Manuscript Details

Manuscript number	JPROCONT_2019_353
Title	Rack-based Data Center Temperature Regulation Using Data-driven Model and Predictive Controller
Article type	Research Paper

Abstract

This work considers the idea of developing a data-driven model based model predictive control (MPC) to regulate temperature for a class of single-rack data centers (DCs). An auto-regressive exogenous (ARX) model is identified for our DC system using partial least square (PLS) to predict the behavior of multi-inputs-single-output (MISO) thermal system. Then an MPC controller is designed to control the temperature inside IT rack based on the identified ARX model. Moreover, fuzzy c-means (FCM) is employed to cluster the measured data set. Based on the clustered data sets, PLS is adopted to identify multiple locally linear ARX models which will be combined by appropriate weights in order to capture the nonlinear behavior of the highly-nonlinear thermal system inside the IT rack. The effectiveness of the proposed method is illustrated through experiments on our single-rack DC and it is also compared with proportional-integral (PI) control.

Keywords	Data-driven model Predictive control Temperature regulation Data centers
Taxonomy	Data Driven Computing, Predictive Control Model, Nonlinear Control System, Multivariable Control System

Corresponding Author	Fengjun Yan
Order of Authors	Shizhu Shi, Jiang Kai, Masoud Kheradmandi, Hosein Moazamigoodarzi, Souvik Pal, Fengjun Yan
Suggested reviewers	Jinxiang Wang, Fei Meng, Pingen Chen

Submission Files Included in this PDF

File Name [File Type]

Hightlights.docx [Highlights]

Manuscript.pdf [Manuscript File]

Conflict of interest statement.docx [Conflict of Interest]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

Temperature regulation in rack-based data center is investigated in this work.

A data-driven model is developed to represent the nonlinear system.

Data-driven predictive control is designed to implement the temperature regulation.

Rack-based Data Center Temperature Regulation Using Data-driven Model and Predictive Controller

Shizhu Shi[a,c], Kai Jiang[a,c], Masoud Kheradmandi[b,c], Hosein Moazamigoodarzi[a,c], Souvik Pal[c], Fengjun Yan[a,*]

[a] *Department of Mechanical Engineering, McMaster University, Hamilton, ON L8S4L7, Canada*
[b] *Department of Computing and Software, McMaster University, Hamilton, ON L8S4L7, Canada*
[c] *Computing Infrastructure Research Centre, McMaster University, Hamilton, ON L8P0A1, Canada*

Abstract

This work considers the idea of developing a data-driven model based model predictive control (MPC) to regulate temperature for a class of single-rack data centers (DCs). An auto-regressive exogenous (ARX) model is identified for our DC system using partial least square (PLS) to predict the behavior of multi-inputs-single-output (MISO) thermal system. Then an MPC controller is designed to control the temperature inside IT rack based on the identified ARX model. Moreover, fuzzy c-means (FCM) is employed to cluster the measured data set. Based on the clustered data sets, PLS is adopted to identify multiple locally linear ARX models which will be combined by appropriate weights in order to capture the nonlinear behavior of the highly-nonlinear thermal system inside the IT rack. The effectiveness of the proposed method is illustrated through experiments on our single-rack DC and it is also compared with proportional-integral (PI) control.

1. Introduction

Due to the rapid and prosperous development of information technology, data centers (DCs) are widely used in every aspect of social life, such as industry, economy or even our daily life. In general, there are three types of DCs. First, the traditional large-scale DCs are usually utilized by cloud providers, such as Amazon or Google. Second, the modular DCs are designed and adopted due to their better performance of energy efficiency, easy assembly and maintenance. Finally, the single-rack micro DC architecture, where IT equipment (ITE) along with its supporting power and cooling infrastructure are all installed within a single IT rack enclosure, has gained significant momentum. These are envisioned as platforms to deploy EDGE computing facilities for a variety of applications. Hence, the focus of work presented herein is single-rack micro DCs.

*Corresponding author
Email address: yanfeng@mcmaster.ca (Fengjun Yan)

Generally, most of the electrical energy consumed by servers in DCs will eventually turn into heat and increase indoor air temperature. So temperature regulation is always a critical issue in all kinds of DCs and our single-rack DC is no exception. It is because that not only we want to reduce power consumption so that we could lower the cost, but we also hope that severs could work normally without any damage in the hot air (temperature of indoor air could reach 40°C during our experiments). Therefore we need to use a cooling system to remove the heat from indoor room to the cold water or ambient air. There are three common types of cooling systems for DCs, air-cooled system, water-cooled system and two-phase cooled system [1] [2]. The cooling system for our single-rack DC is composed of a rack-mountable cooling unit (RMCU) and a chiller. The coolant used here is water that circulates through the cooling system to extract heat from hot IT air and then rejects the heat to a refrigeration system in the chiller. The temperature of cold air coming out of cooling system can be controlled by regulating both water flow rate and fans' speed inside RMCU. Herein we propose a model predictive control (MPC) strategy for modulating these two variables to control the air temperature based on a given set-point.

Different types of control approaches have been employed for temperature regulation. One of the early methods was on-off control [3] [4]. Although on-off is a very cheap form of control, it is not adopted in this work because of the oscillation it causes in the controlled and manipulated variables which will reduce the lifetime of CPU inside servers [3]. Another method is proportional-integral (PI) control which has also been widely adopted. Although PI controller is insensitive to perturbations and does not require complex models, it shows poor control performances for an integrating process or a large time delay process, such as DC's thermal system [5] [6]. Originally, PI controller has also been designed and implemented to control the temperature inside our single-rack DC. However, the performance of designed PI controller is not satisfying because it will lead to large temperature oscillation. Actually, the chips of servers are more easily broken down inside the hot environment with fluctuant temperature. Therefore, PI controller is also excluded. Furthermore, our single-rack DC has a fixed cooling unit location and server configuration which makes MPC an excellent temperature control tool for our DC system because of its better robustness, accuracy and energy-efficiency compared with other methods [7] [8].

An appropriate choice of model is fundamental for MPC. Although physical model could explain the mechanics of thermal system [9], the thermal condition inside DC is very complicated. Moreover, power cables and sensors' wires inside the IT rack might have influence on heat transfer and air flow distribution. Therefore, it is time consuming and impractical to build a relatively accurate first-principle model with simple structure. This is why we choose auto-regressive exogenous (ARX) model for MPC [10] [11]. Based on measured data, we could use partial least square (PLS) and

fuzzy c-means (FCM) to obtain an accurate and combined mathematical model which is suitable for MPC controller design.

The main contributions of this paper are summarized as follows: One combined ARX model is identified based on FCM and PLS and this model can describe most of the nonlinearities of DC thermal system; MPC controller is designed based on the identified ARX model and MPC algorithm to regulate the temperature inside DC system precisely.

