Empirical Modelling in Non-Linear Predictive Control: A Coffee Roaster Application

Cameron E. Bolt

Supervisor: Professor Philip L. de Vaal

CVD 800

June 2023

Empirical Modelling in Non-Linear Predictive Control: A Coffee Roaster Application

completed by
Cameron E. Bolt
17031096

and supervised by
Prof. Philip L. de Vaal

A dissertation submitted in partial fulfilment
of the requirements for the degree

Master of Engineering (Control Engineering)

Department of Chemical Engineering
Process Modelling and Control
Faculty of Engineering, the Built Environment and Information Technology
University of Pretoria

CVD 800

June 2023

Empirical Modelling in Non-Linear Predictive Control: A Coffee Roaster Application

Abstract

This dissertation presents the development and implementation of a model predictive control (MPC) system for a coffee roasting process, to optimise roasting quality while minimising energy consumption. The study involved analysing historical temperature profile data and roaster inputs to develop a hybrid model, combining empirical and first-principles techniques, which predicts the measured bean temperature as a function of the available roaster inputs. The combination of the first-principles model with empirical modelling techniques reduced validation data error by increasing measured temperature prediction accuracy. Subsequently, a nonlinear MPC was designed and tuned through a series of simulations, adjusting prediction and control horizons while limiting input changes relative to the real-time input value. The optimal configuration achieved a **significant reduction in the average usage of liquefied petroleum gas (LPG)** while maintaining a wide input range.

The impact of the intelligent modelling and control system on the reduction of raw material waste, the improvement of the quality of the final product, and the overall efficiency of the roasting process was evaluated, showing significant improvements in all three areas. The proposed system enables operators to perform simulations of roasts and reduce raw material wastage when developing roast profiles, providing a valuable contribution to the coffee roasting industry. Future work includes further investigation of hybrid modelling and nonlinear optimisation techniques.

Keywords: coffee roasting, model predictive control, process optimisation, hybrid modelling.

Acknowledgements

I would like to express my sincere gratitude to my parents. Their constant support and encouragement has been invaluable not only in the production of this work, but also in providing the foundation upon which this work stands.

I am forever grateful for the love and support of my fiancé, Nicole. You motivated me to focus when I could not and encouraged me to exercise balance when I could.

Special recognition is due to my mentor and professor, Prof. Philip de Vaal. Your endless wisdom, insightful discussions, and willingness to discuss all things "control" have been instrumental in this work and in my career as an engineer.

Furthermore, I would like to thank Neil Mareé from Genio Roasters for funding this work and providing the platform upon which this work is based. You taught me to love and understand coffee. Your ambition to be an industry leader is what propels Genio Roasters ahead.

> "If you can't describe what you are doing as a process, you don't know what you're doing."
> — W. Edwards Deming

Contents

List of Figures

List of Tables

Nomenclature

α	Linear regression regularisation parameter	
Δu_{lb}	Minimum change in input	
Δu_{ub}	Maximum change in input	
η	Learning rate	
λ	Latent heat of vaporisation of water	J/g
A	Arrhenius prefactor	s^{-1}
A_{bm}	Beans to metal heat transfer surface area	m^2
A_b	Total surface area of beans	m^2
A_{gb}	Air to bean heat transfer surface area	m^2
A_{gm}	Heat transfer surface area from gas to metal	m
c_b	Coffee bean specific heat capacity	J/(kgK)
c_m	Specific heat capacity of the roaster metal	J/(kgK)
c_w	The partial heat capacity of water in green coffee beans	J/(kgK)
c_{pg}	Specific heat capacity of the gas	J/(kgK)
c_s	The partial heat capacity of dry coffee	J/(kgK)
c_w	The partial heat capacity of water	J/(kgK)
D_b	Average diameter of the coffee beans	mm
E_n	Error of sample n	
F	Ratio of thermal resistances	
G_g	Hot air volumetric flow rate	m^3/s
H_a	Arrhenius activation energy	J/mol
H_e	The amount of heat produced thus far during the roasting process	J
h_e	Air to beans heat transfer coefficient	$W/(m^2K)$
h_{bm}	Bean to metal heat transfer coefficient	$W/(m^2K)$

H_{et}	Total amount of heat produced per kilogram of dry coffee beans	J/kg
h_{gm}	Gas to metal heat transfer coefficient	W/(m^{2}K)
k_3, k_4, k_5	Introduced semi-empirical constants	
K_t	Semi-empirical lag constant	
k_1, k_2	Semi-empirical constants for moisture loss model	
M	Control horizon	
M_m	Mass of the roaster metal	kg
M_{bd}	Mass of dry bean batch	kg
P	Prediction horizon	
P_{bm}	Bean to metal surface area proportion	
Q_{bm}	Heat transfer from the beans to the roaster metal	W
Q_{gb}	Heat transfer from the air to the coffee beans	W
Q_{gm}	Heat transfer from the air to the roaster metal	W
Q_{mb}	Heat transfer from the beans to the roaster metal	W
Q_r	Heat production by exothermic reaction	W
R	Universal gas constant	J/(molK)
r	Reference trajectory	
T_a	Measured coffee bean temperature	°C
T_b	Coffee bean temperature	°C
T_m	Drum metal temperature	°C
T_{gi}	Inlet temperature of the gas in the roaster	°C
T_{go}	Outlet temperature of the gas in the roaster	°C
u	Process input	
W	Vector of input feature weights	
X	Coffee bean moisture content	kg/kg
X	Matrix of input features	
Y	Matrix of output features	

List of Abbreviations

DTR	Decision Tree Regressor
HMI	Human Machine Interface
MPC	Model Predictive Control
NN	Neural Network
PI	Physics Informed
PID	Proportional Integral Derivative
RFR	Random Forest Regressor

1 Introduction

1.1 Background

Coffee roasting plays a critical role in the transformation of green coffee beans into roasted beans that are enjoyed by consumers around the world. A local South African coffee roaster manufacturer produces a range of coffee roasters for sale locally and internationally. The coffee roasting process involves a complex interplay of heat and mass transfer, which influences the flavour profile of the final product. Traditionally, coffee roasting has been considered an art, with operators relying on their experience to control the roasting process using three primary inputs: The flow rate of liquefied petroleum gas (LPG) to the burner, the rotation rate of the drum, and the speed of the blower.

However, the learning curve associated with developing an intuition for the interplay between these inputs and their effect on the beans' heating can result in significant wastage of raw materials and excessive costs. Additionally, the existing control system for the coffee roaster relies on a PID (Proportional Integral Derivative) controller, which only adjusts the LPG input based on the error between the set point temperature and the measured temperature. This approach requires a fully developed roast profile before any quality control can be achieved and does not take advantage of the other available inputs to control the roasting process. To define a roast profile, a roaster operator is required to complete multiple batches of roasting, until the desired degree of roast, and the subsequent desired flavours have been obtained. This is discussed in more detail in Section 1.3.

To address these challenges, there is a need for the development of an intelligent modelling and control system for the coffee roaster. This system should model changes in the measured bean temperature as a function of the roaster inputs, allowing operators to perform simulations of roasts and reduce raw material wastage in developing roast profiles. Furthermore, the developed model should be used in a model-based control algorithm, which can automatically vary all available inputs to reduce the error between the measured bean temperature and the set point of the bean temperature without requiring historical inputs.

1.2 Aims and Objectives

1.2.1 Research Aim

The aim of this research is to develop an intelligent modelling and control system for a local South African coffee roaster manufacturer's roasting process, to reduce raw material wastage, enhance the quality of the final product, and improve the efficiency of the roasting process.

1.2.2 Research Objectives

To achieve the research aim, the following objectives have been identified:

1. Analyse historical temperature profile data and roaster inputs to understand the relationship between input parameters and the final roasted bean quality.
2. Develop a model of the coffee roaster that can predict the measured bean temperature as a function of the available roaster inputs, using both first principles and empirical techniques.
3. Design and implement a model-based control algorithm that can automatically adjust all available roaster inputs to minimise the error between the set point for the measured bean temperature and the actual measured bean temperature.
4. Validate the developed model and control algorithm using experimental data to ensure its accuracy and effectiveness in controlling the roasting process.
5. Evaluate the impact of the proposed intelligent modelling and control system on the reduction of raw material waste, the improvement of the quality of the final product, and the overall efficiency of the roasting process.

By addressing these objectives, this research project seeks to provide a valuable contribution by optimising the roasting process using an intelligent modelling and control system.

1.3 Overview of the Coffee Roasting Process

The coffee roaster of interest consists of a double-walled ceramic coated drum, orientated horizontally on a central axis as shown in Figure 1. The drum rotates above an open-flame LPG burner. The flame heats the inlet air as well as the drum. Air is passed through the roaster by an air blower, located at the outlet of the roaster. As air passes through the drum, smoke, steam and chaff are removed from the drum and separated by a cyclone located at the outlet of the roaster (Rao, 2014). The blower is located at the top of the cyclone to separate the outer skins of the coffee beans (otherwise known as chaff) as well as additional solids. The gas separated from the emissions in the outlet is then passed through the stack and released into the environment (Bolt & Vaal, 2022).

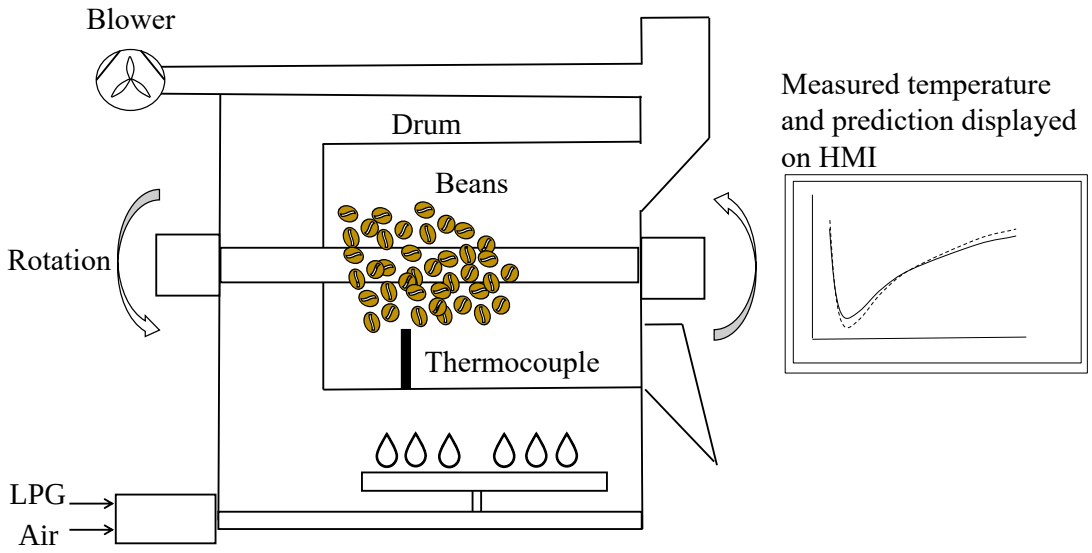


Figure 1: Section of a directly heated drum coffee roaster and temperature profile visualised on a human machine interface (HMI), adapted from Schwartzberg, 2002.

An example of a typical roaster temperature profile is shown in Figure 2. The shape of the temperature profile itself is a result of how the roaster is operated as well as where the measurement of the temperature is made. Before the beans are released from the hopper into the drum, the roaster is primed to a set point temperature. The thermocouple measuring the temperature displayed in Figure 1 (also the temperature controlled to set point during roasting) is located in the drum itself. As roasting is initiated, the temperature measured by the thermocouple is that of the heated air passing through the drum. As the beans are dropped into the drum, the cold beans make contact with the thermocouple. The dynamic lag in the measurement of the thermocouple produces the characteristic non-linear temperature profile shown in Figure 2.

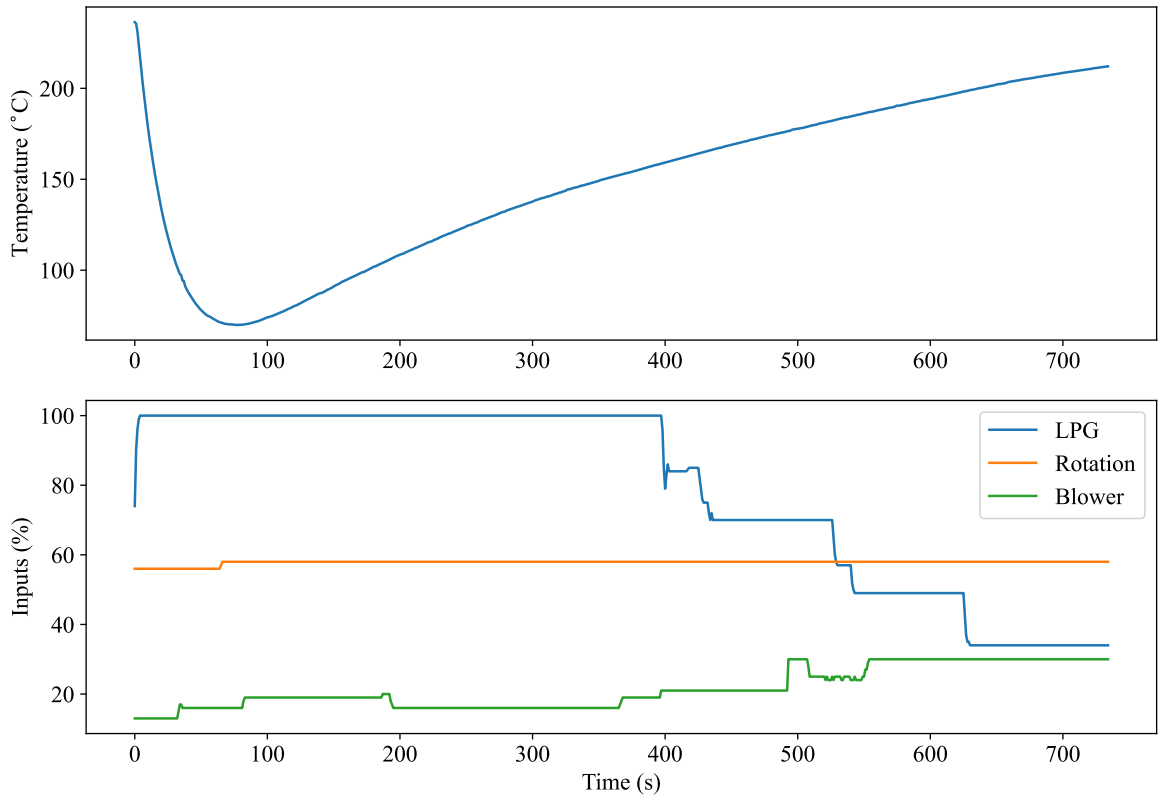


Figure 2: Typical coffee roaster temperature profile including inputs.

