School of Computing & Communications | Lancaster University

Machine learning for smart building energy management

A model predictive control approach

by

Carmen Ka Ki Li

Supervisors:

Denes Csala and Qiang Ni

Dissertation submitted in partial fulfilment for the degree of *Master of Science by Research*

December 2020

Declaration

I declare that this thesis was composed by myself and that the work contained therein is original and my own, unless referenced to the contrary in the text. No part of this thesis has been submitted here or elsewhere for any other degree or qualification. This thesis contains 25,800 words including front matter, appendices, bibliography, footnotes and tables, fewer than the permitted maximum of 35,000.

Carmen Ka Ki Li
December 2020

To my family.

Abstract

We examine the various machine learning methods to be adopted by Q-PLUS, a data-driven smart building energy management system under development by Qbots Energy. Q-PLUS aims to help commercial buildings to save on electricity cost, as well as to generate extra income via the provision of ancillary services, by shifting their electricity demand from the grid away from the peak hours. It is a battery storage, heating, ventilation and air-conditioning (HVAC) control system based on model predictive control (MPC), where machine learning tools are used to predict i), the half-hourly electricity demand of the building and ii), the response of the indoor environment to the HVAC controls. In this thesis, we test and compare different machine learning algorithms to find the most suitable set of tools for the development of Q-PLUS. We also design a battery control algorithm under the MPC framework and fully develop it in Python.

Extended Abstract

We examine the various machine learning methods to be adopted by Q-PLUS, a data-driven smart building energy management system under development by Qbots Energy. Q-PLUS aims to help commercial buildings to save on electricity cost, as well as to generate extra income via the provision of ancillary services, by shifting their electricity demand from the grid away from the peak hours. It is a battery storage, heating, ventilation and air-conditioning (HVAC) control system based on model predictive control (MPC), where machine learning tools are used to predict i), the half-hourly electricity demand of the building and ii), the response of the indoor environment to the HVAC controls. In this thesis, we test and compare different machine learning algorithms to find the most suitable set of tools for the development of Q-PLUS. We also design a battery control algorithm under the MPC framework and fully develop it in Python.

The Q-PLUS battery control system relies on accurate electricity demand forecast for the building, where the half-hourly demand is estimated from the datetime and weather information. We compare two leading machine learning methods for this application, namely random forest regressor and long-short term memory (LSTM), for both short- (hours ahead) and long-term (weeks ahead) predictions. We find that random forest models outperform LSTM models in terms of both accuracy and training time. Thus with a representative set of training data and accurate weather forecast, the random forest regressor can produce a sufficiently accurate, computationally inexpensive and scalable demand prediction for any building.

The Q-PLUS smart HVAC control system can help buildings without battery storage to shift their electricity demand. HVAC loads are flexible loads because buildings have thermal inertia and natural ventilation. In order to exploit their flexibility without violating the safety and comfort requirements though, it is crucial for the system to be able to predict accurately how the indoor environment, including the temperature and CO2 concentration, would change as a function of the power levels of the HVAC systems. However, since conventional building management systems (BMS) operate on simple feedback loops, existing data are bounded within the tight setpoint range allowed by the BMS. Thus it is not possible for any machine learning algorithm to predict accurately what would happen beyond this narrow range, which is where the flexibility comes about. We propose

the use of the Gaussian Process, a Bayesian stochastic method which allows the tracking of the uncertainty of the model. This way the system can probe progressively outside the setpoint range, and with a controllable level of confidence that safety and comfort are not compromised. We show that even with a sparse set of data, the Gaussian Process model is capable of producing reasonably accurate prediction for the zone temperature as a function of the heating coil power in the air-handling unit, and is therefore a viable option for Q-PLUS. Nevertheless, further research is yet to be done to fully integrate the Gaussian Process model into the MPC system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In a report published in 2016 [1], the UK Government's National Infrastructure Commission estimated that a £2 billion worth of network savings can be achieved by 2030 if sufficient levels of storage are installed. Thus instead of building new power plants to meet the ever increasing peak demand for electricity, storage systems can be used to store up energy when demand is low and release it back to the network when demand is high. Storage systems essentially *shift* the demand from generation away from the peak hours, and this has a number of benefits: by smoothing out the demand, power plants can operate continuously closer to their maximum capacity, which is much more efficient than ramping them up and down to follow the demand. This in turn can drive not only the cost of generation down but also the CO2 emission level — gas peaking plants for instance can ramp up very rapidly but at the expense of efficiency, hence their carbon emission rates can be as much as 30% higher than the baseload gas plants. Furthermore, as the penetration of wind and solar energy increases, due to their intermittent and non-dispatchable nature, storage systems will play a crucial role in moderating the supply and demand, storing up surplus generation for later use when demand exceeds generation output.

In addition, the Association for Decentralised Energy reported that further savings can be made if 16% (9.8GW) of the UK's peak electricity demand can be reduced by non-domestic customers, either by shifting demand away from peak periods or using on-site generation [2]. Through the Energy Market Reforms, the UK government identified this as a more compelling option due to the lower investment costs, as opposed to building new power plants which is projected to cost more than £100 billion in the next decade.

Q-PLUS is a smart energy management system that aims to help *commercial buildings* to shift their electricity demand from the grid away from the peak hours.

By doing so, they can not only save on their electricity bills, but also generate extra income by providing ancillary services for the network, as well as contribute to the reduction of CO2 emissions. Most existing building management systems (BMS) for controlling the *heating, ventilation and air conditioning* (HVAC) systems in commercial buildings are based on simple feedback loops: for example, the BMS would activate the heating system when it detects the space temperature has fallen below certain setpoint, and would turn it off when the desired temperature is reached. As such system takes only *current* values into consideration and ignores all important future factors such as weather forecast and electricity prices, it is neither energy- nor cost-optimal. The aim of Q-PLUS is to develop a smart BMS that takes into account all relevant factors that lie in the *future* and optimises the HVAC (and battery) controls by minimising the total cost. The key to building a "smart" BMS that makes cost and energy efficient control decisions is to use *data-driven* methods to develop a control-oriented predictive model of the energy system dynamics within the building. This thesis thus serves as a pilot study for Q-PLUS, with the focus on identifying the most suitable *machine learning* tools to help develop a novel *data-driven model predictive control system* for building energy management.

In order to allow for real-time update of the energy system model, Q-PLUS is developed based on the *model predictive control (MPC)* framework: with reliable and up-to-date information about the weather and operation conditions, *dynamical models* can be used to predict the energy demand of the building over a prediction horizon; the HVAC controls can then be optimised over the prediction horizon based on the prediction. The implementation of MPC in temperature control in commercial buildings was previously studied in [3]; further investigation on finding a suitable mathematical model for the MPC in the same application was carried out in [4], and a cost-benefit analysis of adopting the MPC approach was performed in [5], where the authors identified that although modelling building thermal dynamics is a costly exercise, with the increasing importance of demand side response and rising energy prices, the benefits of an MPC system may outweigh the cost.

The aim of Q-PLUS is to bypass developing such costly *physical* models by adopting machine learning techniques to predict the micro-climate dynamics in the building *from the BMS data*. Machine learning methods have already been applied in the literature to develop dynamical predictive models in MPC-based building energy management systems. For instance in [6], the authors proposed to use an adapted random forest model to partition each *observation* into different equiva-

lence classes (leaves) based on external factors such as outside temperature and datetime information, and then predict the response of the indoor state parameters as a linear function of the *controls* as inferred from the samples in the leaf. Each leaf (or equivalence class) therefore has its own linear regression model. A crucial point to note is that the control variables are *not* passed through the regression trees; by separating out the control variables and not passing them as a *feature* to train the random forest, it ensures that they are still free decision variables in the MPC optimisation. The authors called this approach *data predictive control (DPC)*.

1.2 Structure of the thesis

The structure of this thesis is as follows: in the rest of this chapter, we will be setting up the scene by reviewing MPC and showing how building energy management in general can be treated as an MPC problem. As battery storage will become a crucial and integrable part of the future energy system, the second chapter of the thesis is dedicated to solving the *optimal battery operation*. We will first cast it into an MPC problem mathematically and explore the suitable machine learning tools for predicting the building's electricity demand using weather and datetime information, or *features*. We will compare the performance metrics of the two leading machine learning methods for this kind of time series prediction, namely the tree based, "white-box" random forest model and the neural network, "black-box" Long-Short term memory (LSTM) model. We will implement the battery control algorithm in Python and present some sample result.

We will then switch to the HVAC controls. As reliable BMS data that cover at least one year are not readily available, in chapter three we will look closely at the most complete set of data we could obtain and examine the potential *flexibility* — the ability to shift electricity demand — that can be delivered through the HVAC controls. We will highlight the main challenge in developing a data-driven smart BMS: the lack of comprehensive data that covers a wide enough range of operation conditions and settings for the machine learning tools to "learn" from, as existing BMS data are bounded within the tight setpoint range allowed by the conventional, simple feedback loop system.

In light of this limitation, we then seek a machine learning tool that can ex-

trapolate beyond the tight range of the existing data while allowing us to keep track of the *uncertainty* of the model. Thus in chapter four, we will review the *Gaussian Process*, which is a Bayesian stochastic machine learning method, and discuss how it can be used in the smart HVAC control system. We will show how to formulate mathematically a Gaussian Process model to predict the state variables of the indoor environment as a function of the HVAC controls and the current conditions. We will then demonstrate explicitly how everything fits in with a step-ahead temperature prediction example. Finally, we conclude with a summary and evaluation of our work, and suggestion for future research direction in chapter five.

1.3 Model Predictive Control

The Q-PLUS system is designed following the Model Predictive Control (MPC) framework. In this section we give an overview of MPC, which is also known as Receding Horizon Control (RHC) in the literature. We follow closely the formulation presented in [7]. MPC is a type of feedback control system, it solves a *local* optimisation problem at each time step over a fixed time horizon (i.e. N time steps) in the future, and determine an optimal plan of actions for the fixed time frame, but with only the *first* input being executed in the system. This processed is then repeated at the next time step, solving a new optimisation problem over the same fixed number of time periods in the future but with the time horizon shifted one step forward. In solving the optimisation problem, *estimates* of the quantities in the future based on the current measurements and available data must be taken into account at each time step. These estimates can be obtained in many different ways, such as specialist forecasting, statistical models from historical data, or they can simply be *known* in some special cases. In this thesis, we shall employ *machine learning methods* to obtain the required estimates for the smart building energy management system Q-PLUS.

The first step in setting up an MPC problem is to define the system dynamics. Let time be measured in discrete time steps $t \in \mathbb{Z}$, and let $x_t \in \mathbb{R}^n$ be the vector of measurable state variables of the system; in our case x_t would be the battery state of charge or the indoor temperature and CO2 level; collectively they are referred as the *system state* of the overall system. Let $u_t \in \mathbb{R}^m$ be the vector of control input variables such as the fan speed and schedule information of the building, and let $c_t \in \mathbb{R}^n$ denote the vector of additive Gaussian noise associated with the state vector x_t. Now let us consider a discrete-time linear dynamical

system which can be written as

$$x_{t+1} = A_t x_t + B_t u_t + c_t \ , \tag{1.1}$$

where $A_t \in \mathbb{R}^{n\times n}$ is the dynamic matrix and $B_t \in \mathbb{R}^{n\times m}$ is the input matrix. This is called the *state equation* of the system; the state equation need not take such form, but linear systems are easier to solve. The subscripts t signify the fact that these quantities vary in time in general. Whenever the time step t lies in the future, the quantities need to be *estimated*, but in some cases they could be easily obtained from other sources (e.g. looking up weather forecast) or simply take constant values over time. The state and control variables of our system are subject to a certain set of constraints

$$(x_t, u_t) \in \mathcal{C}_t \ ; \tag{1.2}$$

for example the indoor temperature should be maintained within the comfort range, and ventilation fans cannot operate beyond their full power.

At any time step t, the instantaneous objective or cost function is given by $\ell_t(x_t, u_t)$, which is usually taken to be quadratic

$$\ell_t = x_t^T Q x_t + u_t^T R u_t \ , \tag{1.3}$$

with $Q \succeq 0 \in \mathbb{R}^{n\times n}$ and $R \succeq 0 \in \mathbb{R}^{m\times m}$. The quadratic form guarantees convexity and smoothness so that a unique minimum exists; in fact this kind of linear-quadratic (LQ) control problems can be solved *exactly* by the linear-quadratic regulator (LQR), or the linear-quadratic-Gaussian controller if Gaussian noise is present (see Appendix A for a quick review of LQR). Hence the optimal control plan u_t^* is obtained by finding the *global* minimum of the average cost

$$J = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) \ , \tag{1.4}$$

given that the limit exists. However in practice, we can only minimise the average cost over a *finite* number of time steps N from the current time step τ; that is, to solve for $\{u_t, t \in [\tau, \tau + N]\}$ instead which minimises the total cost over the period $[\tau, \tau + N]$

$$J_\tau = \sum_{t=\tau}^{\tau+N} \ell_t(x_t, u_t) \ . \tag{1.5}$$

In general x_t and u_t, and therefore $\ell_t(x_t, u_t)$, are random variables which follow some distributions; they are then replaced by their expectation values denoted by $\mathbb{E}[X]$.

Since we are optimising over the next N time steps ahead in the *future* from current time τ, the optimal control input sequence $\{u_\tau, ..., u_{\tau+N}\}$ is determined from information available at current time τ, including *estimates* for future quantities which are not known at present. We shall denote estimates with hats, for example $\hat{X}_{t|\tau}$ denotes the estimate of quantity X at time t given all the information available at time τ with $t \geq \tau$, and if the current value of X is known, then $\hat{X}_{\tau|\tau} = X_\tau$.

As mentioned above, the system dynamics are usually described by a linear state equation for the ease of finding a solution; however in most real-life situations, the underlying physical equations governing the system dynamics are highly complex and non-linear, and the same is true also for the constraints and the objective function. Because we need to be able to solve the optimisation problem within a reasonable time frame in order to enable feedback and control, we need to simplify these equations as much as possible into the forms that can be handled easily by standard solvers. Thus the first step in MPC involves building a simplified yet accurate predictive model to forecast the values of the *estimates*

$$\hat{A}_{t|\tau}, \quad \hat{B}_{t|\tau}, \quad \hat{c}_{t|\tau}, \quad \hat{\mathcal{C}}_{t|\tau}, \quad \hat{Q}_{t|\tau}, \quad \hat{R}_{t|\tau} \tag{1.6}$$

from all relevant information available such as building layout, historical data and weather forecast. Conventionally for the purpose of building energy management modelling, these estimates are obtained using white box, physical models made with specialised software packages like EnergyPlus to capture *all* the physical processes within the building, such as heat exchange between the air and the wall and convection current inside the space. However, such models involve labour-intensive surveying and are therefore very expensive and time consuming to build; and since every building is unique, the modelling procedure has to be repeated for every single building, rendering it highly non-scalable and non-economical. Thus we shall follow the black box, data-driven modelling approach in this thesis — we make prediction for the estimates using machine learning techniques based purely on the data available, thereby bypassing all the expensive surveys and physical modelling required, and making the modelling procedure scalable. Throughout this thesis, we may use d_t (standing for *disturbances*) to denote the vector of all

the exogenous variables which the estimates in (1.6) depend on.

The MPC policy then works as follows: at current time τ with the initial state x_τ, consider a fixed time interval extending N steps into the future i.e. $I = [\tau, \tau+1, ..., \tau+N]$. We then perform the following steps:

1. **Build a predictive model**: predict the values of all estimates (1.6) over the time period I using data available at τ. $\hat{x}_t$ and $\hat{u}_t$ are the variables of the problem.

2. **Optimise**: find the solution to $\hat{u}_t \ \forall \ t \in I$ which minimises the objective function, subject to the system dynamics and constraints:

$$\begin{aligned} \underset{\hat{u}_t}{\text{argmin}} \quad & \sum_{t=\tau}^{\tau+N} \hat{\ell}_{t|\tau}(\hat{x}_t, \hat{u}_t) \\ \text{subject to} \quad & \hat{x}_{t+1} = \hat{A}_{t|\tau}\hat{x}_t + \hat{B}_{t|\tau}\hat{u}_t + \hat{c}_{t|\tau} \\ & (\hat{x}_t, \hat{u}_t) \in \hat{\mathcal{C}}_{t|\tau} \\ & \hat{x}_\tau = x_\tau \end{aligned}$$

3. **Execute**: choose the solution sequence $\{\hat{u}_\tau, \hat{u}_{\tau+1}, ..., \hat{u}_{\tau+N}\}$ as the optimal plan of action for the next N time steps, *but execute only the first control input* $\hat{u}_\tau$. At the next time step, the process is repeated with the updated estimates from the feedback of the new current state, pushing the time horizon forwarded by one time step and setting the new current state as the initial state.

It was shown in [8] and [9] that for time invariant (i.e. the quantities in (1.6) are constants) linear-quadratic MPC problems, there exists a closed-from optimisation solution which can be written as a piecewise affine function of the state. Thus the MPC controller can be treated as a collection of conventional linear controllers which switches between the different controls depending on the current state. This means that the set of linear controllers can be computed offline and stored, so that the online algorithm reduces to a quick lookup table search followed by a linear control evaluation.

The MPC controller is reminiscent of the traditional LQR or LQG state feedback controllers in the sense that both have a two-part architecture: an estimator that predicts the future state based on the information available at present, and an optimiser which finds an optimal control plan according to the estimator's predictions. The main difference is, whereas LQR and LQG optimise in a fixed time

horizon and provide a single optimal solution over the entire window, MPC optimises in a smaller, receding time horizon and a new "locally optimal" solution is computed at every time step forward. As a result, the MPC solution is not globally optimal in general, but is nevertheless a very good suboptimal approximation. Another crucial difference is that MPC corrects any deviation between the predicted state and the actual state in the next time step by taking the actual state as the new input and recompute for a new optimal solution, while in LQR and LQG the feedback matrix[1] and the solution control sequence stay fixed over the entire horizon and cannot be updated along the way. In other words, even though an estimator is used to predict the system dynamics in both types of controllers, LQR and LQG are in fact *open-loop* controls without taking any feedback from the actual state, as opposed to MPC which is a *close-loop* control by taking the actual state as feedback. MPC also covers a wider class of control problems as it makes no assumption on the linearity of the system dynamics or quadraticity of the cost function in general, although a closed form solution is not guaranteed then.

[1]The state *feedback* in LQR and LQG is merely a mathematical aspect of the controller, it does not involve any measurement of the actual state being fed back.

Chapter 2

Battery Operation Plan

We shall employ the following notations throughout this chapter:

(continued on the next page)

Nomenclature

Continuous decision variables

x_t^{GD} Energy flow from grid to demand $[kWh]$

x_t^{GR} Energy flow from grid to battery $[kWh]$

x_t^{RD} Energy flow from battery to demand $[kWh]$

x_t^{RG} Energy flow from battery to grid $[kWh]$

Binary decision variables - 0 for off and 1 for on

δ_t^{GR} Whether the battery is charging

δ_t^{RD} Whether the battery is discharging to demand

δ_t^{RG} Whether the battery is discharging to grid

State variable

R_t State of charge of the battery $[kWh]$

Other system variables

D_t Energy demand from building $[kWh]$

N_t Number of partial charge/discharge cycles performed

System parameters

β^c Cycle degradation coefficient [£/cycle]

β^p Power degradation coefficient $[£/kWh]$

η Battery charging efficiency

$\bar{G}$ Contract capacity from the grid $[kWh]$

γ Flow capacity to and from battery $[kWh]$

$\bar{N}$ Maximum number of charge/discharge cycles allowed per day

P_t^b Energy buying price from the grid $[£/kWh]$

P_t^n Non-commodity charges $[£/kWh]$

P_t^s Energy selling price to the grid $[£/kWh]$

$\bar{R}_t$, $\underline{R}_t$ Minimum and maximum state of charge of the battery $[kWh]$

The subscript t labels the half-hourly time step.