This paper is organized as follows: Section II introduces some preliminary knowledge about ARX model, FCM and MPC to make our work more explicit; Section III provides detailed description about the configuration of our hardware system; Section IV emphasizes on ARX model identification using PLS and MPC controller design in Python; Section V presents MPC experimental results under different conditions (total workloads and positions of controlled temperature) and the comparison with PI control.

2. Preliminaries

In this section, some preliminary knowledge and theories about ARX model, FCM and MPC will be introduced.

2.1. ARX Model

In this work, the specific data-driven model expression is based on a parametric identification of an ARX model. ARX model is simple to implement and it can deal with MISO system easily. Basically, ARX model describes the input effects $u(k)$ on the process output $y(k)$ [12]. Mathematically, ARX model is represented by the following structure [10]:

$$y(k) = \sum_{i=1}^{n_y} A_i y(k-i) + \sum_{j=1}^{n_u} B_j u(k-j) + C + v(k) \tag{1}$$

where $y(k)$ and $u(k)$ are the process output and input vectors at sampling instant k; A_i and B_j denote the model parameters; C is a vector representing a bias term, which is important if the model is not identified around an equilibrium point; $v(k)$ is the noise vector; n_y and n_u denote the maximum number of time lags in the outputs and inputs respectively and they also define the order of ARX model.

2.2. Fuzzy C-Means

As we can see, ARX model in the form of Equation (1) is simply a linear difference equation with different orders which might not be able to capture most of the information for our highly-nonlinear DC thermal system. Therefore, FCM is adopted to cluster the data set so as to obtain multiple

locally linear models in this work. Essentially, FCM is a method of clustering which allows one piece of data to belong to two or more clusters [13]. FCM is based on minimizing following objective function [14]:

$$P_m = \sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}^m ||x_i - c_j||^2, 1 < m < \infty \tag{2}$$

where x_i is measure data (M denotes the number of data items); c_j is the center of j^{th} cluster (N is the number of clusters); u_{ij} is the membership function of x_i in j^{th} cluster (m is weighted factor) and subject to $\sum_{j=1}^{N} u_{ij} = 1$; $||*||$ is any norm expressing the similarity between any measured data and the center.

Therefore, we could use iterative optimization of Equation (2) to realize fuzzy partitioning with the update of membership u_{ij} and the cluster centers c_j by [15] [13]:

$$u_{ij} = \frac{1}{\sum_{k=1}^{N}(\frac{||x_i - c_j||}{||x_i - c_k||})^2}, \ c_j = \frac{\sum_{i=1}^{M} u_{ij}^m x_i}{\sum_{i=1}^{M} u_{ij}^m} \tag{3}$$

Iteration will stop when $max_{ij}|u_{ij}^{k+1} - u_{ij}^k| < \varepsilon$, where ε is a termination criterion ($0 < \varepsilon < 1$) and k is iteration step. Finally, this procedure should converge to a local minimum of P_m. Based on this process, the data set could be divided into N clusters with their own centers $c_j, \ j = 1, 2, ..., N$ [16] [17]. Then N locally linear models will be identified using PLS based on these clustered data sets and this topic will be introduced in Section 4.1.

2.3. Model Predictive Control

Essentially, MPC is an advanced process control in which the control action is obtained by solving online iteratively and it is also a finite horizon optimal control problem in which the input sequence is acquired by optimizing objective function [18]. Suppose we are given a model with a general form:

$$x_{k+1} = f(x_k, u_k), \ k = 0, 1, 2, ... \tag{4}$$

where f is a known map which calculates the successor state x_{k+1} using current state x_k and input u_k. Therefore, the state value x can always be obtained as long as we are given the initial state value x_0 and the input sequence $u_k, \ k = 0, 1, ..., N$ by iterating Equation (4).

In MPC, the important concept of prediction horizon which dictates how 'far' we expect controller to predict the future behavior implies that we need to predict for several steps using model equations and obtain the optimal input sequence at each sampling time t_i. Therefore, we need new notation for Equation (4) at each time instant i:

$$x_{k+1|i} = f(x_{k|i}, u_{k|i}), \ k = 0, 1, ..., N \tag{5}$$

where $x_{0|i}$ and $u_{0|i}$ are the initial state and input values at time instant i and $x_{0|i}$ is obtained from the measurement at i; $x_{k|i},\ k = 1, 2, .., N$ are the predicted state values at i; $u_{k|i},\ k = 1, 2, .., N$ are the optimized input values at i.

Up to now, one problem about how to determine the input sequence $u_{k|i},\ k = 1, ..., N$ appears. Actually, this is exactly MPC controller's task, meanwhile, $x_{k|i},\ k = 1, ..., N$ should always be as close as possible to the reference x^{ref}. To this end, the distance between $x_{k|i}$ and x^{ref} is calculated by the objective function $l(x_{k|i}, u_{k|i})$. Generally, objective function is in the quadratic form:

$$l(x_{k|i}, u_{k|i}) = Q(x_{k|i} - x^{ref})^2 + P(u_{k|i} - u^{ref})^2,\ k = 1, ..., N \tag{6}$$

where Q and P are two weighting parameters, x^{ref} and u^{ref} are the reference values for states and inputs, respectively. Based on Equation (6), we are penalizing both state values $x_{k|i}$ and control values $u_{k|i}$ to their set-point. Actually, there might not exist u^{ref} in practical use. So we choose this term to reduce the gradient of input changes during our MPC controller design. At last, the optimizing target in this work is to minimize the following equation:

$$P_N = min \sum_{k=1}^{N} [Q(x_{k|i} - x^{ref})^2 + P(u_{k|i} - u_{k-1|i})^2], k = 1, ..., N \tag{7}$$

The optimizing process above is just for one sampling time t_0. And at this time instant, once the optimization process yields a finite control sequence, the first control action in this sequence will be applied to the plant. Then we will repeat the same procedure at time instants $t_1, t_2, ...$ with new measurements and set these new measurements as the initial values for the prediction and optimization of each time instant [19] [20].

Now we could put forward the MPC algorithm which guides our MPC controller design in this research work [18] [21] [20] [22].

MPC Algorithm: At each sampling time $t_i, i = 0, 1, 2, ...$:

(1) Measure current state value x_i;

(2) Set $x_{0|i} = x_i$;

(3) Solve the optimizing problem via Equation (7) with respect to $u_{lb} < u_{k|i} < u_{ub}$, subject to $x_{0|i} = x_i$ and $x_{k+1|i} = f(x_{k|i}, u_{k|i}), k = 0, 1, ..., N$, where $u_{lb},\ u_{ub}$ are the lower and upper bounds for input values, respectively;

(4) Obtain the optimal input sequence and apply the first value in this sequence to the system;

(5) Repeat procedure (1) ˜ (4) for each sampling instant until the end. □

3. System Description

Our single-rack DC is made up of a cooling system and servers as shown in Fig. 1.