1.4 Structure of the dissertation

The remainder of this dissertation is organised as follows:

1. Section 2 - **Literature review**, presents a comprehensive review of the established literature regarding single input-output control systems, multivariable model-based control systems, semi-empirical coffee roaster modelling and empirical (data-driven) modelling techniques. Additional subsections have been incorporated, which discuss how to construct empirical models using a popular programming language named PYTHON.

2. Section 4 - **Data preparation and modelling methodology**, details the available data and how it was prepared for modelling. Method for the construction and optimisation of each model is provided.

3. Section 5 - **Modelling**, presents and discusses the performance of each developed model. Conclusions are provided on the best-performing models.

4. Section 6 - **Control system design**, discusses the design of the multivariable control system.

5. Section 7 - **Controller implementation**, presents the controller tuning and the performance of various controller configurations.

6. Section 8 - **Conclusions and recommendations**, concludes the dissertation and provides recommendations for future work.

2 Literature review

2.1 Classic control theory

The primary objective of a process control algorithm is to maintain a process at a desired operating condition despite external disturbances (Seborg *et al*, 2011 8:9). The purpose of the coffee control system is to maintain the measured coffee temperature as close to the desired measured temperature as possible to ensure the safe operation of the roaster as well as to maintain product quality. Traditionally, systems of varying complexity are modelled and solved as linear time-invariant systems using the Laplace transform. Understanding how the output of a modelled system behaves as a function of the system inputs naturally leads to the concept of control. Manipulation of inputs to a system to maintain process conditions is often based on the understanding gained through process modelling. Classic control theory is the theory related to the control and modelling of these linear systems through frequency- and time-based analysis.

Assuming that a non-empirical modelling approach is adopted, a typical chemical engineering process can be represented by a set of dynamic mass and energy balances. Often, supplementary relationships are incorporated between state variables through differential or algebraic equations. In instances where the relationship between the outputs and inputs exhibits non-linearity, linearisation of the set of differential equations at a specific operating point is required to apply the Laplace transform. Representing dynamic systems in the Laplace domain simplifies the resolution of the developed set of differential equations and allows the application of linear algebra laws (such as the superposition principle and matrix operations such as addition, subtraction, and multiplication). The application of linear algebra allows for the simplistic representation of multiple systems, and their interrelationships, in the form of block diagrams (Horn & SR Garcia, 2020).

The relationship between a process input, u, and output, y, in the Laplace domain, is commonly referred to as a *transfer function*. A transfer function between the inputs and outputs of the process, commonly denoted as G, is defined as the ratio of the output signal to the input signal:

$$G(s) = \frac{y(s)}{u(s)} \tag{1}$$

Similarly, a linear relationship between a measured process disturbance, d, and the output, y, can be defined as G_d. A block diagram of an arbitrary process is shown in Figure 3.

Implementing the principle of superposition, the output of the open-loop process, y, is the sum:

$$y(s) = G(s)u(s) + G_d(s)d(s) \tag{2}$$

The error, e, between measured response/output, r, and the predicted output as a function of the inputs of the process, is merely the difference between the two. Quantifying this error is an important concept for model fitting and it not to be confused with the control error as discussed in the following section.

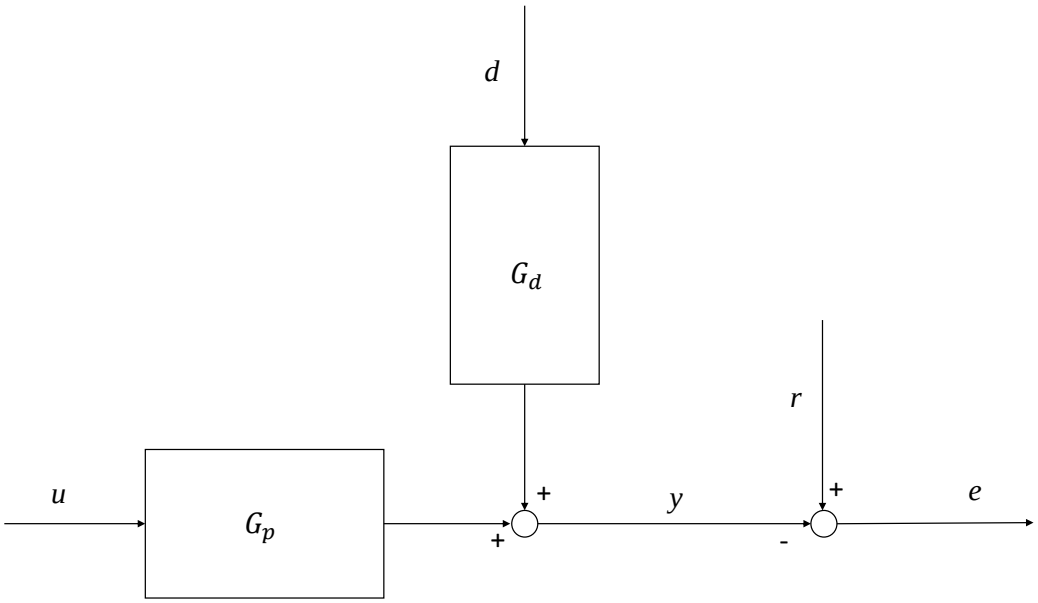


Figure 3: Block diagram of an open-loop linear system.

2.1.1 Feedback control

A block diagram of a closed-loop system is shown in Figure 4. The system is referred to as closed-loop since the residual between the reference and the measured process output is fed back through the system as an input to the controller, K. The control law defines the relationship between the measured error, e, and the process input required to drive the measured error to zero. One such control law is the proportional-integral-derivative (PID) law. The realisable transfer function relating the measured error and output signal of the PID controller is defined as follows:

$$\frac{P(s)}{E(s)} = K_c \left[1 + \frac{1}{\tau_I s} + \frac{\tau_D s}{\alpha \tau_D s + 1}\right] \tag{3}$$

where K_c, τ_I, τ_D and α are parameters. Typically, the derivative filter constant, α, is set to be constant (Seborg *et al*, 2011: 129). As one would deduce based on the name, the control law is calculated based on a proportional multiplier of the measured error, the integral of the measured error, and the derivative of the measured error (Seborg *et al*, 2011 126:130).

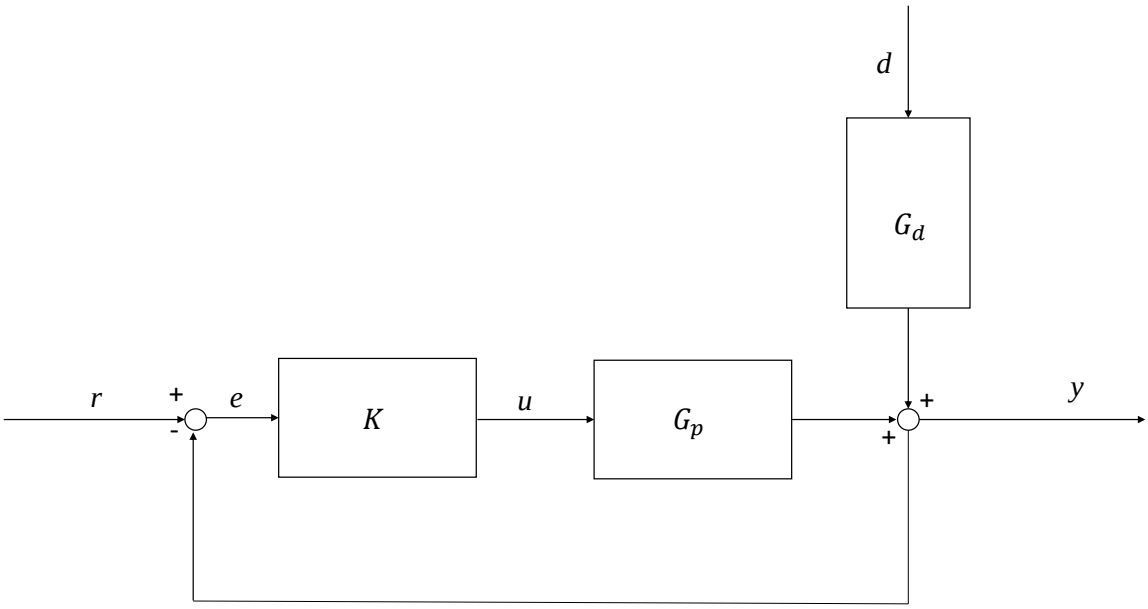


Figure 4: Block diagram of a closed-loop feedback system.

Feedback control forms the basis of control strategies in processing plants worldwide, in most cases, in the form of PID control (Borase *et al*, 2021). Feedback control has the advantage of being able to implement control action as soon as an error is measured, whether this measured error is due to a set point change or a disturbance. PID control serves as a one-size-fits-all solution, since minimal to no modelling of the relationship between process inputs and outputs is required to implement a PID controller (Seborg *et al*, 2011: 271). The widespread application of PID control is due to the simplicity and ease of implementation of the algorithm, as the controller operator only needs to adjust three parameters to improve controller performance (Borase *et al*, 2021).

2.1.2 Feedforward control

Feedforward control is not used in the current coffee roaster control strategy; however, it is worth noting that feedforward control is most often used in conjunction with feedback control. Feedforward control uses a process model between a measured disturbance and the controlled variable (output) to preemptively adjust process inputs so that the effect of the measured disturbance on the controlled variable is mitigated (Marlin, 2000: 483). This combined feedforward and feedback structure addresses inherent disadvantages in both control strategies. The incorporation of feedforward control addresses the reactionary behaviour of feedback control. Feedback control can only take action once deviation (error) of the controlled variable from the reference has taken place. Feedback control addresses disturbance rejection of non-measurable disturbances and its performance does not hinge on model accuracy, as in the case of feedforward control. A block diagram of a combined feedback and feedforward control structure is shown in Figure 5 (Marlin, 2000; Seborg *et al*, 2011: 489,278).

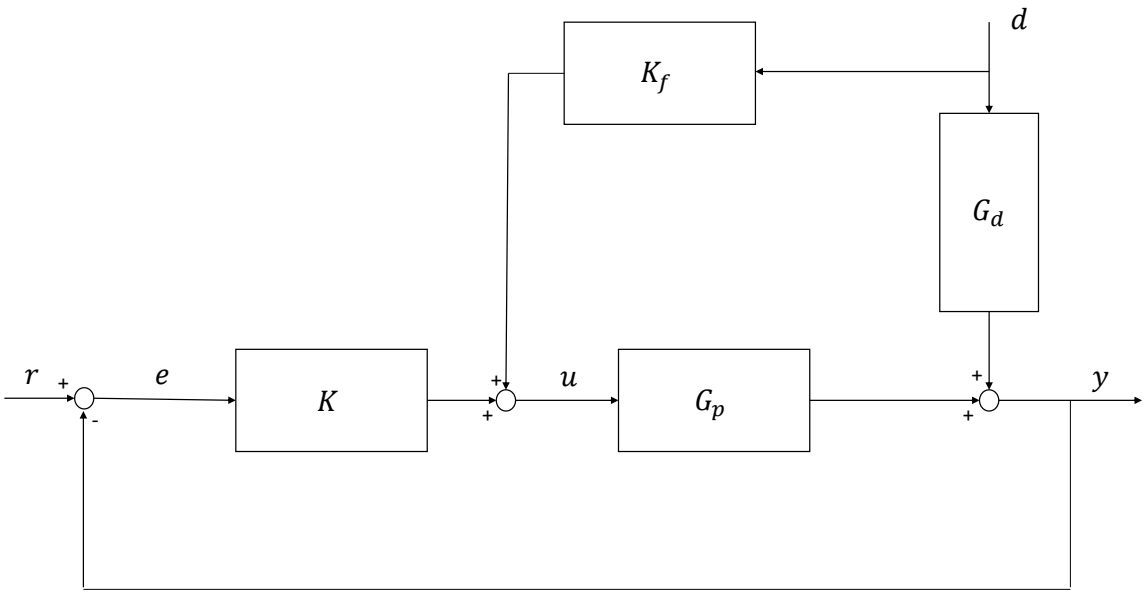


Figure 5: Combined feedback and feedforward control structure.