2.1 The System

Battery storage is the most straight-forward solution to shifting electricity demand from the grid away from peak hours. As battery storage becomes more affordable and accessible, more and more commercial complexes see it as a compelling and cost effective move not just to save on electricity bills, but also to generate extra income by providing ancillary services. Furthermore, as the UK generation mix edges towards wind energy, storage systems will play a crucial role in moderating the mismatch between the intermittent supply and the highly variable demand. Thus in the first part of the Q-PLUS pilot study, we focus on buildings equipped with battery storage system but without on-site generation. The aim is to utilise the battery storage to shift the demand from the grid at times when electricity prices are high to minimise the electricity cost, as well as to maximise the revenue by selling electricity back to the electricity market and providing ancillary services to the electricity network (both the grid and Distribution System Operators). At time step t, let R_t be the state of charge of the battery, D_t be the total demand from the building, P_t^b and P_t^s be the buying and selling prices of energy from and to the grid. Each time step is half hour long, in line with the electricity market. We use $\bar{G}$ to denote the maximum amount of energy the building can take from the grid at each time step, which is fixed by the user's contract with the provider[1]. R_t is bounded above and below i.e

$$\underline{R}_t \leq R_t \leq \bar{R}_t \,, \tag{2.1}$$

and the bounds change over the course of the day depending on whether buffer charge or capacity is required in the battery for the services contracted for. Even outside the contract hours, the battery should maintain a maximum and minimum level of charge in order to preserve its lifespan; there may also be an upper bound $\bar{N}$ on the number of charge/discharge cycles the battery can perform in a day for lifespan preservation.

[1] One can always go over the contract capacity at a penalty cost, but for simplicity we simply cap it at $\bar{G}$ here.

Among the grid (G), battery (R) and the building (D for demand of the building), the flow of energy in the system at time step t is determined by the non-negative *control variables*

$$x_t^i = \{x_t^{RG}, x_t^{GR}, x_t^{RD}, x_t^{GD}\} \geq 0 \ , \tag{2.2}$$

where x_t^{RG} and x_t^{GR} are the flows of energy from the battery to the grid and vice versa, x_t^{RD} is the amount of energy taken from the battery to feed the demand, and x^{GD} is the amount of energy drawn from the grid to supply the demand. At any time t, energy balance must be achieved and the building's electricity demand D_t must be satisfied exactly with

$$D_t = x_t^{GD} + x_t^{RD} \ , \tag{2.3}$$

and the total amount of electricity drawn from the grid at any given time t cannot exceed the contracted maximum $\bar{G}$

$$x_t^{GD} + x_t^{GR} \leq \bar{G} \ . \tag{2.4}$$

Since the battery cannot charge and discharge at the same time, nor can it discharge to both the grid and the building simultaneously, at any give time step t only one of $\{x_t^{RG}, x_t^{GR}, x_t^{RD}\}$ can be positive, or all of them are identically zero with the battery being idle. This logic rule can be imposed *linearly* on the system using the following inequality constraints with the binary decision variables $\{\delta_t^{GR}, \delta_t^{RG}, \delta_t^{RD}\} \in \{0, 1\}$:

$$\begin{aligned} 0 \leq x_t^{GR} &\leq \gamma \delta_t^{GR} \\ 0 \leq x_t^{RG} &\leq \gamma \delta_t^{RG} \\ 0 \leq x_t^{RD} &\leq \gamma \delta_t^{RD} \\ \delta_t^{GR} + \delta_t^{RG} + \delta_t^{RD} &\leq 1 \ , \end{aligned} \tag{2.5}$$

where γ is the maximum charge/discharge capacity of the battery, which depends not only on the battery itself but also the circuitry.

There is always a loss whenever the battery charges or discharges. This is parametrised by the efficiency η, where $\eta < 1$. η in general is a monotonically decreasing function in time because the resistance increases as the battery degrades, but for our proof-of-concept pilot demonstration it is sufficient to use the constant approximation $\eta = 0.95$, and the discharging efficiency shall be taken to be simply the inverse

$1/\eta$. There are two modes of degradation for the battery, one is calender ageing with time and the other is cycle ageing with every charge/discharge cycle. Since calendar ageing is independent of our control decisions, we will only take cycle ageing into account when we calculate the *objective function*, which is defined by the *cost* C_t at time step t. We shall model the cycle degradation as a monetary cost, which is a linear function of the number of (partial) cycles N_t performed and the energy flow into or out of the battery over the half hour, with constant coefficients β^c and β^p respectively. The number of partial cycles performed over the time step is give by

$$N_t = \frac{1}{2}\sum_i \left|\delta_t^i - \delta_{t-1}^i\right| \ , \qquad i = \{GR, RG, RD\} \ , \tag{2.6}$$

if the user imposes a maximum number of cycles permitted per day in order to prolong the battery's lifespan, then N_t must also satisfy

$$\sum_{t=t_0}^{t_0+48} N_t \leq \bar{N} \tag{2.7}$$

where t_0 denotes the beginning of the day.

When buying electricity from the grid, on top of the price P_t^b for the electricity itself, there are also non-commodity charges P_t^n added onto the bill, in fact they make up on average 40% of the non-domestic energy bill. These compulsory non-commodity charges are in force in order to cover the cost of delivering the electricity and balancing the grid. They also include government taxes and levies for supporting renewable energy development and carbon emission reduction. In the UK non-domestic market, the main non-commodity costs are: transmission network use of system (TNUoS), distributed use of system (DUoS), renewable obligation (RO), climate change levy (CCL), contract for difference (CfD), feed in tariff (FiT) and capacity market (CM). While some of the charges like CCL, RO and FiT are fixed (subject to annual revision) and chargeable only per kWh of electricity consumed regardless of *when* it is consumed, the tariffs of some charges such as CM and DUoS change throughout the day by *orders of magnitude*. Hence these time-dependent tariffs are the costs we are aiming to avoid on daily basis by shifting the electricity demand with the help of the battery storage system. For instance, during the winter period from 1/11/2018 to 28/2/2019, N-power charged £89.87/MWh for CM during the peak hours between 4pm and 7pm but

only £0.03/MWh outside the peak hours[2]. Similarly for the DUoS charges, there are three tariff bands — green, amber and red (see figure 2.1), which cost for example 0.641, 2.197 and 5.535p/MWh respectively in 2016/17 for the low voltage half-hourly metered users with Scottish Hydro Electric Power Distribution[3]. Since the tariffs vary from energy provider to energy provider, when and which tariff applies also depends on the region; the table in figure 2.1 shows the time bands for the half-hourly metered properties in South Wales as defined by Western Power Distributions[4]:

S Wales

Time bands for half hourly metered properties			
Time periods	**Red time band**	**Amber time band**	**Green time band**
Monday to Friday	17:00-19:30	07:30-17:00 19:30-22:00	00:00-07:30 22:00-24:00
Weekends	n/a	12:30-13:00 16:00-21:00	00:00-12:00 13:00-16:00 21:00-24:00
Notes	All of the above times are in UK clock time		

Figure 2.1: DUoS tariff time bands for low voltage half-hourly metered users in South Wales.

[2]Source: https://www.npower.com/business-solutions/your-account/billing/charges/
[3]Source: https://www.ssen.co.uk/WorkArea/DownloadAsset.aspx?id=12511
[4]https://www.westernpower.co.uk/downloads/7028

Another non-commodity charge that can be avoided via demand shift is TNUoS. Although the TUNoS charge is calculated based on the consumption only during the "Triad" periods — the three highest half hour periods of demand during the winter months — at a forecasted rate of £54.59/kW for half-hourly metered users in London in the winter of 2018/19[5], it can make up up to 10% of the annual electricity cost. Thus it is important to be able to predict when the Triads will be and include the TNUoS charges in the cost function accordingly, in order to minimise the electricity demand from the grid through our battery planning algorithm. Nevertheless, building managers can subscribe to Triad alerts from their energy providers, therefore it is not necessary for our battery control system to predict the triad dates.

The relationship between the state and control variables is summarised diagrammatically in figure 2.2.

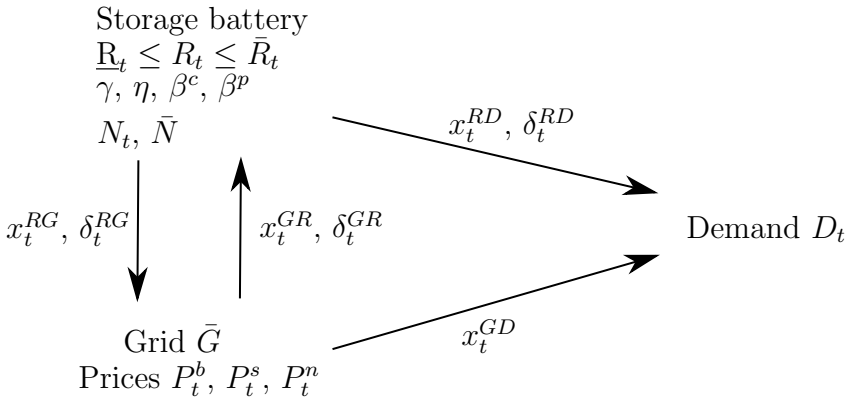

Figure 2.2: Energy flow in our building-battery system.

2.2 Problem Formulation

In order to cast our optimal battery operation plan as a MPC optimisation problem, we need to first define an *objective function*. We shall define our objective in terms of the monetary cost payable to the electricity supplier, and the control objective is therefore to minimise this cost. In fact for our building-battery system, the cost can be calculated straightforwardly by a linear function of the controls. Let x_t denote the vector of the control variables

$$x_t = \left[x_t^{RG}, x_t^{GR}, x_t^{RD}, x_t^{GD}\right] \tag{2.8}$$

[5]https://www.nationalgrideso.com/document/95911/download

and S_t denote the vector of the prices and battery degradation coefficients (measured in terms of monetary cost)

$$S_t = \left[P_t^b, P_t^s, P_t^n, \beta^c, \beta^p\right] \ , \tag{2.9}$$

the net cost at time t is then given by

$$\begin{aligned} C_t\left(S_t, x_t\right) \ &= P_t^b \cdot \left(x_t^{GD} + x_t^{GR}\right) - P_t^s \cdot x_t^{RG} + P_t^n \left(x_t^{GD} + x_t^{GR}\right) \\ &\quad + \beta^c \cdot N_t + \beta^p \cdot \left(x_t^{RD} + x_t^{RG} + x_t^{GR}\right) \ , \end{aligned} \tag{2.10}$$

which is clearly linear in x_t with the parameters given by S_t. The first two terms are the total cost and revenue of buying and selling energy from and to the grid, the third term is the non-commodity cost of buying energy from the grid, and the last two terms take into account of the cost associated to the degradation of the battery. Recall that the number of partial charge/discharge cycles N_t is derived from δ_t and not an independent variable.

Next we need to define the *system dynamics*. In the control theory setting, there is only one state variable in our system — the battery state of charge R_t, which measures the charge level at the *start* of each time step, and the state equation of the battery is simply determined by the charge and discharge operations

$$R_{t+1} = R_t + \eta \, x_t^{GR} - \tfrac{1}{\eta} \, x_t^{RG} - \tfrac{1}{\eta} \, x_t^{RD} \ . \tag{2.11}$$

As we pointed out in the previous section, the state and control variables must satisfy various time-dependent and independent, equality and inequality bounds. These bounds are described by: (2.1), (2.2), (2.3), (2.4), (2.5) and (2.7). Together with the state equation (2.11), they form the set of *constraints* $\mathcal{C}_t$ which the optimiser is subject to. Thus over the time period $1 \leq t \leq T$, the optimal battery plan $x^* = \{x_t^*, \ 1 \leq t \leq T\}$ is obtained by minimising the *total* cost, which can be expressed as

$$x^* = \underset{x_t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(C_t\left(S_t, x_t\right) : \mathcal{C}_t\right) \ . \tag{2.12}$$

In summary, the battery operation optimisation problem over the time period

$1 \leq t \leq T$ can be written as

$$
\begin{aligned}
\underset{x_t}{\text{argmin}} \quad & \textstyle\sum_{t=1}^{T} \left(P_t^b + P_t^n\right)\left(x_t^{GD} + x_t^{GR}\right) - P_t^s x_t^{RG} + \beta^c N_t + \beta^p \left(x_t^{RD} + x_t^{RG} + x_t^{GR}\right) \\
\text{subject to} \quad & R_{t+1} = R_t + \eta\, x_t^{GR} - \tfrac{1}{\eta}\, x_t^{RG} - \tfrac{1}{\eta}\, x_t^{RD} \qquad \forall\, t \\
& \underline{R}_{t+1} \leq R_{t+1} \leq \bar{R}_{t+1} \\
& \{x_t^{RG}, x_t^{GR}, x_t^{RD}, x_t^{GD}\} \geq 0 \\
& x_t^{GD} + x_t^{GR} \leq \bar{G} \\
& D_t = x_t^{GD} + x_t^{RD} \\
& 0 \leq x_t^{GR} \leq \gamma \delta_t^{GR} \\
& 0 \leq x_t^{RG} \leq \gamma \delta_t^{RG} \\
& 0 \leq x_t^{RD} \leq \gamma \delta_t^{RD} \\
& \delta_t^{GR} + \delta_t^{RG} + \delta_t^{RD} \leq 1 \qquad \forall\, t \in [1, T] \ . \quad (2.13)
\end{aligned}
$$

Observe that this is a *linear* optimisation problem: it consists of a linear objective function along with a set of linear constraints. In addition, the battery state of charge constraint (2.1) does not apply at $t = 1$ because we may start from a state of charge outside the buffer charge/capacity bounds after ancillary service delivery. In other words the initial state R_1 is just *given* and so it is not subject to the constraints imposed on the optimiser.

In order to use this method to find an optimal battery operation plan, a number of input parameters need to be defined beforehand. The user or building manager should provide the battery parameters $\{\beta^c, \beta^p, \eta, \gamma\, \bar{R}_t, \underline{R}_t, \bar{N}\}$, the contract capacity from the grid $\bar{G}$, and the pricing information $\{P_t^b, P_t^n, P_t^s\}$[6]. The only quantity that the Q-PLUS system needs to estimate accurately, which is also arguably the most important and non-trivial piece of information, is the electricity demand of the building D_t. As the demand prediction is another central feature of Q-PLUS, we shall discuss the methodology, namely the machine learning techniques we use, in full details in the next section.

Since the delivery of ancillary service can make sudden change to the state of the system R_t, we shall employ *model predictive control (MPC)* for the real time battery control. While the optimiser finds an optimal control plan for the next T time steps, where T may cover the next few hours or the coming 24 hours, only

[6]The spot market prices are highly volatile which requires a separate model to predict; this is outside the scope of this thesis. Besides, users may opt for buying and selling with their energy supplier, in which case all prices are fixed according to the contract.

the *first* control input is executed. This allows the solver not only to update the state of charge of the battery R_t, but also the demand prediction D_t — the further ahead in time, the less accurate the prediction.

Building managers may also use the optimiser to look ahead and help them decide on their monthly ancillary service bids. It can give them an idea of how the battery would operate over the course of the coming month with the required levels of buffer charge and capacity in the battery, depending on the services, and from this they can examine the feasibility and profitability of different ancillary service provision contracts.

2.3 Demand Forecast

The building electricity demand prediction is one of the core services of Q-PLUS. As the building's demand *must* be met, we need an accurate prediction of the demand in order to be able to optimise the battery operations. If the system underestimates the demand during the peak hours and the battery is undercharged prior to the peak hours, the building would then have to buy more than the least possible electricity from the grid. Conversely, if the system overestimates the consumption then the battery may undergo unnecessary charge and discharge cycles and speed up the degradation process of the battery.

One way to forecast the demand is by developing a physical model of the building using specialist software packages such as EnergyPlus to simulate the conduction properties of all the walls and windows, heating and cooling of the whole building due to solar and other thermal radiation, as well as the convection currents circulating around the building. It requires a lot of surveying and calibration work however, which is very costly and labour-intensive. More importantly, since every building is unique, such a model is completely not scalable.

The Q-PLUS demand forecast algorithm on the other hand employs a black box, data-driven approach using machine learning methods. Depending on the interpretability, machine learning models can also be subdivided into black or white box model. Simpler models like linear regression and decision tree are easy to grasp but their predictive power is in general weaker, as they may not be capable of capturing the inherent complexity of the data. For instance, linear regression model assumes that all residuals are independently and identically distributed (iid), thus

is not suitable for modelling datasets which exhibit auto-correlation or collinearity. In contrast, complex black box models like neural networks and gradient boosting models are more accurate in general but lack interpretability. In light of these differences, we examine two of the most widely used forecasting machine learning algorithms, namely the 'white box' regression tree and random forest, and the 'black box' long-short term memory (LSTM), by performing supervised learning on the dataset for Building A in Manchester. The dataset was obtained privately from the University of Manchester and consists of the total half-hourly demand of the building in year 2015. The weather data for model training are obtained from http://rp5.co.uk, using the historical data from the Manchester airport weather station. We sough a machine learning model that can predict the building's total half-hour demand according to the date, time and meteorological information from the weather forecast. The half-hour demand is therefore our target and the features on which the models are trained are: day of the week (d), whether the day is a bank holiday (BH$\in \{0, 1\}$, 1 for bank holiday = true), hour of the day (H), outside air temperature (T), relative humidity (U) wind speed at 10-12m above ground (Ff) and dew point temperature (Td).

2.3.1 Regression Tree and Random Forest Models

Let us begin with the 'white box' regression tree model [10]. It essentially partitions the training samples into different leaves (end nodes) of equivalence classes based on their similarities. Starting from the root node on the top with all the training samples, at each node it determines a binary splitting rule across the feature space which minimises, most commonly the $L2$ error (Euclidean distance), from the mean values of the two descendant nodes. The *impurity*, a measure of dissimilarity of node m containing the set of training data X_m (which is a subset of the set of all training data) with cardinality N_m and target values y_i ($i = 1, ..., N_m$) is thus given by

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y})^2 \tag{2.14}$$

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i \ , \tag{2.15}$$

and the tree finds a cut in one of the feature dimensions which produces the smallest total entropy on both sides of the cut i.e. a splitting rule that minimises $H_{left} + H_{right}$ of the descendent nodes. This process repeats until certain criteria

are met, such as the maximum depth of the tree or the minimum sample size at a node is reached (for instance, only one sample in each node would definitely be an over-fit), then the node becomes a terminal or leaf node. Then in order to make a prediction of the target variable given the values of the features, the model simply passes the features down the tree of splitting rules until it reaches a leaf, where the prediction is made and the predicted value of the target variable is simply the mean of the target variable of the training samples in the leaf. Thus for Q-PLUS, the weather data and datetime information features are passed down the tree and the leaves give the value of the predicted demand.

A *single* tree is highly susceptible to over-fitting — it may learn too well from the training data, including the noise, so that the model is not generalisable. This is where *random forest* [11] comes in: it cures over-fitting by selecting *random samples with replacement* of the training set, training a forest of regression trees and making predictions by taking the average from individual trees. The random sub-sample size is usually taken to be the same as the input sample size; hence if the original training data consists of N observations, it randomly selects N observations but since it does so *with replacement*, the expected number of unique draws is only $N(1-1/e) = N \times 0.63$ and the rest are duplicates. The trees are therefore trained on different, uncorrelated training data, thus any noise from individual trees is averaged out in the forest.

2.3.2 Neural Nets and Long-Short term memory models

An artificial neural network [12] [13] also consists of nodes, which contain an 'activation function', but instead of having a straightforward tree like structure, the nodes are highly interconnected and are organised in layers, resembling network of neurons in the brain, and information is passed back and forth between the nodes. There are three types of layers in each neural network: an input layer where data enter the network, the patterns are then passed onto one or more 'hidden' layers which perform the actual processing via a network of *weighted connections*, and an output layer to show the answer. A neural network 'learns' by updating the weights of the connections according to some learning rule from the patterns it is presented with; this is just like how a child learns to distinguish between a bird and a cat, by seeing examples of birds and cats and updating the weights of their distinctive features such as wings and whiskers.