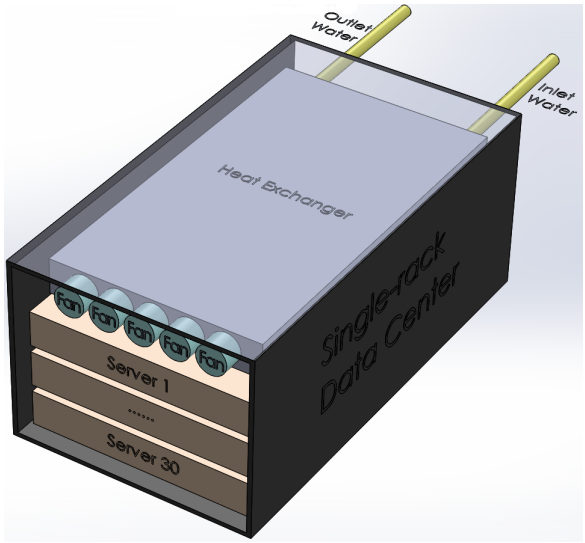

Figure 1: Single-rack Data Center System Configuration

Actually, this kind of single-rack DCs have obvious differences from large scale DCs. The major difference lies on the cooling systems. For most large scale DCs, there exists a computer room air conditioning (CRAC) unit which is used to control the temperature and humidity in the DC in order to make sure the servers could work properly and safely [23] [24]. Different from CRAC which is mounted in an independent space outside the IT racks, our RMCU is situated inside the rack. More specifically, RMCU is mounted on top of the rack during experiments.

3.1. Cooling System

Just as mentioned above, there are two crucial components in our cooling system which are chiller and RMCU. from IT.

Chiller provides chilled water for cooling system and also extract heat from cooling system. It should be pointed out that the temperature of chilled water is not under our control during experiments because chilled water is supplied from building chilled water management system. The inlet water temperature is normally within 12.5°C ˜ 17°C. Therefore, this inlet water temperature is chosen as a measured disturbance since this variable has significant influence on the controlled temperature but we cannot manipulate it.

RMCU consists of one heat exchanger, five identical fans and one valve as shown in Fig. 2. The valve controls flow rate of chilled water coming from the chiller. Then the fans will blow the cold air to the hot aisle which is in front of the rack and suck the hot air from the back of the rack into the heat exchanger. Thus, heat exchanger plays the role of transferring the heat from hot air to cold water and cooling the hot air

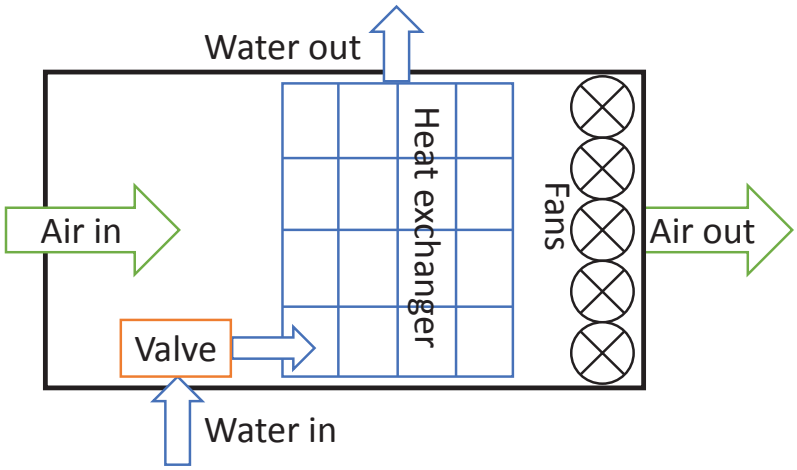

Figure 2: Schematic of Rack-mountable Cooling Unit

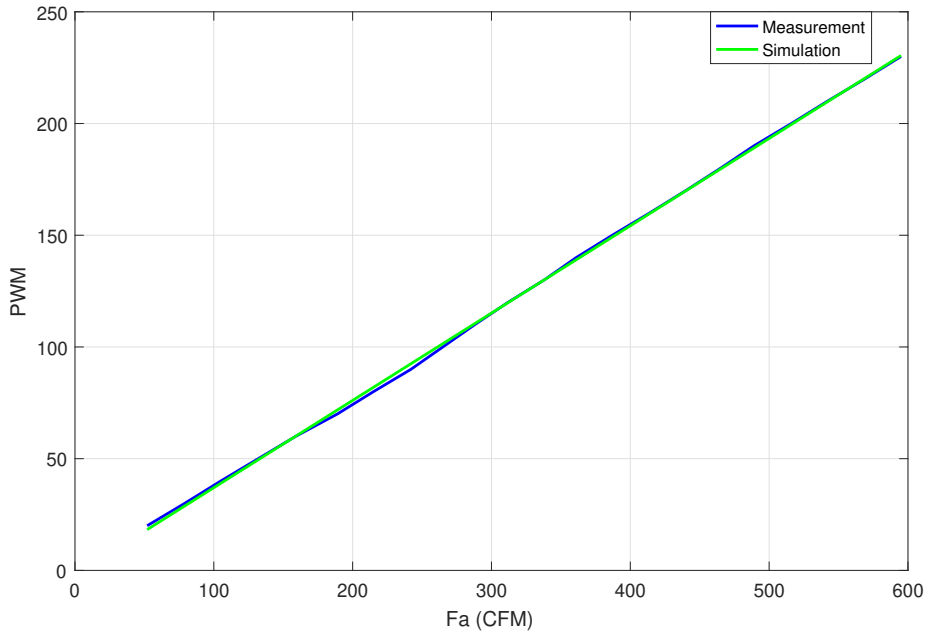

Figure 3: Air Flow Rate and PWM Correlation

As we can see, there are two manipulated variables in RMCU, air and water flow rate. Unfortunately, it is impossible to change these two variables directly. What we could control directly are fans' speed and duration of opening/closing valve. Moreover, fans' speed is adjusted via the signal of pulse width modulation (PWM) generated in Arduino. Essentially, PWM is a technique for getting analog results with digital means and the scale of PWM in Arduino is within 0 ~ 255. Therefore,

