Linear-Quadratic Regulation of Computer Room Air Conditioners

Author:
Johan Aasa

Supervisor:
Damiano Varagnolo
Winston Garcia-Gabin
Jonas Gustafsson

Academic year 2017/2018

Acknowledgments

I would like to thank my advisors Damiano Varagnolo, Winston Garcia-Gabin, and Jonas Gustafsson for giving me the opportunity to work with you and your expert advise throughout the thesis. I would also like to thank Jeffrey Sarkinen and all the staff at RISE SICS North for supporting this work with their time and technical expertise.

Abstract

Data centers operations are notoriously energy-hungry, with the computing and cooling infrastructures drawing comparable amount of electrical power to operate. A direction to improve their efficiency is to optimize the cooling, in the sense of implementing cooling infrastructures control schemes that avoid performing over-cooling of the servers.

Towards this direction, this work investigates minimum cost linear quadratic control strategies for the problem of managing air cooled data centers. We derive a physical model for a general data center, identify this model from real data, and then derive, present and test in the field a model based Linear-Quadratic Regulator (LQR) strategy that sets the optimal coolant temperature for each individual cooling unit. To validate the approach we compare the field tests from the LQR strategy against classical Proportional–Integral–Derivative (PID) control strategies, and show through our experiments that it is possible to reduce the energy consumption with respect to the existing practices by several points percent without harming the servers within the data center from thermal perspectives.

Contents

1 Abbreviations

CRAC Computer Room Air Conditioning

SS State Space

LQR Linear-Quadratic Regulator

LQRi Linear-Quadratic Regulator with Integral Action

MPC Model Predictive Control

SISO Single Input Single Output

MIMO Multiple Input Multiple Output

SIMO Single Input Multiple Output

LTI Linear Time Invariant

RISE Research Institute of Sweden

SICS Swedish Institute of Computer Science

ARX Autoregressive Exogenous

PID Proportional–Integral–Derivative

P Proportional

PI Proportional-integral

EU European Union

PEM Prediction-Error identification Methods

MPC Model Predictive Control

OPC Open Platform Communication

SNMP Simple Network Management Protocol

PLC Programmable Logic Controller

MIB Management Information Database

2 Nomenclature

Variable	Description
t	time index (both continuous and discrete)
$i \text{in} \mathcal{I}$	CRAC index
$j \text{in} \mathcal{J}$	Rack index
$k \text{in} \mathcal{K}$	Server index
$\mathcal{I}$	set of the CRACs indexes
$\mathcal{J}$	set of racks indexes
$\mathcal{K}$	set of the server indexes
$T_{j,k,\text{in}}$	Temperature of the air flow at the inlet of server j,k
$T_{j,k}$	Internal temperature of server j,k
$T_{j,k,\text{out}}$	Temperature of the air flow at outlet of server j,k
$T_{i,\text{in}}$	Temperature of the air flow at the inlet of CRAC i
T_i	Internal temperature of CRAC i
$T_{i,\text{out}}$	Temperature of the air flow at the outlet of CRAC i
$\boldsymbol{T}_{\text{servers,in}}$	vector of all the temperatures of the air flows at the inlet of the various server
$\boldsymbol{T}_{\text{servers}}$	vector of all the internal temperatures of the various server
$\boldsymbol{T}_{\text{servers,out}}$	vector of all the temperatures of the air flows at the outlet of the various server
$s_{i \to j,k}$	Fraction of flow from CRAC i that enters server j,k
$s_{j,k \to i}$	Fraction of air exiting server j,k and returning to CRAC i
$f_{j,k,\text{in}}$	Flow entering server j,k
$f_{j,k,\text{out}}$	Flow exiting server j,k
$f_{i,\text{in}}$	Flow entering CRAC i
$f_{i,\text{out}}$	Flow exiting CRAC i
α_T^s	Parameter summarizing the heat capacity of the air and the thermal conductivity of a generic server (see (10))
α_p^s	Self-heating parameter of a generic server (see (10))
α_f^s	Parameter defining the internal decay of flow within a server (see (3))
α_T^c	Parameter summarizing the heat capacity of the air and the thermal conductivity of a generic CRAC (see (8))
α_p^c	Self-heating parameter of a generic CRAC (see (8))
α_f^c	Parameter defining the internal decay of flow within a CRAC (see (1))
$\boldsymbol{\alpha}_{j,k}^s$	Vector of the thermal conductivities among all the various servers in the data center and server j,k
β_o^*	Substitution parameter for $\alpha_o^* T_s$ in the discrete cases
$p_{j,k}$	Current power usage of server j,k
$u_{j,k}$	Current flow forced by the internal fans of server j,k
p_i	Current power usage of CRAC i
u_i	Current flow forced by the internal fans of CRAC i
$\Delta_{i \to j,k}$	time that it takes for the flow generated by CRAC i to reach server j,k
$\Delta_{j,k \to i}$	time that it takes for the flow generated by server j,k to reach CRAC i
$p_{j,k,avg}$	Average power usage of server j,k
ω_p	The random aspect of the server usage considered as white noise with zero mean
$\mathcal{Z}_i$	The subspace of j,k influenced by CRAC i

Table 1: Notation used throughout the document.

3 Introduction

Data centers are an increasingly significant part of today's society with a growing societal demand for data storage and analysis. They are increasing in number, size and complexity, and as a direct effect of this their aggregated energy usage is also increasing. For example in 2013 data centers in the European Union (EU) alone consumed 11.8 GW on average with a growing rate of 4 % annually. This was roughly 3 % of the total electricity generated across EU which correlates to 38.6 million tonnes of CO_2 emitted [1]. Since the demand for data will not decrease any time soon, there is a great incentive to make more energy efficient data centers. Among the many ways of improving energy efficiency in datacenters, the one that will be the focus of this thesis is improving the way thermal cooling is performed. This is because the energy usage associated to cooling may be very high: according to [2], the percentage of electricity that goes into cooling in a data center may be up to 40% of the total figure. In this thesis we will thus specifically try to improve the control of the CRACs units in air cooled data centers. Investigating minimum cost linear quadratic control strategies for the CRACs. We derive a physical model for a general data center, identify this model from real data. We also apply a black box method of identifying a model from real data by testing different standard models. Then derive, present and test in the field a model based LQR strategy that sets the optimal coolant temperature for each individual cooling unit. To validate the approach we compare the field tests from the LQR strategy against classical PID control strategies. For ease of readability, we describe the steps followed in our work and the intuitions behind them in the next section through a dedicated road-map.

4 Roadmap

The work flow for the project, in the sense of the list of the tasks that had to be done in order to reach the result of obtaining a strategy for controlling the cooling units within a data center that is more effective than the current practice, is the following:

1. start by finding a physical model for the thermal dynamics of the system under consideration, that comprises:
   - inspecting what are the effects of the humidity of the air;
   - understanding how potential time-delays of the volume flows within the built environment can affect the overall thermal dynamics;
2. restructure the physical model found in the previous step in a control-oriented way, that comprises also:
   - discretizing continuous-time models into discrete-time control oriented models;
   - inspecting the connections of these models with classical Autoregressive Exogenous (ARX) models;
   - considering that the aim is to design LQR strategies, derive SS representations of the models above;
3. derive different control strategies, so that it will be possible to draw comparative results. This comprises:
   - deriving classical reactive controllers such as P and PI;
   - derive optimal controllers such as LQR and Linear-Quadratic Regulator with Integral Action (LQRi) (the latter so to cope with potential uncertainties in the model descriptions);
4. implement a system that enables the real time collection of data from the datacenter *and* that allows sending control signals to the infrastructure;
5. perform experiments where we manipulate the various variables involved in the system, so that it is possible to apply suitable system identification algorithms for estimating the parameters of the models defined above. This comprises:
   - designing, running and processing the experiments;
   - performing parameter estimation steps for the various derived models;
6. implement, debug and test the so-obtained controllers on the field;
7. think at what the evidence is saying to us.

5 Models for the thermal dynamics

This section describes the general setup of a general air-cooled data center, presents a description of the dynamics of the volume flows and temperatures in this general data center setup, and eventually combines these dynamics in a discrete physical model.

All the notation is collected in Table 1 on page 7.

5.1 Graphic description of the considered system

The general structure of most air-cooled data centers is that the servers are situated in racks placed in rows that create different aisles with temperature gradients. Air-cooled servers are typically endowed with fans that cool these servers locally; the built environment moreover presents some CRACs controlling the ambient temperature in the server room. Figure 1 depicts this general structure.

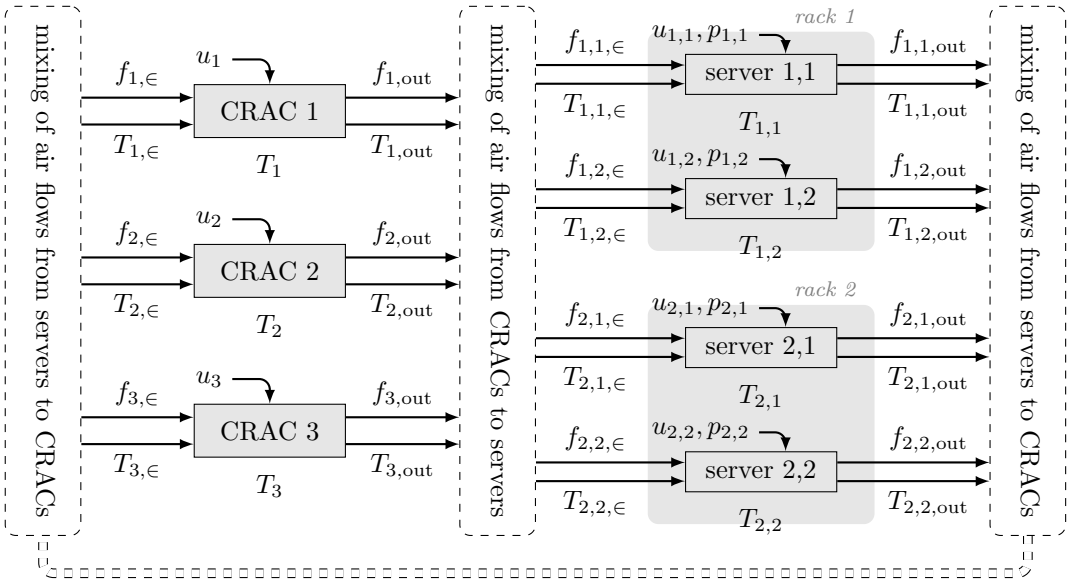

Figure 1: Graphical and simplified description of the system under consideration.

5.1.1 CRACs

As show in Figure 1, the room can be logically divided in four zones. The first zone corresponds to the one defined by the CRAC units. Consider then that an airflow $f_{i,\text{in}}$ with a temperature $T_{i,in}$ enters the top of the various CRACs; the air flow is then altered by the CRAC's internal fan (whose rotational speed

Figure 2: Photo of one of the CRAC units used in our experiments.

is represented by u_i), plus cooled down by the CRAC's cooling coils, whose temperature is T_i. This eventually results in the CRACs outputting a new air flow $f_{i\text{out}}$ at the new temperature $T_{i,\text{out}}$.