Figure 5 illustrates the incorporation of the feedforward controller, K_f into the feedback structure. Importantly, note that within the feedforward structure, the controlled variable, y, is not measured. The ideal feedforward controller K_f is found by setting the numerator of the closed-loop transfer function y(s)/d(s) equal to zero (assuming perfect mitigation of the measured disturbance). The feedforward controller K_f is a function of the disturbance transfer function, G_d and the process transfer function, G_p (Marlin,

2000; Seborg *et al*, 2011: 485,278):

$$K_f = -\frac{G_d}{G_p} \tag{4}$$

The ideal feedforward controller transfer function illustrates that the controller action, and hence overall control performance, is determined by the accuracy of the process models. The influence of model accuracy will become important when considering model-based control techniques such as internal model control (IMC) and model predictive control.

2.2 Multivariable systems

In real-world applications of control systems, it is very rarely the case that a single input single output (SISO) system exists in isolation from other inputs and outputs. The concept of SISO modelling is extended to multiple input, multiple output (MIMO) systems through linear algebra (matrix representation). A vector of n outputs is related to a vector of m inputs through a $n \times m$ matrix of transfer functions:

$$\begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} = \begin{bmatrix} G_{1,1} & \cdots & G_{1,m} \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ G_{n,1} & \cdots & G_{n,m} \end{bmatrix} \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_m \end{bmatrix} \tag{5}$$

Where the matrix element $G_{n,m}$ is the transfer function relating the effects of input m on output n. Single input-output control is extended to the multivariable case with ease, the controller matrix K is a $n \times m$ matrix. If multi-loop control is deployed, only one element in each row of the controller matrix K contains a transfer function (such as a PID control law relating input m to output n).

As the complexity of systems increases, multivariate control techniques become essential to maintain the desired performance. Some advanced control strategies, such as model predictive control (MPC), specifically cater to MIMO systems and are capable of handling constraints and interactions between variables. These advanced control methods, along with other model-based control techniques, play a crucial role in modern industrial control applications.

2.2.1 Model-based control

The PID control algorithm exemplifies a decentralised control approach, as it uses a single measured output to determine the appropriate input value to maintain desired conditions. In a multivariable context, this is known as decentralised or multi-loop control, where multiple single-loop feedback controllers pair one output with one input. However, this method has inherent drawbacks if there are interactions between loops in the multi-loop structure. Centralised control, on the other hand, uses process input and output measurements to determine suitable system inputs (Marlin, 2000: 727).

The model-based control design method, internal model control (IMC), implements a process model in the controller design and operation of the feedback algorithm (CE Garcia & Morari, 1982) as shown in Figure 6. The process input, u, determined by the controller, K^* is passed as an input to both the real process, G, and the process model. The difference between the measured and predicted output eventually passes as an input to the controller K^*. The development of the control law is a two-step process whereby the process model, G_p, is factored into invertible and non-invertible transfer functions. The control law is found by inverting the invertible portion and applying a low-pass filter to the inverted portion of the process transfer function (Seborg *et al*, 2011: 210).

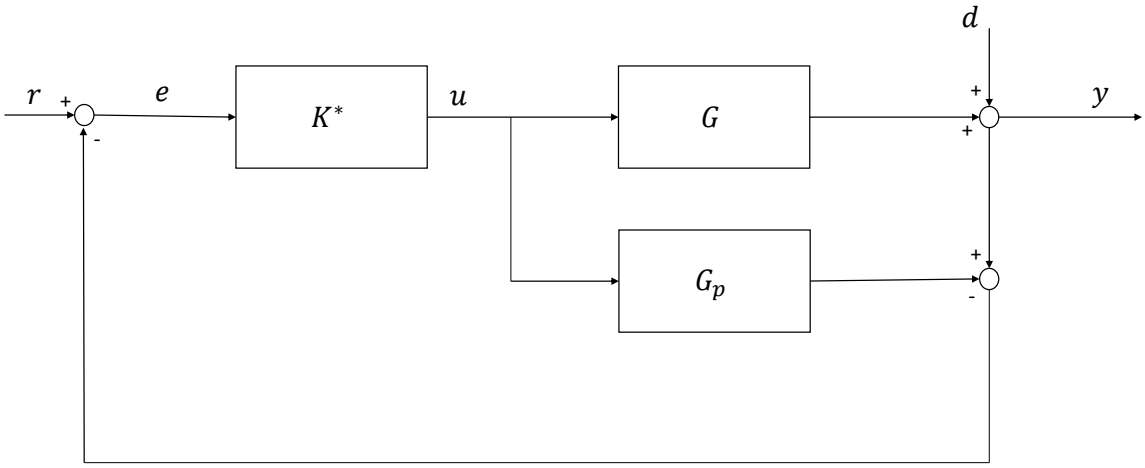


Figure 6: Internal model control block diagram.

The accuracy of the model will have a marked influence on the IMC performance. This was the case when considering the design and performance of the feedforward controller.

The concepts displayed through IMC lay the foundation for advanced control techniques which utilise a process model in the development of the control law.

2.3 The current control algorithm

The current roaster employs a PID feedback control algorithm to minimise the error between the measured bean temperature and its set point, which is typically a time-dependent function in batch systems. Moreover, the measured temperature has been demonstrated to be a non-linear function of inputs. Given that classic control theory relies on linear time-invariant models for the development of control algorithms, a single set of PID tuning parameters would not provide satisfactory control performance.

To address this, an enhanced technique called adaptive PID control was implemented by manufacturers (Pomerleau, Desbiens, Hodouin, *et al*, 1996), which implements a suitable rule to adjust the PID tuning parameters based on the roaster states, such as the measured bean temperature. However, the adaptive tuning method retains the single-loop strategy, achieving the desired measured bean temperature by exclusively manipulating the LPG flow rate.

2.4 Advanced control theory

Advanced process control (APC) typically builds on the base layer of control, which consists mainly of feedback PID control and feedforward control (Seborg, 1987). The base layer aims to achieve process stability and safety through quality and inventory control. Advanced process control drives the process as close as possible to an operating region where the maximum profit may be achieved. An example of this would be driving the throughput of a unit operation as close as possible to stability and safety limits to maximise production whilst ensuring product quality. Advanced process control typically makes use of computationally expensive optimisation and prediction techniques to realise these benefits. The advanced process control algorithm considered and implemented in this work is model predictive control (MPC).

2.4.1 Model predictive control

Model predictive control first made an appearance in the late 1970s, coined as dynamic matrix control and model predictive heuristic control (Cutler & Ramaker, 1980; Richalet

et al, 1978). Model predictive control is a control algorithm that makes use of a process model to calculate the required inputs to force the controlled variables to track a desired reference. The required inputs are calculated as the solution to an optimisation problem. The optimisation problem is posed as a minimisation of a user-defined objective function, which typically includes the sum of the squared error between the reference trajectory of the controlled variables and the predicted trajectory of the controlled variables. The predicted trajectory is obtained using the process model (Camacho & Alba, 2013: 1).

Similarly to classic control theory, much of the initial research surrounding model predictive control assumed the use of a linear time-invariant model of the process either in the continuous or discrete form. This work focuses on applying model predictive control to a black-box empirical model of the coffee roaster. Consider an arbitrary non-linear function which models the relationship between the process inputs and outputs:

$$y = f(x, u, t) \tag{6}$$

The MPC problem is typically formulated as an optimisation problem of the following form:

$$\begin{aligned} \min_{u} \quad & \sum_{j=0}^{P} ||r(t+j) - f(x+j, u+j, t+j)||_n + \sum_{j=1}^{M} \Delta u(t+j-1) \\ \text{s.t.} \quad & \Delta u_{lb} \leq \Delta u \leq \Delta u_{ub} \end{aligned} \tag{7}$$

This can be linguistically described as the following: the solution to the MPC optimisation problem is the set of inputs $u(t+j)$ that minimises the chosen objective function. The objective function consists of two summations, the first being the summation of the vector norm n of the error between the user-defined reference trajectory, $r(t+j)$ and the trajectory predicted by a set of inputs, $f(t+j)$, over a specified period P where $j = 1...P$ (Camacho & Alba, 2013: 2). This period P, is typically named the prediction horizon and is chosen as a tuning parameter. The second portion of the objective function is a penalisation of large input moves and is a summation of the change in input moves over a specified period M. This period, M, is typically known as the control horizon. The control horizon is usually chosen to be less than the prediction horizon to reduce the degrees of freedom during optimisation (Camacho & Alba, 2013: 21). For a time after $j = M$, the control inputs will assume to be unchanged until $j = P$. This formulation can be explained with the aid of a diagram.

Consider Figure 7 shown below, which shows a MPC strategy during a coffee roast. Time has evolved up to some arbitrary point in the roast at which the model predictive controller is solving for the set of inputs which will minimise the user-defined objective function. The reference measured temperature is a profile saved from a previous roast and the predicted temperature is a function of the inputs, LPG flow to the burner, the rotation speed, and the blower speed. The set of inputs which minimises the objective function as shown in Equation 7 is solved through an iterative optimisation routine since the process model is a black box non-linear model. The optimisation routine yields a solution, a set of inputs, in the case of the coffee roaster, a 3 × P-sized matrix of inputs. Only the first control signal for each input i.e. the first element in each row of the solution matrix, is passed to the process as an input.
This means that the optimisation is carried out at each sampling instant and ensures that the process can respond to unmeasured process disturbances.

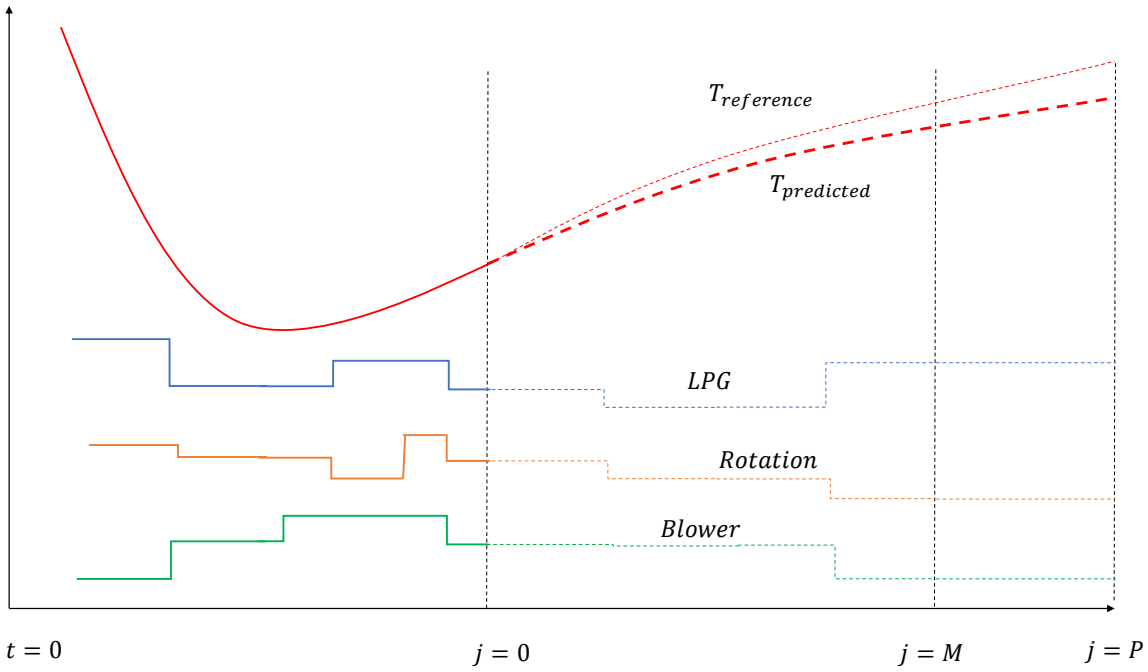


Figure 7: A representation of a model predictive controller acting on the coffee roaster system.

Adjustments to the objective function can be made, such as the introduction of a set point error weight or a move suppression factor. Consider, for example, that should one multiply the n vector norm of the set point error with some constant that inflation of the objective function will occur (compared to the standard case as shown in Equation 7). This will force the optimisation to prioritise set point tracking over move suppression. Similarly, in the case of move suppression factors, the second summation in Equation 7 can be multiplied with a constant to penalise large input changes (for every input); alternatively, a weight matrix can be introduced which penalises moves of a particular

input more than others. In this way, the use of one input may be prioritised over another.

Traditional model predictive control theory makes model predictions using a linear model of the system and represents the objective function using a two-vector norm. The solution to this problem is a well-formulated convex optimisation problem and can be solved using Quadratic Programming (QP) and variations thereof. A large emphasis will be placed on developing a non-linear model of the roaster, and the implementation of a non-linear model into the objective function pivots the optimisation problem from a convex quadratic programme to a nonconvex problem. Naturally, nonconvex optimisation problems are more difficult to solve, and obtaining the global minimum of the objective function in less than the controller sampling time is not necessarily guaranteed. Sequential Quadratic Programming (SQP) can be employed to solve the optimisation problem and approximates the optimisation problem as a quadratic approximation at each iteration. Non-gradient-based black-box optimisation techniques, such as Bayesian optimisation or swarm optimisation, may be considered.

There are advantages that model predictive control provides to the coffee roaster system. It handles the multivariable case, since the optimisation routine will solve for the set of all three inputs, which yields a minimum in the objective function. It directly handles constraints, which means that should it be desirable, the current roaster control strategy can be incorporated into the MPC design. The current control strategy only adjusted the LPG flow rate for set point tracking, this adjustment was done within specific bounds relative to the historical profile input which is used as a reference. In addition, an empirical model of the roaster can be used in the MPC algorithm.