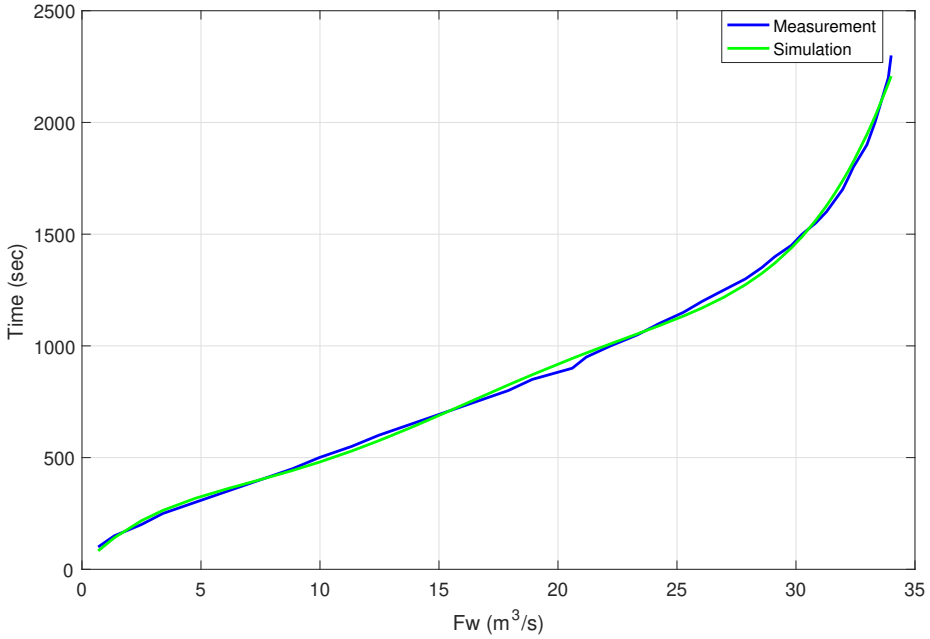

Figure 4: Water Flow Rate and Valve Opening Time Correlation

we need to model valve and fans to derive the correlation between flow rate and directly-controlled variables. Some experiments are conducted in order to get the relationship by changing fans PWM and duration of opening/closing valve so that we could get the corresponding flow rate values. Finally, two polynomials are obtained by curve fitting as shown in Fig. 3 and Fig. 4. And these two polynomials are shown in Equation (8).

$$\begin{aligned} PWM &= 0.3963 \times F_a - 4.0126 \\ Time &= 0.0006 \times F_w^5 - 0.0463 \times F_w^4 + 1.2938 \times F_w^3 - 15.6073 \times F_w^2 + 113.7962 \times F_w + 12.2922 \end{aligned} \tag{8}$$

where F_a and F_w are air and water flow rate, respectively; $Time$ is the duration of opening valve.

3.2. Distribution of Servers

Another important 'input' to the system is the total workload of servers. The temperature inside the IT rack also depends on how many servers are working. For the purpose of our experiment, there are totally 30 servers in the IT rack and the average workload of each server is around $250W$. Temperature control experiments are conducted under three different workloads, $2kW$, $3kW$ and $4kW$. Therefore, some servers should be turned on (we call them as active servers) and others should be turned off (passive servers) to reach a desired load. During the experiments, we also tried to ensure that active servers and passive servers distribute uniformly.

In conclusion, there are three inputs (two manipulated variables and one measured disturbance) in this system which we need to take into account during MPC controller design and all related variables are shown in Table 1.

Table 1: Description of MPC Variables

Symbol	Description	In Single-rack DC
u	Manipulated variable(Input)	Air and Water Flow Rate
y	Measured output	Temperature of front rack
$\bar{y}$	Controlled variable (Output)	Temperature of front rack
r	Set-point (Reference)	Chosen by users
v	Measured disturbance	Inlet water temperature
d	Unmeasured disturbance	Total power of servers

4. Data-driven MPC for Data Center Temperature Control

In this section, the process of identifying ARX model for the system and designing MPC controller using Python will be introduced.

4.1. ARX Model Identification

As mentioned in Section 2.2, FCM is employed to cluster the data. Therefore, if the measured data set which contains inputs and outputs has already been clustered using FCM, we could further use some certain methods to derive multiple locally linear ARX models. And PLS is adopted in this work. Basically, there are several steps to implement PLS:

PLS Algorithm[10] [25][26][27]: (1) Given the original measured data set including X_0 $(N*m)$ and Y_0 $(N*n)$. In other words, there are totally N samples and the numbers of features in X_0 and Y_0 are m and n, respectively. Then we can obtain X and Y by standardizing X_0 and Y_0.

(2) Suppose that the axial vectors (direction of projection) of the first principal component of X and Y are p_1 $(m*1)$ and q_1 $(n*1)$ (note: p_1 and q_1 are still unknown variables which need to be solved later).

(3) Obtain the first principal components of X and Y: $u_1 = Xp_1, v_1 = Yq_1$.

(4) Ensure $\mathrm{Var}\,(u_1) \to \max, \mathrm{Var}\,(v_1) \to \max$ and $\mathrm{corr}\,(u_1, v_1) \to \max$ [28] [29] [30]. We could obtain the optimization objective of PLS by combining these two together:

$$\begin{aligned} &\text{Maximize } \langle Xp_1, Yq_1 \rangle \\ &\text{Subject to: } \|p_1\| = 1, \|q_1\| = 1 \end{aligned}$$

It seems that the objective function of PLS is simple. However, this is not true because p_1 and q_1 are just the first principal components and there are still other components together with their own objective functions needed to be solved.

(5) Solve p_1 and q_1 by using Lagrangian multiplier. It turns out that p_1 and q_1 are the eigenvectors corresponding to the maximum eigenvalues of X^TYY^TX and Y^TXX^TY, respectively. Now we can obtain u_1 and v_1 via $u_1 = Xp_1,\ v_1 = Yq_1$. But this is not enough because we only find the correlation between u_1 and v_1 instead of X and Y. So we need to build the regression equation:

$$\begin{aligned} X &= u_1c_1^T + E \\ Y &= v_1d_1^T + G \end{aligned} \tag{9}$$

where c_1 and d_1 are different from p_1 and q_1 but they have relationship which will be proven later; E and G are residual matrices.

(6) Build the regression equation between u_1 (the first principal component of X) and Y so that we can relate X to Y:

$$Y = u_1r_1^T + F \tag{10}$$

(7) Solve Equation 9 and Equation 10 using ordinary least square (OLS) and the solution is:

$$c_1 = \frac{X^Tu_1}{\|u_1\|^2} \quad ,\ d_1 = \frac{Y^Tv_1}{\|v_1\|^2} \quad ,\ r_1 = \frac{Y^Tu_1}{\|u_1\|^2} \tag{11}$$

Then we can obtain the relationship between p_1 and c_1: $p_1^Tc_1 = p_1^T\frac{X^Tu_1}{\|u_1\|^2} = \frac{u_1^Tu_1}{\|u_1\|^2} = 1$. It is also the same with q_1 and d_1. Generally, p_1 and q_1 are the direction vector used to project original data; c_1, d_1 and r_1 are calculated by OLS to minimize residuals.