5.1.2 From CRACs to servers

The second zone within the logical division that we made for the computer room is the space between the CRACs and the servers, as shown in Figure 3. Here the airflow $f_{i,out}$ with the temperature $T_{i,\text{out}}$ exits the CRACs. This air gets then mixed on its way to the servers and a fraction of the airflow $f_{j,k,\text{in}}$ with the temperature $T_{j,k,\text{in}}$ enters each server. This fraction is determined by the dynamics described by $s_{i\to j,k}$.

5.1.3 Servers

The third zone in our logical division is represented by the servers (graphically shown in Figure 4). Here the airflow $f_{j,k,\text{in}}$ with the temperature $T_{j,k,\text{in}}$ enters a server. This flow is then altered by the servers internal fan, whose rotational speed is represented by $u_{j,k}$, and is moreover heated up by the servers internal temperature $T_{j,k}$. The final result is that the servers output a new air flow $f_{j,k\text{out}}$ with a new temperature $T_{j,k,\text{out}}$.

Figure 3: Photo of the space that lies between the CRACs and the servers for the system that we used in our experiments.

Figure 4: A Dell R430 server.

Figure 5: Photo of the hot aisle of the data center we used in our experiments.

5.1.4 From servers to CRACs

The fourth zone is the space between the servers and the CRACs as shown in Figure 5. Here the airflow $f_{j,k,out}$ with the temperature $T_{j,k,\text{out}}$ exiting each server gets mixed in its way towards the various CRACs. The resulting air flow can be represented through its flow $f_{i,\text{in}}$ and temperature $T_{i,\text{in}}$. In this way the cycle of the air flow is closed.

5.2 Standing hypotheses

In our following derivations we assume that:

- there are no air leakages within servers and within the computer room (notice that high air leakage leads to greater non-linearity);
- at least theoretically, each air flow from each CRAC unit affects each server; the degree of affecting is captured by some parameters that we have to identify from real data.

Notice that some other case-specific assumptions will be introduced when deriving the different models of the thermal dynamics within the data center.

5.3 Dynamics of the volumes of the flows

This subsection describes what happens to the volumes of the flows when they pass through the various zones of the data center room. Referring to Figure 1, the volumes of the flows are modified in four specific points:

1. inside the CRACs;
2. from the CRACs to the servers;
3. inside the servers;
4. from the servers to the CRACs;

More specifically, we postulate that the volumes of the air-flows are modified through the following respective static time-delayed relationships:

$$f_{i,\text{out}}(t) = \alpha_f^c f_{i,\text{in}}(t) + u_i(t) \qquad \text{(inside the CRACs)} \tag{1}$$

$$f_{j,k,\text{in}}(t) = \sum_i s_{i\to j,k} f_{i,\text{out}}(t - \Delta_{i\to j,k}) \qquad \text{(from CRACs to servers)} \tag{2}$$

$$f_{j,k,\text{out}}(t) = \alpha_f^s f_{j,k,\text{in}}(t) + u_{j,k}(t) \qquad \text{(inside the servers)} \tag{3}$$

$$f_{i,\text{in}}(t) = \sum_{j,k} s_{j,k\to i} f_{j,k,\text{out}}(t - \Delta_{j,k\to i}) \qquad \text{(from servers to CRACs)} \tag{4}$$

The relations above are based on the following two concepts:

1. Equation (1) describes the flow out from the CRACs as an function dependant on the flow into the CRACs $f_{i,\text{in}}$ and the flow u_i added by the CRACs internal fans. Where α_f^c is a drag coefficient describing internal flow decay within a CRAC.
2. Equation (2) describes the flow into the servers as a function of the flow $f_{i,\text{out}}$ out from the CRACs. Where $s_{i\to j,k}$ is the fraction of air exiting CRAC i entering server j,k. And $\Delta_{i\to j,k}$ is the time it takes for the flow generated by CRAC i to reach server j,k.

Importantly, since there are mass-transportation effects, we include in the models opportune time-delays. As for the model of these time delays, we postulate that these time delays depend both on the geometry of the computer room and the speed of the various flows through the relation

$$\Delta_{i\to j,k}(t) = \frac{\phi\left(d_{i\to j,k}\right)}{f_{i,\text{out}}(t)} \tag{5}$$

where $d_{i\to j,k}$ is the physical distance that should be traveled when going from CRAC i to server j,k, and $\phi(\cdot)$ is an opportune function that depends on geometrical parameters (e.g., the area of the server inlet). Examples of specific ϕ's for specific types of data centers are reported in Appendix A.1.

5.4 Dynamics of the temperatures of the flows

This subsection describes what happens to the temperatures of the flows when they mix. Referring to Figure 1, mixing of flows happens in four specific points:

1. inside the CRACs;
2. from the CRACs to the servers;
3. inside the servers;
4. from the servers to the CRACs;

More specifically, we postulate that when mixing between CRACs and servers and vice-versa, the mixing is static and time-delayed, with mixing factors that reflect an averaging between the temperatures of the various air-flows. In other words, the models are as follows:

$$T_{j,k,\text{in}}(t) = \frac{\sum_i s_{i\to j,k} T_{i,\text{out}}(t - \Delta_{i\to j,k})}{\sum_i s_{i\to j,k}} \tag{6}$$

$$T_{i,\text{in}}(t) = \frac{\sum_{j,k} s_{j,k\to i} T_{j,k,\text{out}}(t - \Delta_{j,k\to i})}{\sum_{j,k} s_{j,k\to i}} \tag{7}$$

Notice that, as said in Section 5.3, the time delays $\Delta_\star$ at least theoretically depend on the volumes of the air flows $f_{i,\text{out}}$ and $f_{j,k,\text{out}}$.

As for the temperatures of the flows at the output of the components, we consider a mixed static / dynamic model where we exploit the auxiliary internal temperature of the component as a state variable. More precisely, we use a first-order model representing a classical Newton law of cooling comprising convection (and possibly conduction) effects, plus some self-heating terms.

As for the dynamics relative to the CRACs, thus, our postulated model is

$$\dot{T}_i(t) = \alpha_T^c f_{i,\text{out}}(t)\Big(T_{i,\text{in}}(t) - T_i(t)\Big) + \alpha_p^c p_i(t) \tag{8}$$

$$T_{i,\text{out}}(t) = T_{i,\text{in}}(t) + \frac{\widetilde{\alpha}_T^c}{f_{i,\text{out}}(t)}\Big(T_{i,\text{in}}(t) - T_i(t)\Big)\,. \tag{9}$$

Similarly to the CRACs the dynamics relative to the servers are

$$\dot{T}_{j,k}(t) = \alpha_T^s f_{j,k,\text{out}}(t)\Big(T_{j,k,\text{in}}(t) - T_{j,k}(t)\Big) + \alpha_p^s p_{j,k}(t) + \boldsymbol{\alpha}_{j,k}^s \boldsymbol{T}_{\text{servers}} \tag{10}$$

$$T_{j,k,\text{out}}(t) = T_{j,k,\text{in}}(t) + \frac{\widetilde{\alpha}_T^s}{f_{i,\text{out}}(t)}\Big(T_{j,k,\text{in}}(t) - T_{j,k}(t)\Big). \tag{11}$$

Notice that equations (8) and (10) are structurally similar, but different because the latter has an additional term $\boldsymbol{\alpha}_{j,k}^s \boldsymbol{T}_{\text{servers}}$ that takes into account the thermal conductivity between servers (a thermal effect that is not present among the CRACs).

In the case of the CRACs dynamics the self-heating term $\alpha_p^c p_i(t)$ is the cooling effect of the coolant. In most cases this term will be dominant in determining a CRACs internal temperature T_i. If also the cooling power is sufficiently big the CRACs output temperature $T_{i,out}$ will also take on the coolants temperature p_i resulting in a simplification of the dynamics in (8) and (9) so that we can say that

$$T_{i,out}(t) \approx p_i(t) \tag{12}$$

this will become important later on in Section 6 in order to maintain a linear model when setting up the system model in SS form.

5.5 Discretization

In order to later on simulate and implement in the real hardware the to be developed model-based controllers we need to discretize the previously postulated dynamics. In this thesis discretization is always done using forward Euler methods, i.e., by letting

$$y(t+1) = y(t) + T_s f(t). \tag{13}$$

Applying the forward Euler approach (13) to the differential equation (10) and substituting $\beta_o^* = \alpha_o^* T_s$ one then gets the discrete first order linear differential equation

$$T_{j,k}(t+1) = \beta_T^s f_{i,\text{out}}(t)\Big(T_{j,k,\text{in}}(t) - T_{j,k}(t)\Big) + T_{j,k} + \beta_p^s p_{j,k}(t) + \boldsymbol{\beta}_{j,k}^s \boldsymbol{T}_{\text{servers}}. \tag{14}$$

5.6 ARX model

To provide a baseline to our derivations, We consider also a black box model where nothing of the system is know. Then different standard models is applied to a large data set with large variations gatherer from a real world system. We then choose the standard model that gave the best result, the best fit, to our specific real world system. In our case the model that gave the best fit was a first order ARX model that reads as follows:

$$T_{j,k}(t+1) = \beta T_{j,k}(t) + \alpha_1 T_i(t) + \alpha_2 f_{i,\text{out}}(t) + \alpha_3 p_{j,k}(t) + \alpha_4 u_{j,k}(t) \tag{15}$$

6 State space representations for the case constant flow

In this section we derive, from opportune different simplifications of the thermal dynamics defined in Section 5, different SS models of the system with different complexities. In this way we will obtain different to-be-identified parametric models to be used as starting points for our CRAC control strategies. The final aim will then be to check which strategy is the best one from field experiments.

Importantly, we consider here only situations for which the mass flows of the various CRACs is constant in time.

6.1 Notation for SS models

Besides the notation introduced in Table 1 on page 7, we now consider some further notation The simplifications done to the thermal dynamics defined in Section 5 in order to set up the SS models where based on the standing assumptions posed in the relative section. On top of that, firstly we now assume also that the temperature control of the CRACs coolant p_i is sufficiently fast that, according to (12), we can control the output temperatures $T_{i,\text{out}}$ as we want, so that $T_{i,\text{out}}$ becomes our control input. Secondly we assume fixed air flows from the CRACs, so that the various $f_{i,\text{out}}$ are to be considered constant. These assumptions are important since it allows us to decouple the system in a way that the dynamics (8)-(9) can be ignored. In formulas, this means that

$$f_{i,\text{out}}(t) = f_{i,\text{out}} \quad \forall t. \tag{16}$$

Applying (16) into (5) we thus also obtain

$$\Delta_{i\to j,k}(t) = \Delta_{i\to j,k}. \tag{17}$$

Notice that the fact that the time-delays are constants is very important for having a state of the system that keeps a constant dimension.