2.5 Bayes' theorem

Bayes' theorem has found widespread application in the field of applied mathematics and is fundamental to the field of artificial intelligence. Bayesian optimisation is the technique used to optimise all the empirical models developed in this thesis; therefore, Bayes' theorem and its application to an optimisation routine will be discussed in this section. To state it simply, Bayes' theorem is a quantification of updating the probability (commonly referred to as belief) of a particular hypothesis, denoted as H below, given new information or "evidence", denoted as E below.

$$P(H|E) = \frac{P(H) \times P(E|H)}{P(E)} \tag{8}$$

The left-hand side of the equation, $P(H|E)$, stands for the *a posteriori* probability, de-

noting the conditional probability of a hypothesis, H, which is valid upon the availability of evidence, E. The term $P(H)$, located in the numerator on the equation's right-hand side, illustrates the *a priori* probability, which is the primary belief or probability of the hypothesis prior to the introduction of supporting evidence. In the denominator, $P(E)$ conveys the marginal probability of evidence according to the frequentist interpretation. That is, the probability of an event corresponds to the proportion of times the event would occur in an infinitely large number of similar trials (Hajek *et al*, 2019). Concurrently, $P(E|H)$, another term in the numerator, is referred to as likelihood, representing the conditional probability of the evidence given the truth of the hypothesis (Simon, 2006: 73).

2.5.1 Bayesian optimisation

Bayes' theorem provides a framework for the optimisation of black-box objective functions such as in the case of hyperparameter tuning of machine learning-type models. It is especially useful for tuning parameters of models that are computationally expensive to train. Suppose that a problem arises in which a neural network needs to be trained and optimised on a large data set. The performance of this arbitrary neural network is assessed on a data set using the mean squared error between the predictions and the true values. Therefore, an objective function can be constructed that assesses this performance as a function of the model parameters. The parameters of the model i.e. the number of hidden layers and neurons per layer of the network are bounded between a high and low value. Determining which set of parameters produces a minimum in the mean squared error is a cumbersome task considering that the training of the model takes in excess of an hour to train; in other words, it takes an extended period of time to quantify the performance of the model for a single set of parameters. Since the probability distribution between the input parameters and the output of the objective function is not known, traditional gradient descent methods can be applied to find the minimum of the objective function. Testing permutations of each parameter set using a grid search method is not viable from a time- and computational-cost perspective. Bayesian optimisation provides a solution to this problem (Frazier, 2018b).

It is assumed that the relationship between the inputs and outputs of the objective function can be modelled using a set of Gaussian distribution functions. Random samples are drawn from the set of parameters and evaluated on the objective function. A Gaussian regressor is trained on the input-output set to approximate the relationship between the model parameters and the output of the objective function. The Gaussian regressor is commonly referred to as the surrogate model, as it provides a surrogate for the true objective function that requires optimisation (Greenhill *et al*, 2020). The surrogate model

provides a probability distribution for p(mean squared error|hyperparameter), in other words, it provides a quantification of the belief in the hypothesis given the evidence. An illustration of a surrogate model developed based on sampling from an arbitrary objective function is shown in Figure 8. Note that in regions where there is frequent sampling of the objective function, the 95% confidence interval is small - the belief in the distribution of the objective function in this region is high. To update the posterior distribution of the surrogate model, additional points need to be sampled.

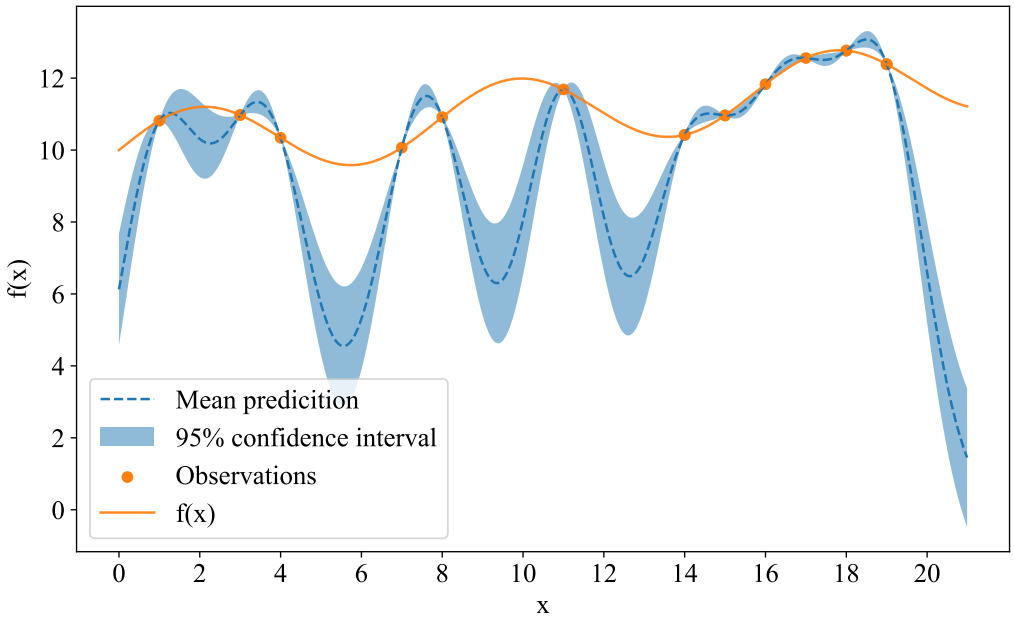


Figure 8: Surrogate model of an arbitrary objective function, $f(x)$.

New sampling points of the objective function are provided by an acquisition function. The acquisition function is chosen by the user and involves a trade-off between local and global optimisation. This translates to searching for an optimum in regions where the belief in the structure of the objective function is high (small confidence interval) versus searching for an optimum in regions where the belief in the structure is small (large confidence interval). There are several acquisition functions that are commonly used, an example of which is the expected improvement function (Torun *et al*, 2018; Greenhill *et al*, 2020). Once a new sampling point has been acquired, the posterior probability distribution can be renewed in light of the new evidence. This process is followed iteratively until a minimum is found in the surrogate model.

2.5.2 Mathematical formulation

In summary, Bayesian optimisation is an iterative method for finding the minimum in expensive to evaluate functions, f. Bayesian optimisation is composed of two primary components, namely, the surrogate model and the acquisition function (Frazier, 2018a). The algorithmic process of Bayesian optimisation can be graphically summarised as shown in Figure 9 where the iteration, n, is incremented until a particular tolerance on the objective function is achieved or the number of iterations has reached some predefined maximum number of evaluations, N.

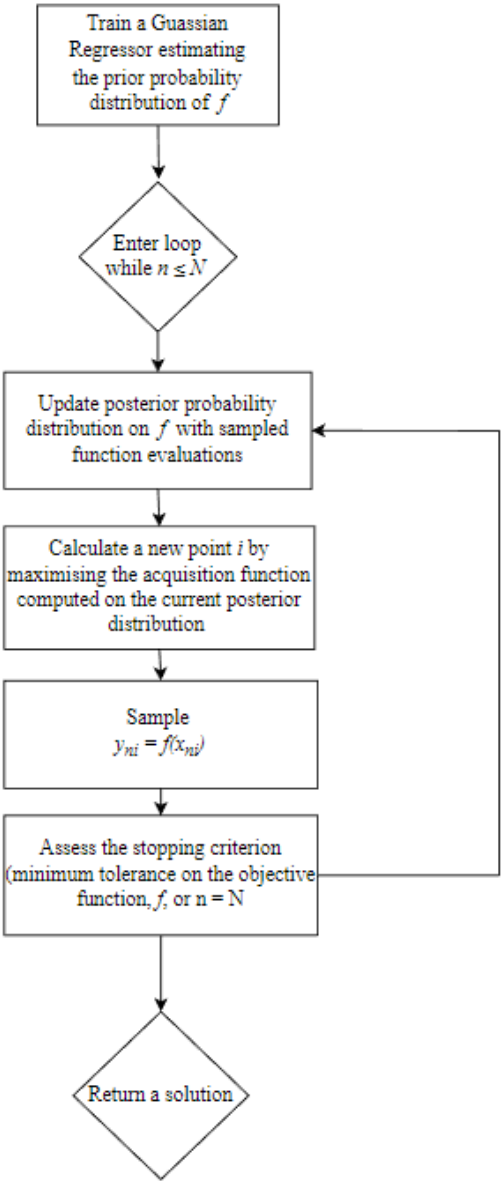


Figure 9: Flow diagram illustrating the Bayesian optimisation algorithm

2.6 Schwartzberg model

The following semi-empirical model, proposed by Di Palma *et al*, 2021, offers a scalable drum roaster model for coffee beans, building on Schwartzberg's (Schwartzberg, 2002 863:882) work by adjusting for roaster mass and volume. A comprehensive derivation of the process equations can be found in Schwartzberg's chapter (Schwartzberg, 2002 863:882), while this section provides a concise overview of the key equations.

The order of the equations discussed below is not necessarily the order in which the equations were originally derived. The rate of change in the temperature of the beans, $\dot{T}_b$, is given by Equation 9. It is simply an energy balance that takes into account heat transfer to the beans (as well as assuming negligible heat losses). The energy balance also provides for the exothermic release of energy during the heating of the beans.

$$\dot{T}_b = \frac{Q_{gb} - Q_{gm} + Q_{mb} + M_{bd}(Q_r + \lambda \dot{X})}{M_{bd}(1 + X)c_b} \tag{9}$$

Equation 9 describes the rate of change in the temperature of coffee beans ($\dot{T}_b$), it is assumed that there is no variation in temperature within the mass of the bean (it is a lumped parameter). This equation is a dynamic energy balance which considers various heat transfer mechanisms, as well as the exothermic heat generated during the roasting process. Specifically, Q_{gb} denotes the heat transfer from the air to the coffee beans, Q_{gm} is the heat transfer from the air to the roaster metal, Q_{mb} is the heat transfer from the beans to the roaster metal, and Q_r denotes the heat generated by the exothermic reaction during roasting. M_{bd} represents the mass of the dry coffee beans and c_b is the specific heat capacity of the coffee beans. X is the moisture content of the coffee beans and $\dot{X}$ is the rate of change of this moisture content. The latent heat from the vaporisation of water is denoted by λ. Therefore, the energy required to vaporise the bound water within the bean is taken into account, and there is less energy available to roast the beans. The equation assumes negligible heat losses from the system, which means that all the heat that is transferred to the beans is used to increase their temperature or evaporate water.

$$\dot{T_m} = \frac{Q_{gm} - Q_{mb}}{M_m c_m} \tag{10}$$

Equation 10 accounts for the rate of change in roaster metal temperature, $\dot{T}_m$, assuming negligible heat losses to surroundings. The total mass of the drum metal is indicated by M_m and the specific heat capacity of the metal is indicated by c_m.

$$\dot{X} = -\frac{k_1 X^2}{D_b^2} \exp\left(-\frac{k_2}{T_b + 273.15}\right) \tag{11}$$

Various assumptions were made in the development of Equation 11 which models the rate of loss of moisture from coffee beans during roast. The differential equation agrees well with the experimental data if the coffee beans' average diameter is 6 mm. The average diameter of the bean, D_b, is measured in mm and the temperature of the bean, T_b is measured in °C for the purposes of Equation 11. Schwartzberg (Schwartzberg, 2002) assumed an initial moisture content of approximately 11% for the simulations performed. It should be noted that Equation 11 contains semi-empirical constants which have been fitted to experimental data on moisture loss.

$$\dot{H}_e = A\frac{H_{et} - H_e}{H_{et}} \exp\left(-\frac{H_a}{R(T_b + 273.15)}\right) \tag{12}$$

Equation 12 models the amount of exothermic heat released during roasting by the so-called "roasting reactions". There are several reactions that take place during the roasting process, such as the Maillard reaction (Yang *et al*, 2016), otherwise known as the caramelising reaction. This equation assumes that the rate of heat generation is proportional to the reactant concentration (as a whole) and that this concentration is proportional to the term $\frac{H_{et}-H_e}{H_{et}}$. Where H_{et} is the total amount of heat produced per kilogramme of dry coffee beans during the roasting process and H_e is the amount of heat produced so far.

$$T_{gi} - T_{go} = \left(T_{gi} - \frac{T_b + FT_m}{1 + F}\right)\left(1 - \exp\left(-\frac{h_e A_{gb}(1 + F)}{G_g c_{pg}}\right)\right) \tag{13}$$

Equation 13 represents the variation in the temperature of the gas as it travels through the roaster, which is dependent on various factors, such as the temperature of the beans T_b, the temperature of the roaster drum metal T_m, and the roaster scaling parameter F. The temperature of the gas inlet T_{gi} is typically determined by the amount of combusted liquefied petroleum gas (LPG).