(8) Then we can use residual E as the new X, residual F as the new Y. So we can obtain the similar principal components: $u_2 = Ep_2,\ v_2 = Fq_2$. Then p_2 and q_2 can be solved as we did in step (5). Similarly, p_2 and q_2 are the eigenvectors corresponding to the maximum eigenvalues of E^TFF^TE and F^TEE^TF, respectively. We can also get the coefficients of regression equations similar to step (6) and step (7): $c_2 = \frac{E^Tu_2}{\|u_2\|^2},\ r_2 = \frac{F^Tu_2}{\|u_2\|^2}$. Then we can get and solve the new regression equations: $E = u_2c_2^T + E',\ F = u_2r_2^T + F'$ by recycling the same procedure until there is not residuals or the values of residuals meet the requirement for accuracy. Eventually, we can obtain following equations:

$$\begin{aligned} X &= u_1c_1^T + u_2c_2^T + u_3c_3^T + \cdots + u_nc_n^T + e \\ Y &= u_1r_1^T + u_2r_2^T + u_3r_3^T + \cdots + u_nr_n^T + f \end{aligned} \tag{12}$$

Moreover, Equation (12) can also be transformed into matrix form:

$$\begin{aligned} X &= UC^T + E \\ Y &= UR^T + F = XPR^T + F = XB + F \end{aligned} \tag{13}$$

Basically, Equation (13) is the final form for the regression equation of $X \to Y$ where $B = PR^T$ is the coefficient matrix. □

Eventually, after using FCM and PLS, all the identified locally linear models can be combined by appropriate weights:

$$y(k) = \sum_{g=1}^{G} w_g [\sum_{i=1}^{n_y} a_{i,g} y(k-i) + \sum_{j=1}^{n_u} b_{j,g} u(k-j) + C_g] \tag{14}$$

where w_g is the weight belonging to g^{th} model of all the models, which is related to the membership function in FCM, C_g is the constant in g^{th} model. The data-driven model of our single-rack DC system could be acquired based on Equation (14):

$$\begin{aligned} T(k) = &\sum_{i=1}^{N_t} a_i T(k-i) + \sum_{j=1}^{N_{u1}} b_j u_1(k-j) + \\ &\sum_{l=1}^{N_{u2}} c_l u_2(k-l) + d_1 T_{in}(k) + d_2 W(k) + C \end{aligned} \tag{15}$$

where k is sampling instant, T is the controlled temperature (the middle or bottom of front rack), u_1 is the first input (air flow rate), u_2 is the second input (water flow rate), T_{in} is the measured disturbance (inlet water temperature), W is servers' total workload and C is a bias term. Experimental data shown in Fig. 5 is used to train and identify ARX model for the single-rack DC system based on FCM and PLS.

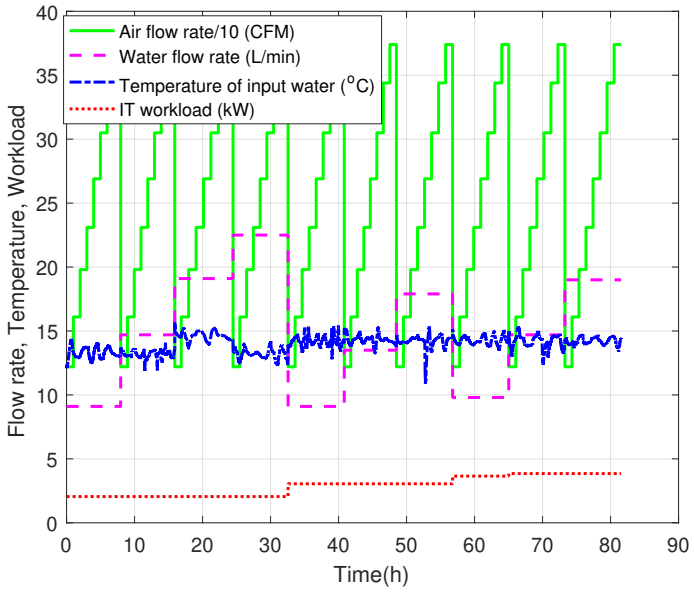

Figure 5: ARX Model Training Data

The identified ARX model using this set of data is:

$$T(k) = 0.9801 \times T(k-1) - 0.0002 \times u_1(k-1) - 0.0096 \times \\ u_2(k-1) + 0.0075 \times T_{in}(k) + 0.0068 \times W + 0.4075 \tag{16}$$

As mentioned before, ARX model could have different orders and we just choose first-order ARX model in this work. The choice of this simple model is based on the fact that the accuracy is already good enough and the computational load is not so heavy for Raspberry Pi in which the computational process of MPC controller is conducted because the computational capability of Raspberry Pi is not as good as computer. Validation result shown in Fig. 6 also reveals that measured data has a good match with simulated result using this first-order ARX model.

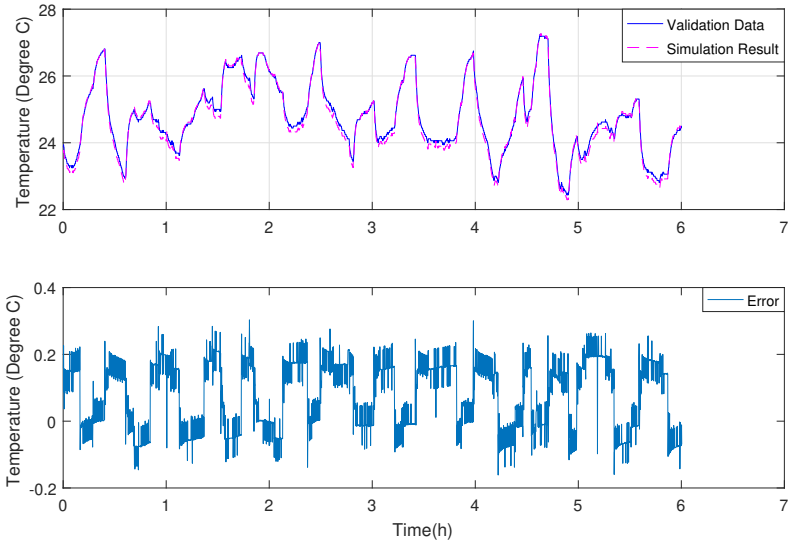

Figure 6: ARX Model Validation Results

4.2. MPC Controller Design

The procedure of MPC controller design follows **MPC Algorithm** as introduced in Section 2.3. And this algorithm is implemented in Python environment. The structure of Python code is shown in Fig. 7. Three functions are defined in Python for different purposes. *Main Function* sets some conditions, such as initial values for predicted inputs and outputs, upper/lower bounds and some constraints for inputs; *Objective Function* defines the cost function which *Main Function* could invoke to use *minimize* so as to obtain the optimal inputs. And *minimize* is a Python command to find a minimum of a constrained nonlinear multi-variable function. Moreover specifically, *minimize* is used to solve *Equation 7* so as to get the optimal predicted input values; *Model Function* is where

we put the ARX model. *Objective Function* could invoke the model to predict future behaviors and calculate cost function using predicted values. Moreover, *Main Function* will use the model to calculate the states of current step. However, this is just for simulation and the current state value can be obtained using sensors in practical use as shown in the red dashed rectangle. The Python code is running in the master computer (Raspberry Pi Zero). In order to control the temperature in the rack, we also need a micro-controller (Arduino Nano) to be the slave computer. On the one hand, Arduino will send commands (PWM and delay time to open/close valve) which obtained from Raspberry Pi to actuators (fans and valve). One the other hand, Arduino also plays the role of collecting data from sensors (temperature sensors and flow meter) and sending them to Raspberry Pi so that MPC controller could know the necessary information about the system.