Our state space model will thus have,

as states:

$$\boldsymbol{x}(t) := \left[\{T_{j,k}(t)\}_{j\in\mathcal{J},k\in\mathcal{K}}\right]$$

as controllable inputs:

$$\boldsymbol{u}_{\mathcal{I}}(t) := \left[\{T_i(t)\}_{i\in\mathcal{I}}\right]$$

depending on the situation, as either controllable inputs or disturbances:

$$\boldsymbol{d}_u(t) := \left[\{u_{j,k}(t)\}_{j\in\mathcal{J},k\in\mathcal{K}}\right]$$

$$\boldsymbol{d}_p(t) := \left[\{p_{j,k}(t)\}_{j\in\mathcal{J},k\in\mathcal{K}}\right]$$

as measurements:

$$\boldsymbol{y}(t) := \begin{bmatrix} \{T_{j,k}(t)\}_{j\in\mathcal{J},k\in\mathcal{K}} \\ \{T_{j,k,\text{in}}(t)\}_{j\in\mathcal{J}_\text{in}\subseteq\mathcal{J},k\in\mathcal{K}_\text{in}\subseteq\mathcal{K}} \\ \{T_{j,k,\text{out}}(t)\}_{j\in\mathcal{J}_\text{out}\subseteq\mathcal{J},k\in\mathcal{K}_\text{out}\subseteq\mathcal{K}} \end{bmatrix}$$

Notice that the flows from the server fans $u_{j,k}(t)$ may not actually manifest themselves as disturbances since there exists a internal feedback loop within the servers controlling the fans. So the fan speeds $u_{j,k}(t)$ are actually a transformation that we may know, or at least be a measurable parameter which could be inserted into the models below.

To obtain the desired dynamics the general strategy is then to use the additional assumptions to simplify opportunely (2), (3), and (6), and then insert them into (14). Eventually the outcome will be a model with the structure

$$\boldsymbol{x}(t+1) = A_\star(t)\boldsymbol{x}(t) + B_\star(t)\boldsymbol{u}(t) + D_\star\boldsymbol{d}_p(t) \tag{18}$$

where $\star = 1,\ldots,3$ indicates different models.

Notice that in general the matrix $A_\star(t)$ may depend on the disturbance $\boldsymbol{d}_u(t)$, on the time delays $\Delta_{i\to j,k}$, on the flows $f_{i,\text{out}}$ imposed by the CRACs, and, obviously, on the various parameters of the model summarized in the notation 1. Notice moreover that, for control purposes, we measure the whole state so there is no need for writing the measurement equation since $\boldsymbol{y}(t) = \boldsymbol{x}(t)$.

We now present several models for the studied thermal dynamics with increasing complexity.

6.2 Model 1 - No time delays, air fractions constant in the whole data center

The first model is also the simplest one; here we assume:

1. the dynamics of the servers temperature changes to be sufficiently slow that the mass-transportation-induced time delays $\Delta_{i\to j,k}$ can be ignored;

2. the fraction of air from CRAC i entering into server j,k to be always the same independently of i, j, and k. This means that, given the notation above, $s_{i\to j,k} = s$.

Notice that with these two assumptions the dynamics of the temperature in (6) simply becomes

$$T_{j,k,\text{in}} = T_{i,\text{out}}.$$

This simplification can then be used to obtain the following state-update equations through opportunely simplifying the dynamics in (14) that become

$$A_1(t) = diag\Big(1 - \beta_T^s\big(\alpha_f^s \sum_i s f_{i,\text{out}} + u_{j,k}(t)\big)\Big) \tag{19}$$

$$B_1(t) = \beta_T^s\big(\alpha_f^s \sum_i s f_{i,\text{out}} + u_{j,k}(t)\big) \tag{20}$$

$$D_1 \boldsymbol{d}_p(t) = \beta_p^s p_{j,k}(t) \tag{21}$$

resulting in the discrete SS model

$$T_{j,k}(t+1) = A_1(t)T_{j,k}(t) + B_1(t)T_{\text{out}}(t) + D_1 \boldsymbol{d}_p(t) \tag{22}$$

with a structure matching the one presented in (18).

6.3 Model 2 - No time delays, but air fractions depending on the server

Still ignoring the time delay $\Delta_{i\to j,k}$, we can now assume to take into account the fact that different CRACs will produce different fractions of air entering each server. This means that

$$A_2(t) = diag\Big(1 - \beta_T^s\big(\alpha_f^s \sum_i s_{i\to j,k} f_{i,\text{out}} + u_{j,k}(t)\big)\Big) \tag{23}$$

$$B_2(t) = diag\Big(\beta_T^s\big(\alpha_f^s \sum_i s_{i\to j,k} f_{i,\text{out}} + u_{j,k}(t)\big)\Big) \frac{s_{i\to j,k}}{\sum_i s_{i\to j,k}} \tag{24}$$

$$D_2 \boldsymbol{d}_p(t) = \beta_p^s p_{j,k}(t) \tag{25}$$

resulting in the discrete SS model

$$T_{j,k}(t+1) = A_2(t)T_{j,k}(t) + B_2(t)T_{i,\text{out}}(t) + D_2 \boldsymbol{d}_p(t) \tag{26}$$

whose structure, like model 1, matches again (18).

6.4 Model 3 - Complete

The third model that we propose mimics the physical dynamics described in Section 5. With respect to the previous model 2, here we remove the assumption that the time delay $\Delta_{i\to j,k}$ can be ignored. When taking this delay into account the system nonetheless becomes more convoluted. Nonetheless, thanks to the assumption that the airflows are fixed the time delays $\Delta_{i\to j,k}$ are constant in time (see also (17)) and this implies that the dimensions of the state-update equations remain constant. More precisely, the model looks like the following:

$$g(j,k;i,\Delta_{max},t) = \begin{cases} diag\Big(\beta_T^s\big(\alpha_f^s \sum_i s_{i\to j,k} f_{i,\text{out}} + u_{j,k}(t)\big)\Big) \frac{s_{i\to j,k}}{\sum_i s_{i\to j,k}} \\ 0 \end{cases} \tag{27}$$

$$D_3 \boldsymbol{d}_p(t) = \beta_p^s p_{j,k}(t) \tag{28}$$

resulting in the discrete SS model

$$\begin{bmatrix} T_{j,k}(t+1) \\ T_{i,\text{out}}(t) \\ T_{i,\text{out}}(t-1) \\ \vdots \\ T_{i,\text{out}}(t-\Delta_{max}+1) \end{bmatrix} = \begin{bmatrix} A_2(t) & g(j,k;1,1) & \dots & \dots & g(j,k;i,\Delta_{max}) \\ 0 & 0 & \dots & \dots & 0 \\ 0 & I & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & I & 0 \end{bmatrix}$$
$$\begin{bmatrix} T_{j,k,\text{out}}(t) \\ T_{i,\text{out}}(t-1) \\ T_{i,\text{out}}(t-2) \\ \vdots \\ T_{i,\text{out}}(t-\Delta_{max}) \end{bmatrix} + \begin{bmatrix} 0 \\ I \\ 0 \\ \vdots \\ 0 \end{bmatrix} T_{i,\text{out}}(t) + D_3 \boldsymbol{d}_p(t). \tag{29}$$

Once again the structure of the state space matches the general one described in (18).

6.5 Model 4 - ARX model

For completeness we also consider the ARX model presented in (15). This can be rewritten in a SS form as

$$T_{j,k}(t+1) = A_4 T_{j,k}(t) + + B_4 T_i(t) + D_4 \boldsymbol{d}_{ARX}(t) \tag{30}$$

where $A_4 = \beta$, $B_4 = \alpha_1$, $D_4 = [\alpha_2\ \alpha_3\ \alpha_4]$ and $\boldsymbol{d}_{ARX}(t) = [f_{i,\text{out}}(t)\ \ p_{j,k}(t)\ \ u_{j,k}(t)]'$. We see that (30) have the same structure as the physical model presented in (18) with the difference being of being Linear Time Invariant (LTI). This similarity will be exploited later on in Section 7 when we derive our LQR control strategies. Indeed the derived controllers will be based on the same structural representation of the SS systems – the unique thing that will change will be the actual dimensions and values of the matrices A, B, C and D.

7 Control strategies design

We here present two different control strategies with increasing complexity strategies that can be used for designing the inputs $\boldsymbol{u}(t)$ in (18). We start with presenting some variations of the classic PID controller, and then move on to LQR controllers. Notice that the latter LQR control strategies, defined in Sections 11.4, will be derived using general formulations, so that it will be possible to implement the different SS models defined in Section 6 using the same formulas parametrized in $A_\star(t)$, $B_\star(t)$ and $D_\star(t)$.

7.1 PID Control

PID controllers are widely used in every industrial settings due to their simplicity in understanding, implementing and tuning them. Interestingly, implementing these controllers does not require any knowledge of the model of the system to be controlled. Indeed PIDs control actions are computed starting only from the values of the measured process variables and the references that they should follow. Some brief theory on PIDs and their implementation is described below in Section 7.1.1-7.1.2.

7.1.1 General theory

A PID controller is a feedback controller that continuously calculates the difference between a desired set point $\boldsymbol{r}(t)$ and the process variable $\boldsymbol{y}(t)$ called the error $\boldsymbol{e}(t)$. The controller consists of three parts: a proportional term, a integral term and a derivative term. The proportional term simply generates a control value that is proportional to the current error $\boldsymbol{e}(t)$. P controllers have the limitation that if the system is subject to disturbances, then the system will present steady state errors. To mitigate this drawback one may then use the second part, i.e., the integral term (whose output is proportional to the sum of the current error $\boldsymbol{e}(t)$ and past errors $\boldsymbol{e}(t-1) + \boldsymbol{e}(t-2) + \ldots + \boldsymbol{e}(0)$). This integral term may reduce steady state errors, but it may also cause the process variable to overshoot its desired set point if the sum of the errors has been charged too much. The third term, i.e., the derivative term, is considered a sort of predictive controller, since it predicts where the process variable is heading by calculating the derivative of the error $(\boldsymbol{e}(t) - \boldsymbol{e}(t-1))\frac{1}{\Delta t}$, and this may be helpful to reduce the overshoot. This can all be summed up to give the control action

$$\boldsymbol{u}(t) = \boldsymbol{K}_p \boldsymbol{e}(t) + \boldsymbol{K}_i \sum_{i=0}^{t} \boldsymbol{e}(t-i)\Delta t + \frac{\boldsymbol{K}_d}{\Delta t}(\boldsymbol{e}(t) - \boldsymbol{e}(t-1)) \tag{31}$$

as a function of the error

$$\boldsymbol{e}(t) = \boldsymbol{r}(t) - \boldsymbol{y}(t). \tag{32}$$

7.1.2 PID controllers for our specific CRAC control problem

In our specific PID implementation case we test two controllers, one with only the proportional term from (31) and one with the proportional term and the integral term. As for the process variable $\boldsymbol{y}(t)$ we consider the average of the various server temperatures $T_{j,k}$, and as a control variable $\boldsymbol{u}(t)$ we consider the various CRAC outlet temperatures $T_{i,out}$ (i.e., the latter ones are controlled to be all equal).