In the first part of Equation 13, the term in parentheses represents the temperature gradient in the roaster. In the second part, the exponential term accounts for the temperature drop in the gas as it moves through the roaster, considering the heat transfer coefficient h_e, the heat transfer surface area from air to beans A_{gb}, the hot air volumetric flow rate G_g, and the specific heat capacity of the gas c_{pg}.

$$F = \frac{h_{gm} A_{gm}}{h_e A_{gb}} \tag{14}$$

The term F in Equation 13, known as the roaster scaling parameter, is defined in the subsequent equation. This parameter is a ratio of the heat transfer rate (determined by the heat transfer coefficients h_{gm} and h_e, and the surface areas A_{gm} and A_{gb}) from the gas to the metal and from the air to the beans.

$$h_e = 0.49 - 0.443e^{-0.206X} \tag{15}$$

Schwartzberg (Schwartzberg, 2002) assumes a constant gas-to-bean heat transfer coefficient, while Di Palma *et al*, 2021 considers it a function of the moisture content of the bean:

$$c_b = \frac{(c_s + c_w X)}{1 + X} \tag{16}$$

The overall bean heat capacity varies as a function of bean temperature and moisture content.

$$c_s = 1.099 + 0.007T_b \tag{17}$$

Here, the partial heat capacity of the beans, c_s, is described as the following function of temperature. The partial heat capacity of the moisture in green coffee beans, c_w, is assumed to not vary as a function of temperature and is specified in Table 1 below.

In summary, this semi-empirical model provides a scalable approach to drum coffee roaster modelling, accounting for key factors such as heat transfer, exothermic reactions, and moisture loss during the roasting process. The equations presented here offer a concise overview of the key relationships that govern roasting dynamics, building on the work of previous researchers.

2.7 Optimised adapted Schwartzberg model

2.7.1 Empirical parameters

To develop a first-principles roaster model, the adapted model (Di Palma *et al*, 2021) will be adjusted by introducing empirical constants that can be fitted to the available roaster data. These constants are introduced into the roaster state variables, which are going to vary largely based on the flow regime. An example of this is the heat transfer coefficient, h_e, between the beans and the hot gas. Another example of a variable that is believed to vary is the bean heat capacity (as a function of species as well as temperature). The so-called "fudging factors" technically known as empirical constants will be fitted to data from a 6 kg, 15 kg, and 30 kg roaster. The adapted Schwartzberg model has been adjusted by the introduction of the following parameters (the empirical constants introduced are shown in red):

$$\dot{T}_b = \frac{Q_{gb} - Q_{gm} + Q_{bm} + M_{bd}(Q_r + \lambda \dot{X})}{M_{bd}(1+X)c_b} \tag{18}$$

$$\dot{T}_m = \frac{Q_{gm} - Q_{bm}}{M_m c_m} \tag{19}$$

$$\dot{X} = -k_5 \frac{k_1}{D_b^2} \exp\left(-\frac{k_2}{T_b + 273.15}\right) \tag{20}$$

$$\dot{H}_e = A \frac{H_{et} - H_e}{H_{et}} \exp\left(-\frac{H_a}{R(T_b + 273.15)}\right) \tag{21}$$

$$h_e = k_3(0.49 - 0.443e^{-0.206X}) \tag{22}$$

$$c_b = \frac{k_4(c_s + c_w X)}{1+X} \tag{23}$$

$$T_{go} = \left(T_{gi} - \frac{T_b + FT_m}{1+F}\right)\left(1 - \exp\left(-\frac{h_e A_{gb}(1+F)}{G_g c_{pg}}\right)\right) \tag{24}$$

In addition to the parameters introduced, the heat transfer coefficient between the gas and the metal (h_{gm}), the heat transfer coefficient between the beans and the metal (h_{bm}) and the proportion of the surface area of the beans in contact with the metal (P_{bm}) will be optimised.

2.7.2 The optimisation problem

Optimisation of any model requires the specification of an objective function, sometimes referred to as a loss/cost function. The goal of the optimisation problem will be to find the set of parameters, specifically $[k_3, k_4, k_5, h_{gm}, h_{bm}, P_{bm}]$, that yields an extremum in the chosen cost function. Typically, for regression problems, the objective function is chosen to be the sum of the absolute error between the predicted bean temperature and the true bean temperature for the entire data set. Alternatively, the sum of the squared error could have been considered. The cost function J^p of batch p (where batch p is a subset of the data set and represents a single roast of length n^p) is calculated for a specific set of chosen parameters and represents the sum of the absolute error for batch p. The optimum set of parameters minimises the cost function, J, the sum of J^p for all p in N roasts.

$$X = [k_3, k_4, k_5, h_{gm}, h_{bm}, P_{bm}] \tag{25}$$

$$J^p(X) = \sum_{i=1}^{n^p} |T^p_{a,i}(X) - T^p_{measured,i}| \tag{26}$$

The optimisation problem can be summarised as follows. The bounds on the optimisation parameters were chosen based on trial and error except in the case of P_{bm}, where a logical upper bound on the proportion of total bean surface area in contact with the drum wall is at most 0.5. Different bounds were chosen for each roaster size. The *python* library *scipy.optimize.minimize* is used to minimise the cost function

$$\begin{aligned} \min_X \quad & \sum_{i=1}^{N} J^p(X) \\ \text{s.t.} \quad & x_{i,lb} \leq x_i \leq x_{i,ub} \quad \forall \quad x_i \in X \end{aligned}$$

2.7.3 Schwartzberg parameter specification

The following parameter specifications were used as supplied by (Di Palma *et al*, 2021; Schwartzberg, 2002).

Table 1: Schwartzberg parameter specifications (Di Palma *et al*, 2021; Schwartzberg, 2002).

Symbol	Description	Value	Units
c_s	The partial heat capacity of dry coffee	1.0	kJ/kg
c_w	The partial heat capacity of water	5.0	kJ/kg
k_1	Semi-empirical constant	4.32×10^{-9}	
k_2	Semi-empirical constant	9889	
K_t	Semi-empirical lag constant	0.01	
M_m	Mass of roaster metal	2000	kg
λ	Latent heat of vaporisation of water	2790	kJ/kg
A	Arrhenius equation prefactor	116200	kJ/kg
D_b	Coffee bean diameter	7.65	mm
H_a/R	Activation energy - gas law constant ratio	5500	K
H_{et}	Total heat of reaction per kg of coffee	232	kJ/kg

2.8 Linear regression

2.8.1 Basis functions

The requirement is to predict the measured bean temperature as a function of the input features. This is a supervised learning regression problem. The final model should predict a continuous numerical value (the measured bean temperature) using the input features. The measured output temperature is used in conjunction with the known input features to construct the predictive model (supervised learning). Linear regression, in its simplest form, is merely a model which predicts the target output variable using a linear combination of the input features. This can limit the model's ability to capture complex non-linear behaviour. Non-linear behaviour can be captured by the linear combination of non-linear functions known as basis functions. This is precisely how one can develop a higher-order model (such as a parabolic function for a single input variable) predicting non-linear output behaviour by linear regression.

The simplest form of a linear model for a scalar input variable x is the following (Zeger *et al*, 2000):

$$y = ax + b \tag{27}$$

This can be easily extended to the multivariate case, including multiple input-output features. Since the final objective is to predict only the measured bean temperature, y will be treated as a scalar output. However, the following can easily be extended to the multivariate case of having multiple predicted outputs, $\mathbf{Y}$.

$$y(\mathrm{X}, \mathrm{W}) = w_0 x_0 + w_1 x_1 + ... + w_D x_D \tag{28}$$

where X is a vector of input features and W is a vector of the model weights/parameters. The parameters are also referred to as weights since they "weight" the contribution of each linear input feature. Note that the weight w_0 is used to offset the prediction due to any constant offset present in the relationship between the inputs and the outputs. In the simple case of a scalar input variable, this is commonly referred to as the "intercept". This would imply that the value of x_0 will always be unity.

At this point, the basis function is introduced. In the simplest case, the basis function $\Phi = \mathrm{X}$. Hence, the output bean temperature is a linear combination of non-linear basis functions of the input features:

$$y(\mathrm{X}, \mathrm{W}) = \mathrm{W}^{\mathrm{T}} \Phi(\mathrm{X}) \tag{29}$$

Examples of basis functions include Gaussian basis functions, Fourier basis functions, sigmoidal basis functions, and polynomial basis functions (Anderson & Rubin, 1949; Yao *et al*, 2004).

Characteristically for all optimisation problems, the goal is to find the set of W that minimises some cost / objective function. In this case, the goal is to minimise the least-squares cost:

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y(\mathrm{X}^{(i)}, \mathrm{W}) - y^{(i)})^2 \tag{30}$$

This is simply a sum of squared errors between the predicted values of y and y from the 1^{st} instance/sample in the data set to the N^{th} instance/sample where X^i is the i^{th} sample and y^{i} is the true value of the output at the i^{th} sample. This summation can be represented in matrix notation where the matrix X is a $\mathrm{N} \times \mathrm{M}$ matrix (M is the number of parameters), the least-squares cost function can then be represented as the following (Hastie, Tibshirani & Friedman, 2001: 44):

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} (\mathrm{XW} - y)^{\mathrm{T}} (\mathrm{XW} - y)$$

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} ((\mathrm{XW})^{\mathrm{T}} - y^{\mathrm{T}})(\mathrm{XW} - y)$$

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}}((\mathrm{XW})^{\mathrm{T}}\mathrm{XW} - (\mathrm{XW})^{\mathrm{T}}y - y^{\mathrm{T}}(\mathrm{XW}) + y^{\mathrm{T}}y)$$

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}}(\mathrm{W}^{\mathrm{T}}\mathrm{X}^{\mathrm{T}}\mathrm{XW} - 2(\mathrm{XW})^{\mathrm{T}}y + y^{\mathrm{T}}y)$$

Since the objective is to minimise the cost function as a function of the weights, the derivative of the cost function with respect to W is taken and set equivalent to 0 in order to obtain the optimum solution.

$$\frac{\partial J}{\partial \mathrm{W}} = 2\mathrm{X}^{\mathrm{T}}\mathrm{XW} - 2\mathrm{X}^{\mathrm{T}}y = 0$$

If we assume that $\mathrm{X}^{\mathrm{T}}\mathrm{X}$ is invertible, W can be solved as

$$\mathrm{W} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}y \tag{31}$$

2.8.2 Gradient descent

As the size of the training data set increases in conjunction with the number of parameters weighting the basis functions so the dimensions of the matrix, Φ, increases accordingly. In high dimensional settings, it becomes computationally inefficient to calculate the numerical values of the weights using the closed-form normal equation, as shown in Equation 31. This can be explained by the fact that while the computation covariance matrix $(\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}$ scales quadratically with the number of features $O(f^2)$ and linearly with the number of samples $O(n)$, the inversion of the matrix itself exhibits a cubic scaling with respect to the number of features $O(f^3)$ (Hastie, Tibshirani, Friedman, *et al*, 2009). Gradient descent is an alternative solution to this problem. It is an iterative optimisation routine that provides approximately the same solution as the closed-form solution (within some tolerance). The gradient descent algorithm calculates the gradient of the loss/cost function (squared error) at every iteration. The loss function or error can be summed over a small batch of data, i.e. 10 samples, and the weights can be updated using this sum in the following way:

$$\mathrm{W}^{(i+1)} = \mathrm{W}^{i} - \eta\vec{\boldsymbol{\nabla}}\sum_{n=1}^{k} E_n \tag{32}$$

The parameter η is known as the learning rate and is a very important hyperparameter which will determine the success or failure of the optimisation routine. As seen above, the learning parameter weights the gradient of the error sum and therefore influences the rate at which the parameters of the W vector change. Should the learning rate be too small, the optimisation routine will take very long to converge, and should the learning rate be too large, the optimisation routine will diverge from the solution. Gradient descent is often preferred over the closed form solution in instances when the feature dimensionality is large or the number of samples causes computational challenges such as memory exceptions (Géron, 2019: 117).

2.8.3 Regularised linear models: Ridge regression

Regularised linear models are linear models that have an additional term added to the standard mean squared error term used as the objective function (the same objective function used for the closed-form solution as well as for gradient descent). The term added distinguishes which regularised type of linear regression is being used. The reason why the standard objective function is regularised is to shrink the parameters toward zero (Hastie, Tibshirani & Friedman, 2001: 59). This is used to prevent overfitting of the developed linear model. The addition of a parameter weighted (α) sum of the weights squared (the square of the l_2 norm of the weight vector, as shown below) provides the Ridge regression model:

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y(\mathrm{X}^{(i)}, \mathrm{W}) - y^{(i)})^2 + \frac{\alpha}{2} \sum_{i=1}^{D} \mathrm{W}_i^2 \tag{33}$$

Increasing the magnitude of α will drive the calculated parameters toward zero.

2.8.4 Regularised linear models: Lasso regression

Lasso regression is a regularised linear regression. Similarly to Ridge regression, it uses the addition of a term to the linear regression objective function. However, in this case, the ℓ_1 norm of the weight vector (as shown below) is added such that the objective function now yields the following (Hastie, Tibshirani & Friedman, 2001: 64):

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y(\mathrm{X}^{(i)}, \mathrm{W}) - y^{(i)})^2 + \alpha \sum_{i=1}^{D} |\mathrm{W}_i| \tag{34}$$

2.8.5 Regularised linear models: Elastic Net regression

Elastic Net regression is a linear combination of Ridge regression and Lasso regression (with regard to the adapted objective function). This is achieved by adding a ratio factor, r, to the norms ℓ_1 and ℓ_2 of the weight vector in the following way:

$$J(\mathrm{W}) = \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y(\mathrm{X}^{(i)}, \mathrm{W}) - y^{(i)})^2 + r\alpha \sum_{i=1}^{D} |\mathrm{W}_i| + (1-r)\frac{\alpha}{2} \sum_{i=1}^{D} \mathrm{W}_i^2 \tag{35}$$

It can be noted that the ratio factor, r, is the proportion of the regularisation term that is constructed using the norm ℓ_1 of the model weights. The remainder of the regularisation term is constructed with the ℓ_2 norm of the model weights.