A neural network model can be seen simply as a function

$$f : X \to Y; \tag{2.16}$$

which can be a distribution over X or both X and Y. The function at a neuron is given by

$$f(x) = K\left(\sum_i \omega_i g_i(x)\right) + b\ , \tag{2.17}$$

where g_i denotes the output functions of its predecessor neurons indexed by i, ω_i is the weight, K is a predefined activation function which depends on the weighted sum of the preceding neurons, and b is the bias which is added to the weighted sum to shift the activation function K when needed. The activation function K is usually taken to be a step-like function such as tanh, the sigmoid function $S(x) = 1/(1 + \exp(-x))$, the rectifier function $K(x) = \max(0, x)$, or the softmax function for vector models, defined by

$$\sigma(\vec{x})_i = \frac{\exp(x_i)}{\sum_(j=1)^m \exp(x_j)} \quad \forall\, i \in 1, ..., m\ , \quad \vec{x} = (x_1, ..., x_m) \in \mathbb{R}^m\ . \tag{2.18}$$

When input is fed into the neural network and output prediction is made, this process is called a forward propagation. In supervised learning, a neural network can learn from the error between its prediction and the actual output value through *back propagation*. This is done by first defining a loss function E such as mean square error, then differentiating the loss function with respect to each weight $\omega_{i,j}$ connecting nodes i and j , and finally updating the weights by descending along the gradient with a step size η which is also called the learning rate; this process is repeated until the error is below certain threshold to avoid over-fitting the training data. Thus the n-th iteration of the gradient descent learning gives

$$\omega_{ij}^n = \omega_{ij}^{n-1} - \eta \cdot \frac{\partial E}{\partial \omega_{ij}^{n-1}}\ . \tag{2.19}$$

The simple, conventional neural networks with only feedforward connections are limited to handle input vectors of *fixed* sizes. In order to train a model to handle lists and sequences, we may add cycles and internal states to an acyclic feedforward network. Such neural networks containing cycles are called recurrent neural networks (RNNs) [14]. In a RNN, the outputs are determined not only by the weights assigned to the inputs but also the *hidden* internal state vector which stores information about the previous inputs and outputs. Therefore, the same input can result in different output depending on the previous input, This is why

RNNs are extremely useful for natural language processing (NLP), as well as time series modelling.

Long short term memory (LSTM) networks [15] are a special type of RNNs which are capable of learning long and short term dependencies. A typical LSTM network consists of four components: a *cell* which stores memory, and three regulators or gates which control the flow of information inside the unit, namely the *input*, *output* and *forget* gates. As the cell state passes down the unit, information can be added or removed from it via interactions with the gates, which are themselves sigmoid neural nets. These gates are powerful tools to select only the *relevant* information to remember, thus enabling the model to learn from both short- and long-term dependencies.

2.3.3 Empirical comparison

We compare the performances of the random forest and LSTM models for 1, two weeks electricity demand prediction based only on datetime and weather (forecast) information and 2, real-time five- and ten-step ahead autoregressive model using datetime, weather and total demand information from the previous five time steps as well as the datetime features and weather forecast for the following five or ten time steps to be predicted, and 3, day ahead prediction analogous to 2, but using information form 12 time steps back. The electricity demand dataset for Building A was merged with the weather data. The period from 2015-9-14 00:00:00 to 2015-9-28 23:00:00 was cut out from the rest of the data for testing purpose.

2.3.3.1 Offline two weeks ahead prediction

Let us first consider the offline, long-term half-hourly demand perdition which is more relevant for the month ahead planning. Here the two-week forecast horizon in our demonstration can easily be extended to arbitrarily long period — as long as the weather forecast is reasonably accurate. First we trained a random forest model to obtain the function

$$D_t^{RF} = D_t\left(d_t, BH_t, H_t, T_t, U_t, Ff_t, Td_t\right) \ . \tag{2.20}$$

We then added time-lag of order two to the model by including the outside and dew point temperatures from the previous hour such that the lagged model gives

the demand as a function of

$$D_t^{RF2} = D_t\left(d_t, BH_t, H_t, T_t, T_{t-1}, T_{t-2}, U_t, Ff_t, Td_t, Td_{t-1}, Td_{t-2}\right) \ . \tag{2.21}$$

Each forest consists of 500 trees with a maximum depth of 30 and the minimum number of samples required to split an internal node set to 20. The forest models were trained with the RandomForestRegressor from scikit-learn. The relative importance of the features are summarised below in table 2.1; it shows that the datetime features are the dominating factors in both models. The relative importance of each feature is given by the *weighted impurity decrease*. At node m with splitting rule s_m that partitions N_m samples into m_L and m_R with cardinalities N_{m_L} and N_{m_R} respectively, the decrease in impurity H is

$$\Delta i(m, s_m) = i(m) - \frac{N_{m_L}}{N_m} i\left(m_L\right) - \frac{N_{m_R}}{N_m} i\left(m_R\right) \ . \tag{2.22}$$

Thus the importance of a feature X_i for predicting Y in a random forest model is computed by summing the *weighted* impurity decrease for all nodes where X_i is used in the splitting s with the weight being the proportion of samples reaching the node out of all N samples, and averaged over all N_{Tr} trees in the forest:

$$\text{Importance}(X_i) = \frac{1}{N_{Tr}} \sum_{Tr} \sum_{m \in Tr, m|s(X_i)} \frac{N_m}{N} \Delta(m, s_m) \ . \tag{2.23}$$

The random forest models are then compared with the LSTM model with time lag of order two over all features

$$D_t^{LSTM} = D_t^{LSTM}\left(X_t, X_{t-1}, X_{t-2}\right) \tag{2.24}$$

where we denote the features collectively as $X_t := \{d_t, BH_t, H_t, T_t, U_t, Ff_t, Td_t\}$. The LSTM model was obtained using the Keras Sequential LSTM model, with 25 epochs and batch size of 72.

The performance of the three models is summarised below in table 2.2 and figure 2.3. The random forest model which also takes the temperatures from the previous hour into account (RF2) produced the most accurate prediction out of the three. Although the training time was almost twice as long, the RF2 model significantly outperformed the straightforward RF model. This is expected, because building has high thermal inertia and it takes time to reach thermal equilibrium. The LSTM model not only took longer to train, it also systematically worsened towards the end of the week giving it a much poorer accuracy than the RF models.

Clearly, the RF2 model is the winner in this test.

Feature	RF	RF2
H_t	0.639231	0.632449
d_t	0.303021	0.294082
BH_t	0.035328	0.034858
Ff_t	0.004493	0.007629
U_t	0.004363	0.007447
T_t	0.009217	0.006443
Td_t	0.004347	0.003463
T_{t-1}	-	0.003135
Td_{t-1}	-	0.002337
T_{t-2}	-	0.004718
Td_{t-2}	-	0.003439

Table 2.1: Relative importance of the features for the random forest models.

Model	CPU time (s)	Wall time (s)	MAE	RMSE
RF	9.97	9.98	2.8332	3.8163
RF2	17.9	18	2.3955	3.1776
LSTM	34.5	20	3.4179	4.6349

Table 2.2: Performance metric of the different models for two weeks ahead offline long-term prediction.

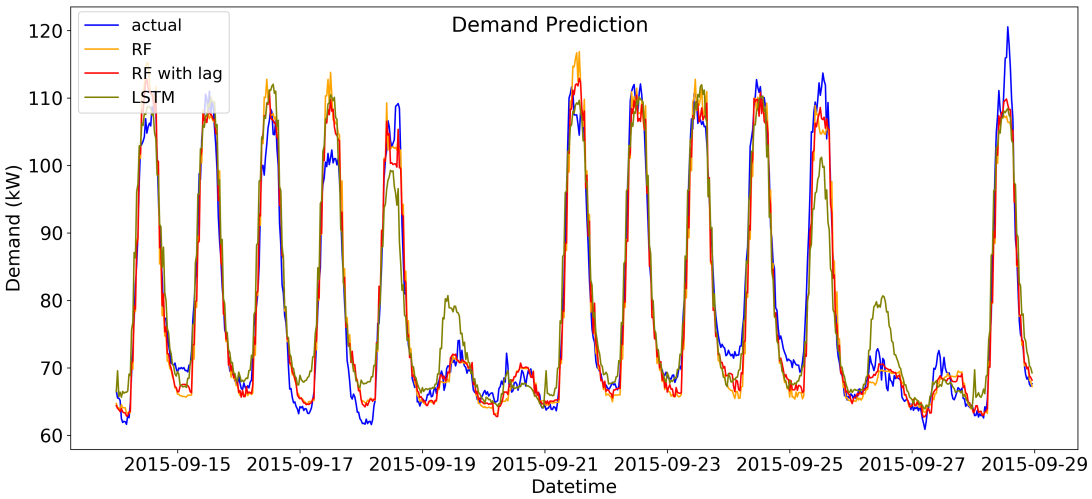

Figure 2.3: Comparison of the outputs from the different models for the two-week ahead prediction.

2.3.3.2 Real-time autoregressive model

Let us now switch to the *online* predictive model for the real-time control of the storage battery. In order to update the model with real-time information as well as to ensure better continuity with the current observation i.e no unexpected big jump in the prediction, autoregressive (AR) models are considered. Thus the previous observed values of the training target, which in our case here is the demand, are also used to train the model as training features. Both the autoregressive order (AR) and the prediction horizon (hr) are set to be five time steps in the first examples, but the Q-PLUS end user is free to choose these parameters according to the characteristics of the building. Autoregression is done by training a model that predicts the hr-dimensional demand vector as a function F of its $AR+1$-dimensional history and $(AR+1+hr)$-dimensional datetime and weather features

$$[D_{t+i}] = F\left([X_{t-j}], [D_{t-j}], [X_{t+i}]\right) \quad i = 1, ..., hr \quad j = 0, ..., AR \tag{2.25}$$

where the square brackets denote time-indexed vectors, e.g.

$$[D_{t+i}] = \begin{bmatrix} D_{t+1} \\ \vdots \\ D_{t+hr} \end{bmatrix} \tag{2.26}$$

is a hr-dimensional vector. Since both input and output are time series, it is reasonable to anticipate that the LSTM model would outperform the random forest model.

For the first example, the test period was randomly chosen to be 11:00 - 13:00 (inclusive) on 2015-09-24, and both models were trained with all data prior to 2015-09-24 11:00:00. As in the previous case, the random forest consisted of 500 trees with a maximum depth of 30 and the minimum number of samples required to split an internal node set to 20. The LSTM model was trained with 25 epochs but this time with a batch size of 32. Table 2.3 summarises the performance of the autoregressive RF and LSTM models. They are also compared against the prediction from the offline non-autoregressive RF2 model from the previous section, which serves as the baseline. In this test, both autoregressive models gave comparable accuracy and both training times were well within the half-hour time frame of the system's requirement, but both were less accurate than the offline RF model which took only weather and datetime information into account and not

the previous values of the demand. Adding recent observations of the target into the model did *not* improve the accuracy in this case. Figure 2.4 shows the output of the different models with the actual target from the data. It shows that the AR RF prediction was an overall overestimation whereas the AR LSTM prediction was an overall underestimation for this particular period.

Model	CPU time	Wall time	MAE	RMSE
AR RF	1m50s	1m50s	2.7048	2.8745
AR LSTM	2m25s	1m55s	2.8564	3.0634
RF2 (baseline)			1.6147	1.9392

Table 2.3: Performance metric of autoregressive RF and LSTM models for the five steps ahead demand prediction for the mid-day period 11:00 - 13:00 on 24-09-2015, as compared to the previous RF2 model. The data used in both AR models are exactly the same. Interestingly, the baseline RF2 model with no autoregression provided more accurate prediction in this period.

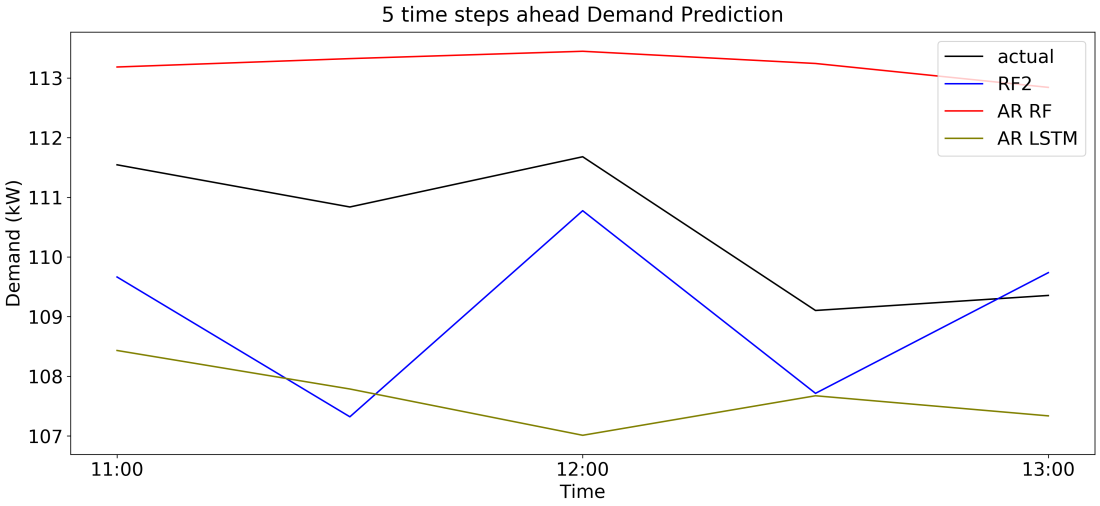

Figure 2.4: Comparison of the five-step ahead predictions from the different models for the mid-day period 11:00 - 13:00 on 24-09-2015.

As a comparison, let us consider a different period on the same day. In this example, we repeat the same procedure as above but for the afternoon peak hours of 16:00-18:00, the time period when the non-commodity charges are the highest and the battery algorithm is expected to discharge the battery to supply the building's demand. The performance of the RF and LSTM models is summarised in table 2.4 and figure 2.5. In this case, autoregression did reduce the error of the model, with the AR RF model outperforming the AR LSTM model in terms of both accuracy and training time. Taking the immediate past observations into account during the hours of falling demand can boost the accuracy of the prediction

precisely by making sure that the projected fall in the coming hours follows the level and trend of the recent observations.

Model	CPU time	Wall time	MAE	RMSE
AR RF	1m59s	1m59s	2.3894	2.7196
AR LSTM	2m30s	1m58s	2.7301	3.0731
RF2 (baseline)			3.2895	4.0102

Table 2.4: Performance metric of the autoregressive RF and LSTM five steps ahead models on the weekday afternoon peak hours from 4 to 6 pm on 2015-09-24. Autoregression did improve accuracy in this case.

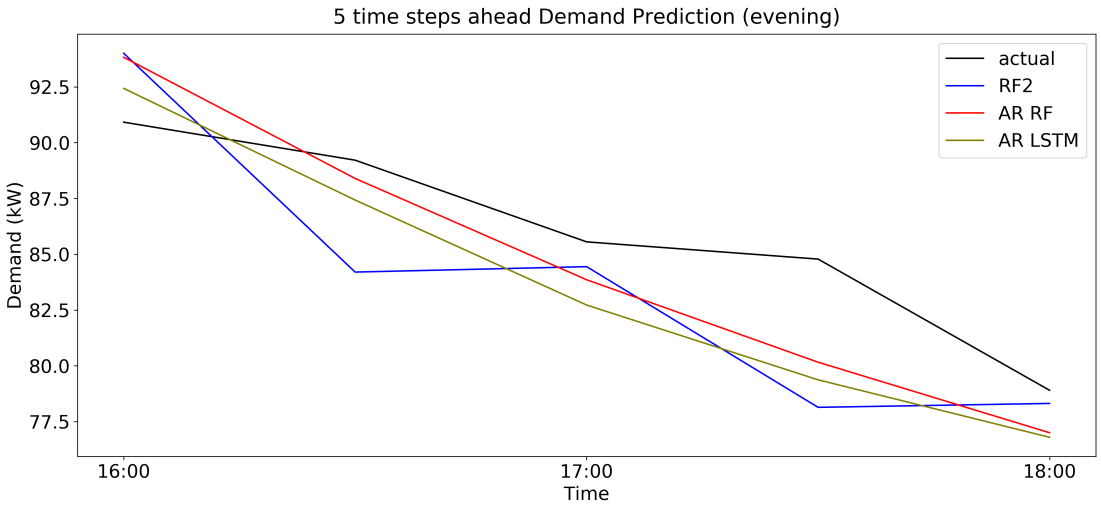

Figure 2.5: Comparison of the five-step ahead predictions from the different models for the afternoon peak period 16:00 - 18:00 on 24-09-2015. The prediction from the baseline offline RF2 model deviates the most from the data.

Now that we have seen how autoregression can make short-term projection more accurate, what happens if we increase the prediction horizon? In the next example, the autoregressive order remains at 5 time steps before but the prediction horizon is now increased to 10 time steps ahead. Since MPC is only a very good approximation to the optimal control, a longer prediction horizon in general results in a better approximation to the global optimum. The test period considered here is 14:30-19:00 on 24-09-2015, the same day as before. The performance of the AR RF and LSTM models is summarised below in table 2.5 and figure 2.6, and they are compared with the baseline RF2 model. In this case, including the autoregressive orders also improved the accuracy of the random forest model. The training times for both the AR RF and AR LSTM models were still just fractions of the 30-minute time frame.

Model	CPU time	Wall time	MAE	RMSE
AR RF	2m46s	2m46s	2.1909	2.5382
AR LSTM	3m28s	1m33s	3.7074	3.9302
RF2 (baseline)			2.3585	3.0254

Table 2.5: Performance metric of the different models for the 10-time step afternoon period on 2015-09-24. The autoregressive random forest model gave noticeably more accurate prediction than the non-autoregressive one.

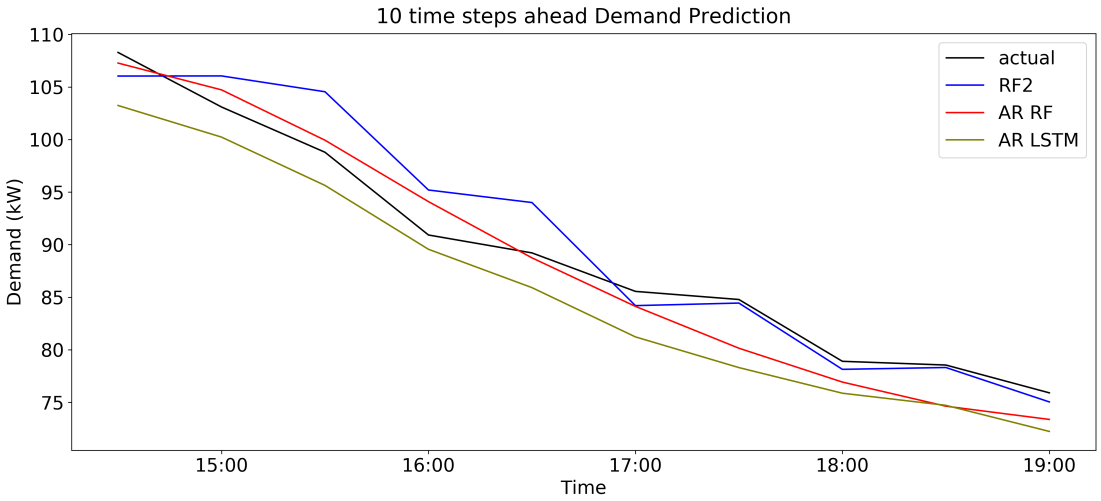

Figure 2.6: Comparison of the different models for the afternoon period of 14:30 - 19:00 on 24-09-2015. Note that the RF2 model prediction was almost spot in the second half.

2.3.3.3 Autoregressive day-ahead demand prediction

The day-ahead demand prediction is particular useful for building managers who wish to participate in the day-ahead wholesale electricity market and avoid price volatility. In this example we test the performance of the autoregressive day-ahead model. Following the same principles as the autoregressive models discussed above, here we expanded the autoregressive order to 12 steps prior and the prediction horizon to 48 steps (one day) ahead. The test period starts from 18:30 on 23-09-2015. The performance of the AR RF and LSTM models, as compared with the baseline model is summarised below in table 2.6 and figure 2.7. Although the two AR models show comparable level of accuracy, the random forest model took almost twice as much CPU time to train as the LSTM model. However both autoregressive models, which took $\mathcal{O}$(minutes) to train, performed significantly *worse* in this test than the baseline random forest model with no autoregression, which took only $\mathcal{O}$(seconds) to train. The inclusion of autoregressive and extra lag features not only increased the complexity of the model, but actually deteriorated the prediction accuracy, which is unexpected, since the learning algorithms should

be able to identify the most relevant factors from redundant inputs to build an accurate predictive model. This is particularly true for the AR LSTM model, as its architecture is designed in particular to handle sequences.