7.1.3 P control

Here only the proportional term from (31) is used resulting in the control signal

$$\boldsymbol{u}(t) = \boldsymbol{K}_p \boldsymbol{e}(t). \tag{33}$$

7.1.4 PI control

Here both the proportional and integral term from (31) is used resulting in the control signal

$$\boldsymbol{u}(t) = \boldsymbol{K}_p \boldsymbol{e}(t) + \boldsymbol{K}_i \sum_{i=0}^{t} \boldsymbol{e}(t-i)\Delta t. \tag{34}$$

7.2 LQR Control

LQR controllers constitute a more advanced feedback control strategy than the PID. With the big difference compared to the PID is that the LQR is a *model-based control strategy*, i.e. it requires a model of the system desired to control

$$\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t). \tag{35}$$

A quadratic cost function is set up based on the model of the system desired to control.

$$J(x_0, U_0) = \sum_{k=0}^{N-1} (x_k' Q x_k + u_k' R u_k) + x_N' Q_f x_N \tag{36}$$

The optimal control feedback $\boldsymbol{u}(t)$ is then found by minimizing the quadratic cost function. This is done by calculating the gradient with respect to $\boldsymbol{u}$, setting the gradient equal to zero and then solving for $\boldsymbol{u}$. Q and R in the cost function is weights. Where Q determines how important it is that the controller keeps the state $\boldsymbol{x}$ at a minimum. While R determines how important it is that the control input $\boldsymbol{u}$ is kept low [3].

A brief summary of the main features for our implementations are described below in Sections 7.2.1 and 11.5.

7.2.1 LQR for our CRAC control problem

In general the models that we derived for our system present time-varying A and B matrices, plus time varying disturbances $\boldsymbol{d}_p$ compared with the linear time invariant LQR problem describe in (35). Our models are thus linear but time-variant of the form

$$\boldsymbol{x}(t+1) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) + D\boldsymbol{d}_p(t). \tag{37}$$

For our control purposes we define the quadratic cost function over the finite horizon N as

$$J(x_t, \bar{U}) = \sum_{\tau=t}^{t+N-1} (x'_\tau Q x_\tau + u'_\tau R u_\tau) + x'_{t+N} Q_f x_{t+N} \tag{38}$$

where $\bar{U}$ is the vector of future inputs $u_t, u_{t+1} \dots u_{t+N-1}$ and x_t is the state vector at the time t. The optimal vector of future inputs $\bar{U}^*$ over the finite horizon N can then be derived trough the batch approach described in [3] as

$$\bar{U}^*(t) = K^x(x_t - r) + K^c\bar{C} + B_t^{-1}(r - A_t r - c_t) \tag{39}$$

where

$$K^x = -(\bar{R} + \bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}})^{-1}(\bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}}^x x_t) \tag{40}$$

and

$$K^c = -(\bar{R} + \bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}})^{-1}(\bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}}^c\bar{C}). \tag{41}$$

The equations for the derivation of $\bar{U}^*(t)$ are presented in appendix A.2. The optimal control input is then simply the first value in the optimal vector of future input, $\boldsymbol{u}^* = \bar{U}^*[1]$ The control algorithm then consists of applying the optimal input on the system, wait until the next sampling time $t+1$ measure the state and repeat by calculating the optimal input again. Implementing the controller in this fashion is also called receding horizon control

7.2.2 LQRi control

To diminish potential detrimental effects on the steady state behavior of the system of imperfect modelling of the system we also consider adding an integral term in the LQR controller defined above. Notice that in this case the control input $\boldsymbol{u}(t)$ is practically the same as the plain LQR plus an integral term, i.e.,

$$\boldsymbol{u}_t^* = K^x(x_t - \boldsymbol{r}) + K^c\bar{C} + B_t^{-1}(\boldsymbol{r} - A_t\boldsymbol{r} - c_t) + K_i\sum_{\tau=0}^{t}\boldsymbol{e}(t-\tau)\Delta t. \tag{42}$$

Notice that this structure shares the integral term as in the PI control – here, nonetheless, K^x and K^c are still the same as in (40) and (41). In any case anti wind-up strategies were implemented for all the controllers using integral actions.

8 SICS ICE data center

Throughout this project we had access to the data center Swedish Institute of Computer Science (SICS) ICE in order to conduct experiments in a real world environment. At SICS ICE we conducted our experiments in their module 2, which is a server room intended for testing of facility and utility innovations[4]. Down in Section 8.1 the set-up of module 2 is described. Access to this data center was provided by Research Institute of Sweden (RISE) SICS North which is a research center situated in Luleå with a focus on data center research [5].

8.1 Module 2

The module 2 server room was set-up with ten racks of servers in two columns of five racks each. The two columns are places so the racks exhaust facing each other creating a hot aisle in the middle of the room seen in Figure 5. The room is outfitted with four CRACs two places in each end of the room seen in Figure 3. They supply the server inlets with cold air creating two cold aisles. So cold air exits the CRACs, enters the servers, heats up, exits the servers, rises and recirculates back to the CRACs, it enters at the top of the CRACs and is cooled down, the cycle continues as described in Section 5 and visualized in Figure 1.

8.1.1 Servers

The ten racks in module 2 were fitted with three different types of servers. Racks 1-3 were fitted with seven *HPE BladeSystem c7000 Enclosures* containing 32 servers each. These enclosures were all turned of during experiments with the exception of one enclosure in rack 1.

The two remaining server models were *Dell poweredge r430* and *Dell poweredge r530*. These two models are similar in hardware, being both outfitted with two CPUs places following each other as in Figure 4. The main difference between the two models is that the r430 is thinner and fitted with 12 40 mm cooling, while the r530 is thicker and fitted with 6 60 mm cooling fans.

Racks 4-7 were fitted with the thinner r430 servers. Each rack housing 30 r430 servers with the exception of rack 4 that only housing 26. While the remaining racks 8-10 were fitted with thicker r530 servers. Each rack housing 16 r530 servers. The total numbers of Dell servers in racks 4 trough 10 summed up to 164 servers. These 164 servers made up our state $\boldsymbol{x}(t)$ or process variable $\boldsymbol{y}(t)$ to be controlled.

8.1.2 Sensors

Module 2 was outfitted with a wide arrange of different sensors. The data we needed were from sensors that correlating with the variables in our models from Section 6. Both CPUs inside the DELL servers had sensors measuring their temperature. We defined the average of these two temperatures as the

server temperature $T_{j,k}$, our state $\boldsymbol{x}(t)$ or process variable $\boldsymbol{y}(t)$. In addition the system comprises sensors for measuring the temperature of the cooling water $p_i(t)$. Notice that, according to (12), we can say that this variable has the same value as the CRAC outlet temperature $T_{i,out}$, i.e., our control variable $\boldsymbol{u}(t)$.

In addition to the previous information, we were allowed to measure the speed (in rpm) of the various servers cooling fans. These variables were defined as the flow forced by the internal server fans, i.e., $u_{j,k}$. Finally we were able to measure also the CRACs fan usage in percent, defining the flow exiting the CRACs $f_{i,out}$. Finally, dedicated sensors measure each server usage $p_{j,k}$ in watts.

Large datasets were gathered from these sensors, and this was then used to identify a numerical estimate of the model parameters, as described in the next Section 9. Appendix **??** describes how the real time monitoring and control of the server room was implemented on top of RISE SICS North building management system.

8.2 Description of the native CRAC control strategy

During the normal operations of the module 2 server room, the cooling water is kept at a constant temperature. The control variable is instead the fan speed of each CRAC, which is controlled via independent P feedback loops starting from the temperature readings from a sensor placed in front of each CRAC at the server inlet side. The sensor can be seen in Figure 3 as the white box at the server inlet.

8.3 Limitations in control

Due to limitations in the infrastructure used for our field experiments, it was not possible to control the temperature of the coolant supplied to the CRACs individually. This implies that the temperature of the air flows from all the different CRACs is structurally limited to be the same, and this reduces for our specific case $T_{i,out}$ to be just T_{out} – which in return limits the possibility of implementing controllers dedicated to our models 2 and 3.

Moreover the cooling water supplied to the CRACs in module 2 had a recommended max/min temperature (respectively $16°C$ and $23°C$) that limited the control signal $\boldsymbol{u}(t)$ usable when implementing the controllers.

9 System identification

Assume to have collected data about server temperatures, usage levels, power consumptions, CRAC cooling levels and their fan speeds through the software interfaces developed in this thesis and by RISE SICS North. It is then possible to use this information to estimate the parameters defined in Section 5 through system identification approaches.

Since the aim of the thesis is to develop control algorithms, the statistical paradigm that should be followed in this estimation step is the standard one of estimating the unknown parameters as that ones that maximize the predictive capabilities of the to-be-identified model. We thus make the standard assumption that the noises corrupting the various measurement and processes are Gaussian, and consider quadratic costs as loss functions. In general, thus, assuming that models such as (14) are used to generate predictions of the temperatures, then the loss has the structure

$$J(\boldsymbol{\theta}) := \sum_{t=1}^{N-1} \Big(y(t+1) - \Psi\left(y(t)\ ;\ \boldsymbol{\theta}\right) \Big)^2 \tag{43}$$

where $\boldsymbol{\theta}$ captures the model parameters, Ψ is the model, and $x(t)$ denotes the output of the system at time t, and N is the number of samples in the training set. This structure of costs leads to the Prediction-Error identification Methods (PEM) estimation strategy

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} J(\boldsymbol{\theta}) \tag{44}$$

with the hypothesis space Θ defined opportunely to guarantee the physical meaningfulness of the various estimated parameters. Once models are obtained from this identification procedure, it is possible to follow the steps described in the previous chapters to define the LQR controllers.

Given that in this thesis we have been proposing several parametric structures for the dynamics of the temperatures, the previous strategy (44) has been specialized and implemented for each of these structures using the `system identification toolbox` in Matlab. Some steps have nonetheless been implemented in the same way irrespectively of the model structure to be identified. More precisely,

- the process of selecting what should be considered inputs and outputs of the system was performed at the beginning of the thesis, and its results are implicitly summarized by the choices performed in Section 5;
- the process of designing the experiments to be performed at SICS was performed independently of the to-be-identified model structure. The experiments, that will be described in more details in Section 11, consisted in a series of randomly placed step-changes of the input signals with random amplitudes;

- collecting and processing of the raw data was the same among all the various identification procedures;
- validation of the identified models was always performed by means of checking fit indexes on independent validation data.