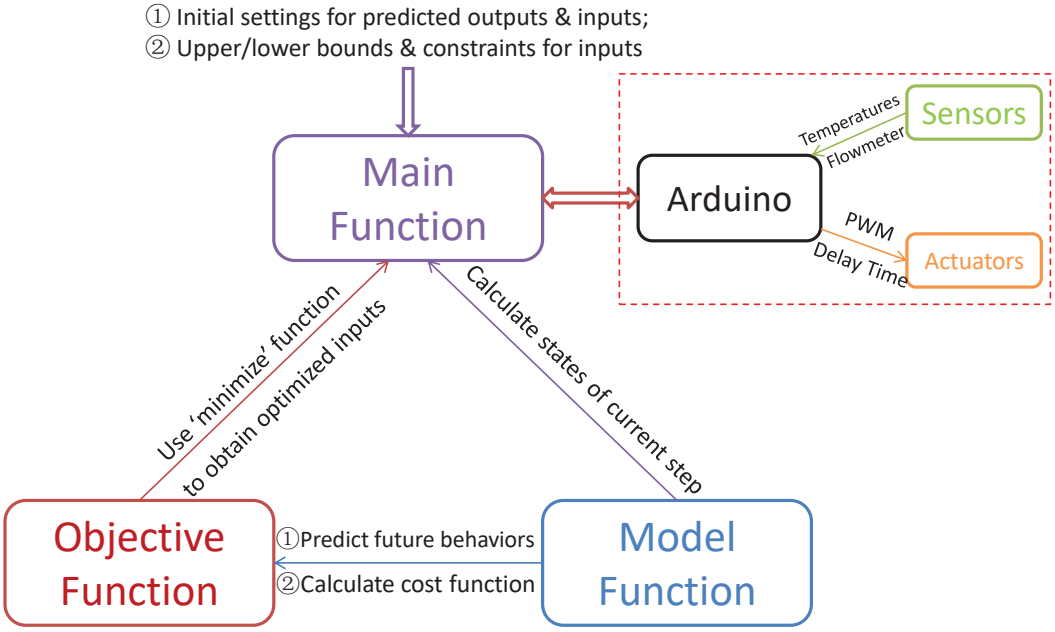

Figure 7: MPC Controller Design in Python

5. Experimental Results

The MPC temperature control experiments are conducted under different conditions in our single-rack DC system. The designed MPC controller is also compared with PI controller through experiments. It should be pointed out that inlet water temperature (T_{in}) has significant impact on the experimental results . If T_{in} is too high/too low, the controlled temperature cannot reach the set-

point although two inputs are already at the maximum/minimum values. The optimal T_{in} lies within 13 ˜ 15 °C based on observation.

5.1. MPC Temperature Control Under Different Workloads

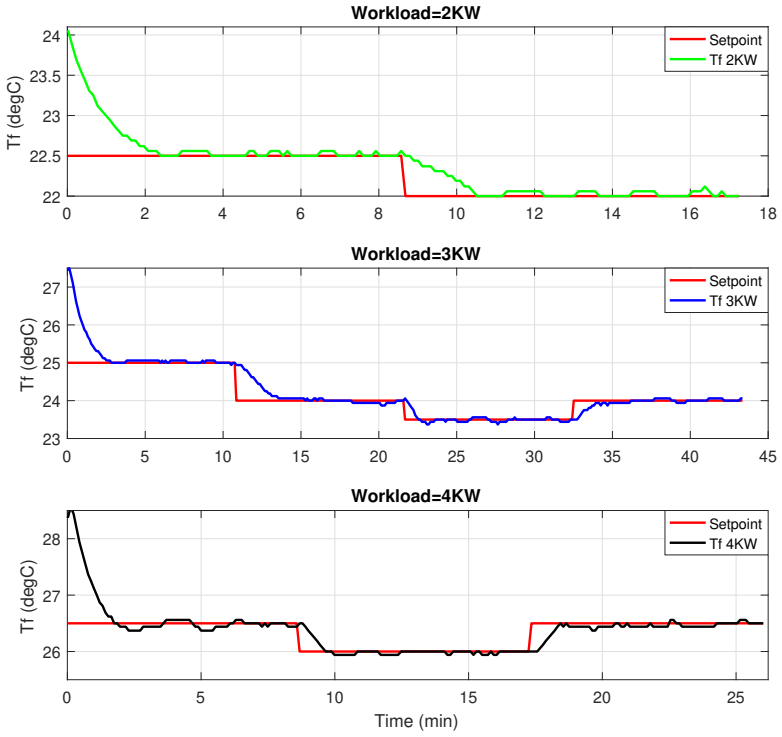

Figure 8: MPC Experiments under Different Workloads

The number of active servers decides on how much heat is generated in the system. Therefore, three groups of MPC temperature control experiments as shown in Fig. 8 have been conducted when the total workload is $2kW$, $3kW$ and $4kW$ so as to verify the performance of MPC controller.

The reason of choosing different set-point values for each workload is that the temperature is higher under a larger workload and vice verse. Thus if we choose the same set-point for $4kW$ as $2kW$, it might not be able to reach that reference. Indexes of evaluating the performance of MPC under these conditions are listed in Table 2.

Based on Fig. 8 and Table 2, the designed MPC controller is feasible and has satisfying performance under different workloads. Mean squared error (MSE) indicates that the temperature control using MPC is very precise. Moreover, the controller can track the reference perfectly and the fluctuation is within ±0.06 °C at steady state for most of the time.

Table 2: MPC Indexes under Different Workloads

	Average Settling Time	Maximum Undershoot	Maximum Overshoot	MSE
2kW	$1min40sec$	0	0	0.0785
3kW	$1min34sec$	0.13°C	0	0.0321
4kW	$1min05sec$	0.13°C	0	0.0130

