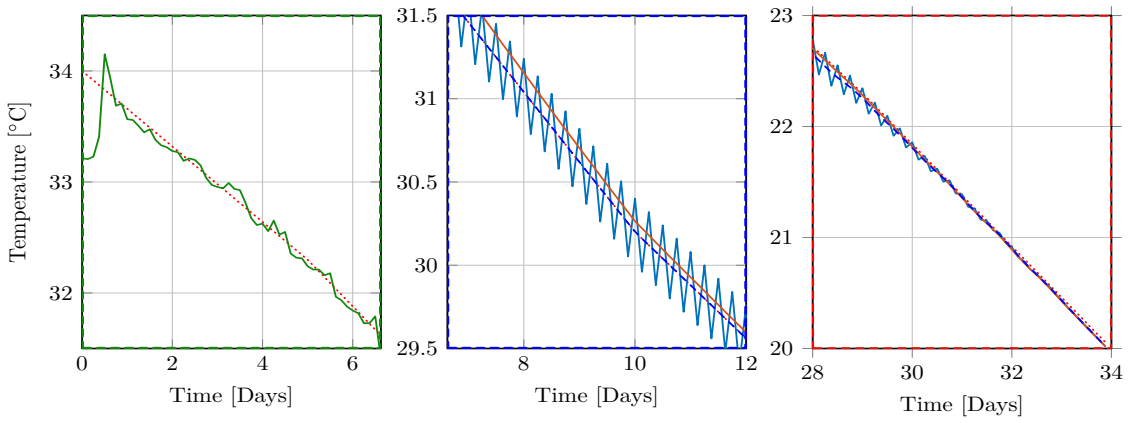

Figure 5.9: **Left:** The reference and the measured temperature curve. **Middle:** A zoom of the ripples which are present the first four weeks of the simulation. **Right:** A zoom at the end of the simulation.

To the left in *Figure 5.9* are the measured temperature and reference presented. The measured temperature is generally close to the desired reference, the initial deviation is very likely to be caused by the broiler insertion as it requires the house to be open during the procedure thus a loss in temperature is inevitable. During the remaining time of the week, form approximately day one is the reference followed in a satisfying manner.

In the middle of *Figure 5.9* is a section of the simulation enlarged and presented in order to clarify the behavior of the MPC. Within the figure four plots are presented, these are the temperature reference $U_{90,\,\mathrm{ref}}$, a simulation with disturbances $U_{91,\,\Delta D}$, a simulation without disturbances $U_{91,\,\Delta D=0}$, and an averaged version of the simulation with disturbances $\overline{U}_{91,\,\Delta D}$. To explain the different simulation types a recap of the simulation environment is given. The simulation is originally defined with a disturbance estimator as shown by the block diagram in *Figure 4.2*, this estimator is however not utilized during the tuning and controller simulation presented in Section *5.1: Simulation*. The omission of the disturbance estimator is chosen due to the usage of the RNN plant model as this model is designed to a specific batch and thus the measured disturbances from the specific batch are utilized. The controller structure is however based on batch-wise variations and thus also batch-wise disturbance variations, see *Equation: (3.15)*. During the tuning and the initial simulation, this combination leads to the usage of a constant disturbance that is zero as previously mentioned. As the live test is commenced the disturbance estimation is utilized to predict the future humidity and outside temperature. This change to a nonzero disturbance has resulted in an unexpected behavior of the MPC where a rippling effect occurs in the suggested temperature. The MPC implementation specifies the upper and lower constraints imposed on the batch-wise temperature change as ± 0.2°C. It is chosen to stay loyal to the implemented control structure such that valid conclusions can be drawn and thus not implement structural changes as the test evolves. It is however not possible to implement the rippling control signal directly due to technical limitations and concern about animal health. Utilizing the eight-point sampling procedure would smooth out the ripples, this is however not the desired way to handle the behavior. A compromise is found by applying a moving average thus obtaining a smooth temperature curve that reflects the overall tendencies of the MPC output. The rippling and the smoothed temperature curves are both presented in the middle and to the right in *Figure 5.9* as a blue line and a blue dashed line respectively.

As comparison is the simulation also conducted without the disturbance estimator,

which is shown as $U_{91,\,\Delta D=0}$, thus reflecting the conditions utilized during the controller tuning. The behavior without disturbances is generally close to both the reference and the smoothed temperature curve. Throughout the following result description will the temperature curve without disturbances be omitted as it does not contribute with a significantly different trajectory and it is desired to keep the chosen model structure utilizing the disturbance estimator.

To the right in *Figure 5.9* an enlargement of the behavior between day 28 and 34 is presented. The enlargement shows that the rippling behavior diminishes and the suggested temperature becomes smooth as a consequence. Considering the temperature changes, it is clear that the algorithm desires to let the temperature follow the same trajectory as initially found. The temperature aspect of the algorithm during week one is now widely covered and a natural progression is an investigation of the broiler weight which is the main focus of the control task. The measured weight, the reference trajectory, and the simulation outputs are presented in *Figure 5.10*.

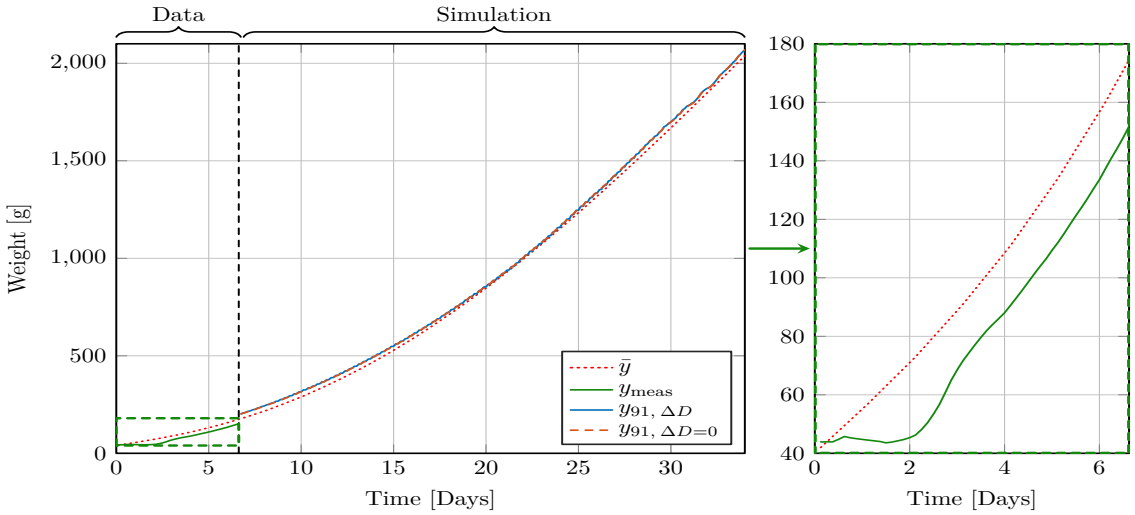

Figure 5.10: Weight reference and measured weight combined with the simulation output.

The measurements obtained during the first week reveal that the broiler weight is significantly lower than expected, as shown in *Figure 5.10*. Even though the weight is not at the desired level is the obtained growth trajectory close to the reference trajectory when disregarding the weight offset. This indicates that the broilers are growing as expected and the weight offset could be caused by a false initial weight. This weight offset could further be affected by the time of broiler insertion and weighting delay. The newly inserted broilers do not wander around the delay is like to be caused by the

To determine whether the current implemented temperature reference should be updated, the simulation results presented in *Figure 5.8* and *Figure 5.10* are considered. These reveal that the future broiler weight will be close to the reference and this will be obtained by following the current temperature reference. No apparent temperature changes are therefore desired and thus the currently implemented temperature reference is retained.

5.2.3 Week Two

The decision of updating the temperature curve during week two is evaluated in the same manner as for week one. However, will the data and results for this and future weeks only be presented as the magnified versions, with focus on the week specific events. The measured temperature U_{meas}, the implemented reference $U_{90,\,\text{ref}}$, the simulation with disturbances $U_{91,\,\Delta D}$, and the averaged version of the simulation with disturbances $\overline{U}_{91,\,\Delta D}$ are presented in *Figure 5.11*.

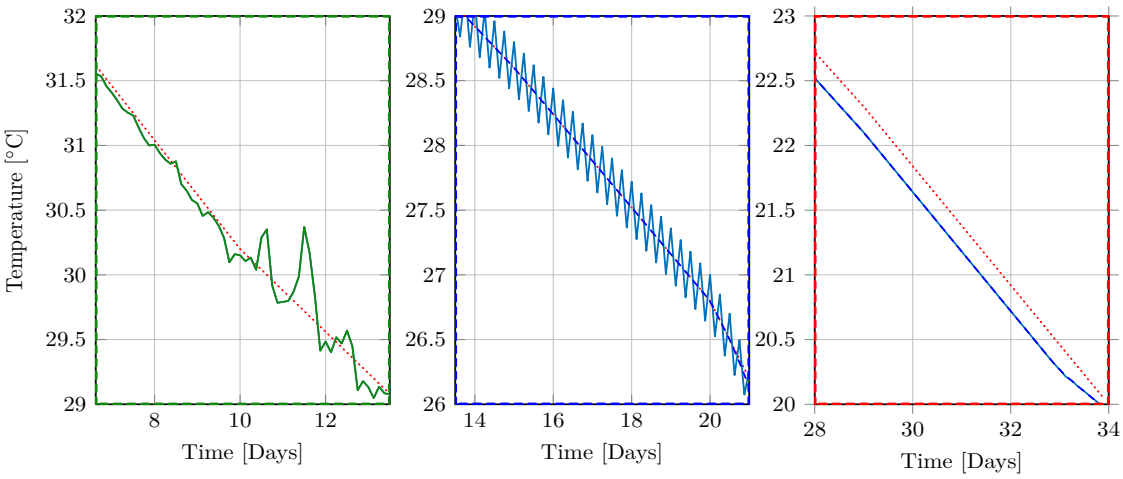

Figure 5.11: **Left:** The reference and the measured temperature curve. **Middle:** A zoom of the ripples which are present until approximately day 21 of the simulation. **Right:** A zoom at the end of the simulation.

Considering the measured temperature displayed on the left in *Figure 5.11*, it reveals that the week is initialized with temperatures sufficiently close to the reference. However on day 10 and onwards it is visible that the climate control is struggling to maintain the reference temperature. The temperature deviations are highly periodic with a clear dip at midnight on day 11. A contributing cause to the deviations might arise from the measured outside temperature, which on both day 10 and 11 peaks at 25 and 28 degrees °C respectively, see Appendix *E: Peripheral Live Test Results*. The enlarged section of the MPC output presented in the middle of *Figure 5.11* reveals that the suggested temperature of the MPC still ripples undesirably. As the rippling behavior vanishes is the lower constraint that is imposed on the MPC activated, and thus is the learning effectively saturated. This split in trajectory occurs approximately at day 21 where the new smoothed version of the control sequence maintains a slightly lower trajectory than the implemented temperature curve throughout the remaining of the batch, as presented in the magnified section to the right.

The third enlargement of the final section of the three temperature sequences is presented to the right in *Figure 5.11*. This magnification reveals that the control algorithm desires to lower the temperature slightly compared to the implemented reference.

This weeks weight measurement presented in *Figure 5.12* shows that the deviations experienced in the previous week are diminishing and the weight generally approaches the reference. However, during the last day of the week is the weight increasingly deviating from the reference yet again. This behavior could be caused by the delayed effect of the temperature fluctuations presented to the left in *Figure 5.11* as the broilers are still highly temperature sensitive due to their young age.

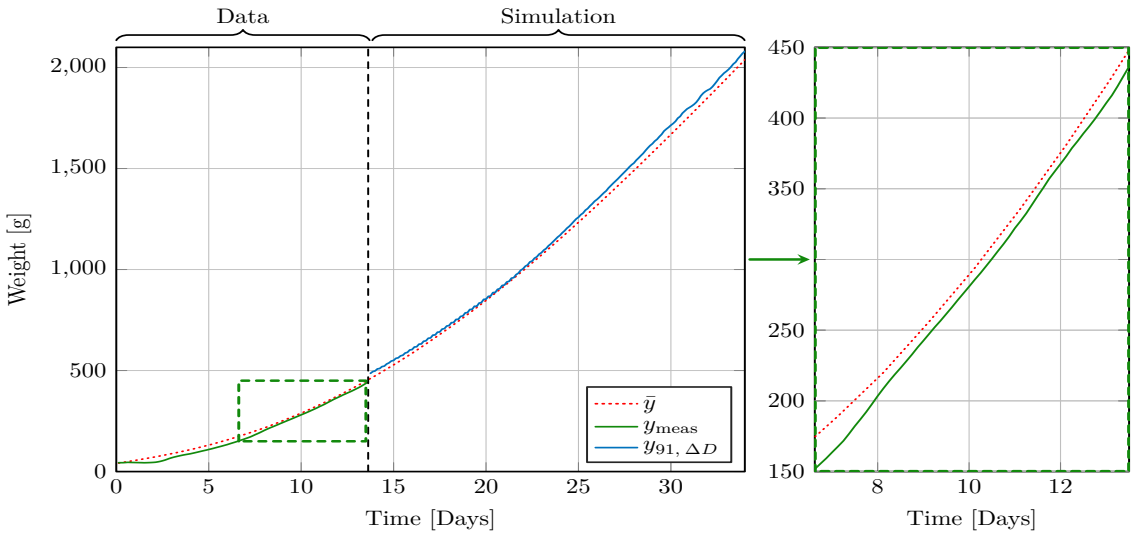

Figure 5.12: Weight reference and measured weight combined with the simulation output.

The effect of the simulated temperature curve is also visible in *Figure 5.12* and shows that the broiler weight throughout the simulation follows the reference trajectory as expected. The deviation present from approximately day 22 and until the end of the batch is largely a result of the limitation caused by the model utilized internally inside the MPC. This behavior is in line with the fit percentages and error trajectories described in Section *5.1: Simulation*.

Summing up the week, the measured temperature has revealed some deviations during the end of the week whereas the weight generally tends to approach the reference as desired. The simulation is suggesting a slightly lower temperature from approximately day 21 and onwards. The next temperature update will take place on day 21 and the simulations suggest to follow the implemented reference until day 21. No changes to the temperature are necessary throughout the following week, it is therefore chosen to retain the currently implemented temperature sequence.

5.2.4 Week Three

The progress of the batch throughout week three is described in the following. The measured temperature, the temperature reference and the temperature suggested by the simulation are presented in *Figure 5.13*.

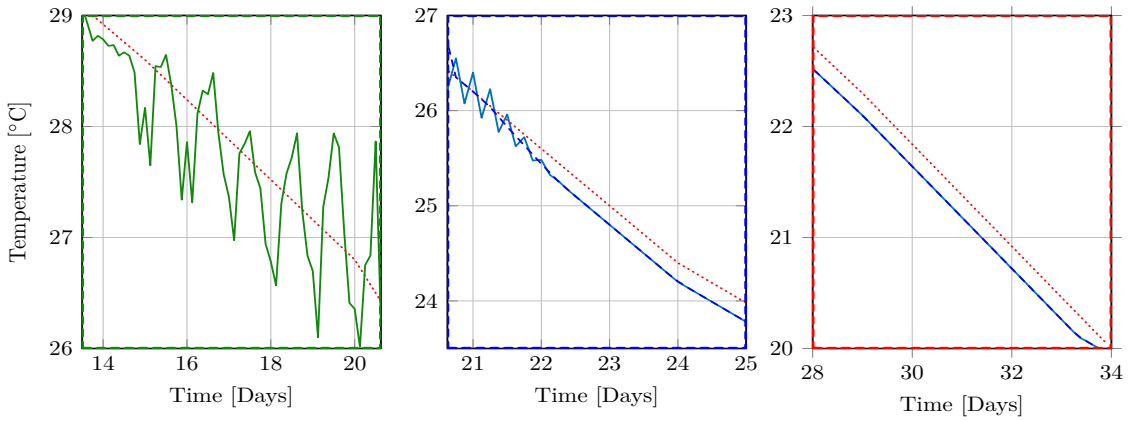

Figure 5.13: **Left:** The reference and the measured temperature curve. **Middle:** A zoom of the ripples which fade away. **Right:** A zoom at the end of the simulation.

The measured temperature data reveals some deviation for the first day of the week and thereafter huge and highly periodic deviations. The deviations measured, are by far extending beyond what is expected to be reasonable and stretch to a peak to peak difference of almost two degrees at the end of the week. The measured peak outside temperatures from day 15 to 20 range from 23.5, 27.9, 22.5, 24.7, 25 and 24.8 degrees °C respectively, see Appendix *E: Peripheral Live Test Results*. It is thus in combination with the increasing biomass likely that the climate control system is challenged and operates close to the limit as the reference temperature is rather close to the outside temperature.

The simulation results reveal a continued desire to lower the temperature slightly. The magnified section presented in the middle of *Figure 5.13* shows that the rippling within the MPC output diminishes quicker than experienced in previous weeks and follows the implemented trajectory with an offset. This behavior continues until the end of the batch apart from a minor temperature increase at the end of the batch, see *Figure 5.13*.

This weeks weight results presented in *Figure 5.14* reveal that the deviation that ended last weeks measurement is diminishing. Furthermore, it is visible that the measured weight follows the trajectory highly accurately until approximately day 19. From day 19 a minor deviation is appearing, this deviation is however within an acceptable range.

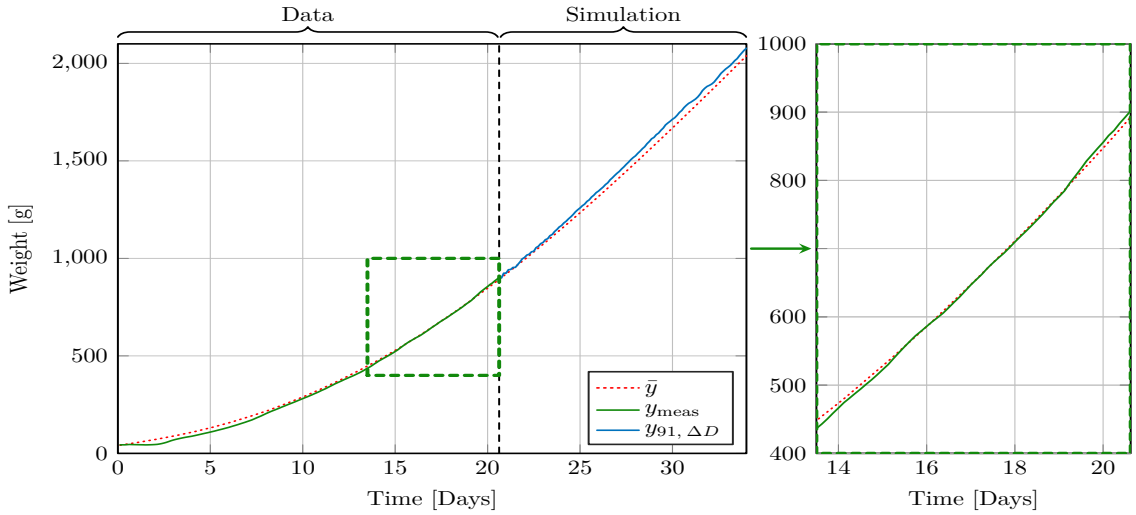

Figure 5.14: Weight reference and measured weight combined with the simulation output.