2.8.6 The bias-variance trade-off

The general goal of the modelling process is to develop a model of the system (the coffee roaster) which generalises well to a wide range of operating conditions. This means that the performance of the developed model should not be hinged on small regions of operation (or on regions representative of the training data set). Regression by least squares leads to an overfitted model if the number of basis functions and the complexity of the basis functions are excessive (Bishop, 2006: 147). This is a model that is termed to have a high variance. However, the complexity of the model (and the number of basis functions) should not be limited to the extent that complex behaviour (such as the non-linear temperature measurement in the roaster) cannot be accurately predicted. This is a model termed as having high bias. The term "generalisation error" is a fundamental concept in machine learning that is constructed of three independent sources. Bias error, variance error, and irreducible error. Bias represents the error that exists between the average prediction over all data sets and the target value (desired). Variance is the measure of how the predictions for an individual data set vary around average prediction. Hence, reducing the variance error favours the reduction in error on a single data set. The variance measures the extent to which the prediction function is sensitive to the data set under consideration. The irreducible error represents the minimum error that can be achieved on a data set due to noise in the measurement.

Practically, this means that if a high-variance model is developed, this will result in a small error in the training data set and a large error in validation data sets. This is because the model is developed to fit the training data set well but cannot generalise to new inputs (beyond that present in the training set). Developing a high bias model is

a model that makes strong assumptions about the relationship between the inputs and the desired output (Géron, 2019: 134). Varying the structure of the developed model by tweaking constant parameters varies the bias and variance of the model. The goal is to find the minimum in the generalised error, which is the sum of the bias and the variance error.

2.9 Building linear models in Python

2.9.1 Ordinary linear regression

Much of the fundamental theory of linear regression has been covered; however, one of the core objectives of this dissertation is to emphasise practical ways in which the above methods can be applied. The SCIKIT-LEARN PYTHON libraries provide an easy-to-understand method of directly building linear models using different basis functions and implementing regularisation in the objective function. The documentation supporting the construction of linear models is extensive (Scikit-learn Linear Models, 2022).

In order to build an ordinary linear model in PYTHON, the following script could be used. The inputs and output of the coffee roaster have been used as an example here.

```
import sklearn.linear_model

X = numpy.c_[LPG ,rotation ,blower] #a matrix of your input values
Y = numpy.c_[Temperature]#a matrix of your output/target values

linear_reg_model = linear_model.LinearRegression()

linear_reg_model.fit(X,y)
Y_test = linear_reg_model.predict(X_test) #test the predicted outputs
                                    against the true values for a
                                    validation/test data set

linear_reg_model.intercept_ ,linear_reg_model.coef_ #intercept and
                                    coefficients
```

The normal equation could be used to obtain the same result. Remember that the basis function (ϕ_0) of the zeroth weight (w_0) is unity; therefore, an additional column of ones has to be appended to the input matrix.

```
X = numpy.c_[LPG ,rotation ,blower]
X_b = numpy.c_[numpy.ones((767 ,1)),X]
theta_best = numpy.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
#theta_best will return the same result as the linear_reg_model.coef_
                                        call
```

2.9.2 Regularised linear regression

Implementing regularised linear models is just as simple. Ridge regression and Lasso regression are implemented in exactly the same way and require the same specified parameters.

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1,solver = 'cholesky')
ridge_reg.fit(X,y)
```

The alpha parameter specified is exactly the same parameter as that present in Equation 33. As the magnitude of the alpha (α) parameter increases, the shrinkage of the parameter weights increases driving the model prediction towards the output mean. The *cholesky* specification of the Ridge function is a solution technique to the normal equation. This specific method is also used in the SCIPY.LINALG.SOLVE function. It uses matrix decomposition to efficiently handle large matrix operations (Géron, 2019: 137).

Note that the Lasso regression function can be implemented in exactly the same way:

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=1)
lasso_reg.fit(X,y)
```

The Elastic Net Regression model can be built in a similar manner. Note that the ELASTICNET function requires a specification for the ℓ_1 ratio. This is the ratio that specifies the ratio of the ℓ_1 norm, i.e., the ratio of regularisation that is Lasso regularisation.

```
from sklearn.linear_model import ElasticNet
EN_reg = ElasticNet(alpha = 1, l1_ratio = 0.5)
EN_reg.fit(X,y)
```

2.9.3 Gradient descent optimisers

The stochastic gradient descent algorithm is one that makes use of Equation 32. The gradient descent algorithm is particularly useful when the data set of samples or measured values is very large, i.e. in the magnitude of hundreds of thousands. It becomes more efficient to consider stochastic gradient descent or batch gradient descent optimisation

algorithms. PYTHON provides a simple and convenient way of implementing the stochastic gradient descent algorithm. The SGDREGRESSOR class can be used to perform ordinary linear regression or regularised linear regression (by specifying the penalty as ℓ_1 or ℓ_2).

The SGDREGRESSOR class is sensitive to the scaling of features. Therefore, it is recommended that the input features be scaled using MINMAXSCALER or STANDARDSCALER. For the case of the coffee roaster data, this is not essential since all of the inputs are scaled between 0 and 100% when recorded. This is, however, good practice and is especially important for the case where the input and output features operate in drastically different scales and ranges.

```
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing import StandardScaler

scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

sgd_reg = SGDRegressor() #the penalty can be specified here as l1 or l2
                                    to perform regularised linear
                                    regression.
sgd_reg.fit(X_train)
sgd_reg.predict(X_test)
```

2.9.4 Determining generalisation error

Determining the bias and variance of a model as a function of specific parameters is an attractive approach to developing a model which generalises well to a large set of operating conditions. This could simply be accomplished by varying model parameters and subsequently calculating the model variance and bias for all data sets used, i.e. the training and validation data sets. The parameter combination that yields a minimum generalisation error will then be chosen as the optimal configuration. This is a simple approach and will work well for model types which require only one or two parameters to be specified. Complicated models such as neural networks will require more advanced optimisation techniques. The MLXTEND library can be used to automatically calculate the bias-variance decomposition.

```
from mlxtend.evaluate import bias_variance_decomp

MSE, bias, variance = bias_variance_decomp(model, X_train, Y_train,
                                    X_test, Y_test, loss = 'mse')
```

2.10 Decision trees

A decision tree is a non-parametric model that exhibits a significant amount of variance, primarily due to its non-parametric nature. In contrast to parametric models, which make specific assumptions about the underlying structure and distribution of the data and involve a finite set of parameters, non-parametric models like decision trees do not make strong assumptions and have a more flexible functional form. Decision trees can be intuitively understood as a series of "if" statements that predict the output value of a set of input features by applying a set of rules ("if" statements) to these features. The method used to develop the set of rules ("if" statements), known as the training algorithm, is what makes the decision tree training distinctive.

Typically, decision trees are used for classification problems, as they are better suited to problems where the input features are characterised by discrete values or attributes rather than continuous variables (Mitchell, 1997: 54). This is somewhat reminiscent of a fuzzy-logic structure, where, for instance, the temperature can be described by a discrete set of linguistic attributes such as hot, mild-hot, mild, mild-cold, and cold (Mitchell, 1997: 54). However, decision trees can also be applied to regression problems. To better understand how a decision tree is trained, it is helpful to analyse how a decision tree makes a prediction.

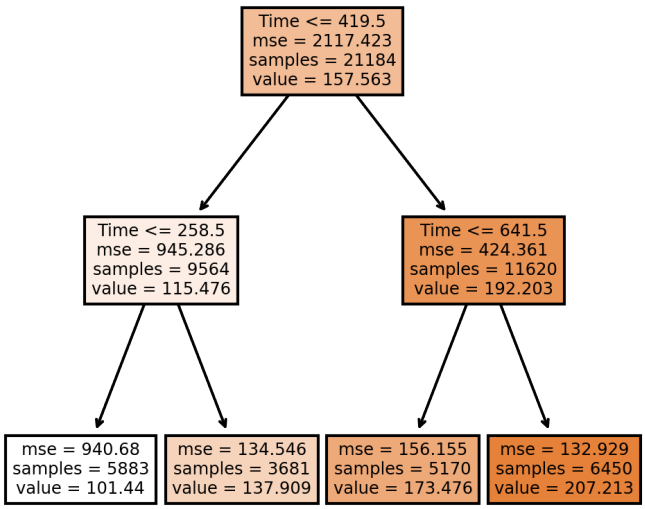


Figure 10: Visualisation of a decision tree for predicting measured bean temperature.

The name "decision tree" becomes clear as each node in the tree represents a decision based on a test applied to the input features. The final node where no further splitting occurs which outputs the outcome of the preceding decisions is referred to as the "leaf node" or simply the "leaf" (Géron, 2019 175:177). In this example, the tree has been limited to a maximum depth of 2, eventually yielding 4 leaf nodes.

Assuming that using time as a predictive input feature is valid and the time is less than 200 seconds, one can track the decision path in the tree shown in Figure 10, applying the tests at each node with a time input of 200 seconds. The first decision node tests whether the time is less than or equal to 419.5 seconds, since 200 seconds is less than the threshold value the subsequent test node to be applied is the one on the left. Similarly, a threshold value of 258.5 seconds as applied until the leaf node is reached. The leaf node predicts a bean temperature of approximately 101.44 °C. This prediction is based on the average bean temperature of 5,883 samples, which yields a mean squared error of 940.68 °C^2. In this case, the selected loss type is the mean-squared error (ℓ_2 loss).

The question remains: How does the decision tree learning algorithm determine which input feature to use in the test at each node, and how does it quantify the value of the selected input feature for each test/rule?

The SCIKIT-LEARN.TREE library employs the CART algorithm (Classification and Regression Tree) to identify the appropriate inequality for the "if" statement test. The CART algorithm uses an optimisation routine that minimises the objective function shown in Equation 36. This algorithm divides the training set into two subsets in order to minimise the selected loss (such as ℓ_1 or ℓ_2 loss) (Géron, 2019: 184). The algorithm continues to split subsequent nodes until a split that reduces the selected loss type beyond a certain tolerance cannot be found. As a result, decision trees are non-parametric models and inherently have high variance.

$$J_{\text{decision tree}}(k, t_k) = \frac{n_{\text{subset 1}}}{n_{\text{total}}}\text{Loss}_{\text{subset 1}}(\text{k}, \text{t}_\text{k}) + \frac{n_{\text{subset 2}}}{n_{\text{total}}}\text{Loss}_{\text{subset 2}}(\text{k}, \text{t}_\text{k}) \tag{36}$$

The loss is calculated using the mean of the training target values (measured bean temperature). This implies that the loss at each node can be calculated as follows (Breiman, 2001; Breiman *et al*, 1984; Scikit-learn Decision Tree Regressor, 2022):

$$\hat{y}_{\text{subset}} = \frac{1}{n_{\text{subset}}} \sum_{i \in \text{ subset}} y^{(i)} \tag{37}$$

$$\ell_1 : \text{Loss}_{\text{subset}} = \sum_{i \in \text{ subset}} |\hat{y}_{\text{subset}} - y^{(i)}| \tag{38}$$

$$\ell_2 : \text{Loss}_{\text{subset}} = \sum_{i \in \text{subset}} (\hat{y}_{\text{subset}} - y^{(i)})^2 \tag{39}$$

As mentioned earlier, the algorithm continues to split subsequent nodes until a split that reduces the selected loss type beyond a certain tolerance cannot be found. To avoid overfitting the training dataset, the number of splits can be limited. This process, often referred to as the decision tree "pruning", completes the organic tree analogy. This is exactly how the decision tree shown in Figure 10 was limited in depth. Limiting the number of splits directly impacts the bias-variance decomposition.

The process of "pruning" is more complex than simply limiting the depth of the tree. Optimal regularisation involves assessing the performance of the developed decision tree on a subset of the input data, termed the validation data set. "Pruning" a particular node involves removing the subtree rooted from the node in question (Mitchell, 1997: 70). This process should only be followed if the performance on the developed model (assessed using an ℓ_1 or ℓ_2 loss between the target/true value and the predicted value) is at the very least equivalent to the original tree structure. This process can be completed iteratively through an optimisation routine, such as Bayesian optimisation.

2.11 Building decision tree models in Python

Building a decision tree regression model in PYTHON is rather simple and limited specification of hyperparameters is required. The diagram of the decision tree can easily be plotted and saved using MATPLOTLIB. The depth of the decision tree is specified using the MAX_DEPTH hyperparameter (Scikit-learn Decision Tree Regressor, 2022). The above decision tree, shown in Figure 10 was created using the following script.

```
from sklearn.tree import DecisionTreeRegressor
import matplolib.pyplot as plt
%matplotlib inline

X = numpy.c_[times_array,LPGs_array,rotations_array,blowers_array]
Y = numpy.array(Temps_array)

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

DT_reg = DecisionTreeRegressor(max_depth=2)
DT_reg.fit(X_train,Y_train)

fn=['Time','LPG','Rotation','Blower']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
sklearn.tree.plot_tree(DT_reg,feature_names=fn,filled=True)

plt.savefig('DT')
```

2.12 Ensemble methods

Ensembling is a concept that many people may be familiar with, albeit unknowingly. For example, consider the game show *Who Wants to Be a Millionaire?*. When faced with a challenging question, a contestant has lifelines they can use, such as *fifty-fifty*, *phone a friend*, *ask the audience*, and *ask one person from the audience*. Assuming that the contestant has no prior knowledge of their friend's expertise on the question, the *ask the audience* option often provides a reliable means of obtaining the correct answer by aggregating the audience's responses. The contestant is likely to choose the answer with the majority vote. This thought experiment captures the essence of ensemble learning, where an ensemble method consists of a group of predictors. In the case of regression, an ensemble method might contain a regularised linear regressor, a decision tree, and a neural network. The average predicted value is used as the final prediction (Géron, 2019: 190).