Model	CPU time	Wall time	MAE	RMSE
AR RF	17m13s	17m14s	5.0157	5.4314
AR LSTM	9m41s	4m41s	4.7587	5.2366
RF2 (baseline)			3.5151	4.1140

Table 2.6: The training time for the random forest model was significantly longer than the LSTM model for the day ahead prediction, although both models produced results with similar level of accuracy. The non-autoregressive baseline model turns out to be more accurate than the autoregressive models.

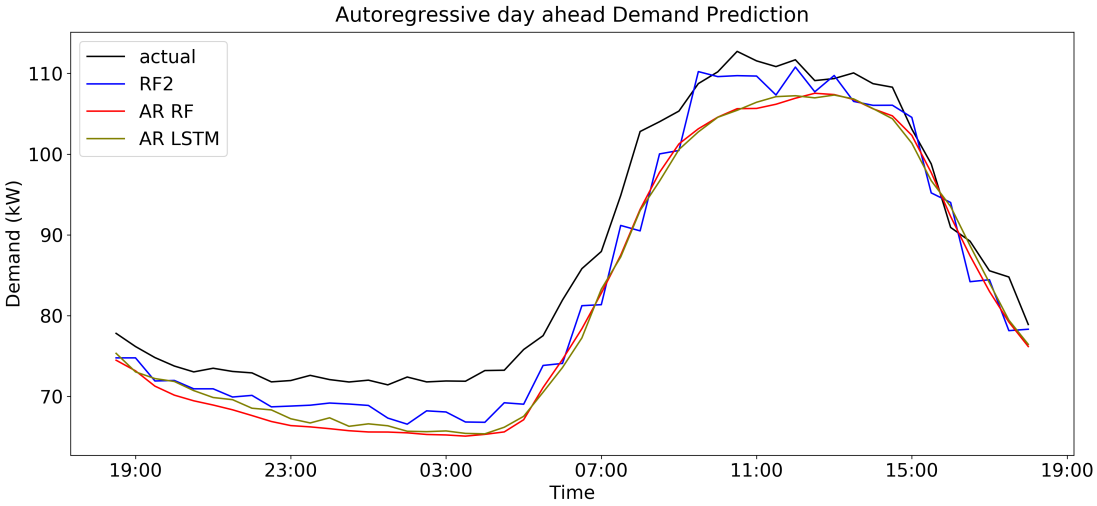

Figure 2.7: Comparison of the day-ahead predictions from the different models for the 24-hour period starting from 23-09-2015 at 18:30. All three models slightly underestimated the demand. The predictions from the AR models also produced smoother curves than the baseline RF2 model.

2.3.4 Analysis and discussion

In this section, we tested the performance of different machine learning models developed from the random forest and LSTM algorithms to predict the half-hourly electricity demand of Building A in various settings. Our first observation is that, despite the common belief that "black box" neural net models perform better then "white box" tree-type models due to their more complex architecture, LSTM models did not outperform random forest ones in any of our tests. In fact the offline LSTM model was very poor with a much higher RMSE than the random forest models in the two-week ahead prediction, with a fall-off towards the

end of each week which was not present in the data. Our second observation is that autoregressive models perform better for short-term online predictions, which makes sense because by taking into account the recent observations, it guarantees better continuity and consistency with the recent history, for example if there is a unexpected surge in the demand observed then the online autoregressive model would produce a forecast that matches with this high level. However, this advantage diminishes as the prediction horizon grows. It is evident from our test that there is no advantage in including autoregression in the model for the day-ahead prediction. Not only did the autoregressive models take much longer to train, the errors were also larger than the baseline offline model. The extra features including the autoregression actually made the prediction worse. Thus for the Q-PLUS demand prediction, the AR RF model with a prediction horizon not longer than 10 to 20 time steps should be used for the real-time online prediction, and the RF2 model is more suitable for the offline prediction for any time periods that span one day or longer.

The main caveat of our experiment is that because it has been conducted entirely with historical data, the weather data used to produce the demand forecasts were real records and not weather forecasts which would themselves be subject to error. Thus the Q-PLUS demand prediction will not have the same level of accuracy as demonstrated here; it will depend also on the accuracy of the weather forecast being used. However, since we want to focus on the errors of the models themselves here, we control for the weather forecast error by using real historical record in our experiment. Another point to note is that like any other data-driven model, the performance of the model does depend on the quality and characteristic of the data. The dataset we have for Building A is an example of "high quality" data in the sense that it covers the full year so there is a complete year for the model to learn from, and also has no missing data. In the next chapter, we will see how the characteristic of the training data could heavily restrict the predictive power of a machine learning model — in particular, if the configuration space is large but the available data only cover a small subset of the total space, then the machine learning model cannot make any reliable prediction outside this subset as there is nothing for it to learn from.

2.4 Optimal Battery Control

Before we conclude the chapter, let us go back to the battery control algorithm and demonstrate an explicit example of optimal control decision. The

optimisation algorithm (2.13) is a mixed integer linear programming (MILP) problem (recall that the δ's in the battery charge and discharge constraints are binary variables). MILP solvers are widely available and we shall use the GNU Linear Programming Kit (GLPK) which is free and open source, to solve for the optimal solution. Our solver is implemented in Python with Pyomo.

In order to demonstrate an example solution to the optimal battery control, let us consider the following: take the actual demand of Building A from 1 to 8pm on 24-09-2015 (15 time steps) and assume that the building does not participate in the spot market for simplicity i.e. different day and night rates only for the electricity prices. Thus $T = 15$ and D_t is copied from the dataset. Since it covers the evening peak, we expect the battery to discharge during the peak hours to feed the building's electricity demand. In the absence of any actual battery characteristics data, let us further assume the following for the battery parameters in the model:

$$T = 15\ , \quad \gamma = 100\ , \quad \eta = 0.9\ , \quad \beta^c = 0.0001\ , \quad \beta^p = 0.0002\ , \quad R_1 = 200\ ,$$
$$\bar{R}_t = 400\ , \quad \underline{R}_t = 40 \quad \forall\, t \in \{1, ..., T\}\ .$$

Thus we assume the building is fitted with a 200kW/400kWh battery which is initialised to be half full, and the state of charge cannot fall below the 10% threshold which is 40kWh. As for the pricing parameters, again in the absence of actual data, we assume a constant selling price for electricity at GBP 0.001 per kWh, and buying prices of GBP 0.2 and 0.1 per kWh for day and night rates. The time series input parameters for the battery control demonstration are shown in table 2.7 below. Note that since we are working at half-hour temporal granularity $\gamma = 200\text{kW} \times 0.5$ hours $= 100$kWh. We have set the selling price to be very low so that it is not favourable to sell electricity to the grid, as we are considering here the case where the building is not participating in the spot market. The optimal solution as given by the solver is summarised in figure 2.8 below; it took the solver 0.0178 seconds to solve. It shows battery does discharge at 16:00 and 17:00 to supply the building's own demand as expected, but it turns out to be not favourable to charge up the battery prior to the peak hours given these parameters.

2.5 Conclusion

In this chapter, we presented a battery control algorithm for Q-PLUS based on the model predictive control framework and implemented it in Python. A crucial element of the data-driven battery control system is the electricity demand

t	D_t	P_t^b	P_t^s	P_t^n
13:00	109.103	0.2	0.001	0.1
13:30	109.355	0.2	0.001	0.1
14:00	110.054	0.2	0.001	0.1
14:30	108.723	0.2	0.001	0.1
15:00	108.284	0.2	0.001	0.1
15:30	103.095	0.2	0.001	0.1
16:00	98.789	0.2	0.001	1
16:30	90.918	0.2	0.001	1
17:00	89.215	0.2	0.001	1
17:30	85.554	0.2	0.001	1
18:00	84.782	0.2	0.001	1
18:30	78.902	0.1	0.001	1
19:00	78.548	0.1	0.001	0.1
19:30	75.903	0.1	0.001	0.1
20:00	74.258	0.1	0.001	0.1

Table 2.7: Input demand and pricing time series for the battery control optimisation example.

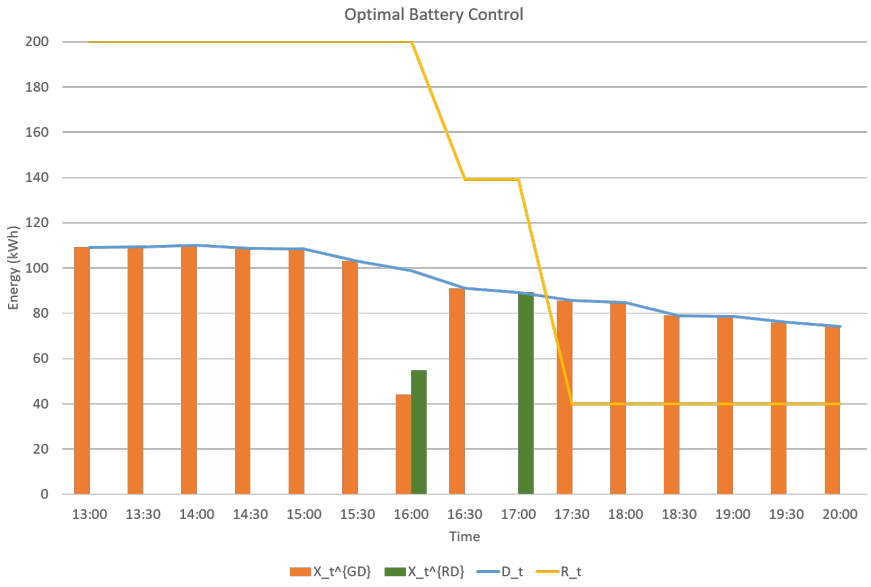

Figure 2.8: The optimal battery operation, showing the battery discharging during the evening peak hours. But given the input parameter we have here, it is not favourable for the battery to charge up prior to that.

forecast. We compared two leading machine learning tools fit for this purpose, namely the "white box" random forest and the "black box" LSTM methods. We found that the random forest model that accounted for the current datetime and weather information as well as the weather conditions in the previous hour was the most suitable for long-term offline forecast, whereas the autoregressive (i.e. demand values from previous time steps are also considered) random forest model

worked better for the real-time prediction provided that the prediction horizon was sufficiently short — shorter than one day (48 time steps) in our case. We also found that despite their sophisticated "black box" architecture, the LSTM models did not outperform the random forest models in any of our settings, with or without explicit autoregression. The offline LSTM model was particularly poor for the 2 weeks ahead forecast. Q-PLUS will therefore employ random forest models for the demand prediction.

Furthermore, our results also highlight the advantages of data-driven machine learning models for electricity demand prediction over the conventional, costly and highly complex physical models: given adequate demand data, the same approach can be applied to any building; the random forest models take only seconds to minutes to build and can achieve high levels of accuracy, without the need to conduct a full survey of the building or compute the numerous parameters that are required by the physical modelling packages like EnergyPlus. Nevertheless, machine learning models are not without any limitations. The validity of a machine learning model depends heavily on the training data. Here for the demand prediction, we have had a sufficient set of half-hourly demand data to work with that covers the full year and without any missing data, so it covers essentially the full temperature range and operation conditions that one can expect to encounter. However, this is not necessary the case for the HVAC control data. As we shall see in the next chapter, BMS data usually cover only a small subset of the very large configuration space consisting of all the possible temperature and control combinations, due to the rigidity of the simple feed-back loop control system that only allows the indoor conditions to vary within some tight setpoint ranges. This clearly poses a huge hurdle for creating a *data-driven* smart energy management system that exploits the HVAC demand *flexibility* by taking the indoor temperature and CO2 level *beyond* the existing setpoint range at certain times of the day.

Chapter 3

Flexibility

3.1 Demand shift without battery

The aim of Q-PLUS is to help commercial buildings shift their electricity demand from the grid away from the peak hours, so they can pay less for electricity as well as generate extra income by providing ancillary services. In the previous chapter, we demonstrated how this could be done with a battery. However, installing a battery storage system in a building a is a huge investment and the expected payback period can easily exceed a decade, this is not a risk every building manager is willing to take. Therefore in this chapter we investigate how, if feasible, buildings without batteries may also shift their demand as if there were a battery.

The idea behind this is straightforward: HVAC systems are *flexible* loads because all buildings have thermal inertia and natural ventilation, just to different extents. For instance, in the winter the building can be slightly overheated before the peak hours so that the heating system can be turned down or even completely switched off during the peak times, because it takes time for the heat to dissipate out of the building. Similarly, the building can be ventilated and get the air exchanged in advance, so that the ventilation system can be turned off during the peak hours without too much CO2 building up inside as the air will still circulate naturally. For those better built buildings which are very well insulated and with good natural ventilation, The HVAC systems may provide sufficient flexibility for the building to provide demand side responses and to generate extra income. Nevertheless, we must ensure that the safety and comfort of the occupants inside are not compromised by doing so. In particular, CO2 can build up to an unsafe level if the air handling unit is switched off for too long and the building relies only on natural convection. Rapid variations in the indoor temperature can also cause discomfort even if the temperature is maintained within the acceptable comfort range; as a rule of thumb, a change of $\pm 0.5°C$ per hour inside the comfort range can generally go unnoticed. Hence in order to make the most of the flexibility, we need to

be able to predict accurately how the *state* variables of the indoor environment, which include the temperature and carbon dioxide level, change with the *controls* of the HVAC system, under different weather conditions and at different times. Our task is then to find the most suitable machine learning tool to do so. However, since a data-driven model cannot be built without adequate data, we need to first assess the characteristics and suitability of the available BMS data before we proceed to model building. Moreover, by examining the BMS data closely, we also want to obtain an estimate of how much potential flexibility we can really get from the HVAC system and justify the idea of "demand shift without battery" for commercial building blocks. This chapter is thus dedicated to the analysis of BMS data.

3.2 A Newcastle case study

Comprehensive BMS data that cover a long enough period so that all the important dynamics and seasonality can be captured are very difficult to find, as most buildings do not keep the record. Fortunately, we have been able to obtain the BMS data for Building B, a building located in the city of Newcastle upon Tyne for our flexibility case study, courtesy of Newcastle University. Completed recently, the building itself was also designed to serve as a big laboratory for smart building experiments, and therefore has very detailed records of its BMS and indoor environment data. The dataset covers one of the plant zones on the third floor and the data runs from 01-01-2018 to 10-10-2018. A closer look at the original technical drawings of Building B reveals that even this single plant zone has very complex dynamics, with 19 data collection points across the whole space plus 16 sets of meter readings for the HVAC system serving the zone. Due to the complexity of the HVAC system and the lack of transparency of the commercial BMS currently being used in Building B, it is not an easy task to fully understand how the individual subsystems interact with each other. Nevertheless, such challenge also motivates exactly the use of data-driven models over physical models. With sufficient data, we hope to be able to infer the interplay of the different subsystems and the indoor environment by extracting meaningful patterns from the data.

Because we are interested in the flexibility and savings achievable from the chill beam water pumps, heat pump and the air handling unit in the plant room that serve the plant zone as a whole, rather than the potentials from individual offices and areas, we shall first aggregate and average the temperature and carbon dioxide readings from the raw data across the whole plant zone, and then compare them

with the operation powers of the various HVAC subsystems.

3.3 Thermal flexibility

Throughout the entire period from 01-01-2018 to 10-10-2018, the average zone temperature had been kept between 20.5 and 25.5 degrees; an example period is shown in figure 3.1 for the month of February. This variation is rather small; the setpoint range can be widened to facilitate higher flexibility, especially in a well-insulated building like Building B. For instance, it is not necessary to keep the building above 20.5 degrees in the evening. Then the office space can be overheated to, for example, 28 degrees early in the morning before the peak hours without causing discomfort to the occupants, as the occupancy is likely to be very low. Depending on the outside temperature and how well the insulation is, the heating systems can then be turned down or completely off for a certain duration of time during the day when the prices are higher. The question is then for *how long* and by *how much* we can turn the heating down.

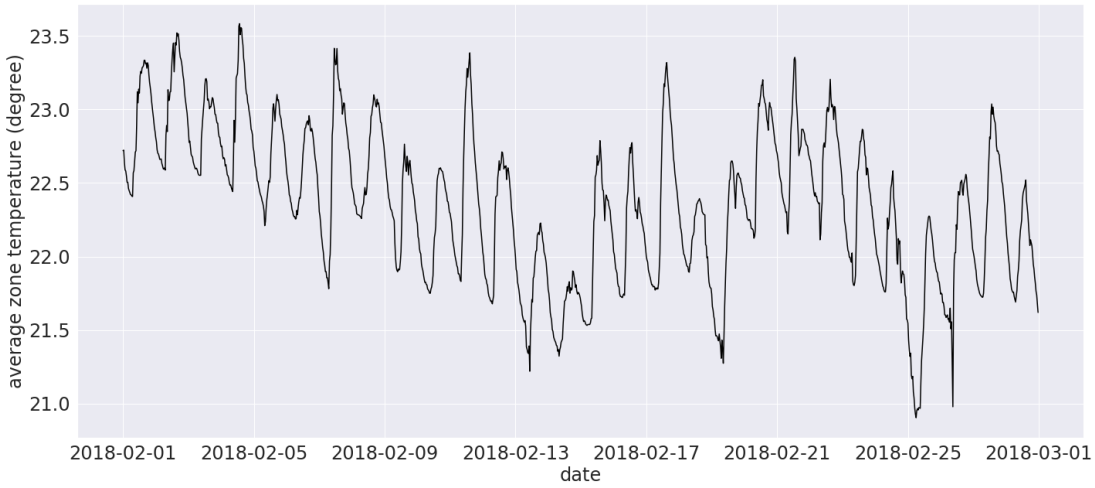

Figure 3.1: Average zone temperature in February 2018. The indoor temperature was kept between 21 and 23.5 all the time. This range can be widened to create more *flexibility* for demand shifting.