Summing up this week, large temperature deviations are experienced, however, the broiler weight reveals the desired progress and is close to the reference. It is due to these observations and the absent effect of the temperature deviations chosen to retain the implemented temperature reference. Even if the change is implemented is any effect on the broiler weight doubtful as the implementable temperature change is limited compared to the experienced deviations.

5.2.5 Week Four

The progress throughout the fourth week of the batch is described in the following. The implemented temperature reference, the measured temperature and the temperature obtained by simulation are all presented in *Figure 5.15*.

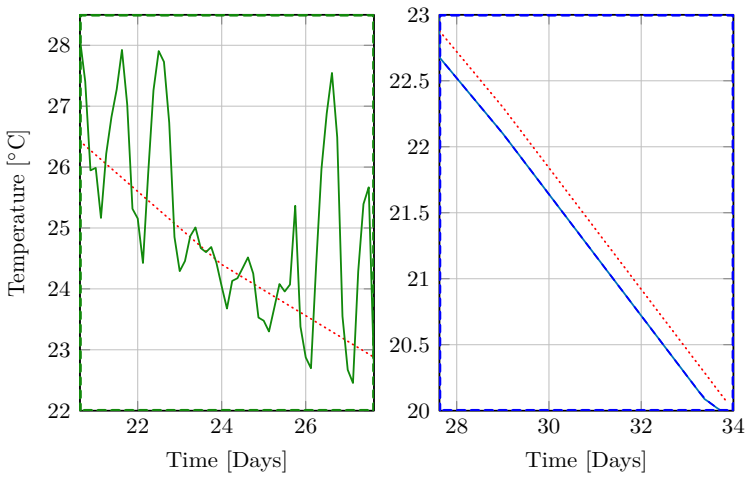

Figure 5.15: **Left:** The reference and the measured temperature curve. **Right:** A zoom at the end of the simulation, which this week is the reamining of the batch.

Compared to the previous week are the experienced deviations increasing even further and at day 27 is the peak to peak temperature difference close to five degrees. The measured temperatures at day 24 and 25 are somewhat satisfying and the deviations are greatly reduced during these days due to lower outside temperatures, see Appendix *E: Peripheral Live Test Results*.

The simulation result reveals that the MPC desires to constantly lower the temperature during the remaining of the batch period compared to the implemented reference. The suggested temperature change reveals that the simulated output follows the bounds imposed on the MPC.

The enlargement presented to the right in *Figure 5.16* reveals that the measured weight throughout this week is increasingly deviating from the desired reference. The trajectory of the measured weight is smooth which indicates that the growth is normal, the broilers are however growing faster than expected.

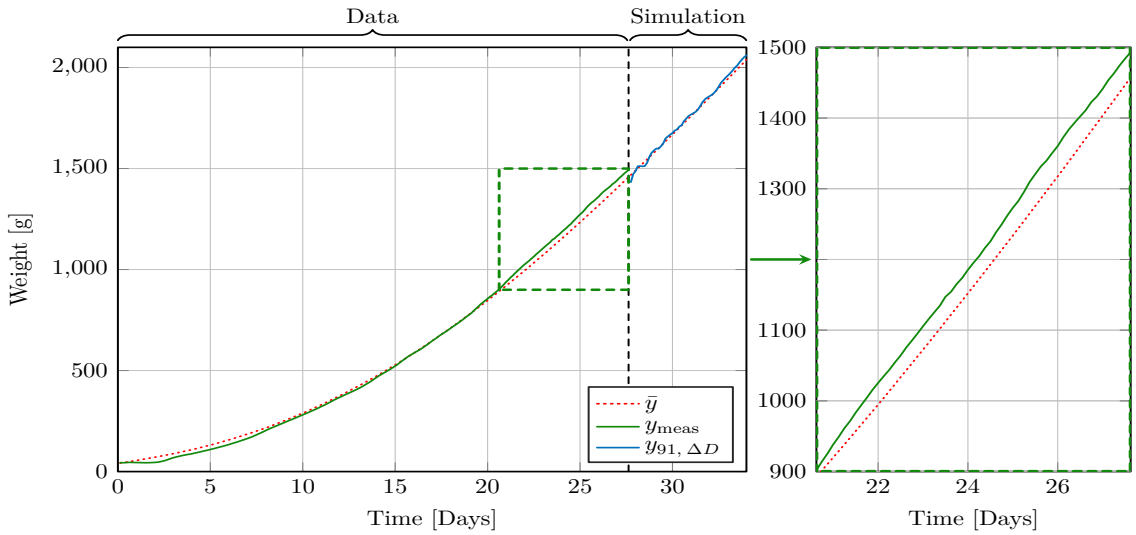

Figure 5.16: Weight reference and measured weight combined with the simulation output.

Summing up the week, the measured temperature reveals increasing deviations compared to the reference but indicate no significant effect on the measured broiler weight. The temperature sequence obtained by simulation desires to lower the temperature along the entire remaining timeline of the batch. It is chosen not to implement the new trajectory as it is highly doubtful that the suggested change of 0.2 degrees °C can achieve any significant weight change. This assessment is based on the previously described observations that the increasing temperature variations of up to 5 degrees peak to peak, tend to have no effect on the weight at this stage of the broilers growth.

5.2.6 Week Five

Week five is the final week of the batch and it is thus not possible to apply any temperature change. The measured temperature and the reference are presented in *Figure 5.17* with a magnification of this weeks temperature placed to the right.

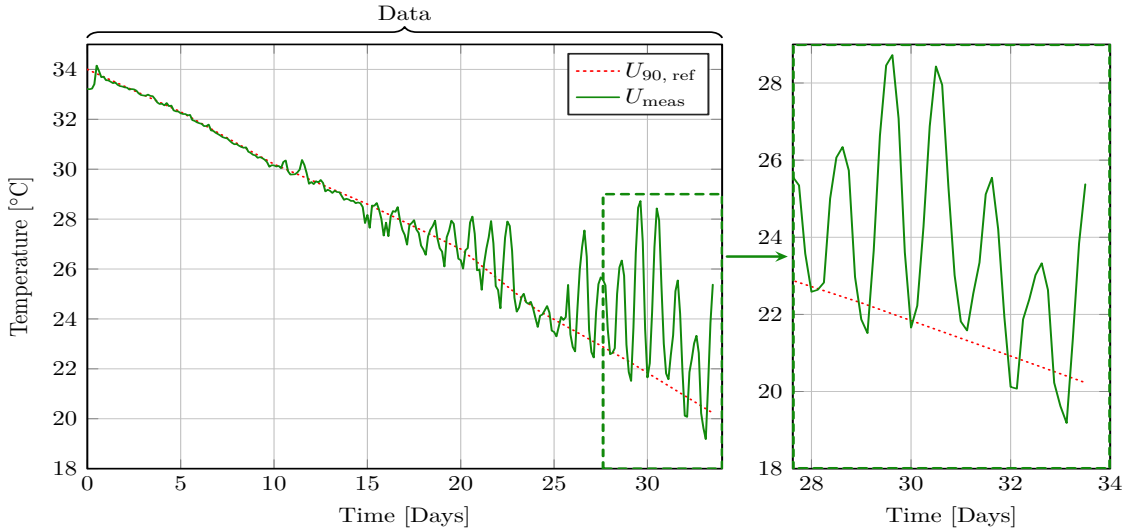

Figure 5.17: **Left:** The reference and the measured temperature curve for the entire batch. **Right:** A zoom of week five.

The measured temperature during this week reveals that the climate control system continues to struggle with the heat as large temperature deviations are measured. The measured outside temperature reveals that the peak daytime temperatures between day 28 and 32 are ranging from 24.5 up to 30.7 degrees °C, see Appendix *E: Peripheral Live Test Results*. The present cooling system consists of vents and fans which this week is an insufficient cooling method for these high temperatures combined with a temperature reference below ambient.

The measured weight presented in *Figure 5.18* reveals a continuation of the trend observed during the previous week. The measured weight is thus steadily increasing and a further deviation from the desired weight reference is revealed.

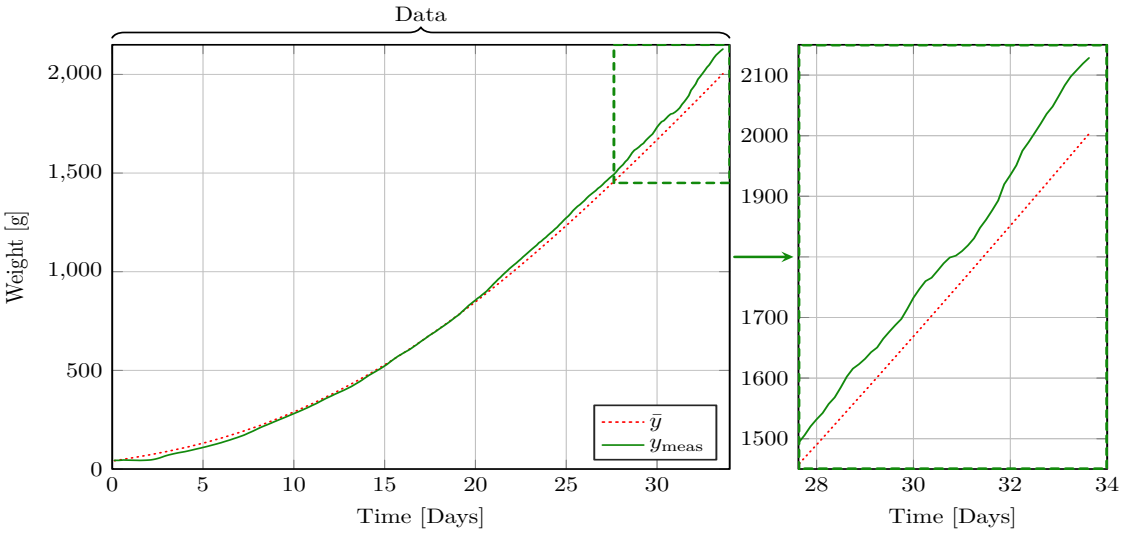

Figure 5.18: **Left:** The weight reference and the measured weight curve for the entire batch. **Right:** A zoom of week five.

The results obtained during week five complete the batch and the final broiler weight is measured to 2128 grams, which is 78 grams above the desired reference of 2050 grams. The slaughter weight reveals an average of 2090 grams which is 38 grams lower than the measured weight and thus indicates the reliability issues concerned with the broiler weight during the last stage of a batch as earlier mentioned, see Section *2.1: Data Description*.

5.2.7 Live Test Discussion

The conducted live test is stretching across 34 days and the designed control algorithm is thus tested under real-life conditions during a complete batch.

During the batch progression, it has become visible that the climate control system is struggling to maintain the desired temperature reference. This behavior occurs the first time during week two and becomes very significant during week three to five due to both the constantly lowering of the reference and high outside temperatures.

The conducted simulations reveal that the MPC for the remaining weeks after week one consistently desires to lower the temperature, this occurs from approximately day 21 and continues during the remaining of the batch. The decreased temperature suggestion reveals that the lower constraint imposed on the MPC is activated as the temperature decrease is specified as 0.2 degrees °C lower than the implemented reference. The activation of the constraint effectively saturates the learning of the MPC and as presented during the

simulation results will it be possible to lower the temperature further during the next batch and thus improve further. This learning process is proven to be functional during simulation, however, to improve the ability of the MPC to reject live disturbances could it make sense to cast the constraint in a different manner.

During the test it was possible to update the temperature curve four times, this was however not utilized. During week one was there no desire to update the temperature reference as the simulation yielded the same result as initially found. During week two was the desire to lower the temperature first occurring on day 21, it was therefore postponed to the following week. During the remaining two weeks where the large deviations in temperature encountered and as the climate control struggled was it assessed that the implementation of a 0.2 degree °C lower temperature would not have the desired effect.

The result of the live test generally reveals that the final weight during week four and five is too high compared to the desired reference. This weight increase and thus also the struggle to maintain the reference could be caused by different factors. The weight reference obtained from the broiler handbook [17]. This reference is selected such that a realistic reference is utilized, however, could it be the case that this reference is too conservative and it, therefore, required excessive temperature changes to retain the reference which is made inaccessible due to the imposed constraints. It is also possible that the model utilized internally in the MPC is too simple and it is necessary to incorporate additional control variables in order to manage the task of reference tracking by the use of climate parameters.

The test is in general inconclusive as multiple factors have affected the final result, this should be seen in combination with the large disturbances encountered. These have thereby further complicated the analysis of the effect imposed by the potential temperature changes suggested by the MPC. The MPC has in general shown that it acts as anticipated, it is however not possible to verify the impact of this behavior by the obtained test results. The handling of the short-term objected is therefore partially verified as the MPC suggested temperature changes, these were however not implemented. The long-term learning objective is not verified by the live test as it requires running additional batches, this objective is only verified during the iterative simulations in the adaption phase.

Part III

Discussion and Conclusion

Discussion 6

The approach taken to model the growth of a broiler by the use of climate parameters resulted in both an integrator model type and an unstable model type. The parameter estimation of these models is conducted with the use of measured data. The ideal operation of a broiler house is achieved by following the smooth descending temperature reference closely, due to this can it be argued that the data should be considered as steady state measurements. It is desirable to excite the system sufficiently during a typical parameter estimation process, this is however not possible due to the health concerns of the broilers. The usage of this data could thus lead to, that the optimal performance of a given model structure is not revealed. It is however not possible to utilize different data, but important to consider when judging the general performance of a given model structure. Instead of a procedure where a general model structure is casted and a data-driven parameter estimation is conducted to improve the model could another approach be based on constructing the initial model from a biologic point of view and thus rely on modeling the biologically related dynamics.

The performed live test revealed that the MPC output ripples undesirably. The MPC did, however, avoid infeasible solutions and thus remained in an operational state. The behavior occurred as the non zero disturbances where included. If any future live tests are to be conducted should this issue be addressed. Conducting an additional live test would also be beneficial in order to verify the controller performance, desirably under more normal circumstances. The outside temperature during the live test reveals that a temperature above 25 degrees °C is measured in 17 of the 34 days. A "summer day" is by meteorological standards categorized as a day where the temperature national-wide is above 25 degrees °C. During an average year in Denmark are approximately 10 "summer days" present, the measured outside temperature cannot be categorized as an official measurement but illustrates that the warm weather is rather unusual [24].

The current model utilized internally in the MPC is based on a linear model structure with a limited number of in- and outputs. The simple structure allows the model to capture the overall process dynamics and eases the computational costs due to the limited complexity. A drawback of such a model is the possibility that the simple linear model structure limits the capabilities of the controller. An indication of this effect can be the reason for the results obtained in Section *5.1: Simulation* where the simulation never reaches fit percentages higher than approximately 97.5%, which is beneath the capabilities of the more advance RNN model. A more advanced model structure might be capable of converging to 100% or at least improve the obtained fit percentage further. Utilizing an advanced and possibly nonlinear model structure requires consideration of the implementation process. This is necessary as the implemented MPC structure only handles linear models, it is thus necessary to linearize any nonlinear model. The linearization will change the operating point of the system during the process. This linearization could be conducted online in order to ensure a representative model is utilized at every time step. This procedure is deemed possible as the sampling time of the process is slow. Even

though a more advanced model might increase the fit percentage, will any disagreement between the system model, used for the controller, and the RNN model, used as the plant result in a deviation of the fit percentage.

The current implemented MPC design features a set of constraints imposed on the control variable. These constraints are casted as a limitation on the change in temperature, $\Delta U_k(t)$ and are implemented as constraints from batch k to batch $k+1$. As $\Delta U_k(t)$ is not constrained from time t to $t+1$, is the system allowed to start rippling uncontrolled throughout the curse of multiple batches, as seen in *Figure 5.5* on page 60. As this is not a desirable behavior will a constraint imposed on the system that attenuates these ripples be desirable. An additional constraint, implemented as a maximum temperature change from time t to time $t+1$, could limit the ripples in a more suitable manner, while still allowing the controller to iteratively learn from batch to batch. Furthermore, could this constraint be made time variant, by increasing the allowed temperature variance from t to $t+1$ as the batch progresses and the broilers grow to become less receptive to temperature variations. Furthermore, cloud the interval specified by the original constraints be increased such that the batch-wise learning can be progressed quickly as the temperature variations concerning broiler health are handled by the new constraint.

Future work within the subject of optimizing the broiler production could be based on the previously mentioned considerations, but other aspects could also be included. The developed structure is centered around temperature as a manipulated control input and the broiler weight as the outcome. This model could be improved by including more manipulated variables and possibly in combination with more advanced model structures. Another option is to change the control objective from the broiler weight to the feed consumption and especially the feed conversion ration (FCR) which indicates the efficiency of the broilers. Considering this performance metric instead of the broiler weight has a number of positive aspects. One of the largest expenses in broiler farming is the cost of feed [25]. Improving the FCR will, therefore, reduce the production cost and utilize the resources in the best possible manner. As the broiler weight is connected to the feed intake will it additionally be possible to affect the broiler weight by feed and FCR control. Furthermore is the measurement of feed intake much more reliable as these are not biased in any kind compared to the weight measurements which rely on the average broiler visiting the scale.

Conclusion 7

The work conducted during this thesis is focused on the development of a model described broiler growth based on climate parameters and trajectory tracking control. The modeling incorporates five different structures where multiple parameter estimations have been conducted in a Monte Carlo fashion. The reference tracking control structure has been specified for a batch environment and incorporates a Kalman filter for state estimation. The control structure has been implemented and tested in a simulation environment and additionally applied on a broiler house where a live test has been conducted.