Incidentally, we report that the best model structure obtained was the ARX one, with a fit on the validation data of 72 percent. The dataset used for training and the simulation results on the validation data are plotted respectively in Figures 6 and 7.

Figure 6: Graph of the training dataset. Where the blue line is the measured server temperature and the orange line is the server usage from the training data set.

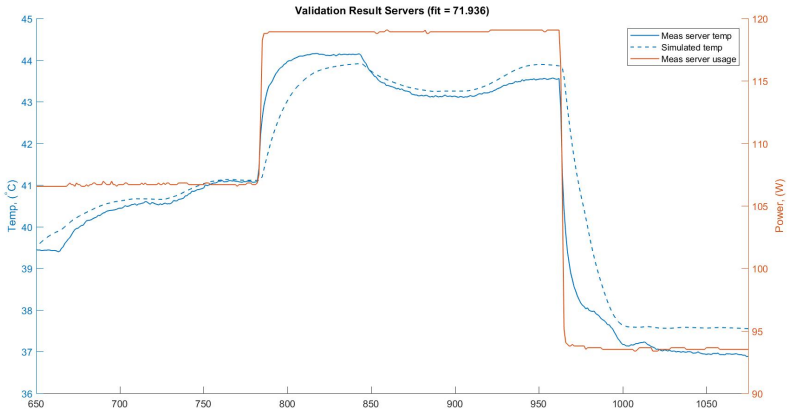

Figure 7: Graph of the simulation results on the validation data. Where the blue line is the measured server temperature and the orange line is the server usage from the validation data set. The blue dashed line is the simulated server temperature that have a fit of 72 % to the measured server temperature.

10 In silico tests

The plan at the start of the project were once the models had been identified from real world data to run simulations in order to test and tune the different controllers. But due to time constraints and the desire to test the different controllers on the real world data center work progressed on to field experiments.

11 Field experiments

Five different scenarios were tested the 4 different controllers discussed in Section 7 the simpler P and PI controllers and the model dependent LQR and LQRi controllers. The final scenario is the reference scenario, the open loop case, described in Section 8 where the data center control variable is the CRAC fans speed that are feedback control by temperature sensors at the CRAC inlets. Experiment set-up was a 2 hours long test were the controllers first ran for 30 min with no server load to reach a steady state then a step in server load up to 30% for 30 minutes then a second step to 60% in server load for an for an additional 30 min and final 30 minutes the system had no load once again. For all 4 controllers the reference value was set to $\boldsymbol{r} = 38$ degrees.

11.1 Open loop

Figures 8, 9 and 10 shows the field experiments results from the open loop reference experiment.

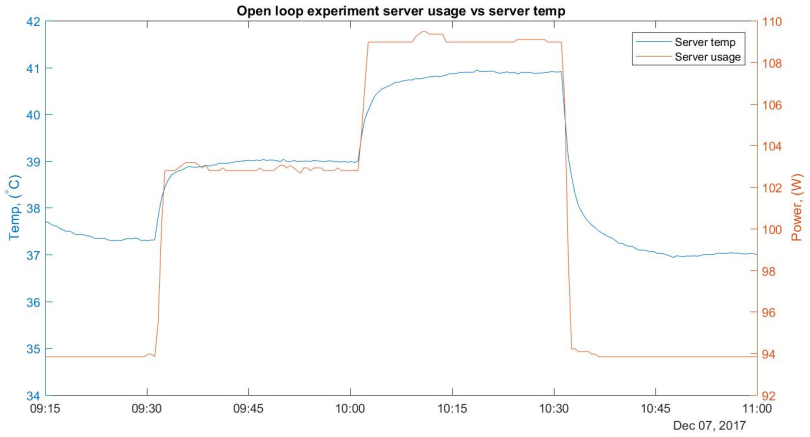

Figure 8: Graph of the server temperature and the server load during the open loop experiment. We that the server temperature increase in a proportional way in relation to the server load.

11.2 P control

Figures 11, 12 and 13 shows the field experiments results from the P control experiment.

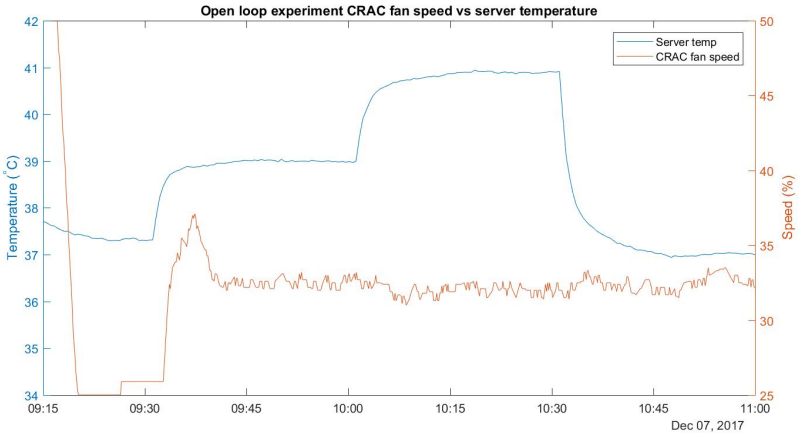

Figure 9: Graph of the server temperature and the CRAC fan speed during the open loop experiment. We have the system not achieving steady state before the first step response, but we still see that a increase in the server temperature doesn't induce a big response on the control variable, the CRAC fan speed

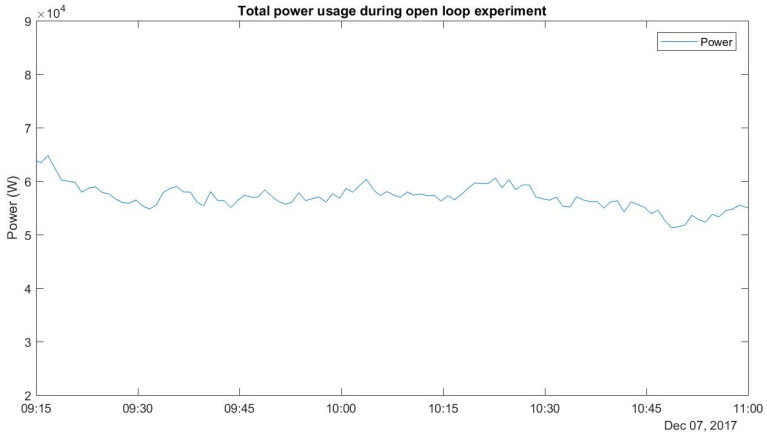

Figure 10: Graph of the total power usage of the data center during the open loop experiment. This includes the energy of the cooling water, the CRACs and all of the IT equipment. We see that the is fairly constant during the entire run of the experiment with the exception of a peak in the beginning when the system is settling into a steady state.

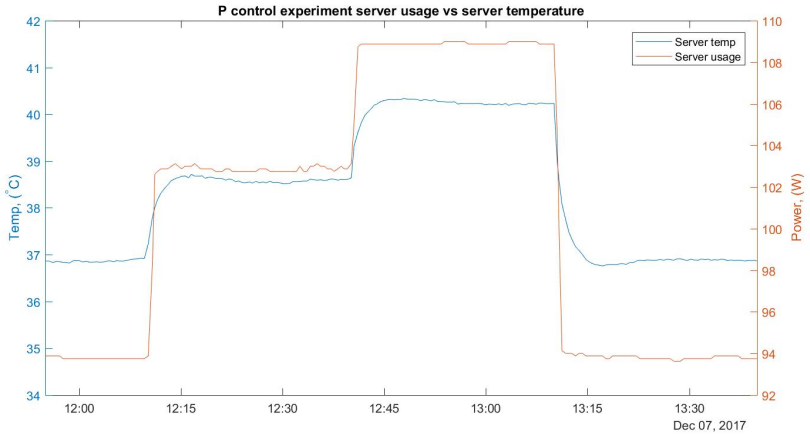

Figure 11: Graph of the server temperature and the server load during the P controller experiment. As to be expected we see a proportional relation between the server temperature and the server usage. The same way as for the open loop experiment but with lower server temperatures.

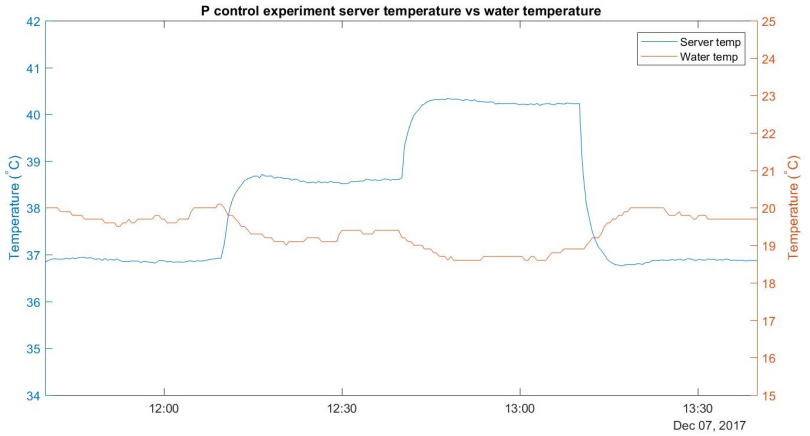

Figure 12: Graph of the server temperature and the water temperature during the P controller experiment. We see quite small changes in control variable, indicating a bigger K_p could have been used.

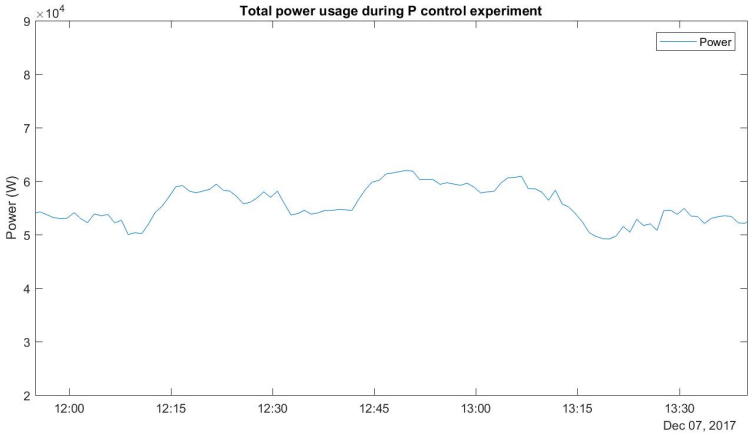

Figure 13: Graph of the total power usage of the data center during the P controller experiment. This includes the energy of the cooling water, the CRACs and all of the IT equipment. We only see small increases in power usage at the points in time when the server load increased and therefore also cooling power.