3.3.1 Hot and chill water pumps

In order to identify flexible load and model it, we need to first identify which part of the power consumption correlates to which element in the system. As shown in the schematic in figure 3.2, heating and cooling of the indoor space is done by water passing through the chill beam. The temperature of the two circuits of hot and cold water is controlled by the heat pump. The circuits are not

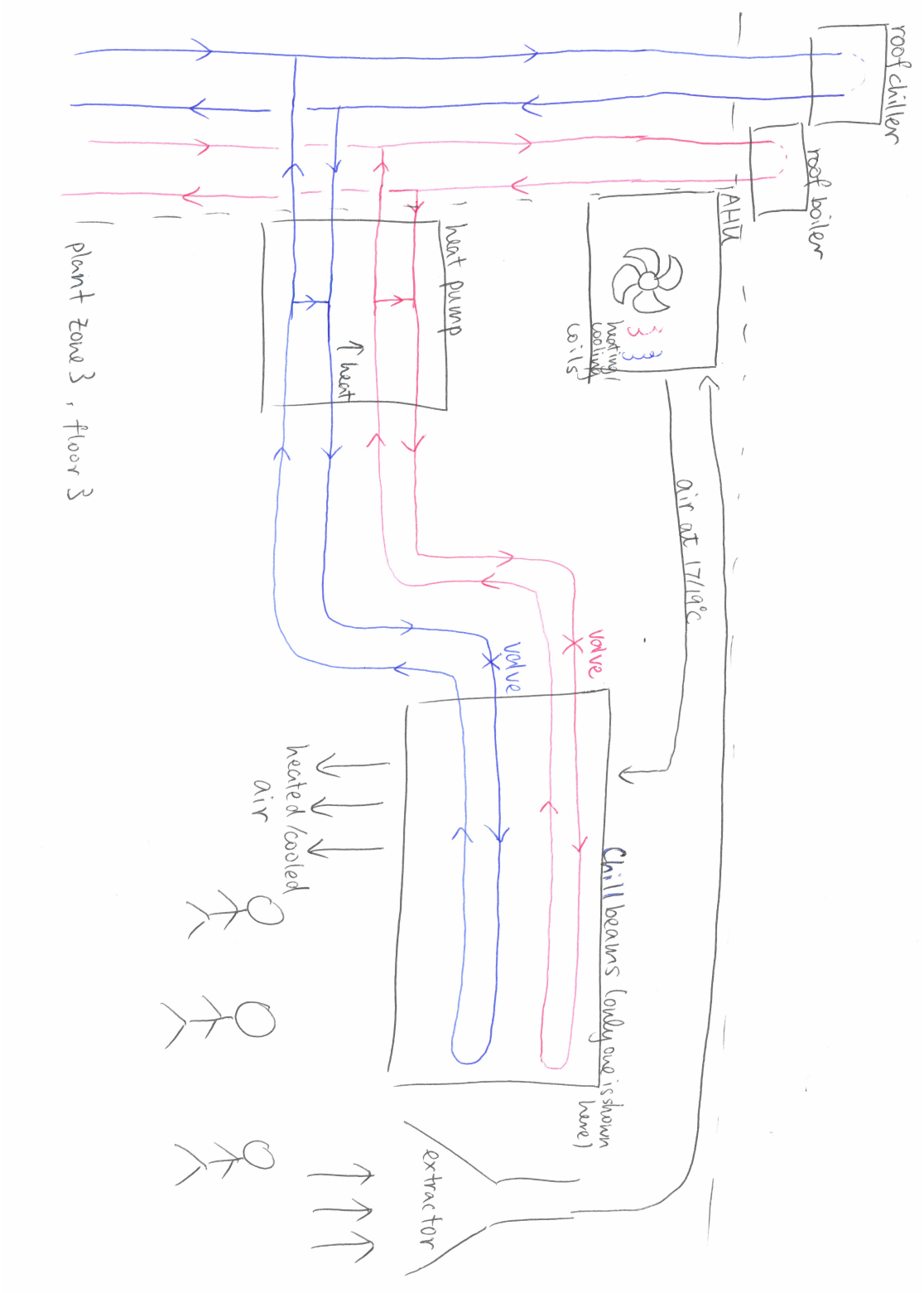

Figure 3.2: The aggregated schematic of the plant zone in Building B.

closed within the zone; there is also hot and cold water coming from the core plant room on the roof of Building B, for which we have no data. The data labels also suggest that there are pumps in the plant room to circulate the water within the zone in the chill beams. Let us first examine how the flow rates varies with the power of the pumps. In order to get a better contrast and a better understanding, we extract a winter period, a summer period and a fall period for the analysis. As the flow rates and the water pump powers are measured in different units, we need to first normalise the different time series with z-normalisation before we make any comparison. We shall use the Keogh distance [16] to measure how similar and *closely related* two time series are: the Keogh distance is the lower bound distance for the dynamic time warping (DTW) of the two time series, thus it takes into account both the shape and relative shift of the waveforms of the time series. Here we employ a *shape based* approach that also allows varying shift in the time domain because the response of the state of the system following a change in the control is highly complicated, thus the response is not necessarily linear with respect to the control and may take different amount of time to manifest under different conditions.

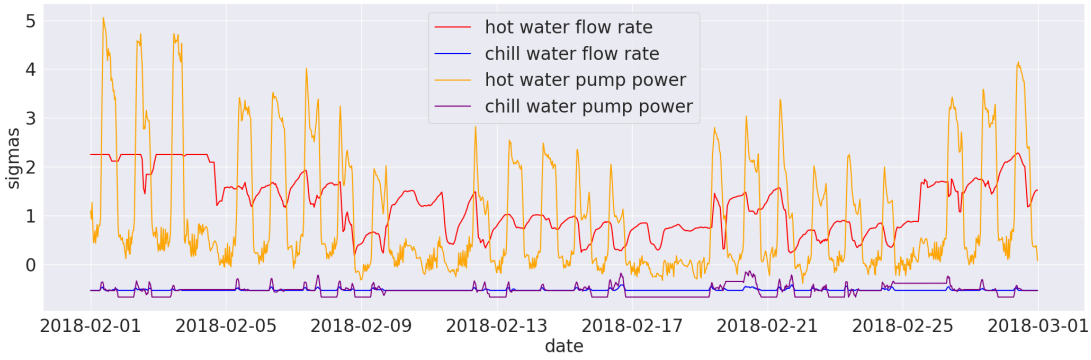

Figure 3.3: The normalised flow rate of hot and cold water in the chill beam verses the power of the pumps in February. The red and blue lines are the hot and chill water flow rates whereas the orange and purple lines are the hot and chill water pump powers. It shows a strong correlation for the cold water circuit but somewhat unexpected behaviour for the hot water circuit.

Let us first look at the winter month of February. As shown in Figure 3.3, there is a strong correlation between the cold water flow (blue) and the power of the cold water pump (purple). Indeed, the z-score Keogh distance between the two curves is 2.85. However, the hot water curves exhibit an rather unexpected anti-correlation: while the hot water pump power (orange) went up during core hours (8am-6pm), the hot water flow rate (red) actually *decreased* slightly over these hours on most days. The z-score Keogh distance between the two curves is 31.61, showing a significant deviation between them. Although the hot water pump

power time series exhibits the expected "5 high plus 2 low spikes" weekly pattern (except for some weekends where there could be some events in the building), the flow rate shows almost the opposite pattern. Figure 3.4 shows the normalised hot water pump power and flow rate on the working days only. The power of the pump does not seem to have any effect on the flow rate at all, which is unexpected. In order to fully capture the heating dynamics of the building, we also need the data from the central core plant as part of the heating energy must have come from there.

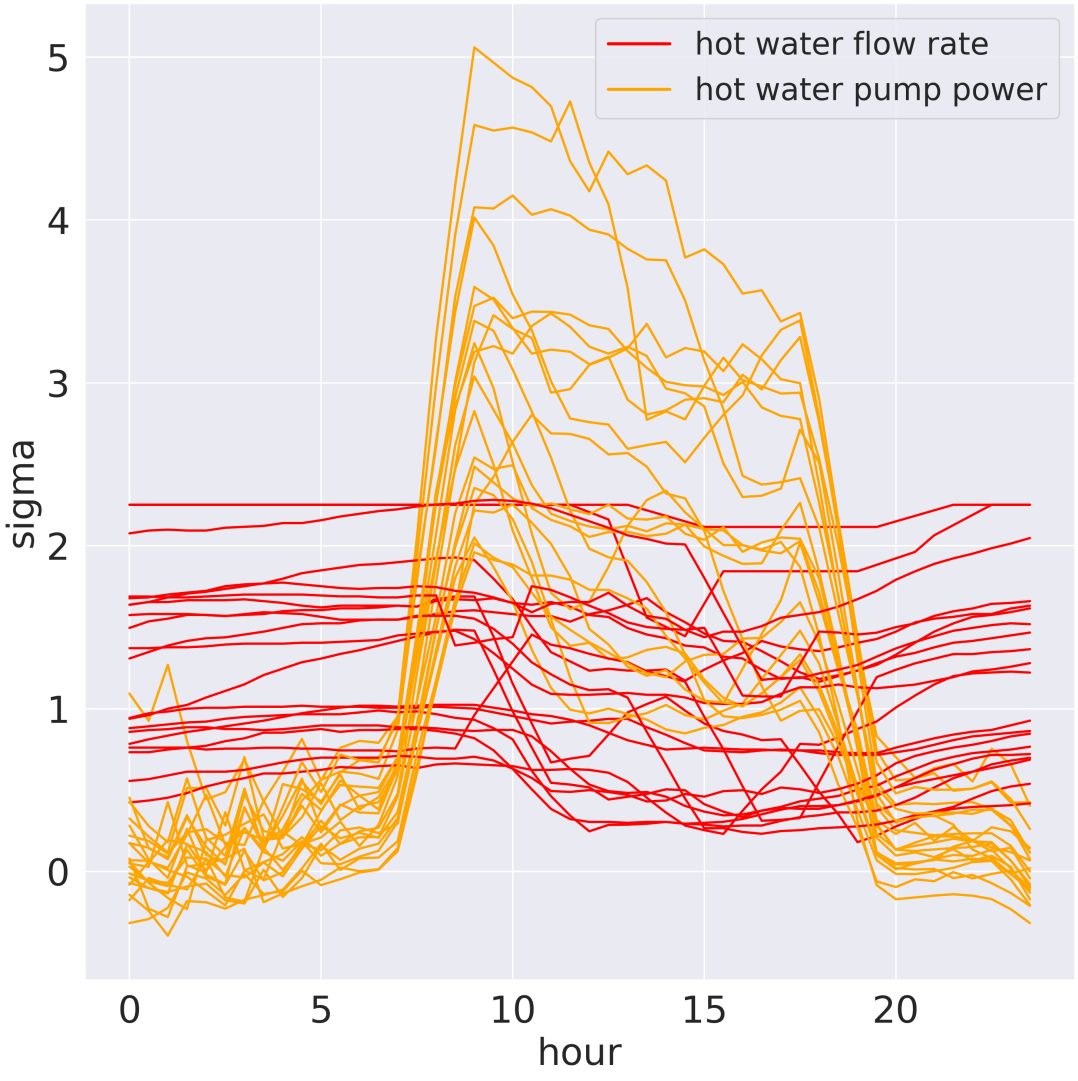

Figure 3.4: Hot water chill beam pump power (orange) vs flow rate (red) on February work days. The pump consumes more power during the core hours of 8am-6pm, but the flow rate remains more or less constant throughout the day, with a slight dip over the core hours. Both sets of curves are z-normalised.

Let us now turn to the pumps and flow rates during the summer period

of July and August (Figure 3.5). This time it is the hot water profiles that show

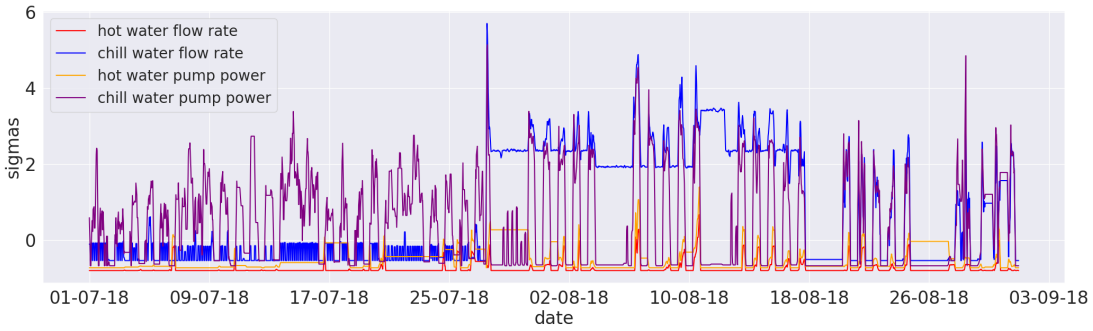

Figure 3.5: Flow rate vs pump power for the summer months; the hot water profiles clearly follow matching patterns, except during the several periods when the flow rate was zero but the hot water pump was still operating at considerable power.

matching pattern but the chill water profiles only match in the second half, with Keogh scores of 18.0 and 74.6 respectively. The flow rate of chill water was somehow very low in July, even though the chill water pump was working hard. This puzzling phenomenon is analogous to what we have just observed above for the hot water pump and hot water flow rate in the winter month of February. Some setpoint was probably changed on July 27 for three weeks where the peaks coincided but a high flow rate was maintained even at the after hours when the pump was off. The main pump was probably pumping hard 24/7 over that period, with some boost of the flow from our measured zone pump which gave the matching spikes. Towards the end the two curves match very well with a z-score Keogh distance of 4.57, suggesting that the main pump was turned down, or even off, and the circulation was done mostly or solely by the zonal pump.

Building B began taking data at the start of 2018; as more and more data were taken, the data quality also improved. Let us now turn to the comparison in fall, towards the end of the data run. Both the hot and chill water flows and pump powers show very strong correlation during this period, with Keogh distances of 6.95 and 17.5 respectively (Figure 3.6). The reason could be that during the transitional seasons of spring and autumn, the building's main heating and cooling systems were turned off or ran at low power, so that most of the temperature regulation in each zone was done locally by the secondary systems in the individual zonal plants.

It is clear that the plant zone itself is not a closed system. There are certainly hidden dynamics coming from the building's main heating system which are

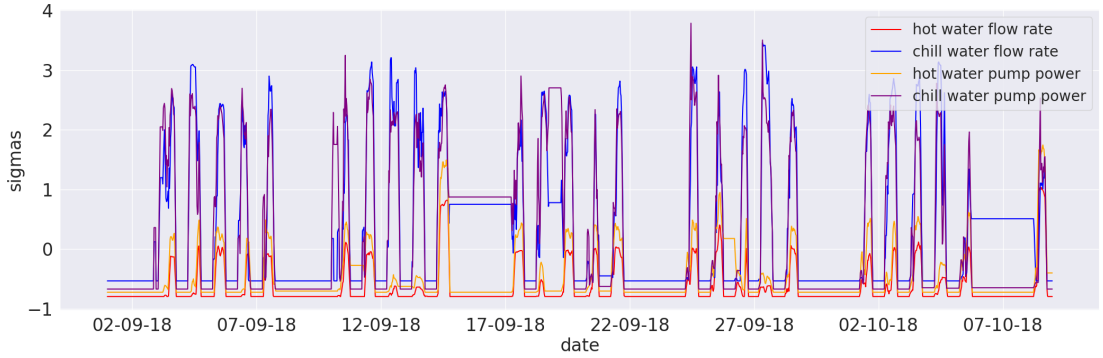

Figure 3.6: Both sets of curves for the hot and cold water circuits exhibit close correlation in fall. It could be that during the transitional season, the core plant was running at low power or maybe be even off, so that heating in cooling in the zone was mostly provided by the zonal plant.

not covered by our data, and this is why the correlation between the hot and cold water pump power and the flow rate varies through out the year. In the winter, most of the heating was done by the main heating system and therefore the hot water flow rate did not quite follow the hot water pump power, whereas cooling demand was low so whatever cooling was needed in the plant zone, it was handled by the zonal cold water pump, and vice versa in the summer. In order to make an accurate model for the Q-PLUS control system though, we *must* take into account these external factors coming from the main plant by including the relevant data in our model, such as the incoming flow rate from the main hot water pipe, even if we do not seek to fully understand the heat transfer and dynamics *physically*. Under the MPC framework, these will be exogenous factors (d_t) as we have no control over the core pumps at the plant zone level. Without any data from the core plant, it is not possible to infer anything meaningful for the hot and chill water pumps in the plant zone. Thus at this stage we are not able to estimate how much flexibility can be harnessed from the hot and chill water pumps that we have been looking at.

3.3.2 Heat pump

Apart from the hot and cold water pumps, there is also a heat pump in the plant room. It is used to regulate the temperature of the hot and cold water circulating in the zone by extracting heat from the cold water and transferring it to the hot water via work. Let us define

$$heating = (supply\ temperature - return\ temperature) \times hot\ water\ flow\ rate$$

and similarly,

$$cooling = (return\ temperature - supply\ temperature) \times chill\ water\ flow\ rate$$

where the supply and return temperatures of the heat pump are recorded in the dataset, and the flow rates are taken to be the same as in the chill beams. Cooling and heating are therefore measured in units of power which also incorporate the specific heat capacity of water. We want to see how the amount of heating and cooling done is related to the power of the heat pump.

As we can see from figure 3.7, in the winter period of February, there was essentially no cooling (blue line). The peaks and troughs of the heating (red) and power (black) curves do coincide although they do not have quite the same shape. Also from the same level of power during the working hours, different levels of heating could be achieved — significantly more heat was clearly being transferred at the beginning and end of February than in the middle, although the heat pump operated at very similar daily profiles throughout this period. Of course, the efficiency of the heat pump also depends on the temperature difference between the hot and cold water; the coefficient of performance decreases with increasing temperature difference. However, as illustrated by the grey curve in figure 3.7, the temperature difference of the water also exhibited regular diurnal pattern with pretty much the same amplitude on the weekdays, thus it wouldn't explain why different amount of heating was achieved by the same level of power. Figure 3.8 further highlights the fact that different amount of heating could be achieved with the same power of the heat pump, and the temperature difference of the water played little role. When the power was around 5kW, any amount of heating across the board was achievable with temperature difference ranging from 15 to 30 degrees. If we do not take the temperature difference into account, the Pearson correlation coefficient between power and heating in February was 0.73, which is not very strong. Indeed, the efficiency of the heat pump is highly non-linear, and it might also be that the water in the heat pump was mixed with the water coming from the main system, whose effect has not been taken into account as we have no data for it. Therefore it is hard to tell what the effect of changing just the power level of the zonal plant heat pump would be on the space heating.

Things were even more bizarre in the summer months of July and August. As shown in figure 3.9, a lot of cooling was done in the first half of August. However, the cooling was done over night, when the temperature was lower and the heat pump was essentially *off* (a non-zero power consumption suggests that

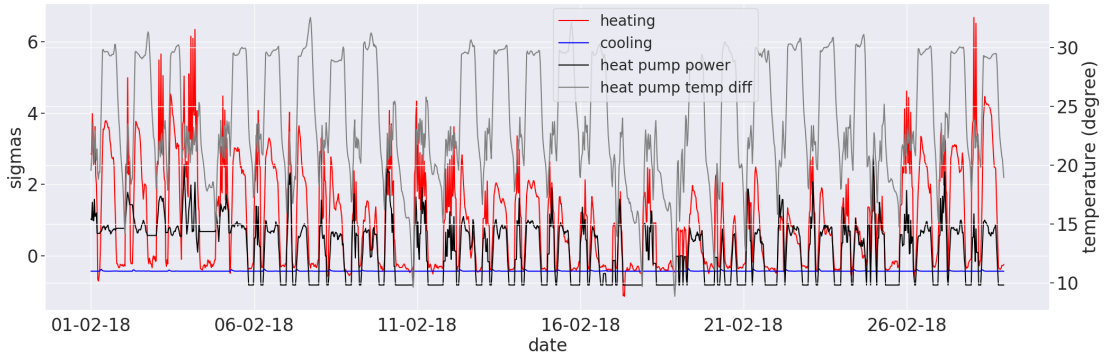

Figure 3.7: The heating peaks mostly coincided but clearly various levels of heating (red) could be achieved with the same level of power (black). The temperature difference between the hot and cold water also played little role in affecting the amount of heating being done.

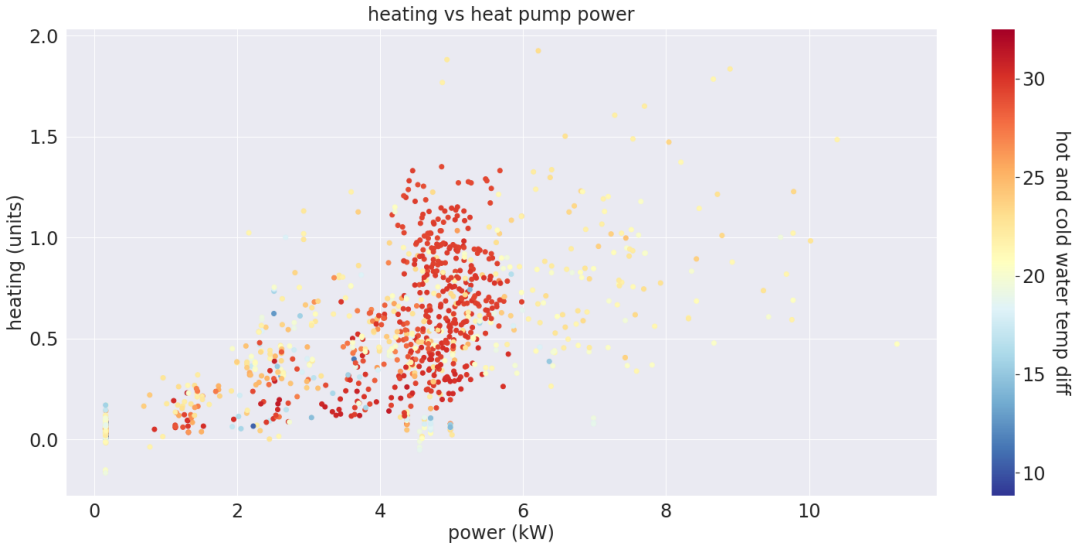

Figure 3.8: Scatter plot of heat pump power vs heating done in February. The correlation coefficient is a moderate 0.73. The temperature difference of the water in the hot and cold water circuits is shown in colour scale.

it was on standby mode). The scatter plot of heat pump power versus cooling in figure 3.10 also indicates that, irrespective of the temperature difference of the water in the heat pump, a wide range of cooling was done when the heat pump was operating at the minimum power level; alternatively, zero cooling could happen even when the heat pump was operating at high power. If we ignore the temperature difference factor, the correlation coefficient between cooling and the power was -0.19 during that period. Thus we cannot find any explicit relationship between cooling and the heat pump power either. It is not possible to estimate how much flexibility one can get from the heat pump from this kind of intuitive analysis of just the plant zone BMS data, without knowing *all* the other components in

play and having their data taken into consideration.