Modeling
The broiler growth has been described by the use of first-order linear state-space models which are based on inputs from climate parameters and include climate based disturbances. The modeling has been attempted by the use of both an integrator based model and an unstable model structure. Parameter estimation for both structures has been carried out with the use of measurement data originating from ten different broiler houses. The parameter estimation has been conducted by dividing the data into four clusters and a set containing all data. Furthermore, have these been divided into training and test sets by a ratio of 70% and 30% respectively. This estimation process has been conducted in a Monte Carlo fashion and the best models have been validated with additional data. During the modeling process has it been determined that the unstable model structure is superior in comparison to the integrator structure. The unstable model structure achieved validation fit percentages of 90.06 % and 88.71 % with and without feedforward respectively.

Controller
The control structure has been based on MPC, this is however adapted to operate in batch-wise fashion and elements from ILC are incorporated into the design. The batch operation of the MPC has been achieved by shrinking the prediction horizon according to the batch progress such that the prediction horizon always corresponds to the remaining batch duration. The internal cost function of the MPC has been specified as the tracking error of the desired weight trajectory, based on the developed model. The internal model utilized within the MPC has been formulated such that it describes the tracking error trajectory based on batch-wise changes of the input and disturbances. The incorporated elements from ILC has introduced two-time scales and thus two control objectives have been considered. The short-term control objective has been specified as the minimization of the tracking error during a batch, where the long-term control objective has been specified as the minimization of the tracking error across batches. In addition to the MPC has a Kalman filter been implemented as this allows to estimate the complete error trajectory. The Kalman filter has been implemented such that it estimated the shrinking error trajectory corresponding to the prediction horizon specified within the MPC.

Simulation and live test
The control structure has successfully been implemented in a simulation environment and the weighting matrices within the MPC have been tuned to obtain the optimal performance. The simulation has been conducted across 90 batches and during these

iterations has it been proven that the control structure minimizes the tracking error. It has furthermore been proven that the error is constantly minimized as the fit percentage increases across batch iterations which indicates that the long-term control objective has been fulfilled. In addition to the achieved simulation results has a live test been conducted. The conducted live test stretched across 34 days and at the end of each week, it has been possible to update the implemented temperature reference. The live test has revealed some undesired behavior by the MPC, it has however been solved by the implementation of a moving average filter such that the suggested temperature changes have been implementable. Throughout week one and two is the measured weight lower than desired, the gap is however reducing during week two. During week three is the measured weight almost identical to the desired reference, however, are the broilers growing significantly faster and thus is the measured weight increasing above the reference during week four and five. The live test is affected by large temperature variations and high outside temperatures. Due to these external disturbances is the temperature reference at some time instances during the batch specified lower than the ambient temperature. The climate control system is therefore insufficient for retaining the desired temperature. Due to the large disturbances are no updates to the implemented temperature sequence made. The capabilities of the controller are therefore not truly tested and verified by the live test as the test remains partly inconclusive. The implemented temperature reference is however based on the conducted simulation. This reference achieved a deviation of 78 grams between the desired and measured weight reference and a deviation of 40 grams between the desired and the slaughter weight.

Part IV

Appendices

Cluster Search A

In this appendix, the cluster search algorithm is explained. Furthermore, it is considered how to choose an appropriate number of clusters.

A.1 *K*-means Algorithm

The batches in the data provided by SKOV A/S are of different length and in order to allow estimation based on these, Matlab requires batches to be of the same length. To solve this all batches are shortened to fit the length of the shortest batch, thus the last samples of the longer batches are lost. In order to use more data during the estimation, it is decided to split up the data into clusters. A cluster search algorithm finds a number of center points and separates the data points according to the closest center point, where each data point is representing a batch. The cluster search is done using the Matlab function $kmeans(\mathbf{Z}, k)$ where $\mathbf{Z}$ is the data matrix and k is the number of clusters. The underlying algorithm is called the k-means algorithm [26]. More specific the algorithm consists of the following five steps:

1. Place k initial centroids.
2. Compute the distance from all data points to all k centroids.
3. Assign all data points to their closest centroid.
4. Compute an average for all data points within each of all k clusters and assign this average as the new k centroid position.
5. Go through step 2 to 4 until the centroids position converges.

The Matlab function `kmeans(Z,k)` leaves multiple options for the user to decide which setting the algorithm should use. Placing the initial centroids in step 1, the placement can be specified in multiple ways. In the case of this project, the initial centroid position is chosen randomly from a uniform distribution within the range of $\mathbf{Z}$. Computing the distance from data points to centroids, step 2, can be done by different distance measures as squared euclidean distances or sum of absolute distances. Because all variations, both positive and negative, in batch length are undesired the squared euclidean distance is chosen. The distance from one data point to a centroid is calculated as seen in *Equation: (A.1)*:

$$d(\mathbf{z}, \mathbf{c}) = (\mathbf{z} - \mathbf{c})(\mathbf{z} - \mathbf{c})^T \tag{A.1}$$

Where

	$\mathbf{z} \in \mathbb{R}^{(1\times n)}$	is a data row of $\mathbf{Z}$,	$[\cdot]$
	$\mathbf{c} \in \mathbb{R}^{(1\times n)}$	is a centroid row vector,	$[\cdot]$
and	n	is the row length.	$[\cdot]$

The last option used decides the number of replicates which makes it possible to repeat the cluster search using new initial centroids. The cluster search is repeated 50 times to ensure that the optimal centroid placement is found [27].

A.2 Choosing Number of Clusters

As mentioned earlier the algorithm works by placing centroids and then assigning the data points to the closest centroid.

The more clusters that are used in this process the more data will be used, but as explained in Section *2.3: Individual Estimations* for each cluster an estimation must be made, and therefore each cluster result in a model with a new set of estimated model parameters. Furthermore, if e.g a cluster is made for each dataset, the estimations are based on only one batch, which would likely result in a model which is bad at reflecting other behavior than that of the batch, originally used to create the model. All of these arguments imply that an optimal number of clusters can be found.

There are no perfect methods to choose the number of clusters, but according to Andrew Ng, Adjunct Professor at Stanford University, there are three general approaches [26]. The first is to look at visualizations of the data. It might be clear that data is split into groups. In the case of this project, no clear separation in the batch length can be seen, see *Figure A.1.*

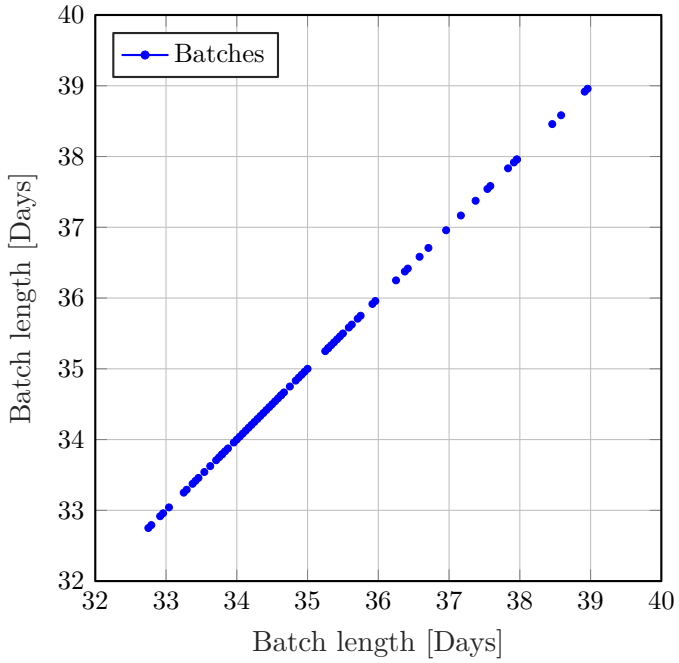

Figure A.1: A plot of the length of all 161 batches to illustrate how they are distributed.

The second approach is to consider the cost or sum of distances from all data point to the centroids. If the cost declines quickly by adding the first number of clusters but the cost then converges as more clusters are added it might be clear what the appropriate number of clusters is. This is called the "elbow effect" as a plot of the cost would look like an elbow. However often the curve is a lot softer and it is not possible to clearly see an elbow joint.

The third approach is to choose based on a physical property. An example of this is to determine how many different t-shits sizes to produce based on data describing the size of a population. Here it is clear that one size might not fit everyone and 7XS to 7XL, a total of 17 different sizes, might be an overestimate of the necessary range of sizes. It should be stressed that this approach does not give a clear answer of how many clusters

to chose. However, it might give an idea of which magnitude the number should be.

For this project the same considerations are relevant. It is hard to specify a final optimal number of clusters. However, the batches are of different length and this could be caused by some measurements halted prematurely and some measurement continued even though there are no broilers within the house. Therefore a minimum of three clusters will ensure that all batches with too few or too many samples are placed in two of the clusters and therefore the third cluster might have a better chance of resulting in a realistic model.

In *Figure A.2* the sum of distances is plotted as a function of the number of clusters, as the orange graph. However, in the case of this project, it is also relevant to consider the average number of batches in each cluster, as an estimation based on very little data would likely result in a model with bad overall behavior. The number of batches per cluster is plotted as the blue graph.

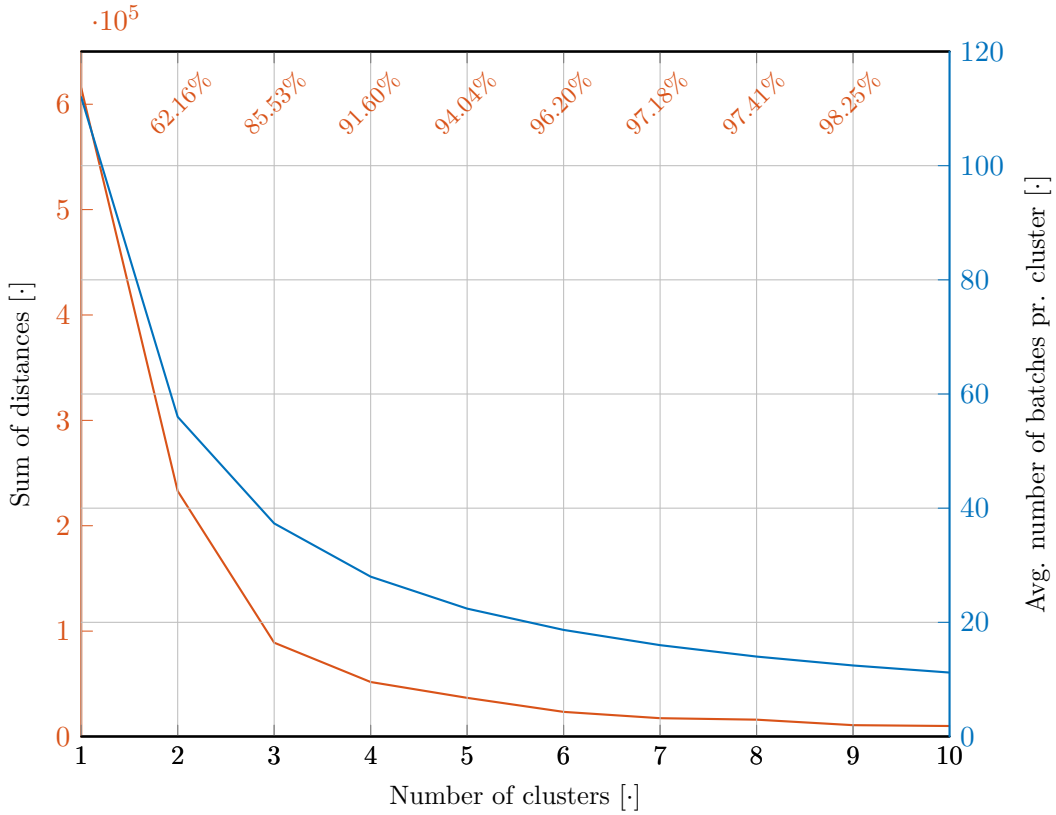

Figure A.2: The left y-axis is the sum of distances from all points to the centroids, the right y-axis is the number of batches per cluster and the x-axis is the number of clusters, using all 161 batches setting aside 30% as test set. Decrease in the summed distances, as clusters are added, is expressed in percentage as the orange number in the top row of the graph.

As it is seen in*Figure A.2* the cost decades a lot by adding the first clusters, cluster two and three. The benefit of adding cluster four and five are not as severe. Considering the number of batches per cluster, three clusters result in 37 batches per cluster and four clusters result in 28 batches per cluster.

All in all, it is a matter of judgment, however choosing four clusters will reduce the sum of distances by 91.6% and still on average leave 28 batches per cluster. Therefore four clusters are deemed appropriate for this project.

A.3 Cluster Search Results

Using four clusters as determined in Section *A.2: Choosing Number of Clusters*, the cluster search yields the result seen in *Figure A.3*.

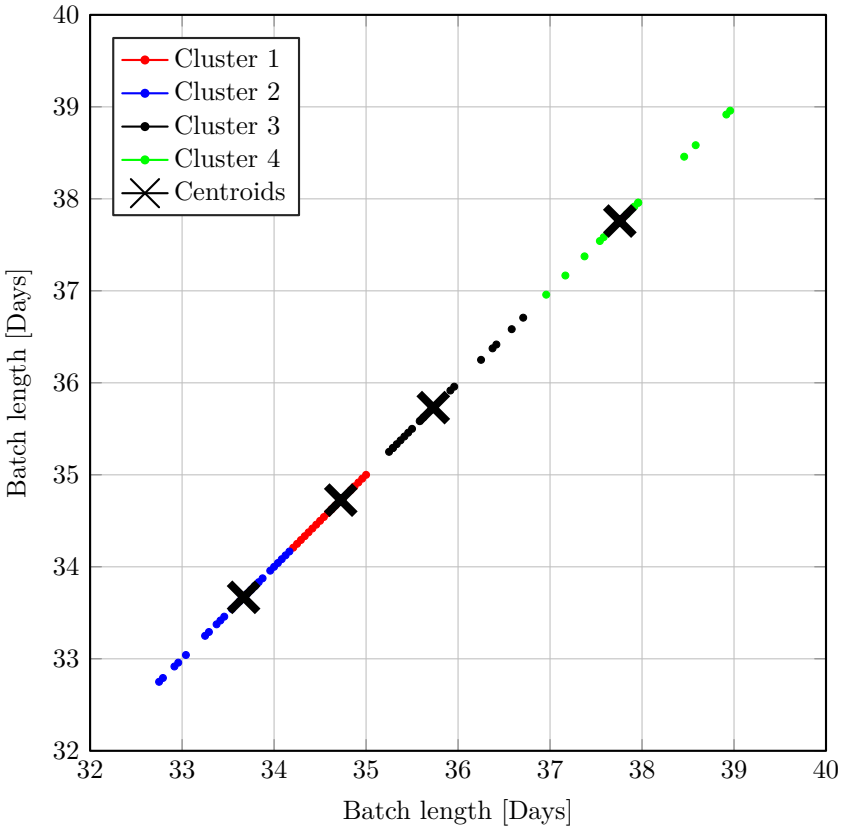

Figure A.3: The batches according to sample length now colored according to how they are split into clusters.

The cluster search is conducted on all 161 batches, described in Section *2.1: Data Description*, and afterward, the cluster is split into a test and a training set. This leaves an average of 28 batches in each training set.

The data is now split up into four clusters which will make it possible to estimate four new models.

Matlab Estimation Toolbox B

In this chapter, the grey box estimator used for estimation is described.

The estimation is conducted using the Matlab grey box estimator `greyest`. The grey box estimator is a linear based estimator that uses an `idgrey model`, an initial model that defines the model structure, and a dataset as input, in the case of this project the training set. The output is the model with estimated parameters. The initial model is the linear state-space model described in Section *2.2: Model Structure*. The output is the model with the estimated elements that yields the best performance on the training set.

It is beside the mentioned input and output possible to set options as a numerical search method, initialization choice of states and modeling disturbances using the function `greyestOptions` [28]. The initial states can be chosen in several ways. The `InitialState` setting contains the options `zero, estimate` and `Vector of doubles`. The option `zero` simply sets the initial states to zero, `estimate` works by handling the initial state values as another parameter to estimate and `Vector of doubles` makes it possible to set the initial states to a specific value. In this project the initial states are set as a random variable normally distributed with a mean of 40 grams and variance of 10 grams, using the `Vector of doubles` setting. This is done to get a positive broiler growth, as the output matrix is also chosen as one.

The search method is set using the setting `SearchMethod`, which has multiple options to chose between. Multiple of these search methods were tried and from this, it was clear that the search method called `fmincon` is the fastest for this specific estimation problem. Therefore `fmincon` is chosen to proceed with.

The search method `fmincon` is a gradient-based nonlinear solver, which works on problems where both the objective function and the first derivative of the objective function are continuous [29]. As described in Section *2.3: Individual Estimations* is the objective to minimize the difference between the measurements and the simulation. This is done by minimizing the sum of the squared error, which can be expressed as:

$$V(\hat{\boldsymbol{y}},\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i(\boldsymbol{\theta}))^T \boldsymbol{I}\,(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i(\boldsymbol{\theta})) \tag{B.1}$$

Where

	$\boldsymbol{y} \in \mathbb{R}^{(j\times 1)}$	is the measurements,	[g]
	$\hat{\boldsymbol{y}} \in \mathbb{R}^{(j\times 1)}$	is the simulation,	[g]
	$\boldsymbol{\theta}$	is the model parameter	[·]
	N	is the number of samples,	[·]
	$\boldsymbol{I} \in \mathbb{R}^{(j\times j)}$	is the identity matrix,	[·]
and	j	is the number of outputs.	[·]

Equation: (B.1) is a quadratic function and therefore is the derivative a linear expression, which means that both the objective function and its derivative are continuous. The

identity matrix in *Equation: (B.1)* could have been chosen as a weighting matrix if some outputs are deemed more important then others. However in the case of this project, there is only one output, the weight of the broiler, and therefore as $j = 1$ the identity matrix $\boldsymbol{I}$ will be reduced to unity.

Minimizing *Equation: (B.1)* with the use of `fmincon` leaves multiple underlying algorithms to choose between. The `interior-point` algorithm is chosen as this is recommended by Matlab [30]. As the name of the algorithm indicates, is the `interior-point` algorithm trying to approach the global minimum of the original problem from within the interior of the feasible set [31] [32]. To ensure that all iterations stray within the interior of the feasible set a barrier function is used. Furthermore the original problem is not solved directly but instead, an approximate dual problem is solved. For the constrained optimization problem all inequality constraints are embedded with a slack variable, resulting in improved feasibility.

Choosing Integrator Model Structure

C

In this section, the results of each estimation using the three different model structures is shown. Furthermore, the processes of deleting outliers are explained.