Although this may sound like a good idea in theory, it raises an important question: Would each predictor within the ensemble method be trained using exactly the same training data set? Would the optimisation/solution routines for the different predictors differ? The concepts of *bagging* and *pasting* address this question by training each predictor using a subset randomly sampled from the training data set. The difference between *bagging* and *pasting* lies in the sampling method. In the *bagging* method, the random subset of training data is sampled with replacement, which means that the sampled training instance is returned to the available sampling population after the random sample is drawn (Breiman, 1996: 123-140). In the *pasting* method, the random sample is sampled without replacement (Géron, 2019: 192).

An ensemble of decision trees, trained using the *bagging* and *pasting* methods, forms a *random forest*. Randomness is introduced into the *random forest* not only through the random sampling of subsets of the training data (for the purpose of training each *tree* within the *forest*), but also in the feature selection at each node of the tree (Breiman, 2001). Instead of selecting the feature that minimises the selected loss (ℓ_1 or ℓ_2) at each node, a feature is chosen from a random subset of input features, which minimises the selected loss. Additional randomness can be introduced by randomising the critical value of the "if" statement that would normally be calculated by optimising the objective function in Equation 36.

2.12.1 Boosting algorithms

The adaptive boosting algorithm (commonly denoted as AdaBoost) is a sequential learning ensemble method first introduced by Yoav Freund and Robert Schapire (Freund & Schapire, 1997). The algorithm was initially developed for application in classification problems with a specific focus on gambling. Note that the fundamental difference between boosting algorithms such as AdaBoost or Gradient Boosting (to be discussed) and other ensemble methods like Random Forests (typically trained by bagging) is the way in which the training of the ensemble method is performed. Boosting algorithms train each predictor in the ensemble sequentially, as opposed to parallel training of each of the predictors in a bagging method. The sequential training algorithm of each of the predictors within the ensemble is what separates boosting algorithms such as AdaBoost and Gradient Boosting.

The concept of boosting was extended to regression problems for the first time by Harris Drucker (Drucker, 1997). The basic premise of boosting remains the same:

- A base predictor (such as a Decision Tree) is chosen as the first predictor in the ensemble and trained on a random subset of the training set. Each instance within the training set is sampled from a distribution with replacement and has an equal chance of being sampled.
- The performance of the predictor is evaluated on the training set by assessing the error on each of the training instances. The sampling probability of the training instances most in error is increased, so that these instances are more likely to be sampled from the distribution in the next predictor in the ensemble method (Freund & Schapire, 1997).
- This forces the next predictor to train on difficult-to-predict training examples (Drucker, 1997). The AdaBoost algorithm weights each of the predictors within the ensemble based on the prediction error on the distribution and not on the test set. This means that the final prediction is a weighted sum of the individual predictors.
- Predictors are added to the ensemble sequentially until a specified number of predictors has been obtained or until a minimum error on the test set is achieved. The variance of the ensemble method can be reduced by limiting the number of predictors or using a regularised base predictor (Géron, 2019: 203).

Gradient Boosting, similar to the AdaBoost algorithm, is a sequential learning algorithm that builds an ensemble method. Originally introduced by Breiman (Breiman, 1997) shortly after the introduction of the AdaBoost algorithm (Freund & Schapire, 1997), Breiman built on the idea of using the performance (or rather failures) of one predictor (as well as previous predictors) within the ensemble to influence the training of the next predictor within the ensemble. The gradient boost algorithm uses a loss calculation (such as the ℓ_1 or ℓ_2 loss) and sequentially adds base predictors that reduce the overall loss of the ensemble method. This is accomplished in much the same way as the gradient descent concept; however, instead of tweaking individual parameters of a single model, an entire model is parameterised and added to the ensemble to reduce the overall loss, thereby reducing the residual loss of all the preceding base predictors in the ensemble method. Géron, 2019 illustrates this concept by training sequential trees, in which each additive tree is trained on the residual error of the preceding tree. The final prediction is the sum of all individual trees within the ensemble method (as opposed to the weighted sum used in the AdaBoost algorithm).

The base estimator used is primarily a decision tree which can be built using exactly the same inputs as discussed in Section 2.11.

In conclusion, boosting algorithms like AdaBoost and Gradient Boosting offer valuable tools in the realm of machine learning, providing an effective means of combining multiple weak predictors to achieve a stronger overall prediction. By carefully selecting the base estimator and managing the potential risk of overfitting, these algorithms can be successfully applied to a wide range of tasks in both the classification and regression domains.

2.13 Building ensemble methods in Python

At this point it is clear that ensemble methods have large potential; however, if built incorrectly, they can lead to models which overfit the training set and result in a high variance. The SCIKIT-LEARN library offers many functions for building ensemble methods with ease; this will be discussed in the content to follow.

An ensemble method trained using the *bagging* concept can easily be built using the BAGGINGREGRESSOR method. The BASE_ESTIMATOR can be specified, as well as the number of estimators within the ensemble through the specification of the hyperparameter, N_ESTIMATORS. If the hyperparameter BOOTSTRAP is set to true, samples are drawn with replacement, and conversely, if set to False, *pasting* is performed (Scikit-learn, 2022c).

```
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor

X = numpy.c_[times_array ,LPGs_array ,rotations_array ,blowers_array]
Y = numpy.array(Temps_array)

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

Decision_tree_ensemble = BaggingRegressor(base_estimator =
                                    DecisionTreeRegressor(max_depth = 2
                                    , max_features = 'auto'),
                                    n_estimators = 20, bootstrap = True
                                    ).fit(X_train,Y_train)
```

The BAGGINGREGRESSOR method provides the opportunity to build an ensemble using different base predictors (referred to as base estimators in PYTHON). Recall that an ensemble of decision trees trained using *bootstrapping* is referred to as a Random Forest. A Random Forest can be built using the BAGGINGREGRESSOR method specifying the base estimator as a Decision Tree (take note that the MAX_FEATURES hyperparameter needs to be specified to 'auto' so that a random subset of the input features is used when determining the Decision Tree node threshold instead of all the input features), alternatively, a Random Forest can be built directly in the following way:

```
from sklearn.ensemble import RandomForestRegressor

X = numpy.c_[times_array ,LPGs_array ,rotations_array ,blowers_array]
Y = numpy.array(Temps_array)
```

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

rf_regressor = BaggingRegressor(n_estimators = 20, bootstrap = True,
                                max_depth = 2).fit(X_train,Y_train)
```

An ensemble method can also be developed using sequential boosting algorithms. The AdaBoost algorithm can be implemented in PYTHON and requires many of the hyper-parameters required in BAGGINGREGRESSOR, and, most importantly, the loss function must be specified in (Scikit-learn, 2022a).

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor()
X = numpy.c_[times_array ,LPGs_array ,rotations_array ,blowers_array]
Y = numpy.array(Temps_array)
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

ada_regressor = AdaBoostRegressor(base_estimator =
                                  DecisionTreeRegressor(max_depth = 2
                                  , max_features = 'auto'),
                                  n_estimators = 20, loss = 'square')
                                  .fit(X_train,Y_train)
```

A gradient-boosted ensemble for the purposes of regression is developed in a similar manner, however, the base predictor cannot be specified, it is assumed that the base predictor is a Decision Tree. Note that the MAX_FEATURES hyperparameter must be specified so that a random subset of the input features is used when determining the Decision Tree node threshold instead of all the input features. Specifying MAX_FEATURES to AUTO will allow all input features to be included in the optimisation routine; a float (fraction) can also be specified. This means that if only two input features are selected at a time of three, then a float of 2/3 should be specified (Scikit-learn, 2022b).

```
from sklearn.ensemble import GradientBoostingRegressor

X = numpy.c_[times_array ,LPGs_array ,rotations_array ,blowers_array]
Y = numpy.array(Temps_array)
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)
```

```
GB_regressor = GradientBoostingRegressor(loss = 'squared_error',
                                         learning_rate 0 .1, n_estimators =
                                         100, max_depth = 2, max_features =
                                         'auto').fit(X_train,Y_train)
```

Clearly, the choice of the number of predictors is an important decision that can affect the bias and variance decomposition of the final model. An ensemble method with an excessive number of predictors will overfit the training data and have a large variance error. There is a number of ways to treat this problem. The first is to use a concept known as early stopping. This means that the ensemble method is evaluated sequentially and training is halted when the validation error reaches a minimum (this is the error on the test set separated during the train-test split). The STAGED_PREDICT method allows monitoring of the validation error at each addition of a predictor, therefore the optimal amount of predictors can be determined:

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
import numpy

X = numpy.c_[times_array ,LPGs_array ,rotations_array ,blowers_array]
Y = numpy.array(Temps_array)
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

GB_regressor = GradientBoostingRegressor(loss = 'squared_error',
                                         learning_rate = 0.1, n_estimators =
                                          100, max_depth = 2, max_features =
                                          'auto').fit(X_train,Y_train)

val_errors = [mean_absolute_error(Y_test, Y_pred) for Y_pred in
                                         GB_regressor.staged_predict(X_test)
                                         ]