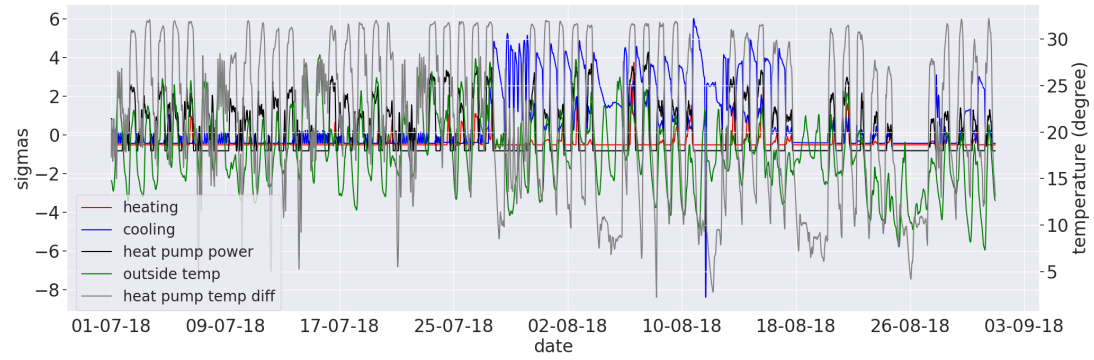

Figure 3.9: Over the summer, most cooling (blue) was done when the heat pump was operating at minimum level in the first half of August. The dip in mid August is most likely due to be some residual error in the data after data cleaning.

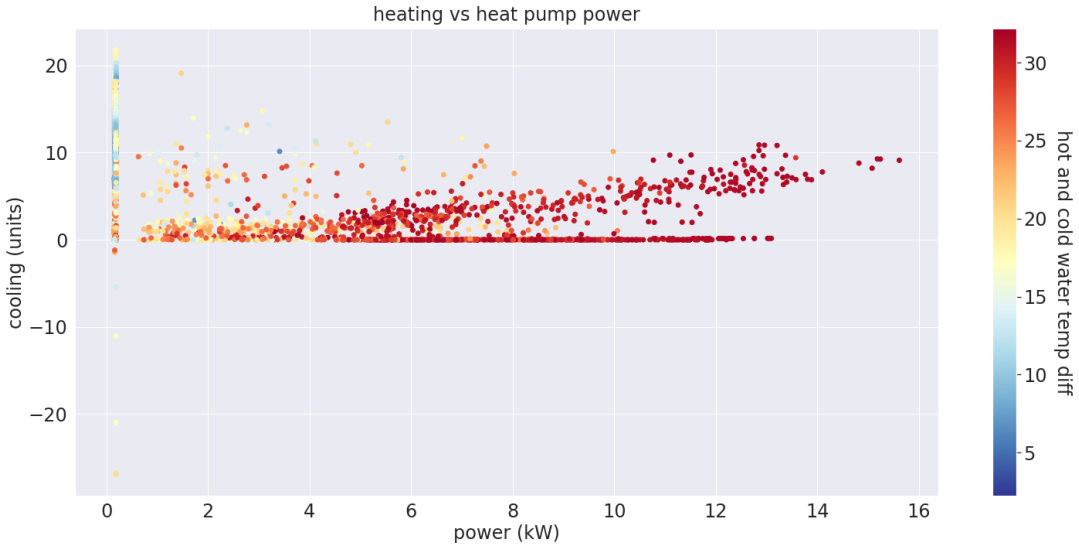

Figure 3.10: Scatter plot of heat pump power vs cooling done in the summer. Any level of cooling was achievable when the heat pump was not operating, whereas any power level could give zero cooling, irrespective of the temperature difference of the water in the heat pump. If we ignore these irregularities, then there is a moderate positive correlation of 0.41 between cooling and heat pump power.

3.4 Air Handling Unit

In order to be able to model how the space state variables change with the control variables *accurately* using machine learning methods, we need to have a large number of data points covering the *entire* target range of the state variables and spreading across the full range of allowed controls. However, since the inside temperature of Building B was always kept inside a narrow 5 degrees band between

20.5 and 25.5 degrees, no data-driven model would be able to provide a reliable prediction on how the indoor temperature well outside this range — all the way down to 15 degrees and up to 30 degrees in order to increase the flexibility for demand shift — would change with the HVAC power levels.

Fortunately during the data run, the air handling unit (AHU) was operating at approximately 20% lower power from 11/09/2018 until the end of the run, as shown in figure 3.11, possibly due to some setpoint change. We can therefore compare the carbon dioxide level in this 4.5 weeks period with the month before and see what the effect was, assuming that all other variables like weather and occupancy were more or less the same throughout the entire two-month period so they had negligible effect on the *change* in the CO2 level.

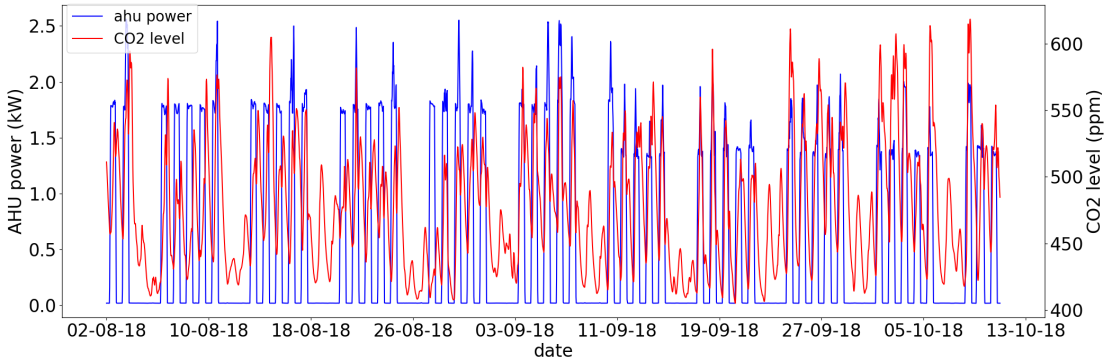

Figure 3.11: The AHU operated at a lower power level after 11/09/18. The CO2 concentration clearly peaked at higher levels on the weekdays after the change.

Let us take a closer look at how the CO2 level responded to a reduced AHU power. Let us focus on the weekdays as it is a matter of safety when the occupants are in; the AHU was already set to off over the weekends anyway. Figure 3.12 shows the daily average of the AHU power and the CO2 level before and after the change. There was a clear increase of about 20ppm in the peak CO2 level after the average power of the AHU was reduced by about 300W, which would translate to around 600kWh of *saving* per year. The increased level of CO2 was still well within the safety bound of 1000ppm; thus there is still room for further savings. This saving is however not the same as *flexibility*. In order to provide demand side response, a flexible load may be turned off completely for a time period ranging from minutes to several hours, and this is likely to happen during the peak hours of 4-7pm. This is fundamentally different from simply running the system at lower power continuously over the course of the day to save energy. According to the data, the AHU was only turned off at 7pm on working days; we cannot estimate from the data how the indoor CO2 level would build up if we were to turn off the AHU

at 4pm instead in order to provide demand side response, because 7pm and 4pm mean completely different operation conditions: at 4pm most occupants would still be inside the building and exhaling CO2 and be affected by its concentration; in fact the occupancy also has an impact on the zonal temperature because people have body heat, but not at 7pm when most of them would have left. Therefore in order to capture how CO2 would build up when people are in, it is necessary to experiment and turn the power down or completely off *during the working hours*. Without knowing the rate at which CO2 builds up at different power levels of the AHU, it is not possible to estimate by how much its energy consumption can be shifted safely.

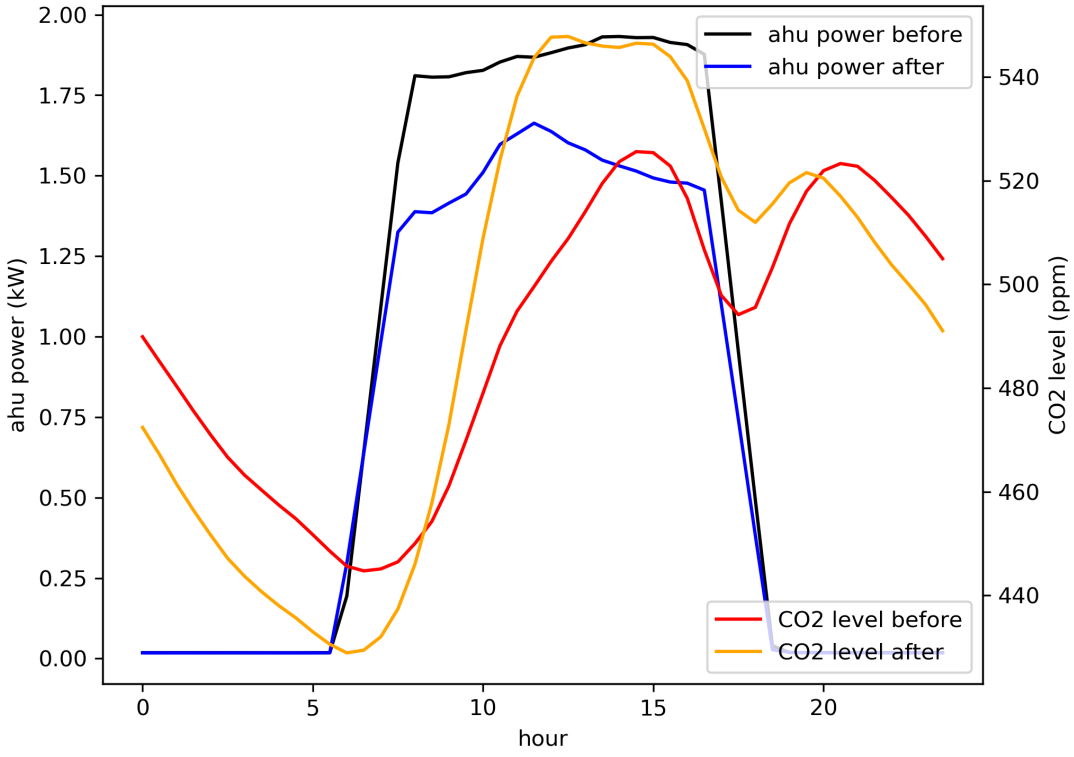

Figure 3.12: As the average AHU power reduced from 1.88kW to 1.52kW during the office hours, the average peak CO2 concentration increased from 526ppm to 548ppm. A concentration of 1000ppm starts causing drowsiness. We can also see that after the AHU was turned off (standby mode, this is why the power is above 0) at 7pm, CO2 concentration started increasing up until around 9pm, as some people could still be working in the building, then it diffused away naturally over night.

3.5 Discussion

In this chapter, we examined carefully the BMS data from a plant zone in a building in Newcastle, in attempt to estimate how much of the HVAC load could

be shifted. We looked for correlations between the power levels of various HVAC units and different state variables of the system, but could not find any consistent inference for the heating or cooling system. We believe this is due to the main heating and cooling systems of the building also contributing to the temperature regulation of the zone, which we have not accounted for as we do not have the data. For the ventilation system, we found that when the mean AHU power was reduced by 19% from 1.88kW to 1.52kW over the working hours, the peak CO2 concentration only increased by 4.2 % from 526ppm to 548ppm, and was still well inside the safety threshold. However, this is just energy saving and not the flexibility we are looking for — we want to see by how much we would need to crank up the AHU before the peak hours so that we can turn it down, or perhaps off from 4pm. In order to estimate this, we need some data to show how the CO2 level would build up if the AHU were to be turned down *during* the working hours when the occupants are in, and not at 5-7pm as it was always done in the data run, because the building would be mostly empty by then.

Indeed this case study highlights the biggest challenge in developing a data-driven MPC-based smart building energy management system — there is insufficient data to learn and build the predictive model from. Conventional BMS that runs on simple feedback loops are bounded to keep the indoor environment to be within the measurement error and some small tolerance of the setpoints. The rigidity of the system means there is barely any data outside this small range of the state space. However, shifting the HVAC load away from the peak hours without a battery relies on the relaxing the indoor environment requirement as written on the setpoints. Therefore without sufficient data that cover more corners of the overall configuration space, it is not possible to harness the flexibility of the HVAC load in a safe and "smart" manner.

Our study also lays bare the deficiency in relying only on the BMS data to understand the dynamics between the HVAC system and indoor environment, in hope that the data will reveal all, especially when the available data do not even represent the full system. Our data only cover one of the many plant zones in Building B and we do not know how the main system serving the entire building interacts with it; so at best we can only account for a subset of all the relevant factors affecting that particular zone. This is why we have not been able to find any meaningful conclusion from the data regarding the potential flexibility. We also lack the engineering knowledge of HVAC system itself — we were hoping that the data would tell how the different subsystems interact with each other, which

turned out to be extremely challenging.

All these observations seem to cast doubt on whether the data-drive modelling approach really is more advantageous than the physical engineering approach in terms of reliability, efficiency and scalability for developing the predictive model for a smart energy management system. While the expert knowledge from engineers are no doubt crucial for building a reliable predictive model, as we shall see in the next chapter, machine learning can serve as a very useful, efficient and scalable complement.

Chapter 4

Temperature prediction with Gaussian Process

As we discussed in the previous chapter, the data collected by existing BMS usually cover only a very narrow range of the state space. In order to fully exploit the flexibility from the HVAC system, ideally we need a model that can extrapolate into the part of the state space where data points are not yet present. However, one cannot expect any data-driven model to give an accurate prediction in regions of the state space not covered by data, therefore it is important to be able to address how *confident* we are about the predictions before we pass them onto the control optimisation algorithm. Thus we need a machine learning tool that predicts not only how the state variables respond to the controls, but also the *error* associated with the predictions, i.e.

$$x_{t+1} = f\left(x_t, u_t, d_t\right) + \varepsilon \ , \tag{4.1}$$

where f is the prediction and ε is the error. One way to do this is via a Bayesian stochastic regression method, namely the Gaussian Process, which can be treated as a distribution over random *functions*. Originally developed by Krige [17], it was reintroduced to the machine learning community by [18], and was first applied within the MPC context in [19]. We shall review the mathematics of the powerful Gaussian Process model and illustrate how to apply it in the Q-PLUS smart HVAC control system in this chapter. We follow closely the formulation and notation used in [20] in our review.

4.1 Bayesian Regression

In regression problems, we learn a map from the input space $\mathcal{X} \in \mathbb{R}^n$ of n-dimensional vectors to an output space $\mathcal{Y} \in \mathbb{R}$ of real valued targets. In a frequentist setting, one would attempt to identify the "best-fit" model of the data to make "best guess" predictions for new inputs, with the assumption that there exists *true* values of the model parameters. This is done by finding model param-

eters that maximise the *likelihood* of the data given the model. The linear least square method is an example of a maximum likelihood estimation (MLE) method. In a Bayesian setting however, only the data are *faithful* and the aim is to find the *probability distributions* that characterise the model parameters. In other words, there are no "true" values of the parameters, just that some values are more probable than the others.

To illustrate the key difference between the two approaches, let $\mathcal{D}$ denote observed data and let $\theta \in \Theta$ be some model or hypothesis of interest characterised by parameters θ. The Bayes' theorem states that

$$p\left(\theta|\mathcal{D}\right) = \frac{p\left(\mathcal{D}|\theta\right) p\left(\theta\right)}{p\left(\mathcal{D}\right)}$$

where $p\left(\theta|\mathcal{D}\right)$ is the *posterior* probability of obtaining the model θ given the observations $\mathcal{D}$, $p\left(\mathcal{D}|\theta\right)$ is the *likelihood* of obtaining $\mathcal{D}$ under the model θ, $p\left(\theta\right)$ is the *prior* probability distribution of the model θ which quantifies our prior beliefs about the reality before making observations, and $p\left(\mathcal{D}\right)$ is the probability of obtaining the observations D, which is effectively a scaling constant. The goal of Bayesian analysis is to find the *posterior* probability distribution, which then allows one to estimate the output y given some input x and with an *error bar*. The likelihood $p\left(\mathcal{D}|\theta\right)$ is the quantity which a frequentist model maximises, where the best fit model is given by the maximum likelihood estimate[1]

$$\hat{\theta}_{\mathrm{MLE}} = \underset{\theta}{\operatorname{argmax}}\, p\left(\mathcal{D}|\theta\right) \ .$$

Since the posterior is proportional to the multiple of the likelihood and prior, there is also a semi-Bayesian method, namely the maximum a posteriori probability (MAP) estimation which is closely related to MLE, but with an augmented optimisation objective which incorporates our prior knowledge of the problem. MAP estimation thus finds the model that best fits the data by finding the mode (maximum) of the posterior distribution, i.e.

$$\hat{\theta}_{\mathrm{MAP}} = \underset{\theta}{\operatorname{argmax}}\, p\left(\theta|\mathcal{D}\right) \ .$$

Note that if the prior distribution is uniform, then MAP and MLE are then the same.

[1] We assume for simplicity there exists a unique solution, which is not always the case.

For a fully Bayesian treatment, let us write $\mathcal{D} := \{(x^{(i)}, y^{(i)}\}_{i=1}^m$ where $x^{(i)} \in \mathcal{X}$ and $y^{(i)} \in \mathcal{Y}$ be a training set of m i.i.d samples from some distribution. Given the prior $p(\theta)$, the *parameter posterior* can be found using Bayes' theorem

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_{\theta'} p(\mathcal{D}|\theta')p(\theta')\mathrm{d}\theta'} = \frac{p(\theta)\prod_{i=1}^m p(y^{(i)}|x^{(i)},\theta)}{\int_{\theta'} p(\theta')\prod_{i=1}^m p(y^{(i)}|x^{(i)},\theta')\mathrm{d}\theta'} , \tag{4.2}$$

where we have explicitly written out $p(\mathcal{D})$ by integrating over the parameters. Let us denote the test input by x^* and the test output we are after y^*. A Bayesian method gives a distribution for y^*, known as the *posterior predictive distribution*, instead of a single guess for the value of y^* as in the frequentist case. Assuming the distribution of *noise* (random fluctuation) does not change so that it is the same for both the training and testing sets, the posterior predictive distribution is therefore given by

$$p(y^*|x^*,\mathcal{D}) = \int_\theta p(y^*|x^*,\theta)p(\theta|\mathcal{D})\ \mathrm{d}\theta\ . \tag{4.3}$$

In general however, the integrals in (4.2) and (4.3) are very difficult to compute.

Clearly, the complexity of the model is related to the number of dimensions of the parameter space Θ. A nonparametric model does not mean there are no parameters. A parametric model assumes a fixed, finite dimensional parameter space Θ; given the parameters (or the distribution of parameters), future predictions of output y^* from input x^* are independent of the observed data i.e $p\left(y^*|\theta, x^*, \mathcal{D}\right) = p\left(y^*|\theta, x^*\right)$, as the finite set of parameters encode *all* the information inferred from the data. This greatly restricts the complexity of the model: the complexity is bounded by the fixed dimension of Θ even though the sample size can be unbounded. A nonparametric model on the other hand, allows the dimension of the parameter space to grow with the dataset by starting with an infinite-dimensional parameter space, and invoking a finite subset of parameters on any given finite dataset, hence giving the model much greater flexibility and the ability to capture more structures as more observations are made.