5.2. MPC Temperature Control at Different Positions

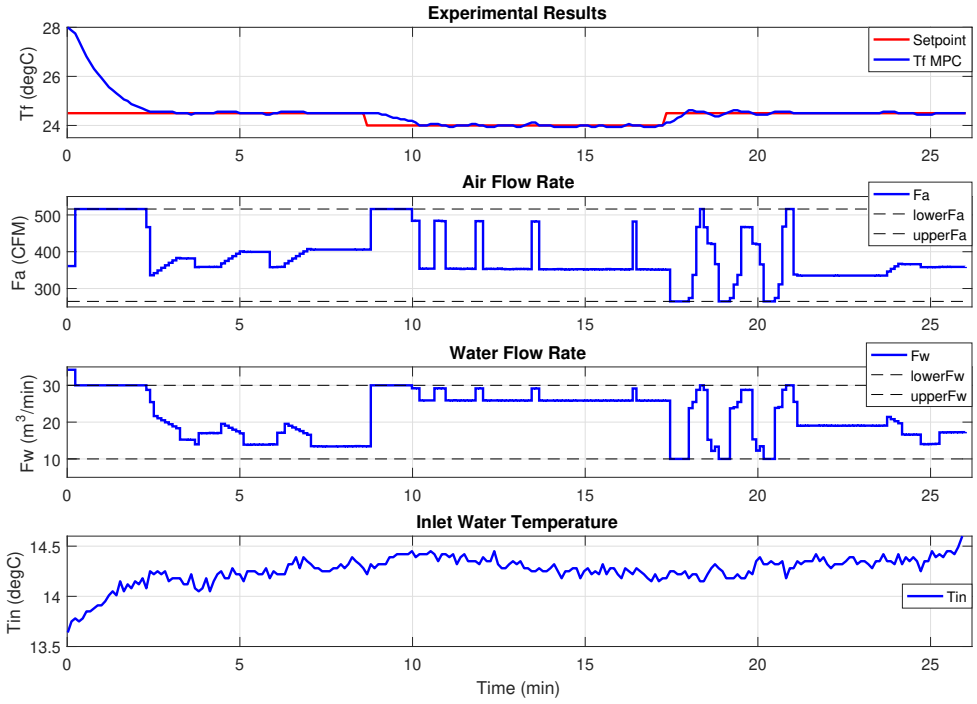

Figure 9: MPC Experiments at Different Positions

The position of controlled temperature shown in Fig. 8 is in the middle of front IT rack. So the control experiment is also performed at the bottom of front rack. Two set-points are chosen and the experimental result is shown in Fig. 9. Some evaluation indexes are shown in Table 3.

As we can see, the designed MPC controller also achieves a satisfying performance at another different position. These indexes are basically the same with the indexes in Table 2 and the error is also within ±0.06°C. Although the controlled temperature fluctuates a little bit due to the inlet water temperature at the third stage, it still reaches the set-point eventually. In conclusion, the

designed MPC controller can be applied to control the temperatures at different positions precisely.

Table 3: MPC Indexes at Different Positions

	Average Settling Time	Maximum Undershoot	Maximum Overshoot	MSE
MPC	$1min25sec$	0°C	0°C	0.0127

5.3. Comparison with PI Control

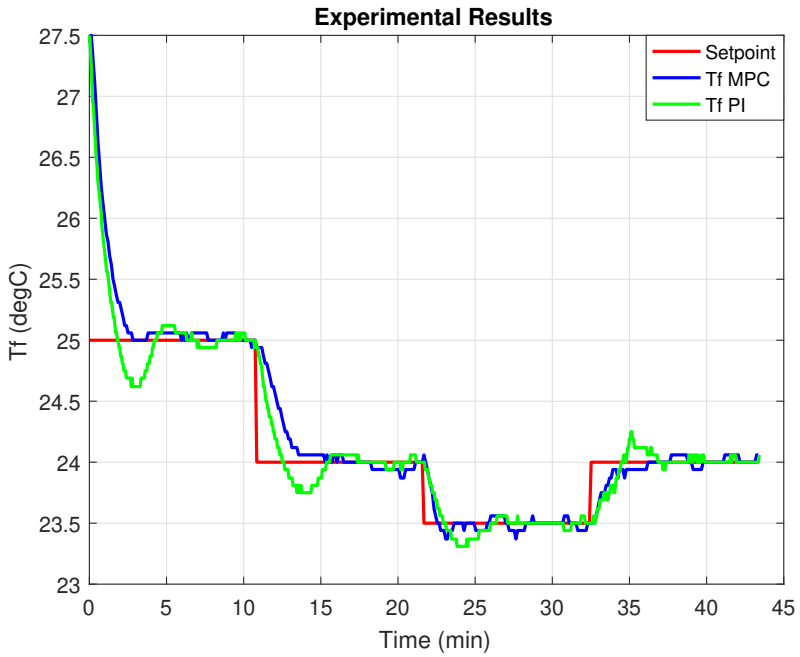

Figure 10: Comparison between MPC and PI

PI controller is also designed in order to have a better understanding of MPC controller's performance and the comparison is depicted in Fig. 10 (workload is 3KW and controlled temperature is in the middle of front rack). The general form of PI controller is:

$$u(k) = u(k-1) + [K_P e(k) + K_I \sum_{i=0}^{k} e(i)] \tag{17}$$

where $e(k) = Setpoint - Measurement$; K_P and K_I are proportional and integral coefficients, respectively; $u(k)$ denotes the input which will be applied to the system. And there are two inputs in our MISO system which are air flow rate (F_a) and water flow rate (F_w). Some important parameters of PI used in experiments are shown in Table 4.

Table 4: PI Parameters

	Initial Input Values u_0	K_P	K_I
F_a	265 CFM	2	0.2
F_w	30 L/min	3	0.2

Some important evaluation indexes of these two controllers are listed in Table 5.

Table 5: Indexes of MPC and PI

	Average Settling Time	Maximum Undershoot	Maximum Overshoot	MSE
PI	$3min40sec$	0.38°C	0.25°C	0.0880
MPC	$1min34sec$	0.13°C	0	0.0321

Two inputs (F_a and F_w) and one measured disturbance (inlet water temperature) are also depicted in Fig. 11.