As described in Section *2.2: Model Structure* the general model structure is written as seen in *Equation: (C.1)*.

$$\begin{aligned} x[t+1] &= [A]x[t] + [B_{11}B_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] + [D_{11}D_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [D_d]w[t] \end{aligned} \tag{C.1}$$

The complete structure of the model leaves multiple choices of which of the model parameters to incorporate. As described in Section *2.2: Model Structure* the structure is chosen by conducting multiple estimations for multiple structures and then comparing how well the model structures fit the data. An estimation is done for the following three structures.

1. Estimation of only B and B_d, in other words without feedforward and leaving A and C as unity.
2. Estimation of B, B_d, D and D_d, thus leaving A and C as unity.
3. Estimation of A, B, B_d, D and D_d, with C as unity, to evaluate on the initial choice of the system being a pure integrator.

The output matrix (C) is chosen as unity for all three structures because estimating C will not add any information, as there is only one state.

The estimations are done as described in Section *2.4: Monte Carlo Estimations*, which results in five models for each structure. Afterward, the results are checked for outliers and if any are found they are removed. This is done because a part of the estimation is to randomize initial guesses, randomize division of training/test sets and because the estimation is a numerical process. Detecting and deleting outliers is done using the Grubbs test for outliers, this is also called maximum normed residual test [33]. Grubbs test is used for detecting single outliers in a dataset of only one variable. An iterative implementation allows detection of multiple outliers within the same dataset.

Grubbs test is a hypothesis test that checks if no outliers exist, by checking if a sample falls within a critical region with a specified significance level. The test calculates a ratio G, which indicate how far each sample is from the others, for each sample. Furthermore, a critical region, describing when a sample is an outlier, is calculated, see *Equation: (C.3)*.

The ratio G is calculated for each sample by:

$$G_i = \frac{|Y_i - \bar{Y}|}{\sigma} \tag{C.2}$$

Where

	G_i	is the ratio describing how far the i^{th} sample is from the others,	$[\cdot]$
	Y_i	is the value of the i^{th} sample,	[%]
	$\bar{Y}$	is the mean of all samples,	[%]
and	σ	is the standard deviation of the dataset.	[%]

The hypothesis that the i^{th} sample is not an outlier, is rejected if G_i is bigger then the critical region, seen in *Equation: (C.3)*.

$$G_i > \frac{N-1}{\sqrt{N}} \sqrt{\frac{(t_{\alpha/(2N),N-2})^2}{N-2+(t_{\alpha/(2N),N-2})^2}} \tag{C.3}$$

Where

	N	is the number of samples,	$[\cdot]$
	α	is the significance level,	$[\cdot]$
and	$t_{\alpha/(2N),N-2}$	is the critical value of the t-distribution, where the t-distribution is used to set a confidence interval.	$[\cdot]$

The used t-distribution is a continuous probability distribution closely related to the normal distribution and these will be almost indistinguishable at high degrees of freedom which for this implementation is given as two minus the sample size. The t-distribution can be interpreted as an estimation of a normal distribution. The t-distribution is often used with small sample sizes or if the standard deviation of the data is unknown. The Grubbs test assumes a normal distributed dataset, therefore this should be checked. In practices, the Grubbs test is conducted using an implementation by Brett Shoelson, [34], and a significance level of 0.05.

C.1 Structure 1: Estimate B

As previously described, the Grubbs test assumes that the data is normally distributed. Therefore this should be checked. In *Figure C.1* is the distribution of 500 estimations, for the four clusters and all batches at once shown. Furthermore, is a normal distribution fitted to the data and shown in red. Here it is seen that the results are close to normal distributed, as desired.

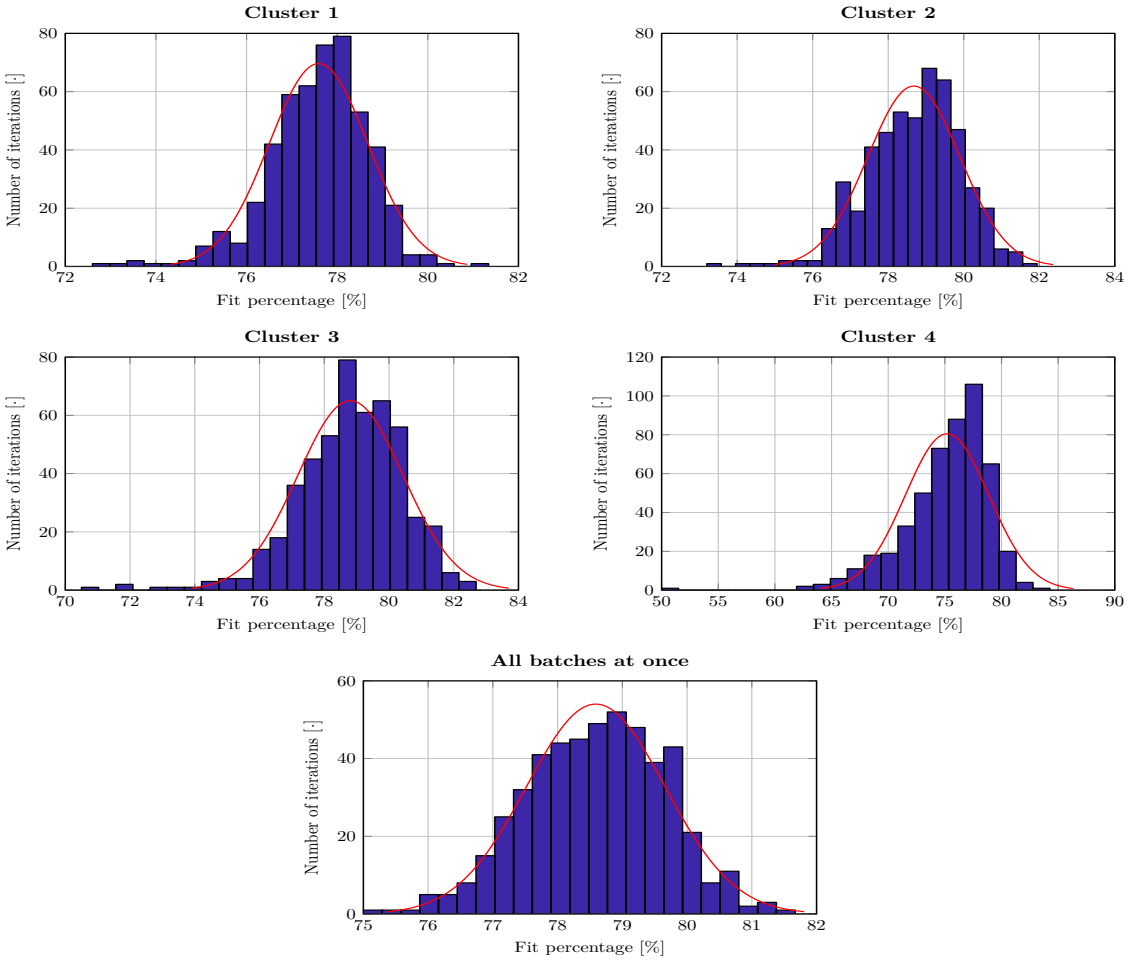

Figure C.1: The distribution of the estimations, when only estimating B. The x-axis shows fit percentage and the y-axis shows number of estimations within the histogram interval.

Now the Grubbs test is used to remove outliers. The Grubbs test for outliers is used in the estimations, leaving only the estimations not categorized as outliers. The remaining estimation results can be seen in *Figure C.2*.

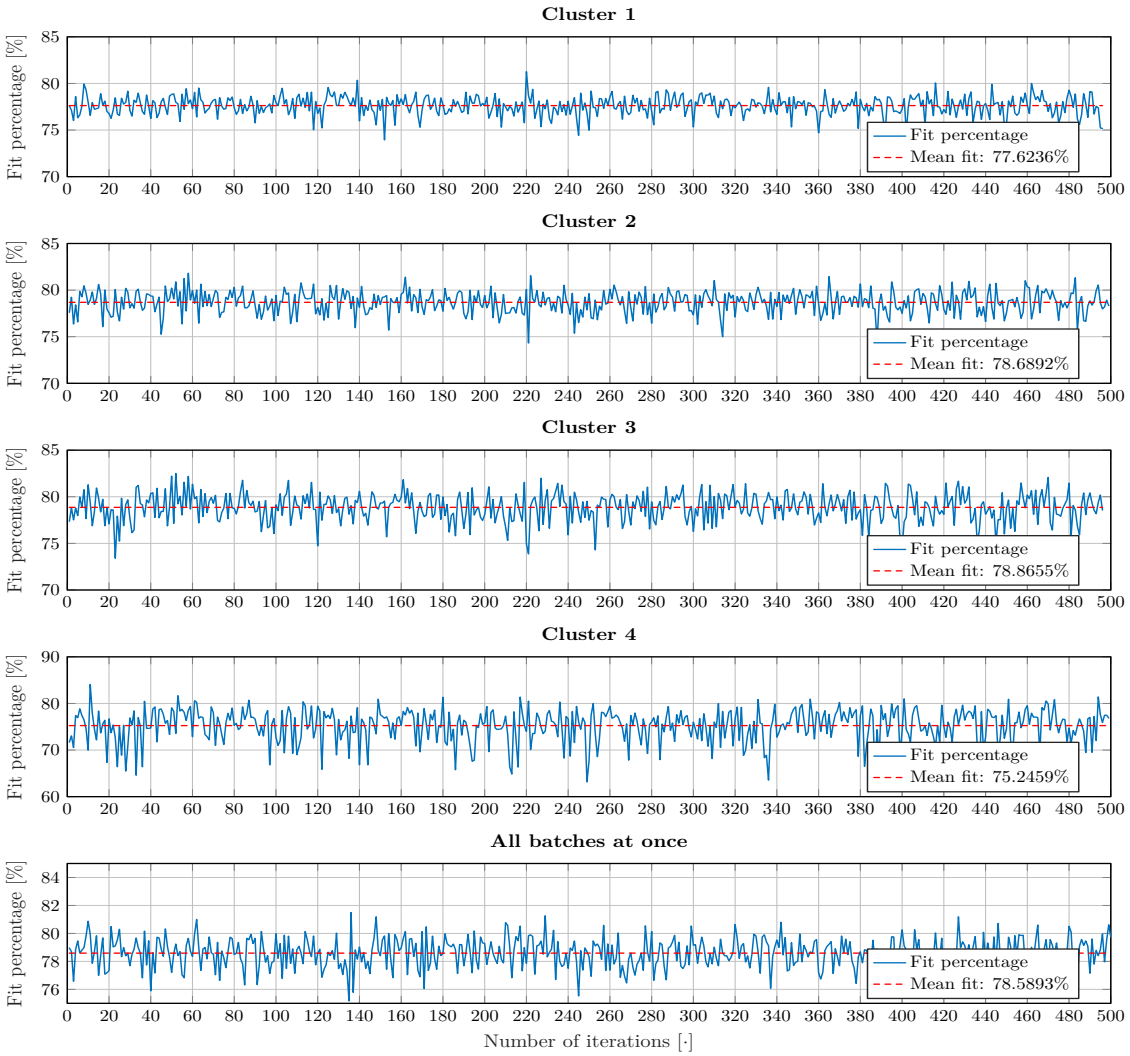

Figure C.2: The fit percentages of all estimations not classified as outliers and the mean fit percent for each of the five models.

For the models in *Figure C.2* is the number of removed estimations four, one, four, one and zero respectively.

C.2 Structure 2: Estimate B and D

When estimating both the input and feedforward matrix is the procedure the same. Therefore it is investigated if the estimation fit percentages are normally distributed and then outliers are removed using the Grubbs test for outliers.

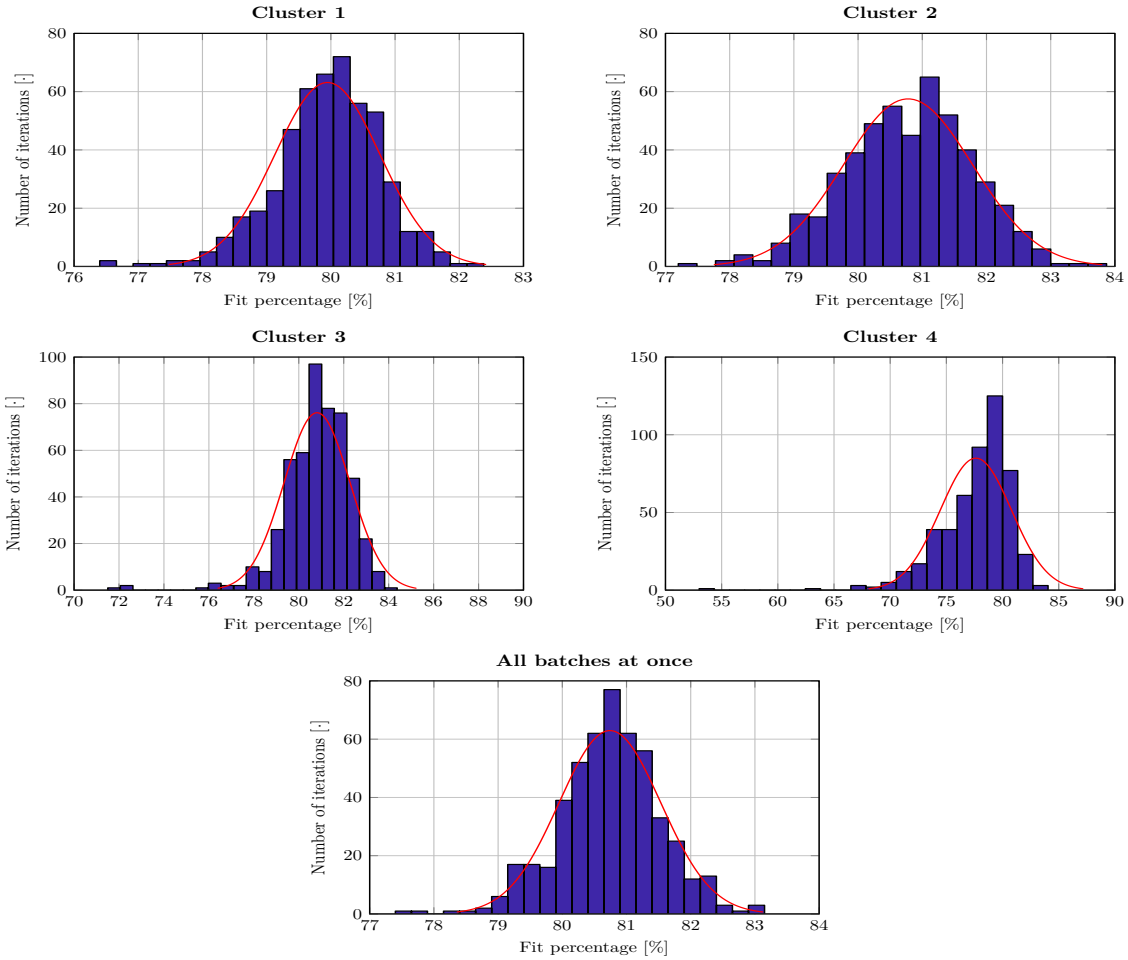

Figure C.3: The distribution of the estimations, when estimating both B and D. The x-axis shows fit percentage and the y-axis shows number of estimations within the histogram interval.

Figure C.3 shows the distribution of 500 estimations, for the four clusters and all batches at once. Here it is seen that the results again are close to normal distributed, as desired. The Grubbs test is again used to remove outliers. The remaining estimation results are seen in *Figure C.4*.

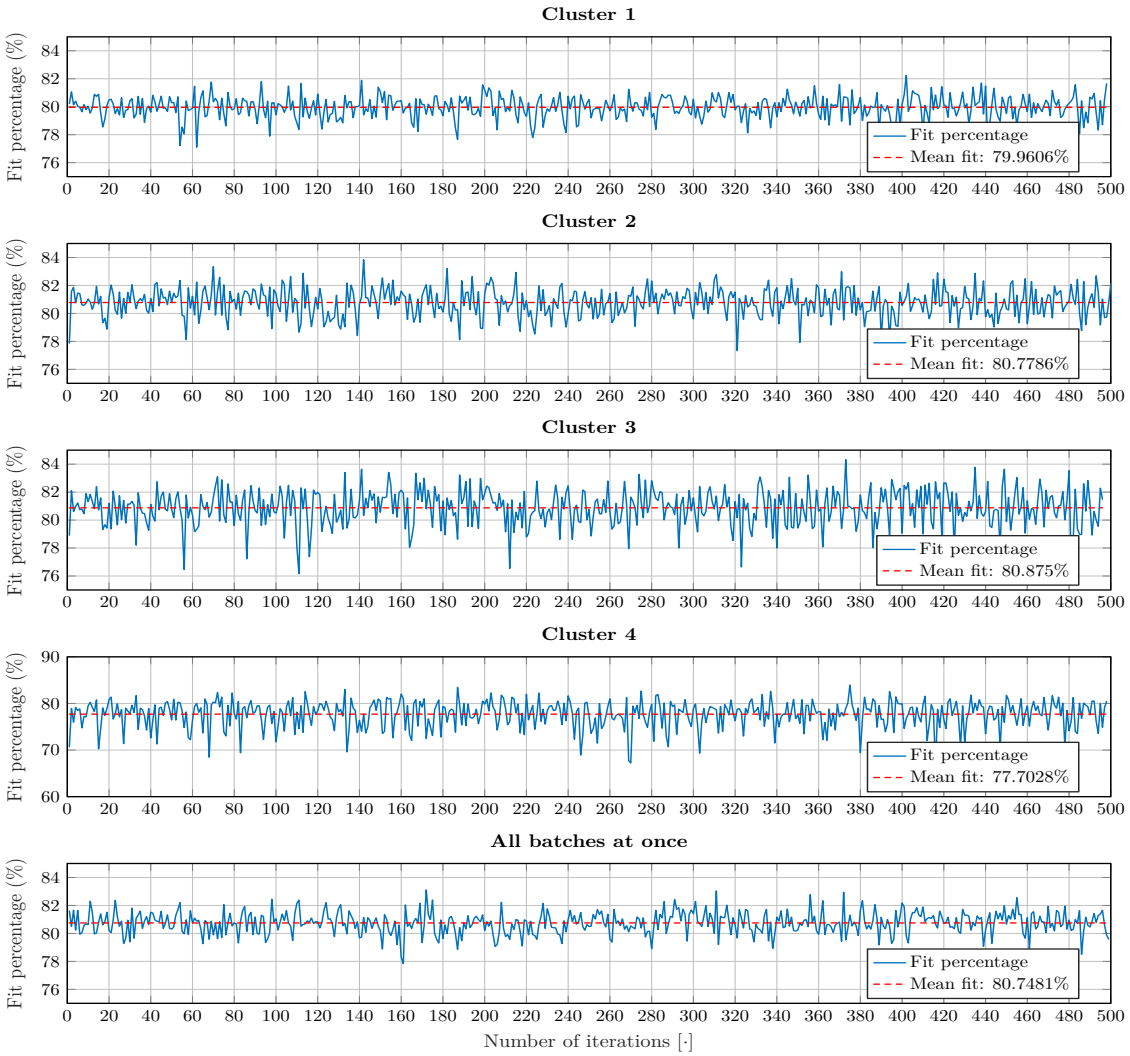

Figure C.4: The fit percentages of all estimations, not classified as outliers and the mean fit percent for each of the five models.