11.3 PI control

Figures 14, 15 and 16 shows the field experiments results from the PI control experiment. We see in figure 14 that the system having problem reaching a steady state before the load loads was applied. Only by the end of the second load the system seem to stabilize around the reference value $\boldsymbol{r}$. This is due to the integral starting at zero, on could run an experiment finding the integral at steady state and the use that as a starting value to reduce ramp-up time, this same problem occurs in the LQRi experiment. We have a big dip in server temperature when the load goes back to zero due inertia the cold air introduces to the system. Used the same K_p as for the P controller and from that we saw that it could have had bigger gain.

11.4 LQR control

Figures 17, 18 and 19 shows the field experiments results from the LQR control experiment.

11.5 LQRi control

Figures 20, 21 and 22 shows the field experiments results from the LQRi control experiment.

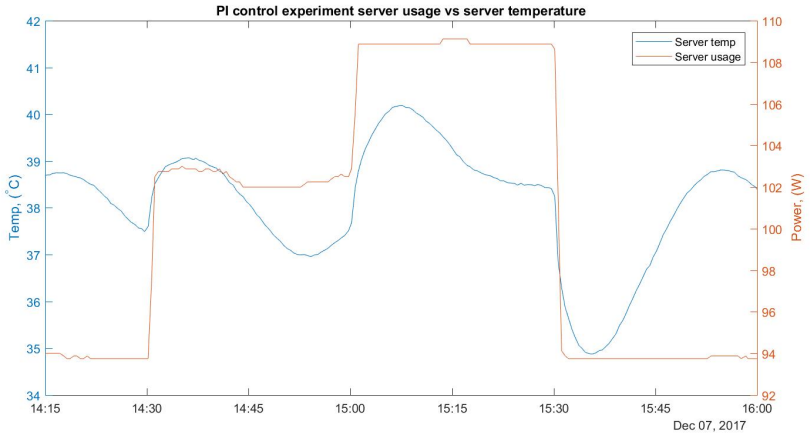

Figure 14: Graph of the server temperature and the server load during the PI controller experiment. We can see there no longer is a proportional relation between the server temperature and server usage.

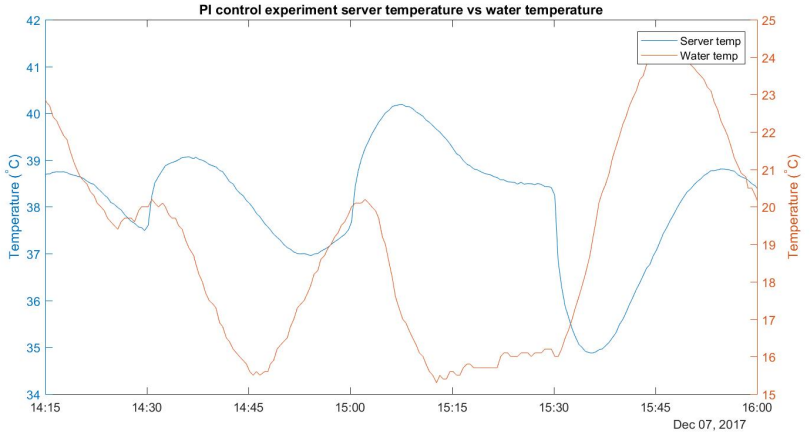

Figure 15: Graph of the server temperature and the water temperature during the PI controller experiment.

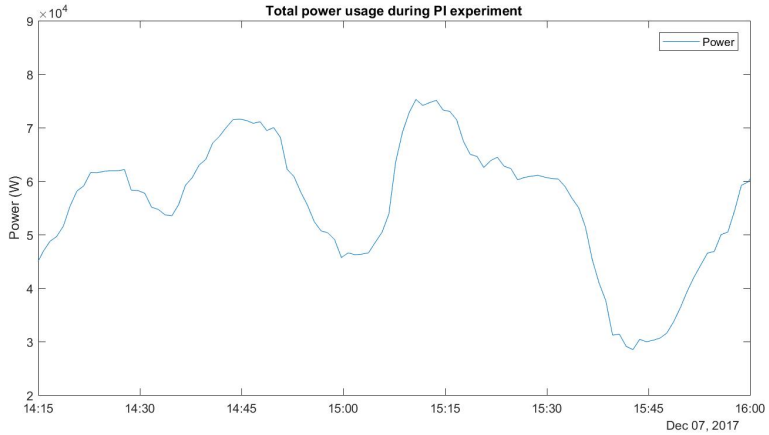

Figure 16: Graph of the total power usage of the data center during the PI controller experiment. This includes the energy of the cooling water, the CRACs and all of the IT equipment. We see a big drop in power usage when the server load decreases since the cooling water don't have to be cooled.

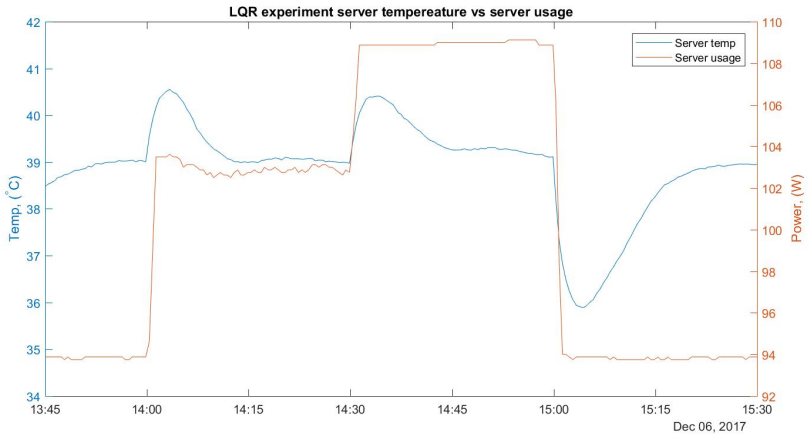

Figure 17: Graph of the server temperature and the server load during the LQR controller experiment. We see despite the changes in server load the controller manages to return the system to a steady state but with a error from the reference $\boldsymbol{r} = 38$.

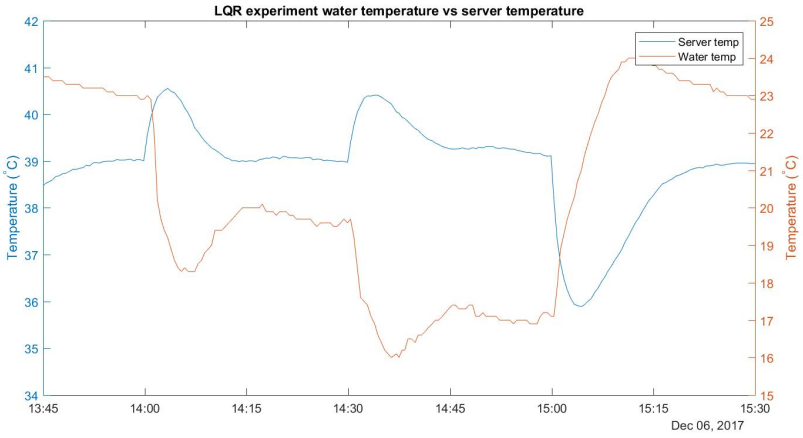

Figure 18: Graph of the server temperature and the water temperature during the LQR controller experiment.

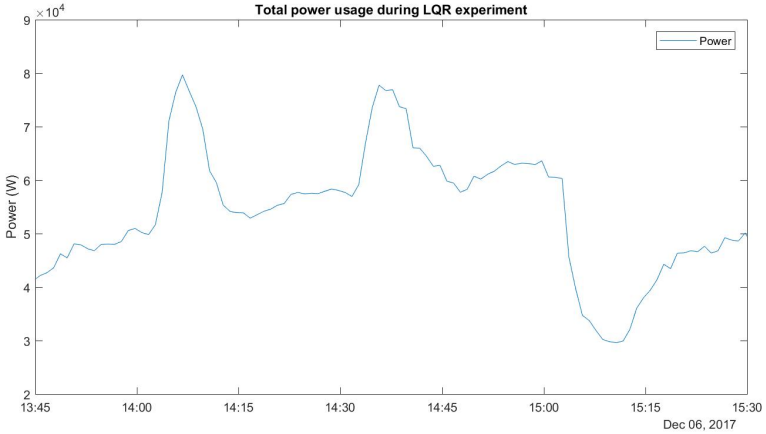

Figure 19: Graph of the total power usage of the data center during the LQR controller experiment. This includes the energy of the cooling water, the CRACs and all of the IT equipment.

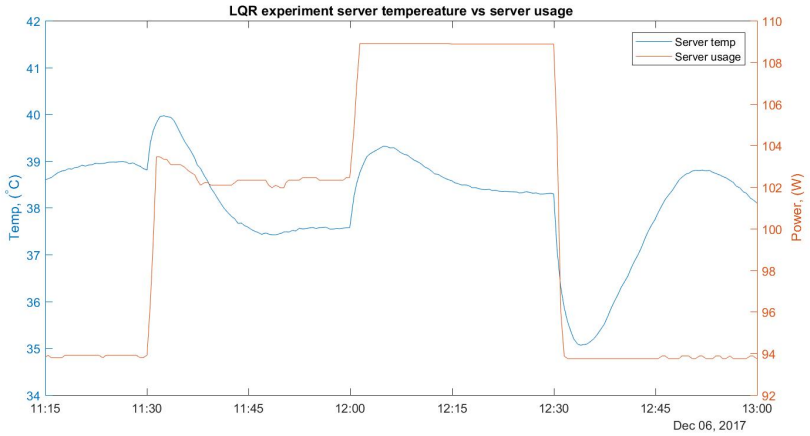

Figure 20: Graph of the server temperature and the server load during the LQR controller experiment.

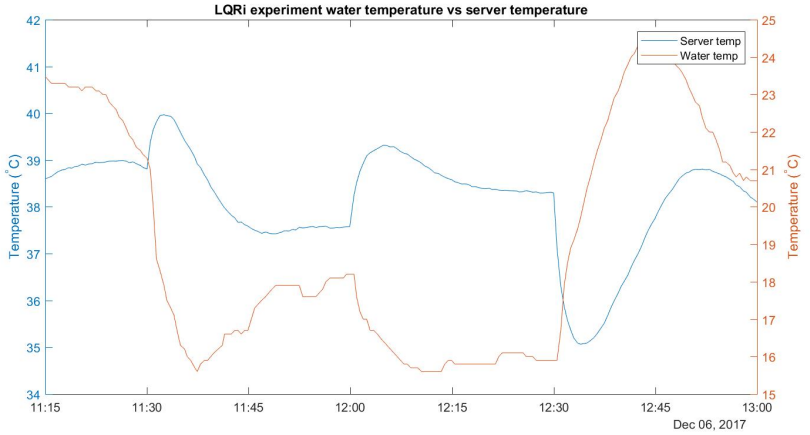

Figure 21: Graph of the server temperature and the water temperature during the LQRi controller experiment.