4.2 Bayesian Linear Regression

Let us demonstrate the principles of parametric Bayesian methods with an example. The standard probabilistic interpretation of *linear* regression from m

samples is that

$$y^{(i)} = \theta \cdot x^{(i)} + \varepsilon^{(i)} \qquad \forall\, i \in [1, m]$$

where $\theta \in \mathbb{R}^n$ is the vector of parameters and $\varepsilon^{(i)} \in \mathbb{R}$ is the i.i.d. noise which follows $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Let us explicitly write

$$X := \begin{bmatrix} \left(x^{(1)}\right)^T \\ \left(x^{(1)}\right)^T \\ \vdots \\ \left(x^{(n)}\right)^T \end{bmatrix} \in \mathbb{R}^{m \times n}\,, \qquad Y := \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m\,, \qquad \varepsilon := \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(m)} \end{bmatrix} \in \mathbb{R}^m\,.$$

To begin with, we need to choose a *prior* distribution over the model parameters. This is typically chosen to be Gaussian $\theta \sim \mathcal{N}(0, \Lambda^{-1})$, where Λ is called the precision matrix or inverse covariance. A common choice for the precision matrix is $\Lambda = \lambda I^{n \times n}$ with $\lambda \in \mathbb{R}^+$ quantifying our prior belief on the concentration of the model parameters θ. In this case the parameter posterior (4.2) and the posterior predictive distribution (4.3) are actually tractable. A good few lines of algebra reveals that, defining $A := \frac{1}{\sigma_\varepsilon^2} X^T X + \lambda I$

$$\begin{aligned} \theta|\mathcal{D} &\sim \mathcal{N}\left(\tfrac{1}{\sigma_\varepsilon^2} A^{-1} X^T Y, A^{-1}\right) \\ y^*|x^*, \mathcal{D} &\sim \mathcal{N}\left(\tfrac{1}{\sigma_\varepsilon^2} x^{*T} A^{-1} X^T Y, x^* A^{-1} x^* + \sigma_\varepsilon^2\right)\,. \end{aligned}$$

Hence the test output y^* follows a Gaussian distribution, with the uncertainty[2] arising from both the random fluctuations ε^* and the uncertainty around the parameters θ.

4.3 Multivariate Gaussian Distribution

Gaussian random variables are extremely useful not only because of the central limit theorem and that noise often exhibits Gaussian distribution, there are many desirable analytical properties of the Gaussian distribution which significantly simplify the calculations, as we have already seen in the previous section. Consider a random vector $x \in \mathbb{R}^n$ with multivariate Gaussian distribution $x \sim \mathcal{N}(\mu, \Sigma)$, i.e.

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^n |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

[2]Recall that y^* is a *distribution* over the target and the spread in the distribution offers a natural measure of *uncertainty*.

with the assumption that the covariance matrix Σ is nondegenerate; this clearly reduces to the usual Gaussian distribution

$$p\left(x;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

for $n = 1$. Now let us consider partitioning $\mathbb{R}^n$ into two subspaces, so that the variables in the random vector $x \in \mathbb{R}^n$ are partitioned into two sets $x_A = [x_1 ... x_r]^T \in \mathbb{R}^r$ and $x_B = [x_{r+1} ... x_n]^T \in \mathbb{R}^{n-r}$, such that μ and Σ are partitioned accordingly:

$$x = \begin{bmatrix} x_A \\ x_B \end{bmatrix}, \qquad \mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \qquad \text{and} \qquad \Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix},$$

where $\Sigma_{BA} = \Sigma_{AB}^T$ as $\Sigma^T = \Sigma$. It follows that:

1. **Marginalisation**. The marginal densities, i.e.

$$\begin{aligned} p(x_A) &= \textstyle\int_{x_B} p(x_A, x_B; \mu, \Sigma)\, \mathrm{d}x_B \\ p(x_B) &= \textstyle\int_{x_A} p(x_A, x_B; \mu, \Sigma)\, \mathrm{d}x_A \end{aligned}$$

   are also Gaussian

$$\begin{aligned} x_A &\sim \mathcal{N}(\mu_A, \Sigma_{AA}) \\ x_B &\sim \mathcal{N}(\mu_B, \Sigma_{BB})\ . \end{aligned}$$

2. **Conditioning**. The conditional densities, i.e.

$$\begin{aligned} p(x_A|x_B) &= \frac{p(x_A,x_B;\mu,\Sigma)}{\int_x^A p(x_A,x_B;\mu,\Sigma)\mathrm{d}x_A} \\ p(x_B|x_A) &= \frac{p(x_A,x_B;\mu,\Sigma)}{\int_x^B p(x_A,x_B;\mu,\Sigma)\mathrm{d}x_B} \end{aligned}$$

   are also Gaussian

$$\begin{aligned} x_A|x_B &\sim \mathcal{N}\left(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\right) \\ x_B|x_A &\sim \mathcal{N}\left(\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}\right)\ . \end{aligned}$$

The fact that the probability density normalises

$$\int_x p(x;\mu,\Sigma)\mathrm{d}x = 1$$

can be handy in computing some complicated integrals which look not too dissimilar to the probability density function of the Gaussian distribution, such as those

of the form

$$\int_x \exp\left(-\frac{1}{2}x^T Ax - x^T b - c\right) \mathrm{d}x$$

by "completing the square". Furthermore, if $y \sim \mathcal{N}(\mu_y, \Sigma_y)$ and $z \sim \mathcal{N}(\mu_z, \Sigma_z)$ are two *independent* vector-valued Gaussian random variables of the same dimensionality, the their sum is also Gaussian with

$$y + z \sim \mathcal{N}\left(\mu_y + \mu_z, \Sigma_y + \Sigma_z\right) \ .$$

4.4 Gaussian Process

As the system dynamics between the indoor environment and the HVAC systems are highly non-linear, Bayesian linear regression would be over restrictive for our purpose. Since we do not know the functional form of f in our sough after state equation (4.1), we would like to consider *all* possible functions in our prior and examine the most likely ones.

How are probability distributions parametrised over functions? Let us first consider a finite set of m elements $\mathcal{X} = x_1, ..., x_m$ and let $\mathcal{H}$ be the set of all possible functions mapping $\mathcal{X}$ to $\mathbb{R}$. In this case we may represent an example function $f_1(\mathcal{X}) \in \mathcal{H}$ as a m-dimensional vector $\vec{f_1}$ and specify a probability density over each $\vec{f}$. For instance, if $\vec{f} \sim \mathcal{N}(\vec{\mu}, \sigma^2 I^{m\times m})$, then the probability distribution over the random functions f is given by

$$p(f) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(f(x_i) - \mu_i)^2\right) \ . \tag{4.4}$$

Observe that after a random function f_0 is drawn from $\mathcal{H}$ *probabilistically*, f_0 is a *deterministic* mapping $f_0 : \mathcal{X} \to \mathbb{R}$. It is not the output of f_0 that is probabilistic, but the drawing of f_0. Here the domain of the functions f has finite size, so how can we extend the domain size to be infinite?

Recall that a stochastic process is a set of random variables $\{f(x) : x \in \mathrm{X}\}$ indexed by the elements x from some index set $\mathcal{X}$. We denote the random variable by $f(x)$ because in regression problems, the index set is often taking to be the d-dimensional input space i.e. $\mathcal{X} = \mathbb{R}^d$. A Gaussian process is a stochastic process such that *any finite sub-collection* of random variables has a multivariate Gaussian distribution: if the collection $\{f(x) : x \in \mathrm{X}\}$ is said to be drawn from a Gaussian

Process with mean function $m(x)$ and covariance function $k(x, x')$, denoted by

$$f(x) \sim \mathcal{GP}\left(m(x), k(x, x')\right) \ , \tag{4.5}$$

then for any finite set of elements $x_1, ..., x_m \in \mathcal{X}$, the corresponding set of random variables $f(x_1), ..., f(x_m)$ follows the distribution

$$\begin{bmatrix} f\left(x_1\right) \\ \vdots \\ f\left(x_m\right) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m\left(x_1\right) \\ \vdots \\ m\left(x_m\right) \end{bmatrix}, \begin{bmatrix} k\left(x_1, x_1\right) & \cdots & k\left(x_1, x_m\right) \\ \vdots & \ddots & \vdots \\ k\left(x_m, x_1\right) & \cdots & k\left(x_m, x_m\right) \end{bmatrix} \right) \ . \tag{4.6}$$

Just like the multivariate Gaussian, the *mean function* and *covariance function* have the interpretations

$$m(x) = \mathbb{E}\left[f(x)\right] \tag{4.7}$$

$$k\left(x, x'\right) = \mathbb{E}\left[\left(f(x) - m(x)\right)\left(f(x') - m(x')\right)\right] \ . \tag{4.8}$$

Thus a function $f(x)$ drawn from a Gaussian process can be treated as a very high-dimensional vector representing $f(x)$ drawn from a very high dimensional multivariate Gaussian, each dimension corresponding to an index x in the index set $\mathcal{X}$. Due to the nice marginalisation property of multivariate Gaussians, the marginal distribution of any sub-collection of $f(x)$ is also a multivariate Gaussian. This is the key for making predictions in Gaussian processes, as we shall see below.

Supposed we have a training dataset of m i.i.d samples $\mathcal{D} := (X, Y) = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ where $x^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in \mathbb{R}$ (thus $X \in \mathbb{R}^{m \times n}$ and $Y \in \mathbb{R}^m$). For computational convenience, let us further assume that the training set is already z-transformed, so it is zero-meaned and dimensionless. We want to fit a regression model

$$y^{(i)} = f\left(x^{(i)}\right) + \varepsilon^{(i)} \tag{4.9}$$

where $\varepsilon^{(i)} \sim \mathcal{N}\left(0, \sigma^2\right)$ is some i.i.d. noise which follows an independent Gaussian distribution. Now instead of specifying a prior distribution for the model parameters θ of some given functions as in the Bayesian linear regression model, in a Gaussian process regression model, we specify the prior distribution of the random function $f(x)$ to be a Gaussian process

$$f(x) \sim \mathcal{GP}(0, k(x, x')) \tag{4.10}$$

for some *appropriate* covariance function $k(x, x')$. We shall come back to what makes a covariance function *appropriate* after deriving the Gaussian process predictions.

Let $X^* = \{x^{*(j)}\}_{j=1}^{l} \in \mathbb{R}^{l \times n}$ be a set of l test inputs and $Y^* = \{y^{*(j)}\}_{j=1}^{l} \in \mathbb{R}^{l}$ be the corresponding test outputs. We may assume that the test set $T = (X^*, Y^*)$ and the data $\mathcal{D}$ are mutually independent. Our goal is thus to find the posterior predictive distribution $p(Y^*|X^*, \mathcal{D})$, and this is where the nice partitioning property of multivariate Gaussian comes into play: since any function $f(x)$ drawn from a Gaussian Process prior over *any* set of input points x belonging to the index set $\mathcal{X}$ must follow a joint multivariate Gaussian, the same must also be true for the composite set of inputs $X + X^*$. Furthermore, since $X + X^*$ is naturally partitioned, we have

$$\left.\begin{bmatrix} f \\ f^* \end{bmatrix}\right| X, X^* \sim \mathcal{N}\left([0], \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*X^*) \end{bmatrix}\right) , \tag{4.11}$$

where

$$\begin{aligned} f &= \left[f(x^{(1)}) \cdots f(x^{(m)})\right]^T \in \mathbb{R}^m \\ f^* &= \left[f(x^{*(1)}) \cdots f(x^{*(l)})\right]^T \in \mathbb{R}^l \\ (K(X,X))_{i,j} &= k(x^{(i)}, x^{(j)}) , \qquad K(X,X) \in \mathbb{R}^{m \times m} \\ (K(X,X^*))_{i,j} &= k(x^{(i)}, x^{*(j)}) , \qquad K(X,X^*) \in \mathbb{R}^{m \times l} \\ (K(X^*,X))_{i,j} &= k(x^{*(i)}, x^{(j)}) , \qquad K(X^*,X) \in \mathbb{R}^{l \times m} \\ (K(X^*,X^*))_{i,j} &= k(x^{*(i)}, x^{*(j)}) , \qquad K(X^*,X^*) \in \mathbb{R}^{l \times l} . \end{aligned} \tag{4.12}$$

Similarly for the i.i.d. noise, we have the decomposition

$$\begin{bmatrix} \varepsilon \\ \varepsilon^* \end{bmatrix} \sim \mathcal{N}\left([0], \begin{bmatrix} \sigma^2 I^{m \times m} & [0]^{m \times l} \\ [0]^{l \times m} & \sigma^2 I^{l \times l} \end{bmatrix}\right) , \tag{4.13}$$

with $\varepsilon \in \mathbb{R}^m$ and $\varepsilon^* \in \mathbb{R}^l$. Then by the summing property of multivariate Gaussian, we have

$$\left.\begin{bmatrix} y \\ y^* \end{bmatrix}\right| X, X^* = \begin{bmatrix} f \\ f^* \end{bmatrix} + \begin{bmatrix} \varepsilon \\ \varepsilon^* \end{bmatrix} \sim \mathcal{N}\left([0], \begin{bmatrix} K(X,X) + \sigma^2 I^{m \times m} & K(X,X^*) \\ K(X^*,X) & K(X^*X^*) + \sigma^2 I^{l \times l} \end{bmatrix}\right) . \tag{4.14}$$

Last but not least, using the conditioning property, the posterior predictive distribution can be found straightforwardly:

$$y^*|X^*, X, y \sim \mathcal{N}(\mu^*, \Sigma^*) \tag{4.15}$$

with

$$\begin{aligned} \mu^* &= K(X^*, X)\left(K(X, X) + \sigma^2 I^{m\times m}\right)^{-1} y \\ \Sigma^* &= K(X^*, X^*) + \sigma^2 I^{l\times l} - K(X^*, X)\left(K(X, X) + \sigma^2 I^{m\times m}\right)^{-1} K(X, X^*) \end{aligned} \tag{4.16}$$

Although the derivation of Gaussian process regression is nice and neat, the main computation challenge lies in the matrix inversion $(K(X, X) + \sigma^2 I^{m\times m})^{-1}$. As it is a $m \times m$ matrix, its inverse is of $\mathcal{O}(m^3)$ complexity.

While the prior mean can be taken logically and straightforwardly from the training sample, we still need to specify an appropriate covariance function $k(x, x')$, also known as the kernel, which plays a central in the Gaussian Process. In fact, it encodes all the information about how different points in the input space covary with each other in the output (or target) space, thus completely defines the behaviour of the Gaussian process. As it specifies the covariance matrix of the multivariate Gaussian, it must be positive semidefinite. The positive semidefiniteness also ensures that the Gaussian process can be represented as an infinite linear combination of orthogonal functions by the Karhunen–Loève theorem, analogous to the Fourier series decomposition of a function on a bounded interval. Kernels can be multiplied and combined to make new ones, which is extremely useful in regressing different types of features in the input space. For our purpose, we expect the state equation to be smooth such that nearby points in the input space should be mapped to nearby points in the output space. Other properties to be considered include stationarity, isotropy and periodicity. A covariance function is stationary if $k(x, x') = k(x - x')$, isotropic if $k(x, x') = k(|x - x'|)$, and periodic if $k(x, x') = k(\sin(x - x'))$. The most commonly used covariance function is the square-exponential (SE) kernel, also known as the radial based function (RBF) kernel,

$$k_{SE}(x, x') = \exp\left(-\frac{1}{2\ell^2}||x - x'||^2\right) \tag{4.17}$$

where ℓ is the characteristic length scale of variation. The square-exponential kernel is smooth, stationary and isotropic. In fact most of the standard kernels

are stationary; the simple linear kernel

$$k_L(x, x') = x \cdot x' \tag{4.18}$$

alone is just Bayesian linear regression but it is nevertheless non-stationary, therefore it can be useful in introducing non-stationarity in combined kernels.

Coming back to our specific problem of HVAC control, by comparing the state equation (4.1) and the general regression model (4.9), it is easy to see that we can apply Gaussian process regression to obtain (4.1) by taking the regression output y to be the state at the next time step, and the model input to be the state, control and external disturbance at the current time step, i.e. $y = x_{t+1}$ and $x = \{x_t, u_t, d_t\}$.

4.5 Step-ahead temperature model

Building C is another building in Manchester. Unlike Building B, as shown in figure 4.1, Building C's HVAC system is rather simple, which is more suitable for a proof-of-concept demonstration of the application of Gaussian process regression in data-driven building energy modelling. The data also has a temporal granularity of 15 minutes, which is more suitable for HVAC control than the half-hour temporal steps the electricity market operates in.

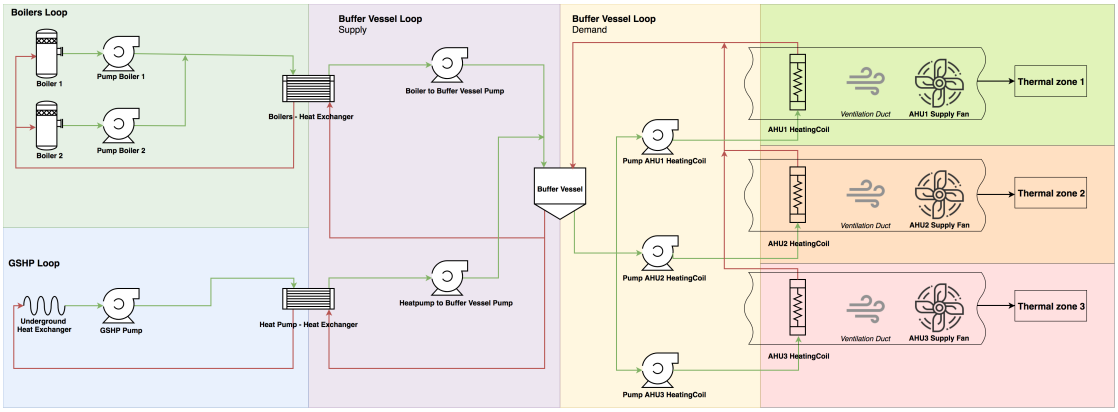

Figure 4.1: Schematic of the HVAC system in Building C (courtesy of the University of Manchester).

We shall focus on the flexibility of AHU1, and treat the boiler and underground heat exchange systems on the left hand side of figure 4.1 as external factors. In fact we only need to take into account the supply and return temperatures in the buffer vessel on the supply side as the all the effects of the boiler and GSHP loops

manifest in the buffer vessel. The raw time series data we obtained privately from the University of Manchester are indeed synthetic data generated with EnergyPlus, based on the weather and some other relevant data in the year of 2002. The dataset has been validated by engineering experts, therefore we may treat it as the ground truth even though it was simulated.

Our task is therefore to come up with a regression model that predicts how the temperature in zone 1 responds to the power of the AHU1, thus the state we want to predict is simply the one-dimensional scalar zone temperature, the control u is just the power of AHU1 heating coil, and the disturbances d will account for all other relevant factors such as outside temperature, humidity and the temperatures in the buffer vessel loop.

Since detailed records (or in this case, simulation) of BMS data are hard to obtain, before we set off and draw up a list of features to be fed into the Gaussian Process, let us first look at what is available from the data. It is clear that not all the features recorded in the raw data file are relevant to our problem of finding the flexibility of AHU1. As illustrated in figure 4.1, the AHUs act as a temperature boost to the buffer vessel demand loop. We may think of the part of heat transfer done by the buffer vessel being proportional to the temperature difference between the supply and return temperatures and treat it as one of the external factors in d. The weather data is however not present. We therefore use the 2002 weather data from Manchester airport as this is geographically the closest weather station to Building C where the 2002 data are publicly available. The weather data is taken from the website https://www.wunderground.com. In order to capture the effect of day-time factors, with which the occupancy, a very important feature for building energy model but is unfortunately usually unavailable, is strongly correlated, we also input the extra engineered feature of whether or not the data point is in the working hours.

Thus the input we use for the Gaussian Process (GP) model are: zone 1 air temperature Z, AHU1 heating coil power P, buffer vessel supply and return temperatures S and R, their difference $\Delta = S - R$ outside temperature T, dew point temperature D, humidity h, wind speed W, pressure p and weather condition C (fair, cloudy, windy). The direct sum of these 11 features defines the 11-dimensional input space $\mathcal{X}$, and the output is simply the zone 1 air temperature at the next

time step denoted by Z', which is a scalar, given by the Gaussian Process

$$Z'(x) \sim \mathcal{GP}\left(m(x), k(x, x')\right) \tag{4.19}$$

where the kernel is chosen to be the composite kernel $k = k_C \cdot k_{SE}$, the multiple of the constant kernel and the square exponential kernel for its smoothness. Note that the time index in the original time series plays no part here — in this formulation we only want to obtain the zone temperature for the *next* time step, therefore it is the underlying *physical conditions* that matter.