For the models in *Figure C.4* are two, zero, four, two and one estimations removed respectively.

C.3 Structure 3: Estimate A, B and D

When estimating both the system, input and feedforward matrix the procedure is the same. Therefore it is investigated if the estimation fit percentages are normally distributed and then are outliers removed using the Grubbs test for outliers.

The initial condition of the system matrix is set to a random variable in the interval $[0.01, 1]$. This is chosen as the output matrix is positive and then a positive system matrix will result in a positive growth of the broiler, which corresponds to the correct behavior.

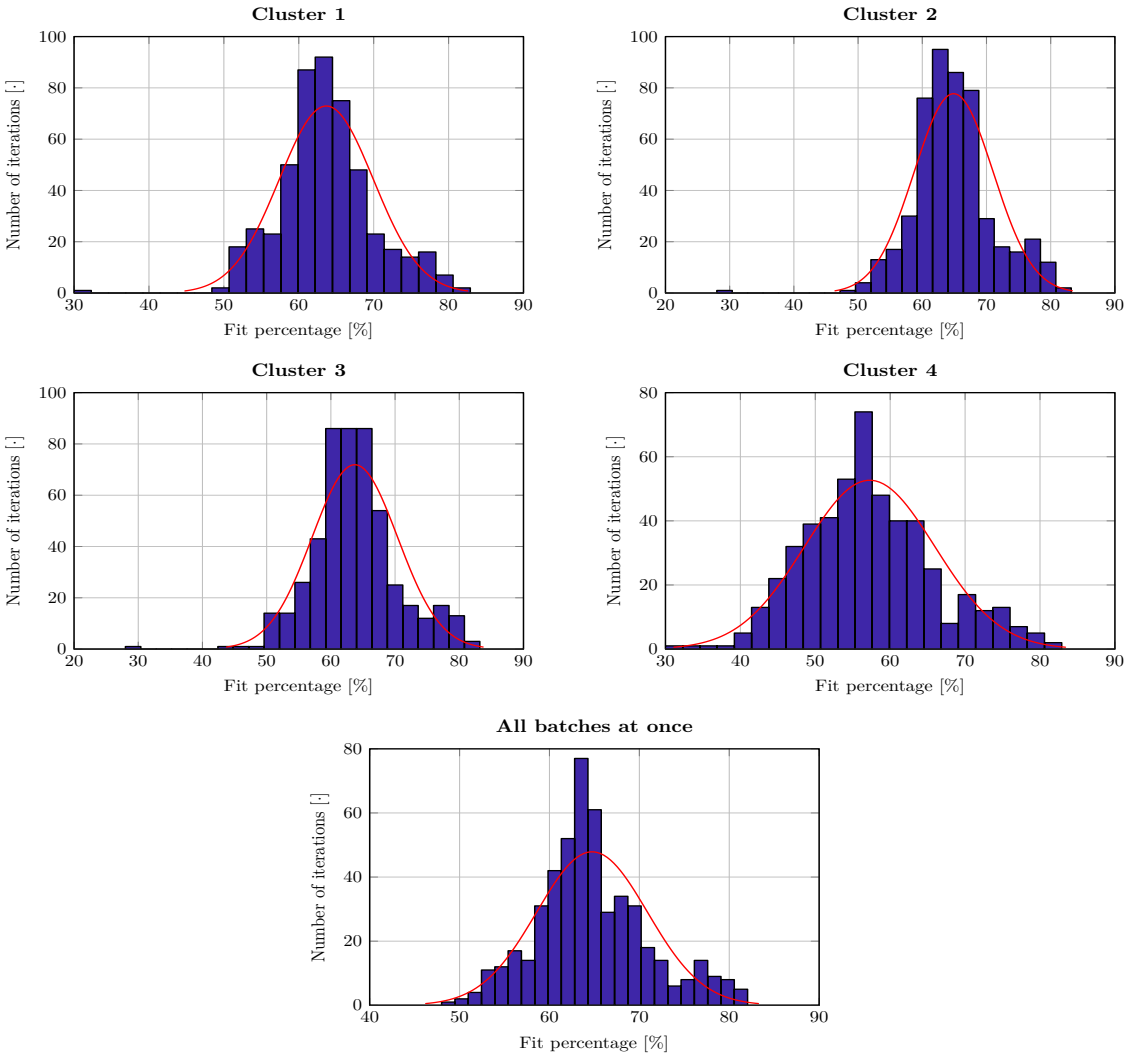

Figure C.5: The distribution of the estimations, when estimating both A, B and D. The x-axis shows fit percentage and the y-axis shows number of estimations within the histogram interval.

Figure C.5 shows the distribution of 500 estimations, for the four clusters and all batches at once. Here it is seen that the results again are close to normal distributed, as desired. The Grubbs test is used to remove estimations outside the specified significance level. The remaining estimation results can be seen in *Figure C.6*.

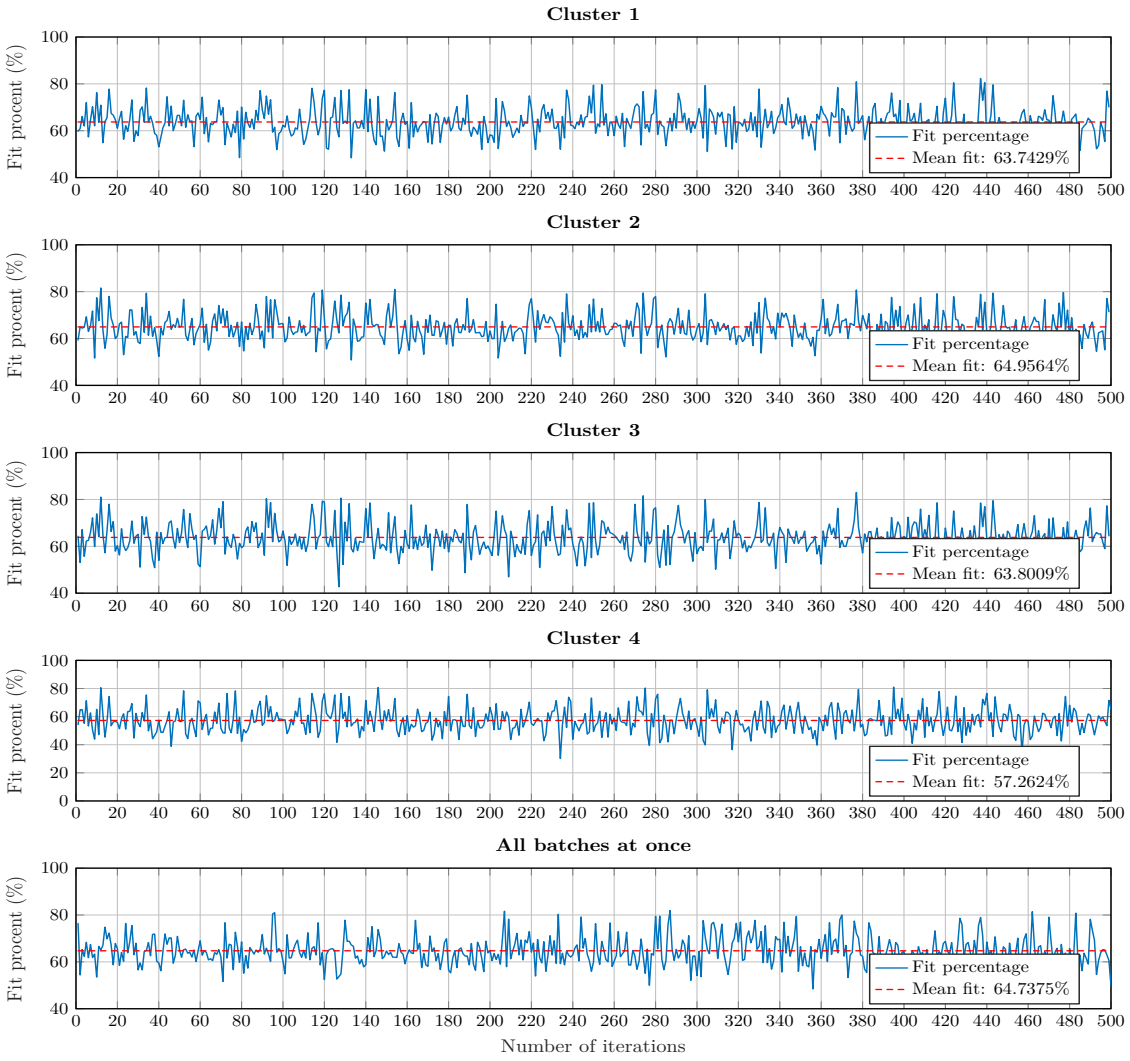

Figure C.6: The fit percentages of all estimations not classified as outliers and the mean fit percent of the five models.

For the models in *Figure C.6* are one, one, one, zero and zero estimations removed respectively.

C.4 Model Structure Conclusion

Estimating both the system, input and feedforward matrix resulted in fit percentages in the range of 57 to 65 %. Estimating only the input matrix resulted in fit percentages in the range of 75 to 79 % and estimating the input and feedforward matrix resulted in fit percentages in the range of 77 to 81 %.

From this, it can be concluded that the best model structure is model structure two. The better fit when using feedforward could be caused by the slow sample times, resulting in some of the dynamics of the input matrix being delayed.

Choosing Unstable Model Structure

D

In this section, the results of each estimation using two different model structures is shown. This section utilizes the exact same procedure as in Appendix *C: Choosing Integrator Model Structure*. As described in Section *2.2: Model Structure* the general model structure is written as *Equation: (D.1)*.

$$\begin{aligned} x[t+1] &= [A]x[t] + [B_{11}B_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [B_d]w[t] \\ y[t] &= x[t] + [D_{11}D_{12}]\begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [D_d]w[t] \end{aligned} \tag{D.1}$$

It is deemed desirable to investigate unstable version of the current model structure. Such models could improve the results found in Appendix *C: Choosing Integrator Model Structure* as they, with the same inputs, will allow a more exponential behavior due to their unstable nature. The instability of the model should be limited an thus only be slightly unstable as it otherwise could cause the system equations to blow up within a short time.

Finding the optimal pole placement of the system matrix is done with the help of the estimation toolbox. As only a slightly unstable model is desired, is the pole placement constrained to the interval [1.0001,1.100] within the z-domain. Based on this constraint, parameters for two model structures will be found by estimation.

The two model structures contain the estimation of:

1. A constraint to $[1.001, 1.100]$, B and B_d, thus leaving C as unity and D and D_d as zero.
2. A constraint to $[1.001, 1.100]$, B, B_d, D, and D_d, thus leaving C as unity.

The estimations are conducted as described in Section *2.4: Monte Carlo Estimations*, which corresponds to the procedure used in Appendix *C: Choosing Integrator Model Structure*. The method results in five models for each structure. The results are checked for outliers and if any are found they are removed. This is done because a part of the estimation is to randomize initial guesses, randomize division of training/test sets and because the estimation is a numerical process. Detecting and deleting outliers is done using the Grubbs test for outliers [33]. Grubbs test is applied in an iterative implementation thus allowing detection of multiple outliers within the same dataset. The full process is found in Appendix *C: Choosing Integrator Model Structure*. The results presented throughout this section are all treated with Grubbs test and do not include outliers.

D.1 Structure 1: Estimate A and B

The average fit for each of the five estimations is calculated and leads to a converging fit percentage. The converging fit percentages for each estimation is presented in *Figure D.1*.

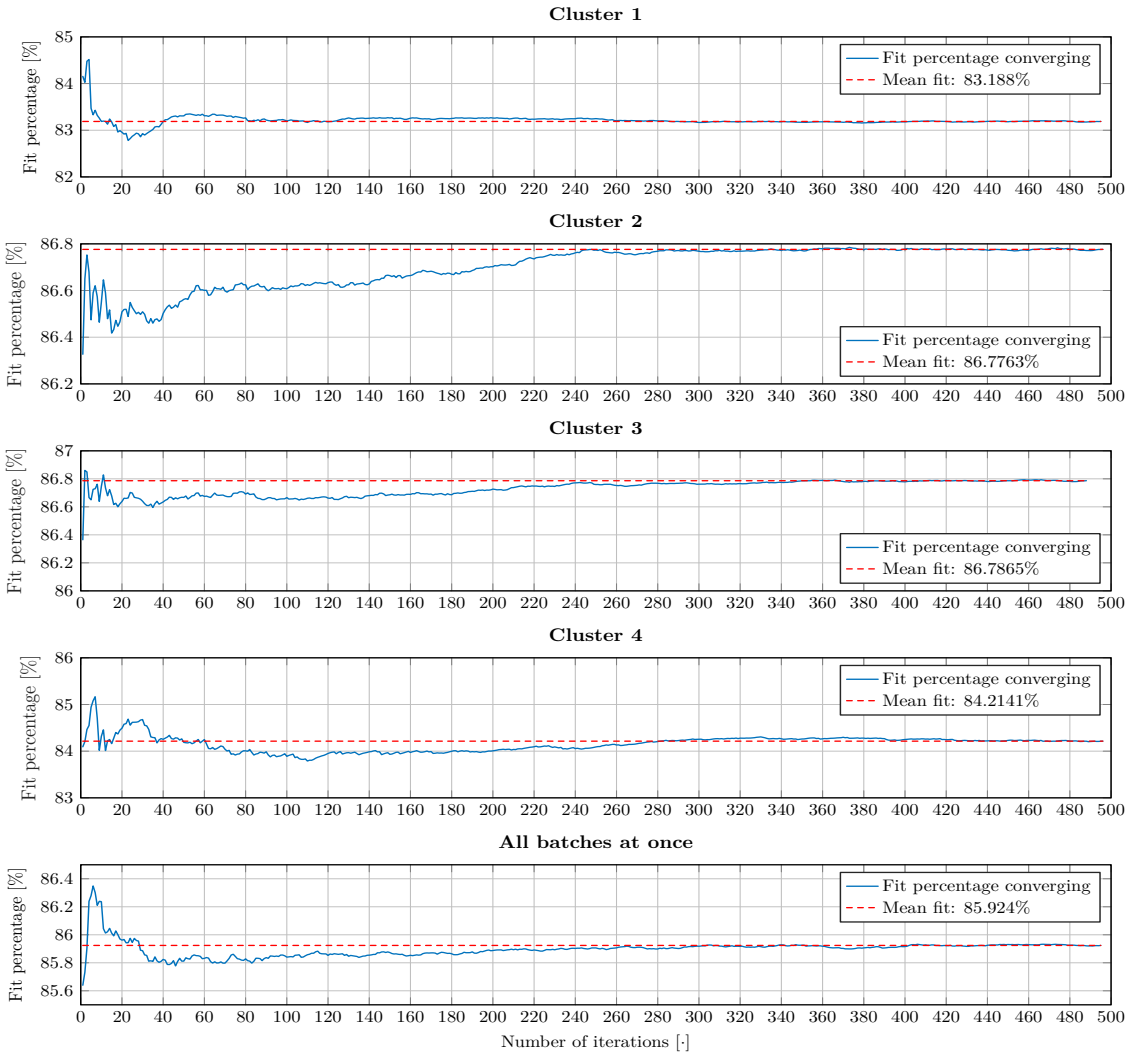

Figure D.1: Converging fit percentages for each of the five models estimated.

Across all estimations, the fit percentages have converged after roughly 300 iterations. The choice of running 500 iterations is reused from the previous procedure. The converging fit shows that the 500 iterations should be enough to reveal the full potential of the model structure, even with the randomized selections of training and test batches.

The results are, as the previous estimations, obtained by five estimations with each 500 iterations, resulting in a total of 2500 models. To determine which of the 2500 sets of parameters to select, the strategy used previously will be used again. From each of the five estimations, three models are selected. To recap the procedure, the models are selected as follows.

1. The parameter set, of all 500, that yields the highest fit percentage, during estimation.
2. The average of all 500 parameter sets.
3. The parameter set, of all 500, that yields the lowest fit percentage, during estimation.

Evaluating the performance of the selected 15 models is done by simulating each model with all 161 batches and comparing the fit. The performance of the 15 models across all batches is shown in *Figure D.2*.