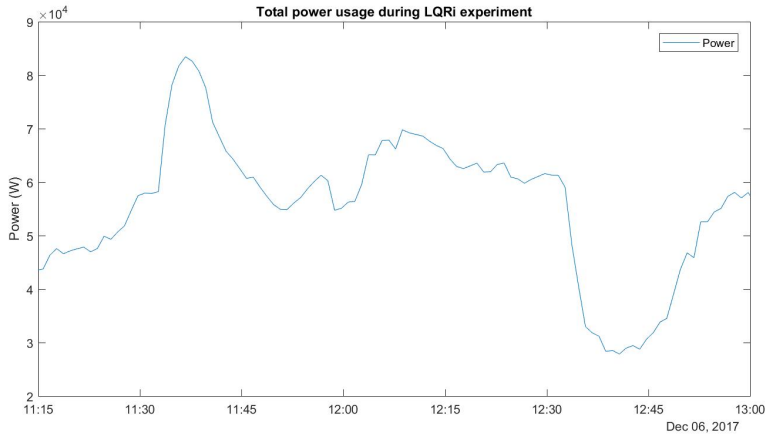

Figure 22: Graph of the total power usage of the data center during the LQRi controller experiment. This includes the energy of the cooling water, the CRACs and all of the IT equipment.

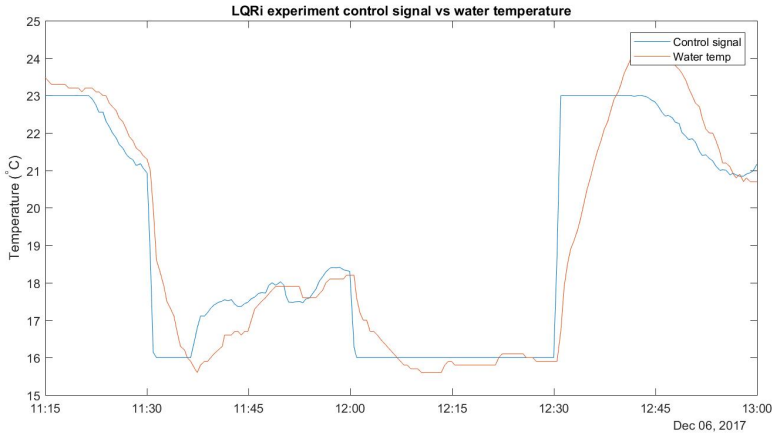

Figure 23: Graph of the water temperature versus the actual control signal sent. We see that the water temperature lags behind, this due to how fast it is able to actually change.

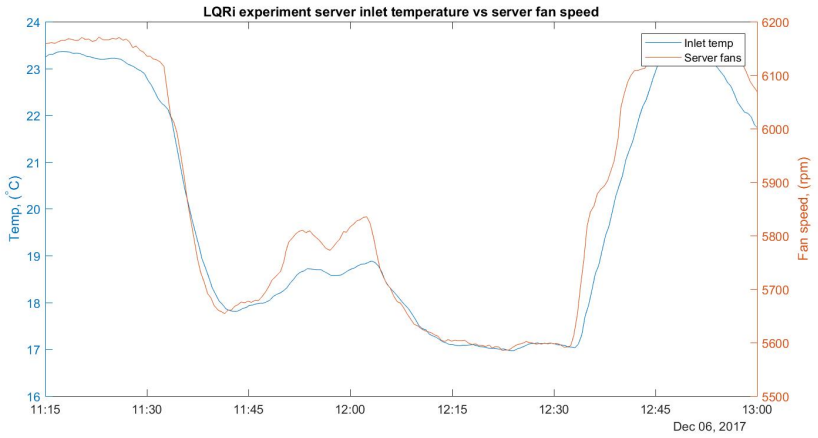

Figure 24: The server fan speed vs server inlet temperature. We clearly see that the server inlet temperature have a dominating effect on the server fan speed.

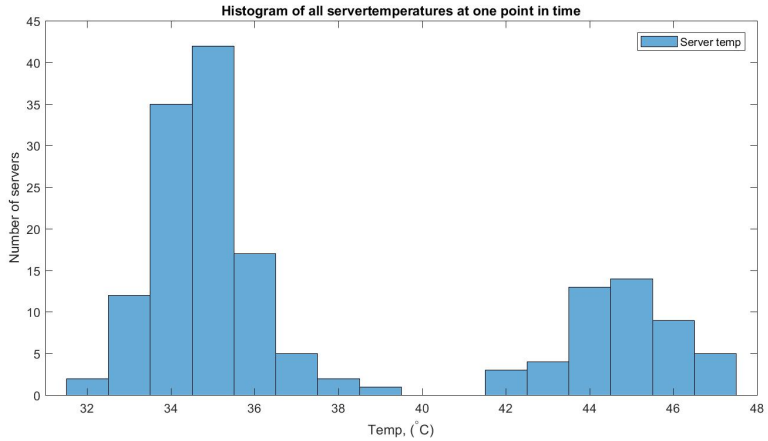

Figure 25: Histogram of all the server temperatures at one point in time. We clearly see two separate peaks corresponding to the two different type of servers in the data center.

12 Conclusions and future work

12.1 Conclusions

We derived and identified several models of the thermal dynamics within air-cooled data centers, and with these dynamics we designed and tested with field experiments ad-hoc LQR and LQRi controllers for CRAC units. Testing these controllers against reactive ones (P and PI) we notice that the newly developed controllers respond faster to changes in the server temperatures, and that enable reducing the risk of over-cooling the computer rooms. This in its turn can be associated to energy savings, the goal that we set for ourselves at the beginning of the work.

Despite successful, the controllers haven't been tuned accurately – some experiments indicated indeed that performance could have been further improved having had some more time for testing different parametric setups. Nonetheless the fact that the developed controllers performed acceptably even if without having been extensively tested indicates that the proposed solution is, at least in the system where we performed experiments, robust.

The correlation between the server inlet temperature and server fan speed is something that makes sense in the open loop case. Since the reference temperature the CRACs tries to keep is the server inlet temperature. But for the controllers we tested, where the reference temperature the CRACs instead tries to keep a certain server temperature, this doesn't make sense. Because when the server temperatures are low and our CRAC controller pushes hotter air the server fans will ramp up and when the servers run hot the server fans will slow down. This behaviour is quite the opposite of what you would like. So also implementing a controller for the server fans with the server temperature as reference would probably further improve performance. One specific thing that would decrease is the spikes in temperature at the step in server load since the local server fans has a much faster response then the CRACs.

12.2 Future works

This work – and the associated implementation efforts – opens the possibility of implementing even more advanced control strategies. For example, we suspect that Model Predictive Control (MPC) strategies would be even better suited than LQR ones, since these would allow more flexibility in managing the server temperatures, instead of the LQR case where the controller constantly tries to reach a certain temperature and does not exploit information about potential constraints in the control actions or in the state of the system.

An other source of improvement would be running longer experiments and performing a more thorough system identification of the system. This would indeed lead to better estimates of the parameters of the physical models, and this is expected to translate into better control.

As mentioned in sec 12.1 developing controllers for the local server fans that work in sync with the our CRAC controller rather then in opposition.

Finally we think that it is devisable to run further tests and tuning of all the aforementioned controllers (i.e., P, PI, LQR, LQRi, and MPC), so to benchmark their performance in a structured way.

A Appendix

A.1 Derivation of model a case specific time-delay (5)

$$\Delta_{i\to j,k}(t) = \frac{\phi\left(d_{i\to j,k}\right)}{f_{i,\text{out}}(t)}$$

This is a derivation of the time delay $\Delta_{i\to j,k}$ is for the geometry found in module 2 which also is a common geometry for datacenters. The time delay we define as the time it takes for the air exiting the outlet of CRAC i to reach the inlet of server j,k. The time-delay will then be equal to the inverse of the air velocity integrated over the distance travelled from CRAC i to server j,k

$$\Delta_{i\to j,k} = \int_0^{d_{i\to j,k}} \frac{1}{v_{i\to j,k}(d)} dd. \tag{45}$$

If we assume that the total volumetric flow out from the CRACs $f_{i,out}$ will remain constant on its travel to the servers inlets. It is also possible to assume that the velocity function is linear

$$v_{i\to j,k}(d) = kd + m. \tag{46}$$

Using (46) the integral in (45) becomes

$$\left[\frac{1}{k}\ln(kd+m)\right]_0^{d_{i\to j,k}} = \frac{1}{k}\big(\ln(kd_{i\to j,k}+m) - \ln(m)\big). \tag{47}$$

Knowing the velocity $v_{i_j,k}$ at two points along the distance $d_{i\to j,k}$ the linear velocity function (46) can be determined. We can describe the air velocity using known entities at two points, the CRAC outlet $d = 0$ and the server inlet $d = d_{i\to j,k}$. The velocity by which the air leaves the CRAC can be described as the volumetric airflow divided by the CRACs surface area

$$v_{i,out} = v_{i\to j,k}(0) = \frac{f_{i,out}}{a_i}. \tag{48}$$

The velocity at the inlet of the servers is can be described as the fraction of the airflow entering the server inlet divided by servers inlets surface area

$$v_{jk,in} = v_{i\to j,k}(d_{i\to j,k}) = \frac{s_{i\to j,k} f_{i,out}}{a_{j,k}}. \tag{49}$$

Using (46), (48) and (49) we get our velocity function based on measurable entities

$$v_{i\to j,k}(d) = \frac{\frac{s_{i\to j,k} f_{i,out}}{a_{j,k}} - \frac{f_{i,out}}{a_i}}{d_{i\to j,k}}\, d + \frac{f_{i,out}}{a_i}. \tag{50}$$

Now substituting (50) back into (47)

$$\begin{aligned}
&\Delta_{i\to j,k} = \\
&= \frac{d_{i\to j,k}}{\frac{s_{i\to j,k}f_{i,out}}{a_{j,k}} - \frac{f_{i,out}}{a_i}} \left(\ln\left(\frac{s_{i\to j,k}f_{i,out}}{a_{j,k}}\right) - \ln\left(\frac{f_{i,out}}{a_i}\right) \right) \frac{1}{f_{i,\text{out}}(t)} = \\
&= \frac{d_{i\to j,k}}{\frac{s_{i\to j,k}f_{i,out}}{a_{j,k}} - \frac{f_{i,out}}{a_i}} \left(\frac{s_{i\to j,k}f_{i,out}a_i}{f_{i,out}a_{j,k}} \right) \frac{1}{f_{i,\text{out}}(t)} = \\
&= \frac{d_{i\to j,k}}{\frac{s_{i\to j,k}}{a_{j,k}} - \frac{1}{a_i}} \ln\left(\frac{s_{i\to j,k}a_i}{a_{j,k}}\right) \frac{1}{f_{i,\text{out}}(t)}
\end{aligned} \tag{51}$$

A.2 Solution to the LQR-problem via Batch approach

Using the Batch approach described in [3]

Derivation for LQR of a discrete linear time-variant system with a time-variant disturbance. The system we wish to control

$$\boldsymbol{x}(t+1) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) + D\boldsymbol{d}(t) \tag{52}$$