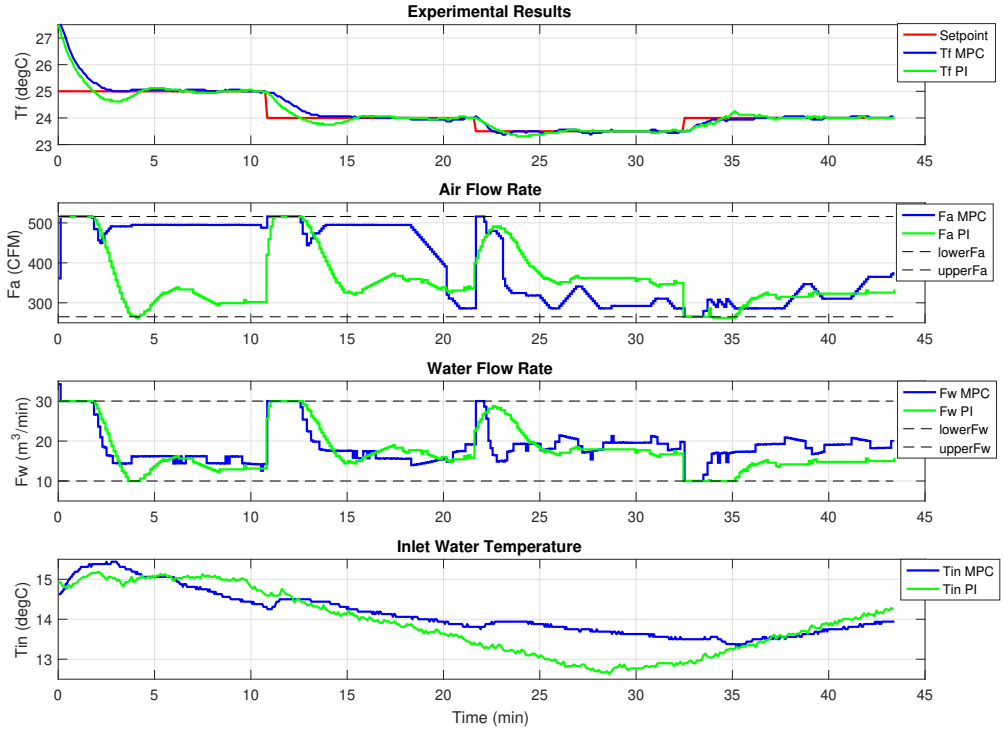

Figure 11: Inputs and Disturbance of MPC and PI

Generally, MPC controller makes obvious improvements on almost every aspect compared with PI controller. But it should also be noticed that the fans' speed of MPC controller is always higher which indicates MPC controller will consume more power. There are several explanations for this phenomenon, the difference of inlet water temperatures between these two groups of experiments, the system has been working for a long time when MPC control experiment is conducted, etc. Anyway, power consumption management will be an important topic for future work by adding power consumption of fans and chiller to the optimization process. Eventually, we hope that not only the set-point can be tracked properly, but the power consumption will also reach an optimal value by using MPC controller.

6. Conclusion

This paper proposed a feasible advanced temperature control method (MPC) for a class of single-rack DCs with RMCU. This control strategy is based on a combined ARX model which is identified using FCM and PLS. This kind of combined ARX model has a better description for the nonlinear DC thermal system than conventional linear ARX model. The experimental results and some evaluation indexes reveal that the designed MPC controller works properly under different workloads and has a good performance. Moreover, the MPC controller is also compared with PI controller and the results indicate that MPC controller improves the control performance in settling time, oscillation, control precision (MSE). However, the power consumption of fans is not so satisfying while using MPC. Therefore, our future work will emphasize on optimizing power consumption using MPC, meanwhile, the controller can still track the reference.

Acknowledgment

Financial support from the Natural Science and Engineering Research Council (NSERC) of Canada and equipment support from CINNOS company are gratefully acknowledged.

References

[1] K. Ebrahimi, G. F. Jones, A. S. Fleischer, A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities, Renewable and Sustainable Energy Reviews 31 (2014) 622–638 (2014).

[2] H. Zhang, S. Shao, H. Xu, H. Zou, C. Tian, Free cooling of data centers: A review, Renewable and Sustainable Energy Reviews 35 (2014) 171–182 (2014).

[3] A. Ryniecki, J. Wawrzyniak, A. A. Pilarska, Basic terms of process control: the on-off control system, Przemysł Spożywczy 69 (11) (2015) 26–29 (2015).

[4] Understanding on-off temperature controllers, https://www.west-cs.com/news/understanding-on-off-temperature-controllers/, accessed February 20, 2019.

[5] K. J. Aström, R. M. Murray, Feedback systems: an introduction for scientists and engineers, Princeton university press, 2010 (2010).

[6] S. W. Sung, I.-B. Lee, Limitations and countermeasures of pid controllers, Industrial & engineering chemistry research 35 (8) (1996) 2596–2610 (1996).

[7] H. Chen, F. Allgöwer, Nonlinear model predictive control schemes with guaranteed stability, in: Nonlinear model based process control, Springer, 1998, pp. 465–494 (1998).

[8] J. M. Maciejowski, Predictive control: with constraints, Pearson education, 2002 (2002).

[9] H. Moazamigoodarzi, S. Pal, S. Ghosh, I. K. Puri, Real-time temperature predictions in it server enclosures, International Journal of Heat and Mass Transfer 127 (2018) 890–900 (2018).

[10] S. Aumi, B. Corbett, P. Mhaskar, T. Clarke-Pringle, Data-based modeling and control of nylon-6, 6 batch polymerization, IEEE Transactions on Control Systems Technology 21 (1) (2013) 94–106 (2013).

[11] S. Aumi, P. Mhaskar, Integrating data-based modeling and nonlinear control tools for batch process control, AIChE Journal 58 (7) (2012) 2105–2119 (2012).

[12] Y. Chetouani, Using arx approach for modelling and prediction of the dynamics of a reactor-exchanger, in: INSTITUTION OF CHEMICAL ENGINEERS SYMPOSIUM SERIES, Vol. 154, Institution of Chemical Engineers; 1999, 2008, p. 297 (2008).

[13] A tutorial on clustering algorithms, https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html, accessed February 20, 2019.

[14] J. C. Dunn, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters (1973).

[15] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013 (2013).

[16] P. V. Gorsevski, P. E. Gessler, P. Jankowski, Integrating a fuzzy k-means classification and a bayesian approach for spatial prediction of landslide hazard, Journal of geographical systems 5 (3) (2003) 223–251 (2003).

[17] M. J. Li, M. K. Ng, Y.-m. Cheung, J. Z. Huang, Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters, IEEE transactions on knowledge and data engineering 20 (11) (2008) 1519–1534 (2008).

[18] J. B. Rawlings, D. Q. Mayne, Model predictive control: Theory and design, Nob Hill Pub. Madison, Wisconsin, 2009 (2009).

[19] E. B. Lee, L. Markus, Foundations of optimal control theory, Tech. rep., Minnesota Univ Minneapolis Center For Control Sciences (1967).

[20] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, Dynamic programming and optimal control, Vol. 1, Athena scientific Belmont, MA, 1995 (1995).

[21] G. Lars, P. Jürgen, Nonlinear model predictive control theory and algorithms (2011).

[22] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. Scokaert, Constrained model predictive control: Stability and optimality, Automatica 36 (6) (2000) 789–814 (2000).

[23] S. Greenberg, E. Mills, B. Tschudi, P. Rumsey, Myatt. best practices for data centers: Results from benchmarking 22 datacenters, in: Proceedings of the 2006 ACEEE Summer Study on Energy Efficiency in Buildings, Vol. 16, 2006 (2006).

[24] Y. Fulpagare, Y. Joshi, A. Bhargav, Rack level forecasting model of data center, in: 2017 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), IEEE, 2017, pp. 824–829 (2017).

[25] P. Geladi, B. R. Kowalski, Partial least-squares regression: a tutorial, Analytica chimica acta 185 (1986) 1–17 (1986).

[26] R. Rosipal, N. Krämer, Overview and recent advances in partial least squares, in: International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection", Springer, 2005, pp. 34–51 (2005).

[27] H. Abdi, Partial least square regression (pls regression), Encyclopedia for research methods for the social sciences 6 (4) (2003) 792–795 (2003).

[28] I. Jolliffe, Principal component analysis, Springer, 2011 (2011).

[29] G. Saporta, N. Niang, Principal component analysis: application to statistical process control, Data analysis (2009) 1–23 (2009).

[30] D. R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural computation 16 (12) (2004) 2639–2664 (2004).

Conflict of interest statement

The authors declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Fengjun Yan

McMaster University