In order to illustrate how well the Gaussian Process model works and how it fits into the MPC problem, let us focus on a small region of the 'exogenous' input space $\mathcal{X}'$, the subspace of the input space $\mathcal{X}$ without the control input P. This small region thus contains the very similar instances, and ideally they would spread across the P dimension so that the Gaussian Process gives a nicely fitted regression when these exogenous dimensions are collapsed onto the P-Z' plane (i.e. each point on the P-Z' plane is 10-dimensional), which can then be treated simply as the function $Z' = Z'(P)$ with just one variable P, if all points are close enough in $\mathcal{X}'$. Then when we encounter an instance with similar inputs in $\mathcal{X}'$, we can just look up this function $Z' = Z'(P)$ and use it as the state equation for the MPC.

Since the dimensionality of the input space is high and we only have one year of data, we cannot look too closely into any specific region or else there would be no data point to work with. Let us focus on the peak hours between 4 and 7 pm on the weekdays; these are the time periods when we want to turn down or off the AHU to avoid the high prices. Because we are controlling the heating coil in the AHU which acts as a boost to the supply air temperature, the control should be highly dependent on the temperature difference in the buffer vessel Δ and the outside dew point temperature D. Let us divide each of these two dimensions in to 10 equidistant bins and select the 5th bin. By further selecting the data points when the weather condition was 'cloudy', we are already left with just 22 data points, with D ranging from 2.22 to 3.89 degrees, Δ from 15.78 to 19.4 degrees, P from 0 to 68W, and Z, Z' ranging from 18.91 to 21.75 degrees. By taking the 22nd point as the test point and the rest to train the Gaussian Process, we obtain model shown in figure 4.2. Despite the fact that we only have 21 data points to train the 11 dimensional model, the target is within 3 standard deviations from the prediction. Note that the observations (red dots) are not expected to sit on the mean function (black line) as they are projected onto the P-Z' plane with

the other 10 dimensions taking the values of the target. In other words the mean function shown here is the prediction for the zone temperature as a function of the power of the AHU when *all* other conditions, including the outside temperature and buffer vessel loop temperatures are the same as the blue target data point. The mean model function in figure 4.2 is almost flat, which means that there is

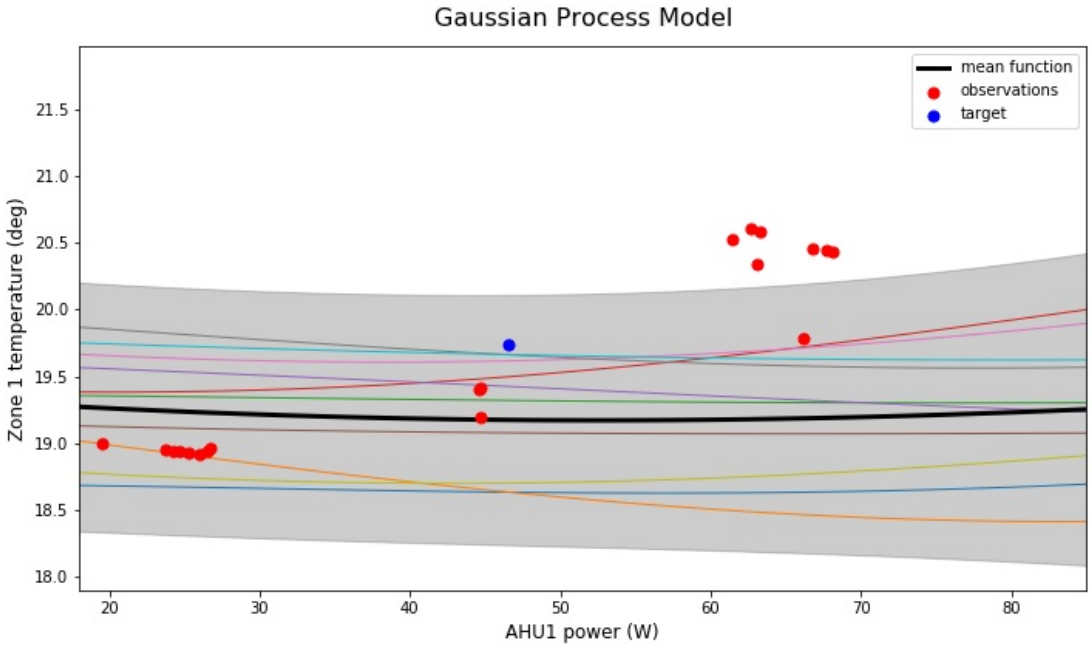

Figure 4.2: Gaussian Process model for one step ahead (15 minutes) zone temperature prediction as a function of the power of AHU1. The shaded region is the 3 standard deviations uncertainty range. The rainbow lines are posterior sample functions. Note that only half of the sample functions are increasing in the P-Z' plane as one would expect (i.e. the higher the power, the warmer). The model was implemented using *sklearn.gaussian_process.GaussianProcessRegressor*. The optimised kernel used for prediction was $1.21 \cdot RBF(\ell = 4.53)$.

huge flexibility from the AHU1 under these operating conditions. The predicted zone temperature stays very close to 19.25 deg no matter what the AHU1 power is. This is not so surprising as this is just one-step, 15 minutes ahead model but the building has huge thermal inertia; however, this is still a promising start for future work on extension to multi-step ahead prediction for the Q-PLUS system.

4.6 Discussion and future work

In this chapter, we reviewed the mathematical properties of the Gaussian Process model and demonstrated how it can be used to predict the zone temperature as a function of an HVAC control parameter given different operation conditions. We showed that even with only 21 data points (in a 11-dimensional

input space, which makes the training dataset *sparse*), the model can make good in-sample prediction with error within 3 standard deviations, thus the Gaussian Process model looks to be a promising option for predicting the indoor state in the Q-PLUS system. Of course, this really only is a starting point because we have only considered a one-step ahead model and the time step is only 15 minutes long. Further work is still needed to look into expanding the prediction horizon to multiple-step ahead, covering at least the coming hour, as well as the integration into the MPC framework. Another avenue for further research is to find the most suitable covariance function or kernel for the specific purpose of smart HVAC control. Here we have only considered a simple composite kernel, and we observed that the mean function in the example was not monotonically increasing in the P-Z' as we had expected. One may therefore try and emphasise the positive correlation between P and Z' by modifying the covariance function accordingly.

Going back to the basic question of whether machine learning models are indeed less complex to build and more scalable than physical models, several aspects need to be addressed. First of all, the complexity of Gaussian process notoriously scales cubically with the number of samples because of the matrix operations involved. In our example, we have only considered some very specific operation conditions and so there were only 21 training samples for the model. It will take further research to investigate how best to select a subset of all the available data points for model training in order to balance the trade off between accuracy and complexity. Secondly, the real test lies in the out-of-sample prediction — extrapolation into the regions of the state space forbidden by the current control system — which requires actual experimentation inside a building. Until then, it is not possible to fully assess the accuracy of either approach. Nevertheless, since machine learning models can be updated easily with new observations, their accuracy can always improve over time. Thus machine learning clearly holds advantage over physical modelling in terms of scalability and flexibility.

Chapter 5

Conclusion and future work

In this thesis, we conducted a pilot study for Q-PLUS, a data-driven model predictive control smart building energy management system for commercial buildings. In particular, we looked at several machine learning algorithms and examined their applications in the various functions of Q-PLUS. In the first chapter, we presented the optimal battery control as a model predictive control problem and implemented the optimisation algorithm in Python. Using the historical demand data as well as the corresponding weather and datetime information, we trained and compared various random forest and LSTM models for the half-hourly electricity demand prediction for the MPC. We tested their performances on both long-term (2 weeks plus) and short-term (up to one day ahead) forecasts. We found that for the long-term demand forecast, the random forest model that took into account the current datetime and weather information, as well as the weather conditions in the hour before (i.e. two previous time steps), gave the most accurate prediction, much better than its LSTM counterpart despite the more sophisticated architecture of the latter; the LSTM model also took almost twice as long to train. On the other hand, autoregressive random forest models were shown to be the most suitable for short-term demand prediction of up to 10 steps (5 hours) ahead for the real-time control. Interestingly, we also found that as we increased the forecast horizon to 48 steps (one day) ahead, autoregressive models gave less accurate predictions than the baseline non-autoregressive random forest model used for offline long-term forecast.

Autoregressive models that take also the immediate past observations of the demand into account can give more accurate demand predictions in the near future by making sure that the projected demand follows the same trend and level as in the previous few hours, so that there is better continuity with the "reality" as opposed to an offline model which cannot be updated. For example, a conference event being held in the building would cause a sudden surge in the electricity demand. An offline model that has learnt mostly from normal operation days would fail to predict this surge, whereas autoregression allows an online model to correct for the higher than usual demand on the go. Of course, special events are

usually scheduled in advance and the building manager usually has some idea of the number of attendees, so even an offline model should be able to capture such spikes by including occupancy as a training feature. However, it is actually a very challenging task to count the occupancy in a building accurately; this is why we have not been able to include it in any of our models. In fact, it is an active area of research to estimate the occupancy inside a building by using various proxy indicators such as CO2 level or the number of devices connected to the WiFi system.

We then investigated the possibility of shifting a building's HVAC demand without a battery in chapter two. Using the data from a building in Newcastle, we attempted to quantify the amount of flexibility the HVAC system could provide by exploiting the thermal inertia and natural ventilation in the building. However, due to the fact that conventional BMS only allows the indoor temperature and CO2 level to vary within some narrow setpoint range, we did not have sufficient data to cover a diverse enough set of operation conditions to estimate the flexibility. Indeed, the lack of data to "learn" or "train" from is the main obstacle to developing a machine learning predictive model for the smart building energy management system. Our analysis also highlighted the limitations of a purely data-driven approach to modelling the indoor climate as a function of the HVAC controls. Without expert engineering knowledge of the full system, there is little one could infer about the dynamics between the various subsystems and the indoor environment just from the data.

To remedy the lack of diverse enough data to learn from and to address the need to extrapolate the model into unprobed region of the state space, we proposed the application of Gaussian Process model in chapter three. Treating the sought after function as a high-dimensional vector where any finite collection of its samples follows a multivariate Gaussian distribution, the Bayesian stochastic method allows us to keep track of the uncertainty of its prediction. Thus we may take the indoor state gradually outside the existing setpoint range, and be confident at the level as reflected by the uncertainty range that the comfort and safety requirements would not be violated. Then as more data points are collected, especially in regions of the state space not covered previously, the accuracy of the model improves progressively. We demonstrated how to apply the Gaussian Process in the context of temperature control with a step-ahead example and showed that it gave promising result. However, further work needs to be taken to expand it into a multi-step ahead model as well as to fully incorporate it into the MPC framework. Computational cost is another issue that needs addressing as the complexity of Gaussian

Process models grows cubically with the training sample size; kernel engineering to find the most suitable covariant function for our specific application would be another interesting and important research problem. Furthermore, a real experiment in an actual building is the only way to assess the out-of-sample prediction of the Gaussian Process model. Hence a lot more work is yet to be done along this line of research both theoretically and empirically; what we have achieved in the thesis is only a promising starting point.

Before we close our discussion, let us revisit the fundamental question of whether machine learning approach is really *better* than physical modelling approach for developing the predictive model in an MPC based smart building energy management system. According to this study, the answer is an affirmative yes for the electricity demand forecast of the whole building, which is useful in optimising the control of the battery storage system for buildings fitted with batteries. We showed that machine learning methods such as random forest regressor can produce highly accurate predictive models that are simple, interpretable, scalable, and computationally inexpensive to run. On the other hand, much research is yet to be done for a smart HVAC control system. Without sufficient data that cover a diverse set of operation conditions, it is very challenging to develop a reliable and accurate machine learning predictive model. One clear advantage of a machine learning model though is that it can be updated easily with the new incoming data, thus the accuracy will always improve over time. On the contrary, physical models rely on first principles and are therefore highly complex to build and not scalable; any update of even a single parameter is likely to require much more human intervention and a comparatively expensive re-run of the whole complicated model; although relying on first principles also means that a physical model is much more likely to be correct than a machine learning model in regions of the state space not covered by data. All in all, as we illustrated in the Newcastle case study, one cannot go far with just the data and without the full knowledge of the dynamics of the HVAC system itself. In order to make the most of the existing BMS data via machine learning, it is crucial to first formulate the right question to ask with expert engineering insight. Thus until a full scale experiment can be conducted to enable pure data-driven modelling, the way forward to develop a reliable smart HVAC control system is by exploiting in combination the robustness of a physical model, as well as the adaptability, flexibility and scalability offered by the machine learning tools.

Appendix A: Linear-Quadratic Regulator

In this appendix we review the linear quadratic regulator, one of the most fundamental problems in control theory. Consider the general continuous time control problem starting from initial time t_0 and finishing at a fixed termination time t_f:

$$\begin{aligned} \text{minimise} \qquad & J = \psi\left(x\left(t_f\right)\right) + \textstyle\int_{f_0}^{f_t} L\left(x(t), u(t), t\right) \mathrm{d}t && (5.1) \\ \text{subject to} \qquad & \dot{x} = f\left(x(t), u(t), t\right) && (5.2) \\ & x\left(t_0\right) = x_0 \end{aligned}$$

where $\psi\left(x\left(t_f\right)\right)$ is the termination cost and x_0 is the initial state which is known. In addition, we assume that L is non-negative. The general procedure for solving for an optimal control solution is by applying the calculus of variations. Using a Lagrange multiplier $\lambda(t)$ associated with the state equation constraint (5.2), which is also called the costate, the cost is augmented into

$$\bar{J} = \psi\left(x\left(t_f\right)\right) + \int_{t_0}^{t_f} \left(L + \lambda^T\left(f - \dot{x}\right)\right) \mathrm{d}t \ , \tag{5.3}$$

and so

$$\begin{aligned} \delta\bar{J} \quad & = \psi_x\left(x\left(t_f\right)\right) \delta x\left(t_f\right) \\ & + \int_{t_0}^{t_f} \left(L_x \delta x + L_u \delta u + \lambda^T f_x \delta x + \lambda^T f_u \delta u - \lambda^T \delta\dot{x}\right) \mathrm{d}t \ . \end{aligned} \tag{5.4}$$

Using integration by parts, the last term can be rewritten as

$$-\int_{t_0}^{t_f} \lambda^T \delta\dot{x} \ \mathrm{d}t = \lambda^T\left(t_0\right) \delta x\left(t_0\right) - \lambda^T\left(t_f\right) \delta x\left(t_f\right) + \int_{t_0}^{t_f} \dot{\lambda}^T \delta x \ \mathrm{d}t \ , \tag{5.5}$$

which gives

$$\begin{aligned} \delta\bar{J} \quad = \quad & \left[\psi_x\left(x\left(t_f\right)\right) - \lambda^T\left(t_f\right)\right] \delta x\left(t_f\right) + \int_{t_0}^{t_f} \left(L_u + \lambda^T f_u\right) \delta u \ \mathrm{d}t \\ & + \int_{t_0}^{t_f} \left(L_x + \lambda^T f_x + \dot{\lambda}^T\right) \delta x \ \mathrm{d}t + \lambda^T\left(t_0\right) \delta x\left(t_0\right) \ . \end{aligned} \tag{5.6}$$

The last term has to vanish because the initial state is fixed; as we can vary u, x and $x(t_f)$ independently, the remaining three components must also vanish independently. Hence

$$\begin{aligned} \dot{\lambda} &= -\left(L_x + \lambda^T f_x\right) \\ \lambda^T(t_f) &= \psi_x(x(t_f)) \\ L_u + \lambda^T f_u &= 0\,. \end{aligned} \tag{5.7}$$

Note that the costate λ evolves in reverse time — it propagates backwards from the final state.

Now let us consider explicitly the linear time-invariant convex optimal control problem with linear dynamics

$$\dot{x} = Ax + Bu \tag{5.8}$$

and quadratic cost

$$L = \tfrac{1}{2}x^T Q x + \tfrac{1}{2}u^T R u \qquad\qquad Q, R \succeq 0 \tag{5.9}$$

and let us consider the case where there is no terminal cost i.e. $\psi = 0$ for simplicity. Then (5.8) becomes

$$\dot{\lambda} = -Qx - A^T\lambda \tag{5.10}$$

$$\lambda(t_f) = 0 \tag{5.11}$$

$$Ru + B^T\lambda = 0\,. \tag{5.12}$$

Let us try $\lambda = Px$, then $\dot{\lambda} = \dot{P}x + P\dot{x}$. (5.11) thus becomes

$$0 = \dot{P}x + P\dot{x} + Qx + A^T P x\,. \tag{5.13}$$

Substituting for $\dot{x}$ using (5.8) gives

$$0 = \dot{P}x + PAx + PBu + Qx + A^T P x\,. \tag{5.14}$$

Finally, we can use (5.12) to substitute for u so that we have

$$\dot{P}x + PAx - PBR^{-1}B^T P x + Qx + A^T P x\,. \tag{5.15}$$

Since this must hold for all x, it reduces to

$$0 = \dot{P} - PBR^{-1}BP + PA + A^T P + Q \tag{5.16}$$

which is a matrix Riccati equation for P. Thus P is solved by integrating *backwards* from the boundary condition $P(t_f) = 0$.

For a steady state solution, $\dot{P} = 0$ and (5.16) becomes the algebraic matrix Riccati equation

$$0 = -PBR^{-1}BP + PA + A^T P + Q \ . \tag{5.17}$$

Finally by back substituting λ into (5.12), the optimal control solution is therefore given in *feedback gain* form by

$$u = -R^{-1}B^T P x \ , \tag{5.18}$$

where the matrix $K := R^{-1}B^T P$ is called the *feedback gain matrix*.

Bibliography

[1] National infrastructure Commission, "Smart power." https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/505218/IC_Energy_Report_web.pdf, 2016. Accessed June 2019.

[2] The Association for Decentralised Energy, "Flexibility on demand - Giving customers control to secure our electricity system." https://www.theade.co.uk/assets/docs/resources/Flexibility_on_demand_full_report.pdf, 2016. Accessed June 2019.

[3] J. Drgoňa and M. Kvasnica, "Comparison of mpc strategies for building control," in *2013 International Conference on Process Control (PC)*, pp. 401–406. June, 2013.

[4] E. Žáčeková, Z. Váňa, and J. Cigler, "Towards the real-life implementation of mpc for an office building: Identification issues," *Applied Energy* **135** (2014) 53 – 62. http://www.sciencedirect.com/science/article/pii/S0306261914008071.

[5] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith, "Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis," *IEEE Transactions on Control Systems Technology* **24** no. 1, (Jan, 2016) 1–12.

[6] A. Jain, M. Behl, and R. Mangharam, "Data predictive control for building energy management," in *2017 American Control Conference (ACC)*, pp. 44–49. May, 2017.

[7] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems Magazine* **31** no. 3, (June, 2011) 52–65.

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica* **38** no. 1, (2002) 3 – 20. http://www.sciencedirect.com/science/article/pii/S0005109801001741.

[9] A. Bemporad and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *Journal of Optimization Theory and Applications* **117** no. 1, (April, 2003) 9–38.

[10] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC Press, 1984.

[11] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pp. 278–. IEEE Computer Society, Washington, DC, USA, 1995. http://dl.acm.org/citation.cfm?id=844379.844681.

[12] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics* **5** no. 4, (Dec, 1943) 115–133. https://doi.org/10.1007/BF02478259.

[13] S. C. Kleene, "Representation of events in nerve nets and finite automata," 1951.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature* **323** no. 6088, (Oct, 1986) 533–536.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation* **9** no. 8, (1997) 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[16] E. Keogh, "Chapter 36 - exact indexing of dynamic time warping," in *VLDB '02: Proceedings of the 28th International Conference on Very Large Databases*, P. A. Bernstein, Y. E. Ioannidis, R. Ramakrishnan, and D. Papadias, eds., pp. 406 – 417. Morgan Kaufmann, San Francisco, 2002. http://www.sciencedirect.com/science/article/pii/B9781558608696500433.

[17] D. Krige, "A statistical approach to some basic mine valuation problems on the witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy* **52** no. 6, (1951) 119–139. https://journals.co.za/content/saimm/52/6/AJA0038223X_4792.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the 2004 American Control Conference*, vol. 3, pp. 2214–2219 vol.3. 2004.

[20] Chuong B. Do and Honglak Lee, "Gaussian processes." http://cs229.stanford.edu/section/cs229-gaussian_processes.pdf, 2008. Accessed March 2019.