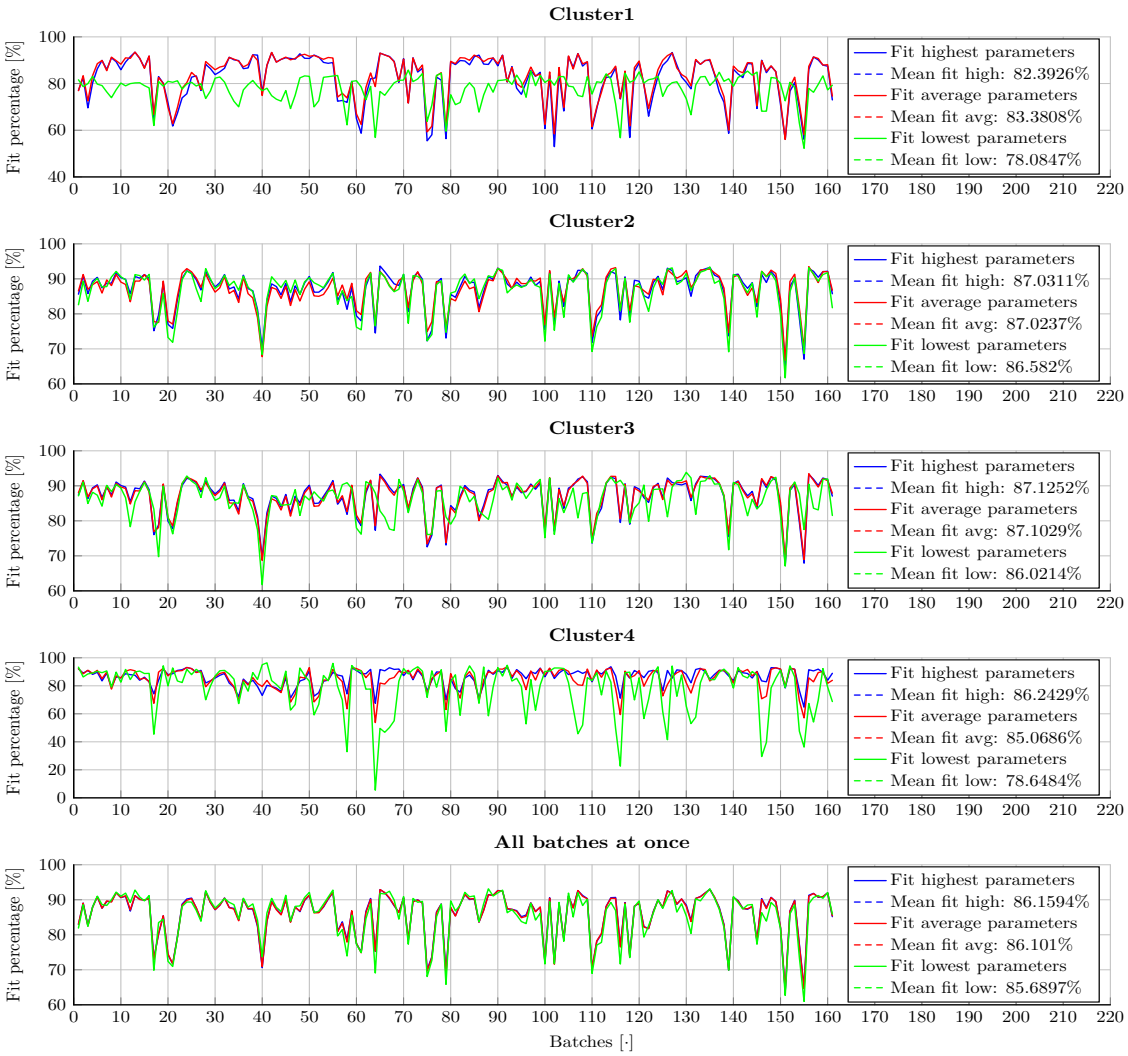

Figure D.2: The fit percentages of the three different models for all five estimations without outliers.

The comparison reveals that the highest overall fit, 87.12%, is achieved by the model that is based on the parameters that yielded the highest fit during estimation from cluster three. The overall second best model yielding a fit of 87.10% is based on an average of all parameter sets from cluster three. The third overall best model yields a fit of 87.03% and is based on the parameter set that provided the highest fit during estimation from cluster two.

Further investigation of the best performing models is conducted by comparison of the parameters throughout the 500 estimations. Parameters contained in the three best models based on average parameters are compared. These are named model A to C and originate from the *average parameters from cluster 2* with a fit of 87.02%, *average parameters from cluster 3* with a fit of 87.10% and *average parameters from all batches* with a fit of 86.10%, respectively. The compared parameters are shown in *Figure D.3* where each element in the system, input, and disturbance matrices are shown, according to iteration number and originating model.

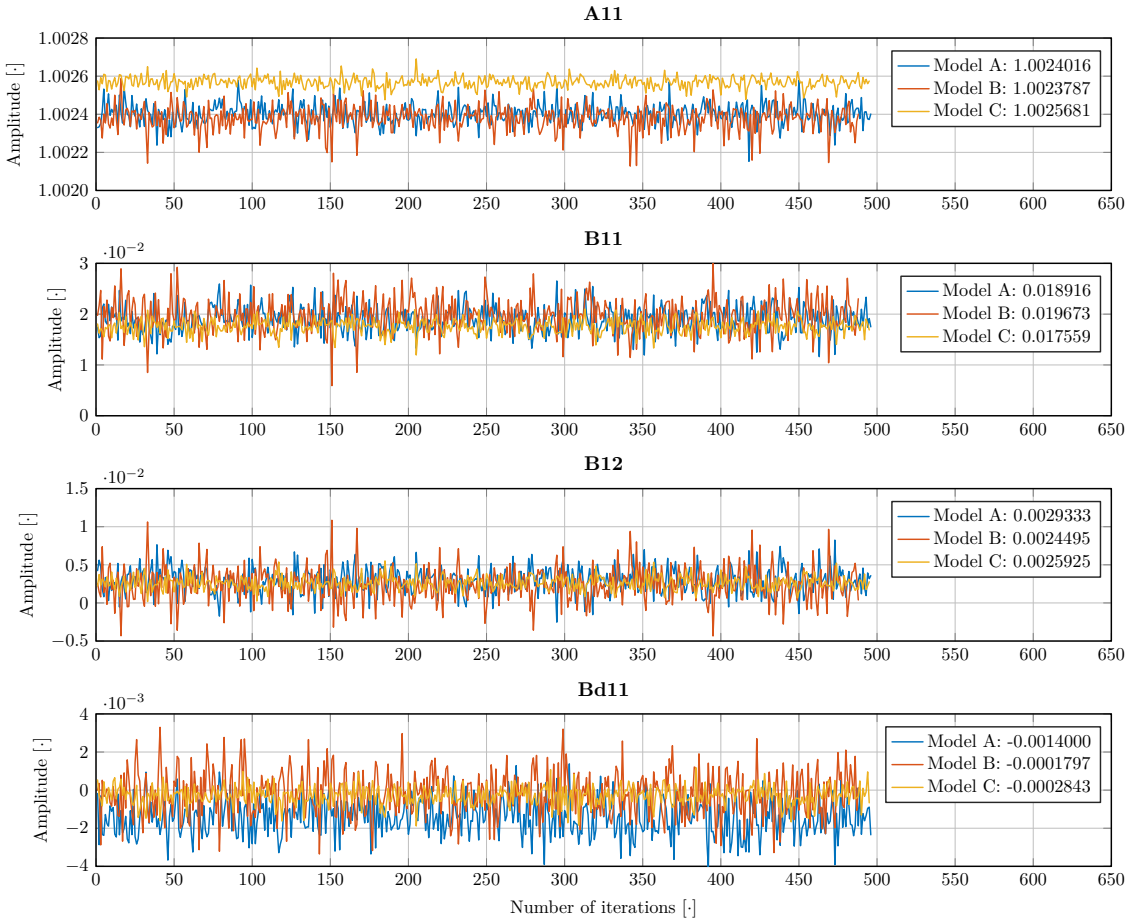

Figure D.3: The parameters for all iterations, from the three best performing average models. The value shown in the legends is the mean of all iterations.

The average value of the parameters across the three model is within a small span, thus the models obtain similar fit percentages. The variations of the parameters for each model throughout the 500 estimations seem to differ. The parameters of model B vary the most whereas the parameter variation of model C is the smallest. Despite this fact is model C still the worst performing of the three models. Overall are the three models behaving similarly and they all agree on the sign of each parameter.

The overall best performing model is based on the parameters from cluster three that yielded the highest fit. This model is named model D and the parameters are:

$$\begin{aligned} x[t+1] &= [1.0024]x[t] + [0.019900 \;\; 2.6000 \cdot 10^{-3}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-2.9221 \cdot 10^{-4}]w[t] \\ y[t] &= x[t] \end{aligned} \tag{D.2}$$

The overall third best performing model is based on the parameters from cluster two that yielded the highest fit. This model is named model E and the parameters are:

$$\begin{aligned} x[t+1] &= [1.0024]x[t] + [0.019100 \;\; 3.2000 \cdot 10^{-3}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [1.0000 \cdot 10^{-3}]w[t] \\ y[t] &= x[t] \end{aligned} \tag{D.3}$$

Comparing the parameters of model D and E with the parameters of the average based model shown in *Figure D.3* shows that they feature parameters with similar values. The only greater difference is that model E has a positive $Bd11$ entry, whereas the other investigated models contain a negative value. As the fit percentages of model B and E are very similar is model E discarded and model B is kept due to the sign structure being in line with the other compared models.

A comparison of the three overall best performing model for this structures, that is model A, B, D, is conducted and each model is simulated with an average of all batches. This comparison is shown in *Figure D.4*

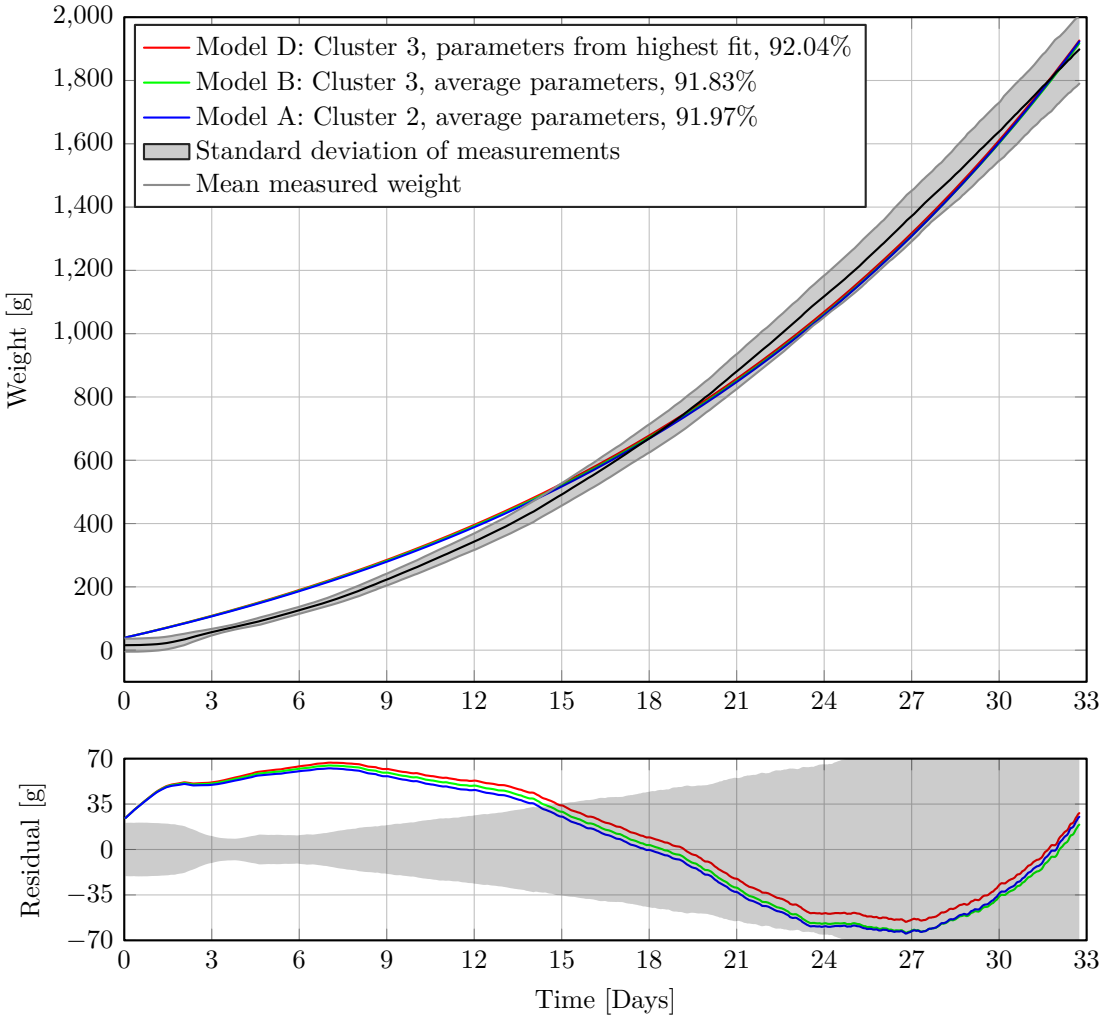

Figure D.4: The mean of all batches, the standard deviation, and the three best models. A relative short batch period is represented, as a mean for all batches only exits for the length of the shortest batch.

The simulation shows similar performance across the three models, but model D performs slightly better and is thus stated as the resulting model for model structure four.

D.2 Structure 2: Estimate A, B and D

The average fit for each of the five estimations is calculated and leads to a converging fit percentage. The converging fit percentages for each estimation is presented in *Figure D.5*.

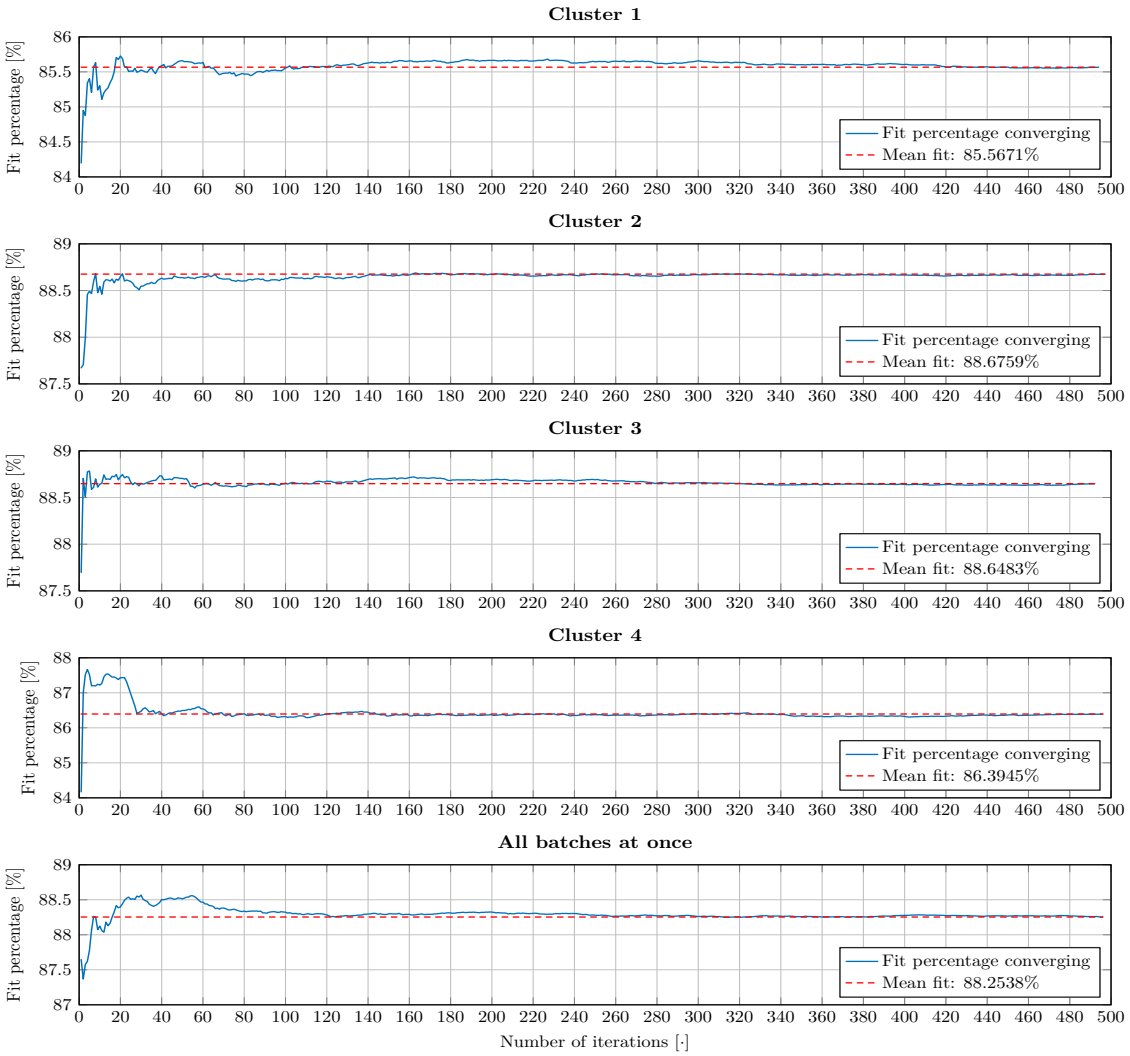

Figure D.5: Converging fit percentages for each of the five models estimated.

To determine which of the 2500 sets of parameters to select, three models are selected from each of the five estimations and compared. To recap, the models are selected as follows.

1. The parameter set, of all 500, that yields the highest fit percentage, during estimation.
2. The average of all 500 parameter sets.
3. The parameter set, of all 500, that yields the lowest fit percentage, during estimation.

The performance evaluation of the 15 selected models is conducted by simulating each of the models with all 161 available batches and comparing the obtained simulation fits. The following simulations are conducted with results where outliers are removed. The performance of the 15 models across all batches is shown in *Figure D.6*.

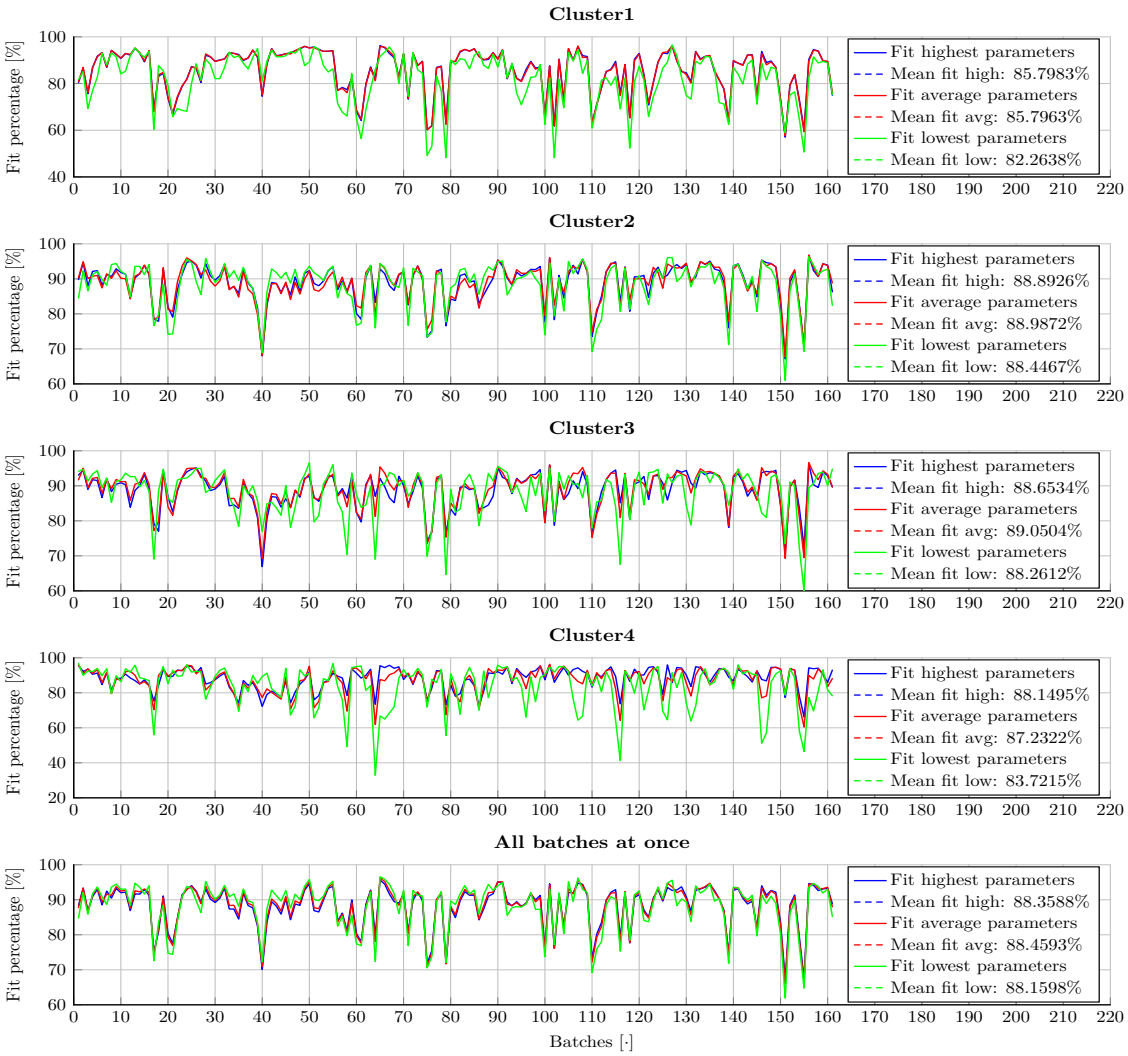

Figure D.6: The fit percentages of the three different models for all five estimations without outliers.