Applying a sequence of N inputs $\bar{U} = [u_t', u_{t+1}'...u_{t+N-1}]'$ at the time t to the system model

$$x_{\tau+1} = A_\tau x_\tau + B_\tau u_\tau + Dd_\tau \tag{53}$$

A quadratic cost function can be defined over N steps into the future called the horizon

$$J(x_t, \bar{U}) = \sum_{\tau=t}^{t+N-1} (x_\tau' Q x_\tau + u_\tau' R u_\tau) + x_{t+N}' Q_f x_{t+N} \tag{54}$$

Using (53) all the future states $x_t, x_{t+1}...x_{t+N}$ of the system can be expressed as a function the current state x_t the future inputs $u_t, u_{t+1}...u_{t+N-1}$ and the

future disturbances $d_t, d_{t+1}...d_{t+N-1}$

$$\begin{bmatrix} x_t \\ x_{t+1} \\ \vdots \\ x_{t+N} \end{bmatrix} = \begin{bmatrix} I \\ A_t \\ \vdots \\ A_{t+N-1} \end{bmatrix} x_t + \\ \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ B_T & 0 & \cdots & \cdots & 0 \\ A_{t+1}B_t & B_{t+1} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ A_{t+N-1}\dots A_{t+1}B_t & A_{t+N-1}\dots A_{t+2}B_{t+1} & \cdots & \cdots & B_{t+N-1} \end{bmatrix} \begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+N-1} \end{bmatrix} + \\ \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ D & 0 & \cdots & \cdots & 0 \\ A_{t+1}D & D & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ A_{t+N-1}\dots A_{t+1}D & A_{t+N-1}\dots A_{t+2}D & \cdots & \cdots & D \end{bmatrix} \begin{bmatrix} d_t \\ d_{t+1} \\ \vdots \\ d_{t+N-1} \end{bmatrix} \tag{55}$$

By defining a notation for the matrices in expression (55) it can be rewritten in a compact form

$$\bar{X} = \bar{\boldsymbol{S}}^x x_t + \bar{\boldsymbol{S}}^u \bar{U} + \bar{\boldsymbol{S}}^c \bar{C} \tag{56}$$

Then applying the same notation to the cost function

$$J(x_t, \bar{U}) = \bar{X}' \bar{Q} \bar{X} + \bar{U}' \bar{R} \bar{U} \tag{57}$$

Substituting (56) into (57) and preforming some algebra

$$\begin{aligned} &J(x_t, \bar{U}) = (\bar{\boldsymbol{S}}^x x_t + \bar{\boldsymbol{S}}^u \bar{U} + \bar{\boldsymbol{S}}^c \bar{C})' \bar{Q} (\bar{\boldsymbol{S}}^x x_t + \bar{\boldsymbol{S}}^u \bar{U} + \bar{\boldsymbol{S}}^c \bar{C}) + \bar{U}' \bar{R} \bar{U} = \\ &x_t' \bar{\boldsymbol{S}}^{x\prime} \bar{Q} \bar{\boldsymbol{S}}^x x_t + \bar{U}' (\bar{R} + \bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}}^u) \bar{U} + 2 x_t' \bar{\boldsymbol{S}}^{x\prime} \bar{Q} \bar{\boldsymbol{S}}^u \bar{U} + 2 x_t' \bar{\boldsymbol{S}}^{x\prime} \bar{Q} \bar{\boldsymbol{S}}^c \bar{C} + \\ &2 \bar{U}' \bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}}^c \bar{C} + \bar{C}' \bar{\boldsymbol{S}}^{c\prime} \bar{Q} \bar{\boldsymbol{S}}^c \bar{C} \end{aligned} \tag{58}$$

Calculating the gradient of (58) with respect to $\bar{U}$

$$\nabla_{\bar{U}} J(x_t, \bar{U}) = (\bar{R} + \bar{\boldsymbol{S}}^{u\prime} \bar{U} \bar{\boldsymbol{S}}^u) \bar{U} + x_t' \bar{\boldsymbol{S}}^{x\prime} \bar{Q} \bar{\boldsymbol{S}}^u + \bar{C} \bar{\boldsymbol{S}}^{c\prime} \bar{Q} \bar{\boldsymbol{S}}^u \tag{59}$$

Setting the gradient equal to zero and solving for $\bar{U}$ gives the array of optimal control

$$\bar{U}^* = -(\bar{R} + \bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}})^{-1} (\bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}}^x x_t + \bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}}^c \bar{C}) \tag{60}$$

The first value in the vector $\bar{U}$ is our optimal control u_t^*.

A.2.1 Following reference signals

The optimal control in (60) can be written more compact by updating the notation in order to make further calculations more comprehensible.

$$K^x = -(\bar{R} + \bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}})^{-1} (\bar{\boldsymbol{S}}^{u\prime} \bar{Q} \bar{\boldsymbol{S}}^x x_t) \tag{61}$$

$$K^c = -(\bar{R} + \bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}})^{-1}(\bar{\boldsymbol{S}}^{u\prime}\bar{Q}\bar{\boldsymbol{S}}^c\bar{C}) \tag{62}$$

Using this notation equation (60) now looks like

$$\bar{U} = K^x x_t + K^c\bar{C} \tag{63}$$

Considering we want the system to reach a different equilibrium, say (x_d, u_d) the optimal control then becomes

$$u_t^* = K^x(x_t - x_d) + K^c\bar{C} + u_d \tag{64}$$

Need to determine x_d and u_d

When the system reaches desired steady state (x_d, u_d) the system looks like

$$x_d = A_t x_d + B_t u_d + c_t \tag{65}$$

Want the output to track the reference signal r substitute $u_d = N_u r$ and $x_d = N_x r$ into (65). $N_x = 1$ since our desired state is the same thing as our reference signal $x_d = r$ so the substitution gives

$$r = A_t r + B_t N_u r + c_t \tag{66}$$

Solving (66) for $N_u r$

$$N_u r = B_t^{-1}(r - A_t r - c_t) \tag{67}$$

Substituting (67) back into (64) and we get the optimal control following the reference signal r

$$u_t^* = K^x(x_t - r) + K^c\bar{C} + B_t^{-1}(r - A_t r - c_t) \tag{68}$$

A.3 Datacollection/closing the control loop

In order to control the system we need to be able to measure the process/state variable $T_{j,k} = \boldsymbol{x} = \boldsymbol{y}$ and be able to set the input $T_i = \boldsymbol{u}$ in real time. For the LQR controller a model of the system is also required. And in order to model the system we need to gather large amounts of data of the different variables effecting the system. Those being, in addition the the ones mentioned above, the server load p_j, k, the server fan speed $u_{j,k}$ and the CRAC fan speed $f_{i,out}$. We also require real time collection of these variables when implementing the LQR controller. The way this was done is what will be described in this appendix

A.3.1 Datacollection required for system identification

In order to get a good and robust model that is capable of operating i many different scenarios we need a large dataset with big variations. So experiments are run in SICS ICE Module 2 were we vary the different variables that we could set at will. Those being the CRACs fan speed, the water temperature to the CRACs and the server load. The remaining variables in the system (the server temperature and server fan speed) could not be set. The server temperature because it's a function of all the other variables and the server fan speed because there were no access to the Dell server software. The datset is collected after the experiment from a Kairos-database at SICS that logs all sensor measurements.

A.3.2 Datacollection required for system control

In order to control the system one doesn't just need to collect the data but one needs to the gather, or sample, the data live and with as little delay as possible. All this was done with a few MATLAB and Python scrips.

A.3.3 OPC

Open Platform Communication (OPC) is a software interface standard that allows Windows programs to communicate with industrial hardware devices. OPC is implemented in server/client pairs. The OPC server is a software program that converts the hardware communication protocol, i our case Modbus see section A.3.4, used by a Programmable Logic Controller (PLC) into the OPC protocol. The OPC client uses the OPC server to get data from or send commands to the hardware [6].

OPC were used to read and write data to and from the CRACs and the cooling water, that both were controlled by an PLC.

The OPC client the OPC server were done in Python using the freeopc package. It was written as a function, thereby reading and writing could be done from MATLAB using a simple function call.

During my tenure at SICS the OPC server were under heavy load at times. This heavy load would lead to OPC request timing out which i return lead to the freeopc package used returning an error crashing the main MATLAB script. Diving into the freeopc package was something there neither was time or knowledge to do. So after having multiple experiments fail due to this problem reading and writing from and to the CRACs and the cooling water was changed to Modbus, see section A.3.4 below.

A.3.4 Modbus

The Modbus Protocol is a messaging structure used to establish master-slave/client-server communication between intelligent devices [7].

Due to the problems with OPC reading and writing from and to the CRACs and the cooling water were switched to Modbus. This was a more direct way of communication since Modbus communicates directly with the PLCs in the data center.

The Modbus implementation was also done in Python. It used the pymodbus package and written as a function in similar fashion to the OPC implementation. After the switch to Modbus the control of the CRACs became more robust and faster.

A.3.5 SNMP

Simple Network Management Protocol (SNMP) is an application–layer protocol for exchanging management information between network devices. SNMP consists of of the managed devices (servers, switches, routers...) all running an SNMP Agent that collects the various variables from the devices locally and

making it available to be queried for by a SNMP Manager. For the manager to be able to request specific information from the clients all the agents have a Management Information Database (MIB) describing all the device variables available, the manager is then loaded with the matching MIB [8].

SNMP was implemented in a python scrip using the PySNMP package. It was at first written as a function like the OPC scrip. The function failed to work when called from MATLAB despite working when called from cmd prompter. The script was rewritten saving the data to a matfile that could be read from MATLAB. The script sent the SNMP requests asynchronously i.e. sent all the requests at once instead of waiting for the previous one to be completed. An synchronous implementation would have been to slow considering the sampling time and the number of servers.

SNMP were used to gather all the data needed from the servers. Those being the power usage of each server, the temperature of both CPUs in each server and the speed of one fan in each server. The choice to only gather fan speed data from one fan per server were due to time constraints, the time to collect more would have taken to long time compared to our sampling time. The data were already gathered through asynchronous SNMP in order to improve performance.

References

[1] Pedca final report summary. technical report. 2014.

[2] Y Wen M Dayarathna and R Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys and Tutorials*, 18(1):732 – 794, 2016.

[3] A. Bemporad F. Borelli and M. Morari. *Predictive Control for linear and hybrid systems*. 2015.

[4] Sics ice. `https://www.sics.se/projects/sics-ice-data-center-in-lulea`.

[5] Rise sics north. `https://www.sics.se/groups/rise-sics-north`.

[6] What is opc. `http://www.opcdatahub.com/WhatIsOPC.html`.

[7] Modbus faq. `http://www.modbus.org/faq.php`.

[8] What is snmp. `https://www.manageengine.com/network-monitoring/what-is-snmp.html`.