These results have shown that the overall highest average fit, 89.05%, is obtained by the model that is based on an average of all parameters from the 500 models belonging to cluster three. The second overall best model with an average fit of 88.98% is obtained by a model based on average parameters from cluster two. The third overall best model with an average fit of 88.89% is obtained by the model that had the highest fit within the estimations from cluster two. The third best model based on average parameters is based on the estimation of all data and achieves an average fit of 88.45%.

Further investigation of the best performing models is conducted by comparison of the parameters throughout the 500 estimations. Parameters contained in the three best models based on averaging parameters are compared. These are named model A to C and originate from the *average parameters from cluster 2*, *average parameters from cluster 3* and *average parameters from all batches* respectively. The compared parameters are shown in *Figure D.7* where each element in the system, input, feedforward and disturbance matrices are shown, according to iteration number and originating model.

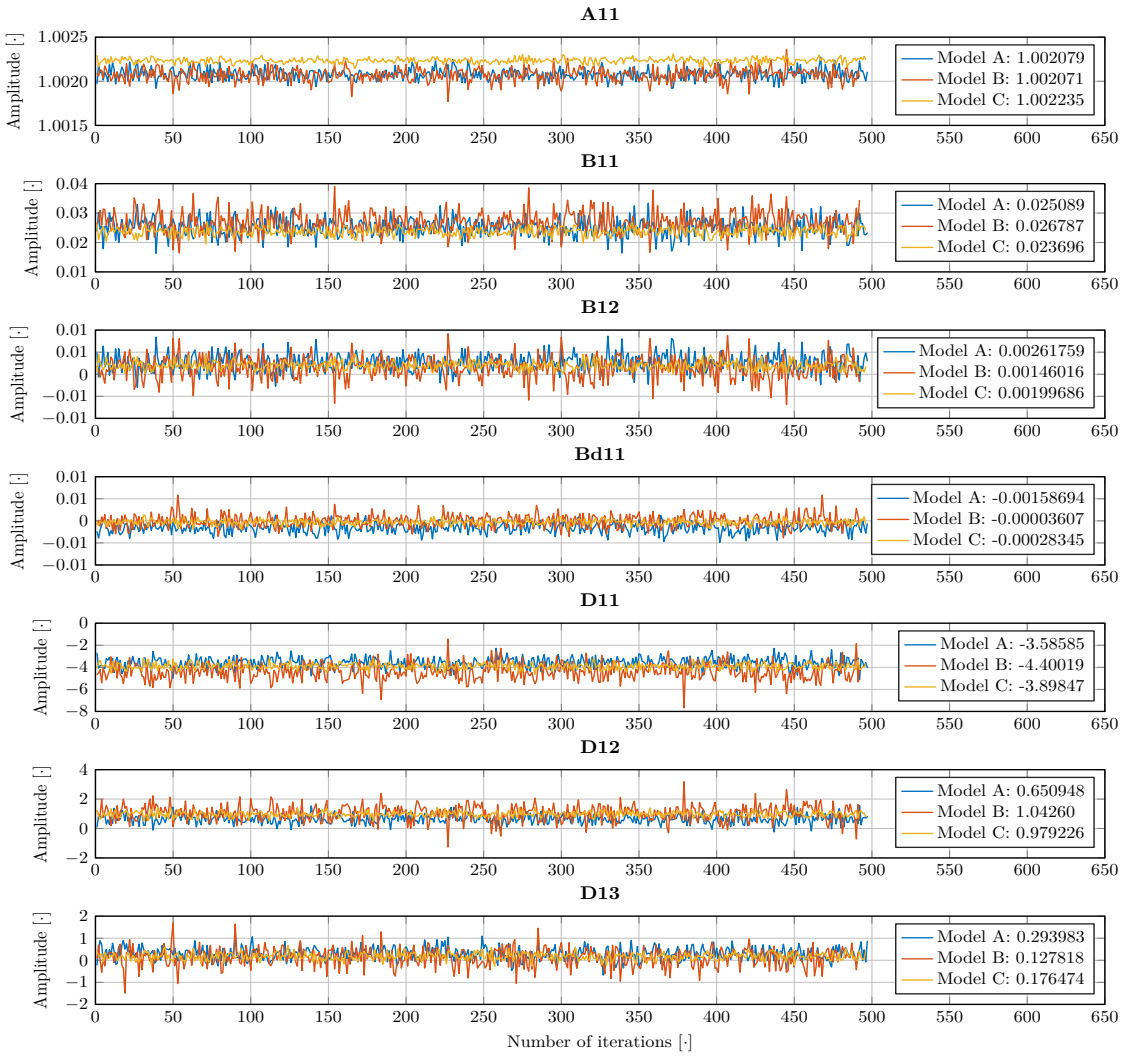

Figure D.7: The parameters for all iterations, from the three best performing average models. The value shown in the legends is the mean of all iterations.

The comparison reveals variations throughout all parameters but an agreement of the signs, thus the models agree on which parameter entry should be negative and which are weighed positive. Investigation of the individual model parameters has shown that apart from the parameter variations are the models closely related, which correlates with the similar behavior and fit percentages during simulation.

The previous parameter comparison does not include the third overall best performing model, as it is based on only one parameter set, that is the best performing model from cluster two. This model is for convenience called model D and the parameters are shown in *Equation: (D.4)*.

$$\begin{aligned} x[t+1] &= [1.0021]x[t] + [0.028800 \;\; 4.2423 \cdot 10^{-4}] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [-4.7552 \cdot 10^{-4}]w[t] \\ y[t] &= x[t] + [-3.0780 \;\; 0.32120] \begin{bmatrix} u_1[t] \\ u_2[t] \end{bmatrix} + [5.6000 \cdot 10^{-3}]w[t] \end{aligned} \tag{D.4}$$

The parameters of model D follow the same sign convention as the three best average parameter based models. The parameter values of model D deviate slightly, compared to

the average values shown in *Figure D.7*. The entries $B12$, $Bd11$ and $D13$ are significantly lower than the values for the three average based models. Despite the parameter, variations are model D still the third best performing model overall for this structure.

The three overall best models, that is model A, B, and D, are simulated with an average of all batches and their performance is compared.

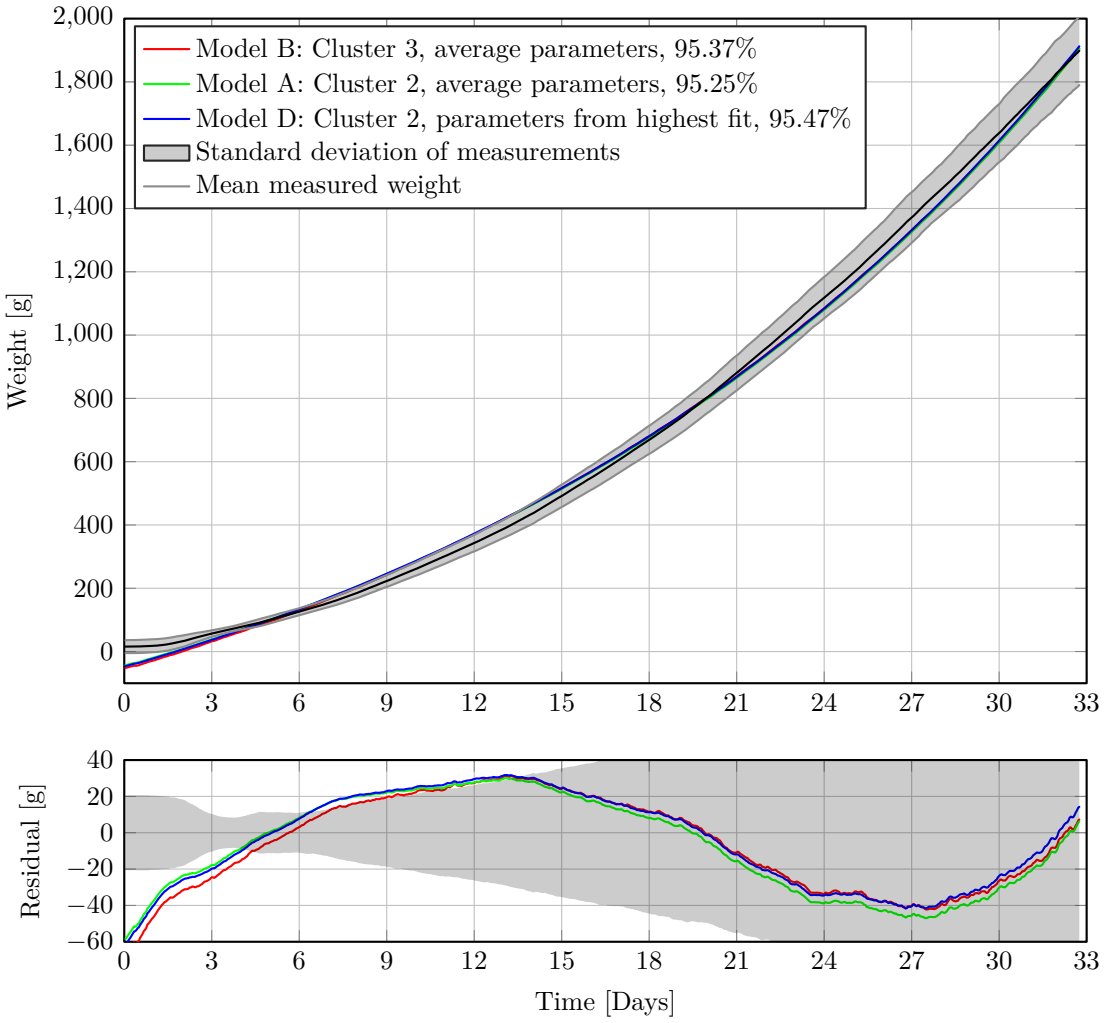

Figure D.8: The mean of all batches, the standard deviation, and the three best models. A relative short batch period is represented, as a mean for all batches only exits for the length of the shortest batch.

Based on the conducted simulation are all three models performing very similar. However, model D yields a marginally higher fit and is thus stated as the resulting model for model structure five.

D.3 Model Structure Conclusion

The estimation of the system and input matrix resulted in a maximum fit percentage of 92.04 % based on the final performance comparison. Estimating the state, input and feedforward matrix resulted in a maximum fit percentage of 95.47 % based on the final performance comparison.

Peripheral Live Test Results

E

In this appendix, the peripheral results obtained during the live test are presented. Furthermore are the obtained results compared to a batch average generated by the 10 most recent batches.

E.1 Temperature

The temperatures presented in *Figure E.1* consist of the implemented reference during the live test, the measured temperature and the average temperature based on the 10 most recent batches.

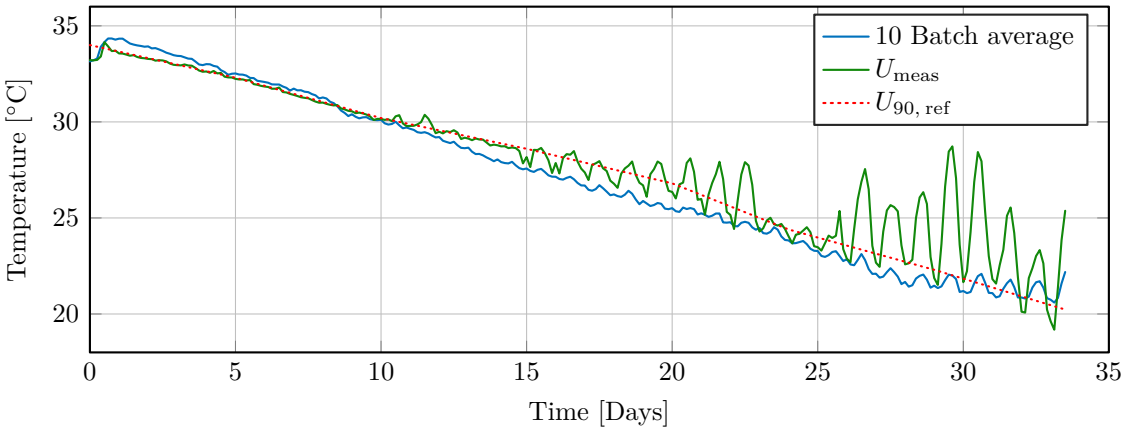

Figure E.1: The implemented reference in comparison with the actual measurement and a 10 batch average temperature.

The comparison between the measured temperature and the ventilation amount is presented in *Figure E.2*.

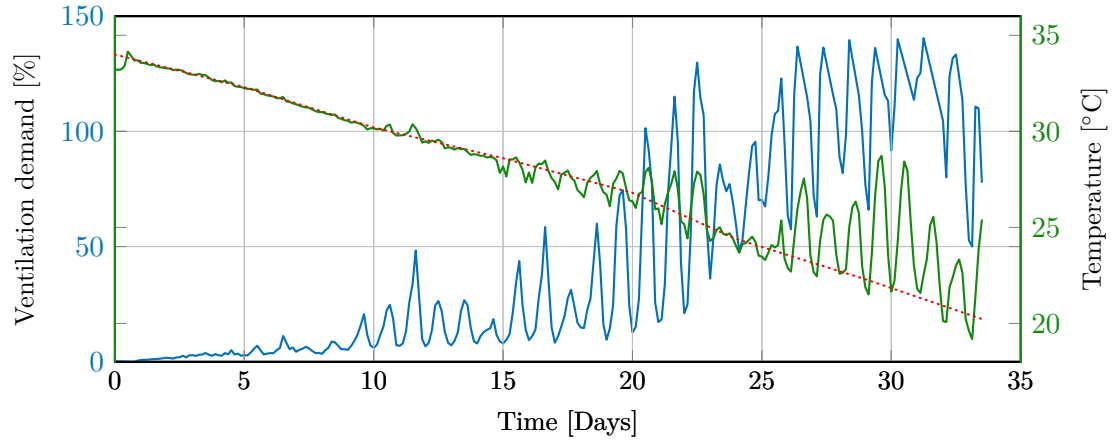

Figure E.2: The comparison of the measured temperature and the ventilation demand.

The figure indicates that the ventilation demand can exceed 100 % which is non-intuitive, the results are however correct and the reason for the slightly different use of a percentage scale is due to the specific implementation and reporting system utilized by SKOV A/S.

E.2 Weight

The weight presented in *Figure E.3* consists of the desired weight trajectory, the measured weight and the average measured weight based on the 10 most recent batches.

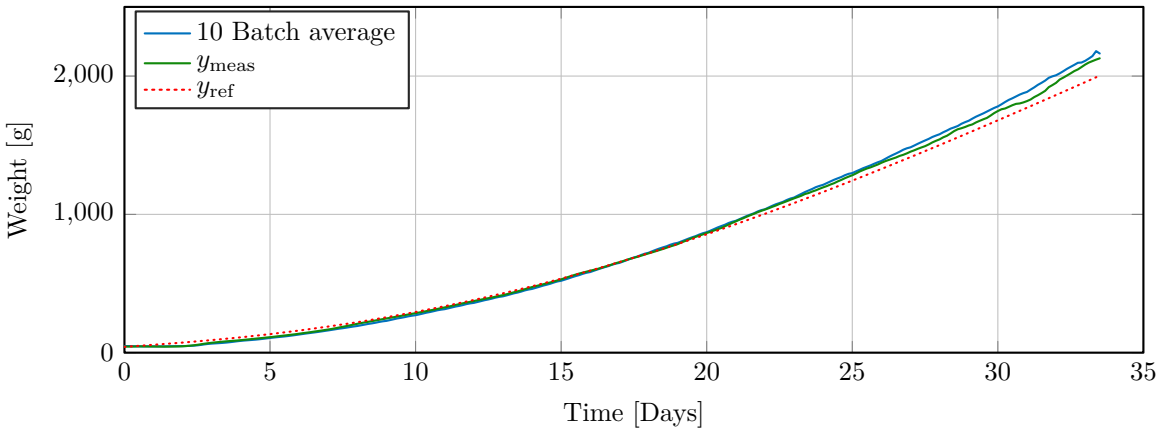

Figure E.3: The reference weight compared to the measured weight and a 10 batch average.

E.3 Disturbances

The measured disturbances are presented in *Figure E.5*, these are the humidity inside the broiler house and the measured outside temperature.

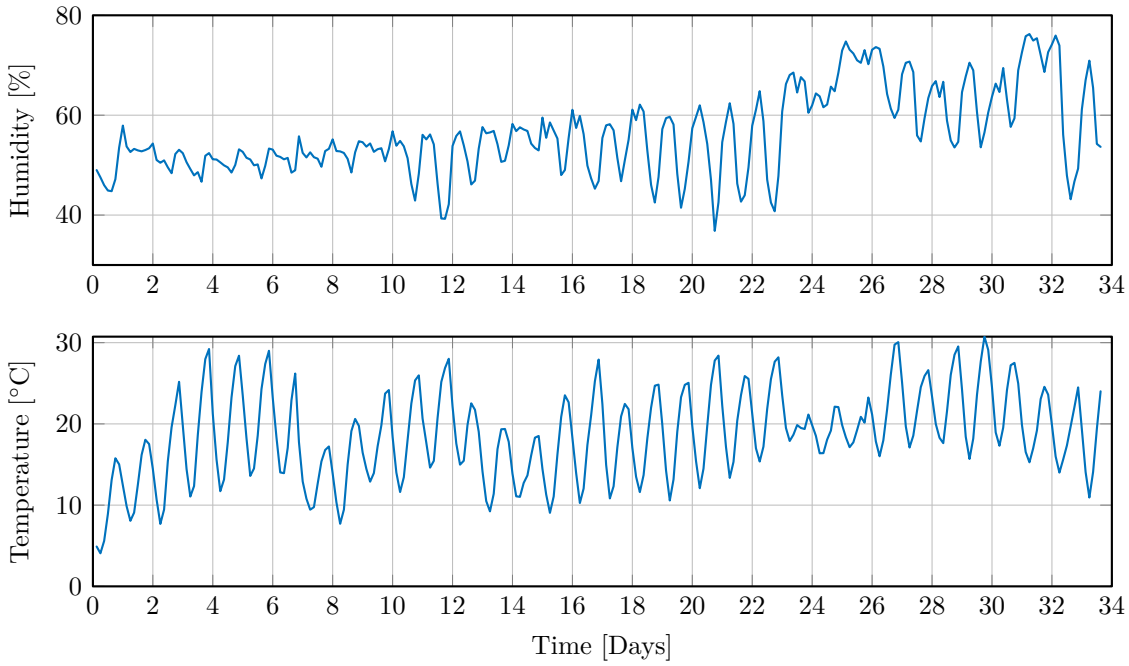

Figure E.4: The measured humidity and outside temperature.

The difference in disturbance from the last to the current batch is presented as small signal values in *Figure E.5*

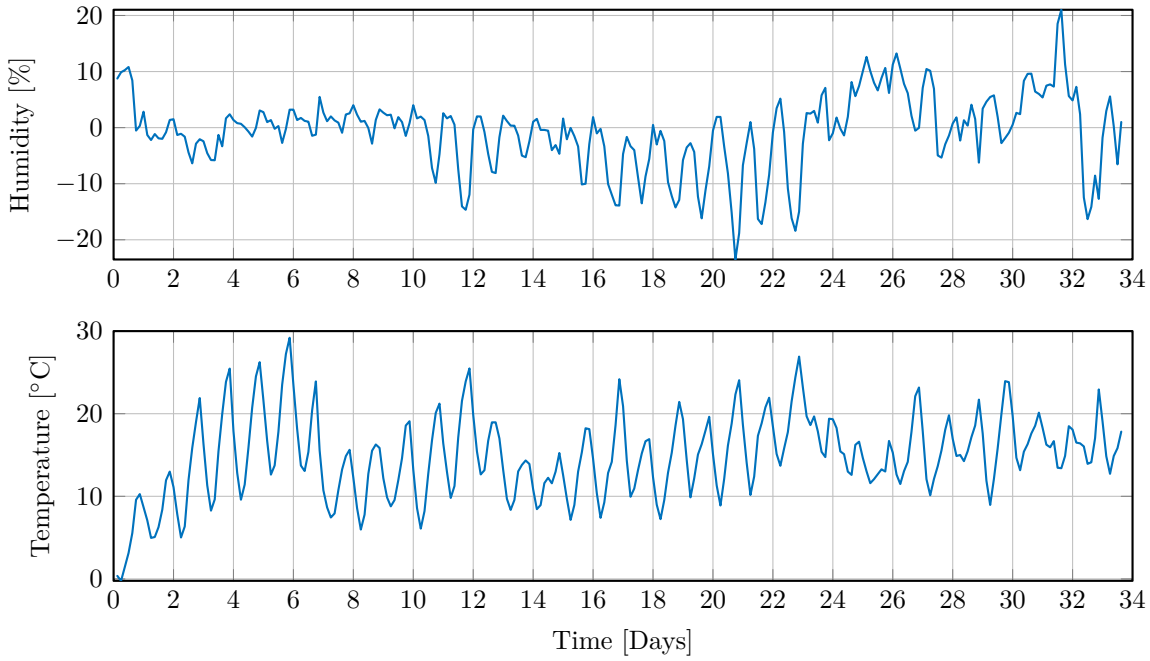

Figure E.5: The small signal difference in humidity and outside temperature compared to the simulation.

Estimation Flow Graphs

F

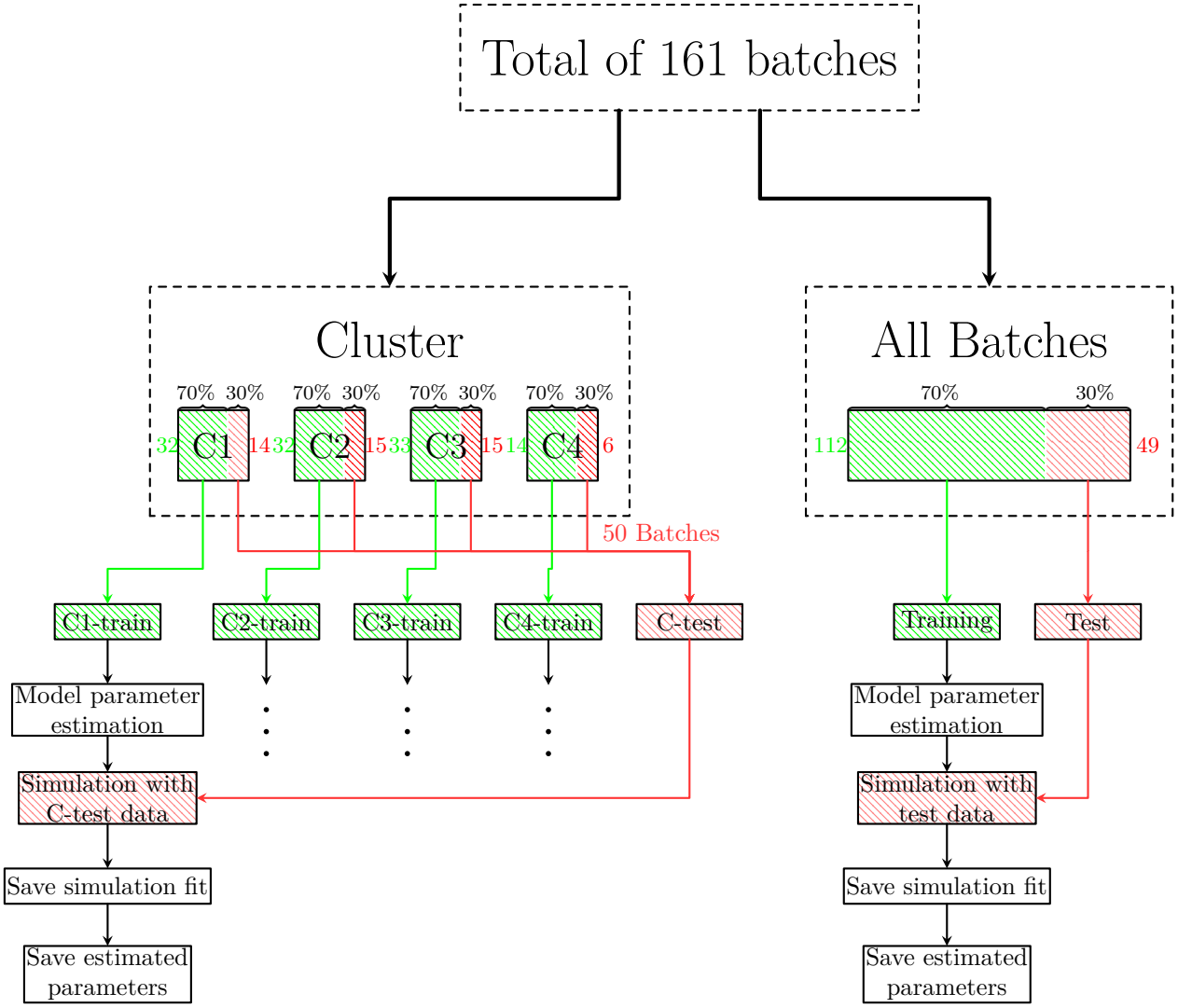

Figure F.1: Describing the final estimation flow with all 161 batches.

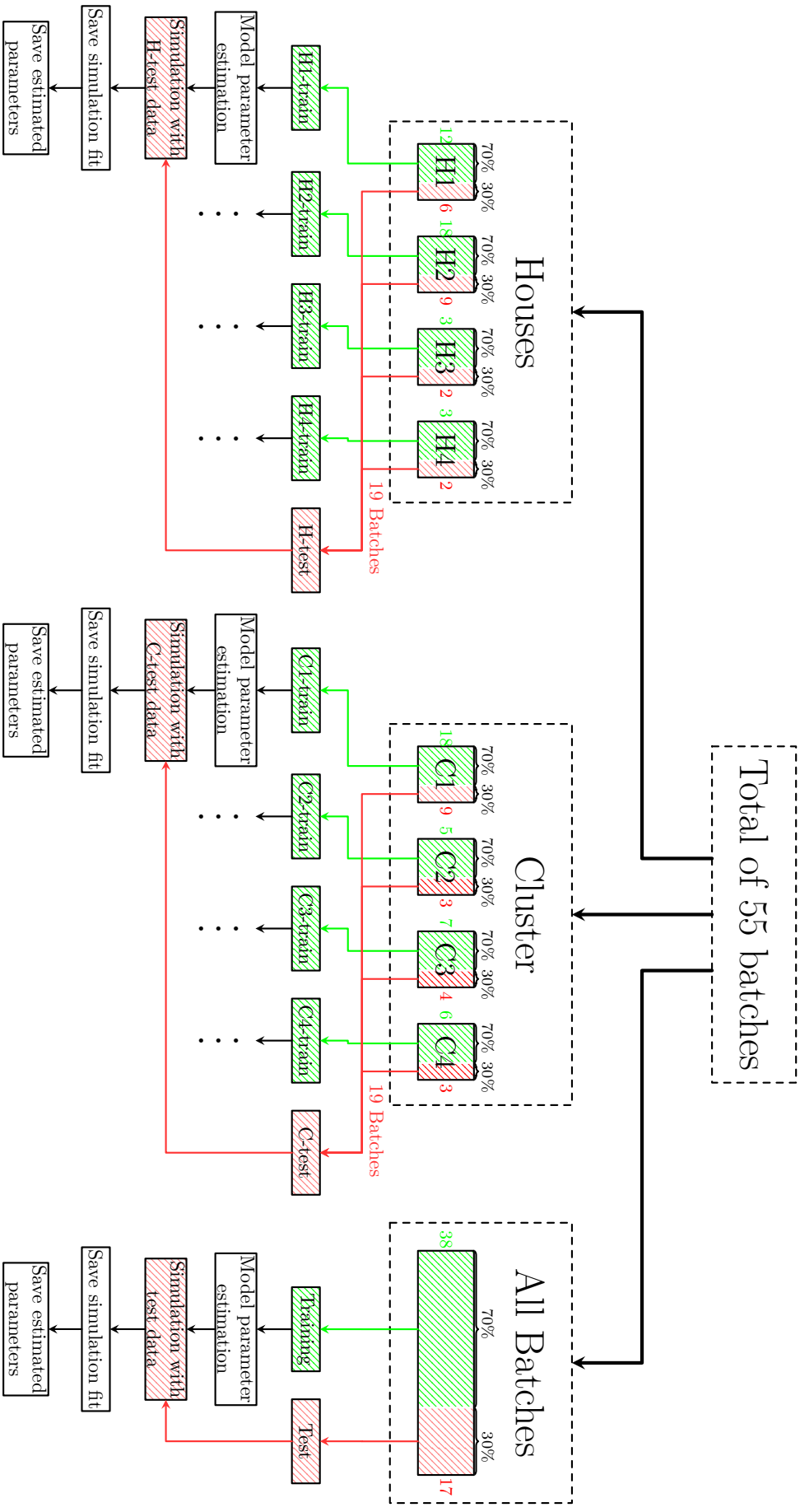

Figure F.2: Describing the initial estimation flow with the first 55 available batches.

Bibliography

[1] [OECD and FAO], "Agricultural outlook 2017-2026." Last visited 20-12-2017, http://dx.doi.org/10.1787/19991142.

[2] N. C. Council, "Per capita consumption of meat." Last visited 20-12-2017, http://www.nationalchickencouncil.org/about-the-industry/statistics/per-capita-consumption-of-poultry-and-livestock-1965-to-estimated-2012-in-pounds/.

[3] A. SKOV, "Give your birds the best start." Last visited 15-11-2017, https://www.skov.com/da/poultry/Sider/Give-your-birds-the-best-start.aspx.

[4] S. V. Johansen, J. D. Bendtsen, M. R.-Jensen, and J. Mogensen, "Broiler weight forecasting using dynamic neural network models with input variable selection," 2017. Preprint submitted to Journal of Neural Networks.

[5] J.-M. Aerts, C. Wathes, and D. Berckmans, "Dynamic data-based modelling of heat production and growth of broiler chickens: Development of an integrated management system," *Biosystems Engineering*, vol. 84, no. 3, pp. 257 – 266, 2003.

[6] H. Talpaz, J. de la Torre, P. Sharpe, and S. Hurwitz, "Dynamic optimization model for feeding of broilers," *Agricultural Systems*, vol. 20, no. 2, pp. 121 – 132, 1986.

[7] S. V. Johansen, J. D. Bendtsen, M. R.-Jensen, and J. Mogensen, "Data driven broiler weight forecasting using dynamic neural network models," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5398 – 5403, 2017. 20th IFAC World Congress.

[8] Aviagen, "Ross broiler management handbook," 2014. p. 16 tab. 2.

[9] Aviagen, "Ross broiler management handbook," 2014. pp. 99-100.

[10] C. V. Caldwell, E. G. Collins, and S. Palanki, "Integrated guidance and control of auvs using shrinking horizon model predictive control," in *OCEANS 2006*, pp. 1–6, 2006.

[11] M. Steinbuch and R. van de Molengraft, "Iterative learning control of industrial motion systems," *IFAC Proceedings Volumes*, vol. 33, no. 26, pp. 899 – 904, 2000. IFAC Conference on Mechatronic Systems, Darmstadt, Germany, 18-20 September 2000.

[12] J. Hätönen, T. Harte, D. Owens, J. Ratcliffe, P. Lewin, and E. Rogers, "Iterative learning control - what is it all about?," *IFAC Proceedings Volumes*, vol. 37, no. 12, pp. 547 – 553, 2004. IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP 04) and IFAC Workshop on Periodic Control Systems (PSYCO 04), Yokohama, Japan, 30 August - 1 September, 2004.

[13] L. K. S., C. In-Shik, L. H. J., and L. J. H., "Model predictive control technique combined with iterative learning for batch processes," *AIChE Journal*, vol. 45, no. 10, pp. 2175–2187.

[14] X. Liu and X. Kong, "Nonlinear fuzzy model predictive iterative learning control for drum-type boiler–turbine system," *Journal of Process Control*, vol. 23, no. 8, pp. 1023 – 1040, 2013.

[15] Z. Xiong, J. Zhang, X. Wang, and Y. Xu, "Integrated batch-to-batch iterative learning control and within batch control of product quality for batch processes," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 273 – 278, 2005. 16th IFAC World Congress.

[16] I. Chin, S. Qin, K. S. Lee, and M. Cho, "A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection," *Automatica*, vol. 40, no. 11, pp. 1913 – 1922, 2004.

[17] Aviagen, "Ross 308 broiler: Performance objectives," 2014. p. 3.

[18] E. F. Camacho and C. Bordons, *Generalized Predictive Control*, pp. 47–79. London: Springer London, 2007.

[19] N. national meteorological institutes, "Long-term weather forecast for a specified place." Last visited 09-05-2018, https://api.met.no/weatherapi/locationforecastlts/1.3/documentation.

[20] MathWorks, "Interpolation function." Last visited 09-05-2018, https://se.mathworks.com/help/matlab/ref/interp1.html#btwp6lt-1-method.

[21] D. M. Institut, "10 års vejrnormal, 2006-2015." Last visited 09-05-2018, https://www.dmi.dk/vejr/arkiver/normaler-og-ekstremer/klimanormaler-dk/vejrnormal/.

[22] H. Schwetlick and T. Schütze, "Least squares approximation by splines with free knots," *BIT Numerical Mathematics*, vol. 35, pp. 361–384, Sep 1995.

[23] T. Schütze and H. Schwetlick, "Constrained approximation by splines with free knots," *BIT Numerical Mathematics*, vol. 37, pp. 105–137, Mar 1997.

[24] K. Sarup, "Hvor mange sommerdage har vi egentlig?." Danmarks Meteorologiske Institut, Last visited 09-05-2018, https://www.dmi.dk/nyheder/arkiv/nyheder-2004/hvor-mange-sommerdage-har-vi-egentlig/.

[25] Éva Gocsik, S. D. Brooshooft, I. C. de Jong, and H. W. Saatkamp, "Cost-efficiency of animal welfare in broiler production systems: A pilot study using the welfare quality® assessment protocol," *Agricultural Systems*, vol. 146, pp. 55 – 69, 2016.

[26] A. Ng, "Machine learning, week 8: Unsupervised learning," 2013. Stanford University, Last visited 14-11-2017, https://www.coursera.org/learn/machine-learning/home/week/8.

[27] MathWorks, "k-means algorithm." Last visited 07-11-2017, https://se.mathworks.com/help/stats/kmeans.html.

[28] MathWorks, "greyestoptions description." Last visited 20-11-2017, https://se.mathworks.com/help/ident/ref/greyestoptions.html.

[29] MathWorks, "fmincon description." Last visited 20-11-2017, https://se.mathworks.com/help/optim/ug/fmincon.html.

[30] MathWorks, "Choosing algorithm for fmincon." Last visited 20-11-2017, https://se.mathworks.com/help/optim/ug/choosing-the-algorithm.html#bsbwxm7.

[31] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, pp. 149–185, Nov 2000.

[32] R. Byrd, M. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[33] T. N. I. of Standards and U. S. D. o. C. Technology, "Grubbs test for outliers." Last visited 04-12-2017, http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm.

[34] P. A. E. f. M. Brett Shoelson, "Grubbs test implementation." Last visited 04-12-2017, https://se.mathworks.com/matlabcentral/fileexchange/3961-deleteoutliers.