best_estimators = numpy.argmin(val_errors)+1
```

2.14 Neural networks

Since the introduction of the neural network in 1943 (McCulloch & Pitts, 1943) as a mathematical model of the neural pathways within the brain, the application of this empirical modelling technique to a multitude of scenarios has seen many successes as well as failures. Advances in computer processing power, partially attributed to the thriving computer gaming industry and gamers' insatiable appetite for better video rendering (Manning-Smith & Mustain, 2019), along with access to "Big Data", have led to a resurgence in the application and performance of neural networks. The application of neural networks has reached impressive heights in recent years and has been instrumental in the development of technology such as autonomous vehicles and fraud detection algorithms (Chatterjee, 2022).

The ease of application of empirical modelling techniques, including neural networks, is in part why these techniques have gained popularity within the control engineering community. The "model" of a process is fundamental to all model-based control techniques. In most cases, model development from first principles is difficult and time-consuming. Industrial processes often have access to vast databases covering a multitude of operating regions as well as modern cloud-based computing environments. All the tools required to build intelligent models are available. This raises the question: Why are engineers working in industrial applications apprehensive of machine learning models? Perhaps this apprehension is rooted in the desire of engineers to understand physical phenomena. Often, black-box models such as neural networks or ensemble methods have difficult-to-interpret parameters that rarely hold any physical significance towards the modelling problem (Dobbelaere *et al*, 2021). Analysing the thousands of weights associated with a neural network is undoubtedly a strenuous task. Moreover, applications of machine learning models within the chemical engineering community have found many unsuccessful applications due to the development of high-variance models and poor pre-processing of training data, both closely related to the lack of education in the field (Dobbelaere *et al*, 2021). The danger associated with the ease at which machine learning models can be developed and implemented is that the engineer developing the relevant machine learning models need not understand the inner workings of the empirical model being developed. This results in a "plug and play" scenario where engineers simply develop models with the click of a button.

The following sections discussing neural networks will aim to demystify the empirical regression method and provide practical methods for reducing variance.

2.14.1 Mathematics of the neural network

As previously mentioned, the neural network was inspired by the structure of the human brain (Kellher, 2019), thus the structure of the neural network is appropriately composed of layers of interconnected neurons or nodes (McCulloch & Pitts, 1943). In a fully connected neural network, each neuron in each layer of the network is connected to every neuron in the preceding and following layers. The number of layers in the network and the neuron content of each layer are modelling parameters. The transfer of information from one neuron to the next is weighted, with the value of this weight assigned and calculated through a training process, which will be explained in the following paragraphs. Consider a single neuron, named the perceptron (Rosenblatt, 1958), with inputs chosen to be relevant to the coffee roaster.

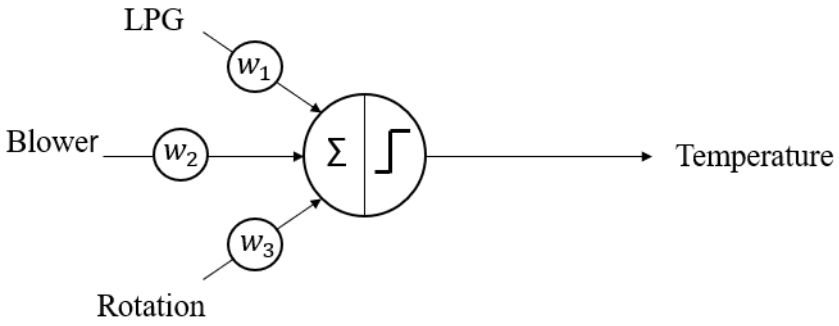


Figure 11: The perceptron model (Rosenblatt, 1958) - a roaster perceptron.

Figure 11 illustrates a perceptron with three inputs, namely the LPG feed input, the blower speed input, and the rotation speed input. These inputs have been scaled as a percentage of their range. The objective is to model the effects of changes in these inputs on the target output, namely, the measured bean temperature. Two mathematical operations are performed on the inputs to map the respective inputs to the target output. Firstly, the weighted sum of the inputs and a neuron bias are taken, and then a transform function, commonly referred to as the activation function, is applied to the weighted sum. The choice of activation function depends on whether the modelling problem is a regression or a classification problem. The summation of the inputs can be represented mathematically using matrix algebra if one assumes that the bias weight is equivalent to one (1):

$$\text{sum} = [w_1, w_2, w_3, w_4] \begin{bmatrix} \text{LPG} \\ \text{Blower} \\ \text{Rotation} \\ \text{Bias} \end{bmatrix} \tag{40}$$

Subsequently, the activation of the summed weight produces the neuron output:

$$\text{neuron output} = \text{activation}(\text{sum}) \tag{41}$$

The operation of a single neuron is rather simple. Thus far, it is clear that several choices define the neuron model. Besides the correct choice of inputs and target outputs (an initial step in the hierarchical process of modelling), the selection of input weights and activation function has a quantifiable effect on the neuron output. Considering the effect of these weights and activation functions is crucial for training and optimising neural networks, which can consist of thousands or even millions of neurons.

Given a particular set of inputs, weights, and activation function, a specific output can be calculated and compared to the target value of the output. In the case of the roaster, this would involve evaluating the difference between the true measured temperature and the predicted temperature. Evaluating the difference takes the form of error measurements common in the control engineering field, such as mean squared/absolute error (MSE, MAE). Using the chain rule, a fundamental principle in the field of calculus, the partial derivative of the calculated error with respect to a particular weight can be calculated. The gradient descent optimisation technique can then be used to drive each weight to its respective optimum value to achieve a minimum in error between the predicted and target output value.

Consider the single neuron (perceptron) model in Figure 11 and the implementation of the derivative chain rule to the input of LPG and the associated weight, w_1.

$$\frac{\partial \text{error}}{\partial w_1} = \frac{\partial \text{error}}{\partial \text{activation}} \times \frac{\partial \text{activation}}{\partial \text{sum}} \times \frac{\partial \text{sum}}{\partial w_1} \tag{42}$$

This implies that the activation function needs to be a differentiable function with respect to its input, the sum of the weighted inputs. Once the partial derivative of the calculated error with respect to the weight (w_1) has been calculated, the weight can be updated for the next training iteration ($k+1$) using the generic gradient descent algorithm:

$$w^{i+1} = w^i - \eta \times \frac{\partial \text{error}}{\partial w} \tag{43}$$

It has not been stated mathematically above, but the bias of each neuron is updated in the same manner, first by determining the partial derivative of the error function with respect to the particular bias and then by updating the bias using gradient descent. Once again, the choice of the learning rate (η) will have a marked influence on the success of the training optimisation process. An aggressive learning rate will result in divergence, and a conservative rate will result in slow training. Although the above example represents the simplest case of a neural network, built-in PYTHON packages such as KERAS apply the chain rule (referred to as backpropagation) and gradient descent optimisation to every weight within a network consisting of many interconnected neurons, and therefore many chains. Much of this computational load is reduced by using linear algebra.

It is worth noting the behaviour of this algorithm at this point. The algorithm builds the neural network (given a set of neurons and layers) in such a way that the calculated error function is minimised. This type of approach is often referred to as greedy. A neural network with an excessive number of neurons, trained with excessive iterations (referred to as epochs), can lead to a high-variance model that will perform poorly on test data and exceptionally well on the training set of data. Techniques such as *early stopping*, *dropout*, and ℓ_1 *and* ℓ_2 *regularisation* have been developed as a means of reducing the variance of the model (overfitting).

2.14.2 Activation functions

It has been established that the activation function is a chosen function applied to the sum of the weighted inputs at each neuron with a neural network. It is the choice of a non-linear activation function that will allow a neural network to model non-linear input-output mappings (Kellher, 2019: 74). This makes sense since if a linear activation function were chosen (or no activation function at all) the mapping between the output and input would be restricted to some weighted linear combination of the inputs and subsequently the individual neuron outputs. It has been established that it is a requirement for the activation function to be differentiable to make use of the backpropagation, gradient-descent training algorithm. Therefore, in theory, there are multiple choices available when choosing an activation function.

The activation functions available for implementation in KERAS are the most popular (Keras, 2022):

- Logistic/Sigmoid function:

$$f(x) = \frac{1}{(1 + e^{-x})}. \tag{44}$$

- Hyperbolic tangent function:

$$f(x) = \tanh(x) \tag{45}$$

- Linear function:

$$f(x) = x \tag{46}$$

- Softplus function:

$$f(x) = \log(e^x + 1). \tag{47}$$

- Exponential linear unit:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases} \tag{48}$$

- Rectified linear unit:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{49}$$

Historically, sigmoid and hyperbolic tangent activation functions found popularity. However, these activation functions suffered from a limitation known as the *vanishing gradient problem* identified by Sepp Hochreiter (Hochreiter *et al*, 2001). The *vanishing gradient problem* is rooted in the backpropagation of errors through a network during training, especially for deep networks.

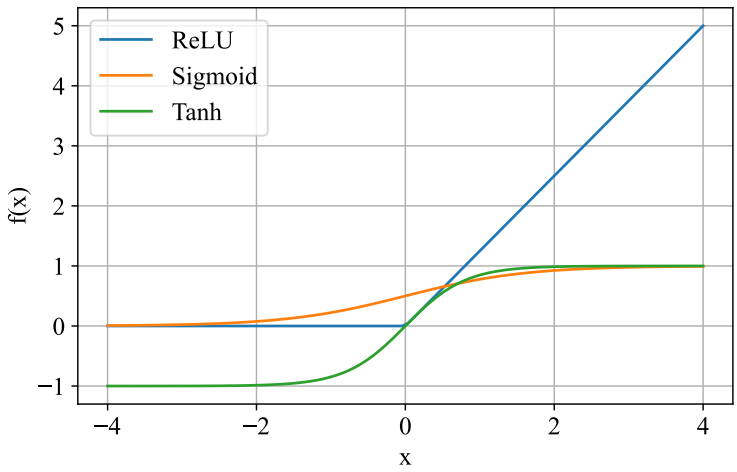


Figure 12: Comparison of the ReLU, sigmoid and tanh activation functions for the same input.

As the name suggests, the *vanishing gradient problem* is a phenomenon observed when the error gradient calculated during backpropagation becomes extremely small as the algorithm moves towards the input layer in a deep neural network. This, of course, will affect the ability of the gradient descent algorithm to find an optimum since a small gradient (a gradient tending towards zero) will result in a very slow training algorithm, which will likely fail to converge. This problem was exacerbated by the differentiable qualities of the common activation functions at the time, such as the tanh and sigmoid functions.

In Figure 12 a comparison of the activation functions of ReLU, sigmoid, and tanh is made for the same input. Notice that the tanh and sigmoid functions saturate at both a lower and upper bound, meaning that for large and small inputs the derivative of the tanh and sigmoid functions tends to zero. This means that during backpropagation saturated inputs at a particular node will force the error signal to zero, and lower layers will consequently receive virtually no gradient information. This problem was partially alleviated by using nonsaturating activation functions such as the Rectified Linear Unit (ReLU) and adjusted weight initialisation techniques (Glorot & Bengio, 2010). It is for this reason that non-saturating activation functions such as ReLU and adaptations of ReLU such as the exponential linear unit (ELU) have found widespread popularity in modern neural network training techniques.

However, the ReLU activation function suffers from the introduction of *dead neurons* due to lower bound activation of zero as observed in Figure 12. If a negative sum is calculated as an input to the ReLU function, the output of the neuron will be zero. This can produce large portions of the neural network with no output. This prompted the introduction of ReLU adaptations which do not have a hard lower bound of zero, but rather an exponentially decaying lower bound such as in the case of the exponential linear unit (ELU) and the scaled exponential linear unit (SELU).

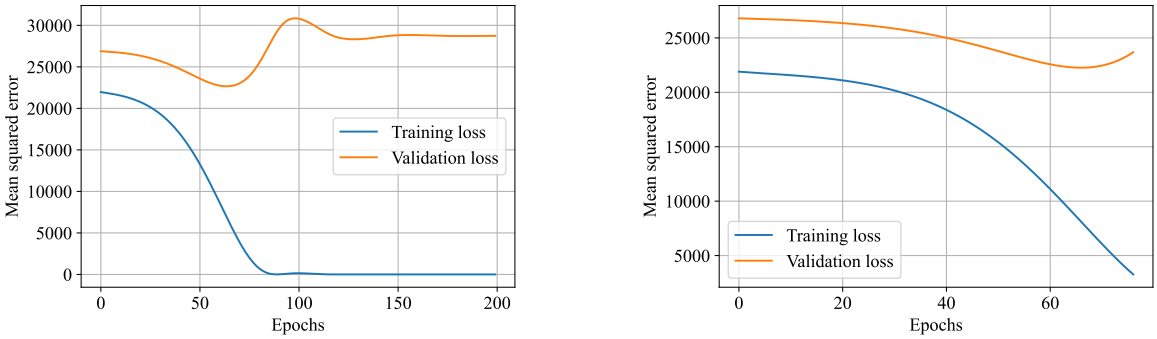
It has become apparent that the choice of activation function has a marked effect on the performance of the final model. There are heuristics that are commonly applied, depending on the modelling problem at hand. Typically ReLU and adaptations thereof are applied to regression problems, such as in the case of the roaster problem. The final output neuron is then typically chosen to have linear output activation. In the case of classification problems, finite integer outputs representing a particular category or class is a logical option.

2.14.3 Regularisation techniques for neural networks

Taking into account all the design parameters for a simple feedforward neural network demonstrates the flexibility of the neural network. However, this flexibility is often a pitfall for the naive modelling engineer. Iterating/training a deep neural network with many hidden layers and neurons with non-linear activation functions often yields very good performance on the training set of data and very poor generalisation on the validation/test set of data. As mentioned, the neural network is a greedy algorithm, so an effort needs to be made to find the trade-off between model bias and variance. There are several techniques that can be easily implemented using the KERAS API, the theory of which is discussed in the following paragraphs.

The concept of *early stopping* links directly to the trade-off between model bias and variance. At the start of neural network training, the number of neurons and hidden layers are specified and the weights are randomly initialised (random initialisation is most often handled by the API of choice such as KERAS and TENSORFLOW). At every iteration (meaning one forward pass to calculate the model output and one backward pass, performing backpropagation and weight updates by gradient descent), a calculation of the model performance on the training and validation data set can be performed. Tracking the progression of performance in the training and validation data set is the core of the concept of *early stopping*. At some point a minimum in the validation error set will be obtained, training the neural network beyond this point will result in over-training and an increase in validation error beyond the minimum will be observed. The *early stopping* technique requires a specification of the number of iterations to perform after the minimum validation error is observed. Once this specification is met, the training is halted. This technique can spare significant amounts of time in the early stages of model development since the number of training iterations to perform on a neural network (with a specified structure) would otherwise have to be approached as a trial-and-error method.

In order to illustrate this point, consider the training and validation mean squared error (loss) on a neural network consisting of 3 layers (1 hidden layer) and a learning rate of 0.01. Note that it is not necessary to use a fixed learning rate, and in actual fact, it is encouraged to schedule the learning rate throughout training (to be discussed). Consider the progression of the calculated mean squared error on the training and validation data from the coffee roaster as shown in Figure 13a. For illustration purposes, it is not important to elaborate on the chosen activation or inputs, however, a simple linear activation was chosen. A reduction in training loss and validation loss is observed up to approximately 75 epochs before the validation loss begins to increase while the training loss continues to reduce. This is a clear illustration of overtraining and the loss of generalisation as a consequence.



(a) Model without *early stopping*. **(b)** Model with *early stopping*.

Figure 13: Comparison of the effect of *early stopping* on an arbitrary neural network model

Early stopping has been implemented using the internal *keras* class *keras.callbacks* which actively monitors the model performance on the training and validation data sets and stops the training after the specified number of iterations (to perform after the minimum in validation error) is observed. These results are shown in Figure 13b. The training was halted 10 iterations after the minimum validation loss was observed. It is in this way that a trade-off between model bias and variance is achieved.

In addition to *early stopping*, a fairly common regularisation technique is known as *dropout*. The *dropout* technique was designed to mimic the effects of training multiple neural networks in such a way as to construct an ensemble method. An ensemble of neural networks would require unlimited computational power since the ideal way to regularise a network is to find the mean prediction of all possible choices of the network parameters (Srivastava *et al*, 2014). This is achieved by randomly dropping nodes during the training procedure at each iteration and effectively recreating the desired advantage of training multiple networks with different architectures without the added cost and effort (Brownlee, 2018: 305). A fixed probability for each neuron within a layer to remain within that layer is chosen. Should the neuron be randomly removed from the network all inputs and outputs from that neuron are nullified. The remaining network is called a thinned network since neurons have been shed from the network. There are in theory 2^n networks that are possible where n is all the neurons which have a probability of being removed. *Dropout* provides major performance and regularisation enhancements over other techniques such as ℓ_1 and ℓ_2 weight regularisation (Srivastava *et al*, 2014).

Dropout rate of 50% was implemented on a neural network consisting of 3 layers (1 hidden layer) and a learning rate of 0.01. This is the same neural network used to illustrate the advantages of implementing *early stopping*. However, *dropout* was implemented after the input and hidden layer. A probability of 50% was chosen since this is suggested as a good starting point in most cases (Srivastava *et al*, 2014). The results of the example

implementation are shown in 14. Comparison of the final validation loss in Figure 14 and Figure 13b illustrate the advantages that *dropout* provide for model regularisation. However, it is interesting to take note of the noisy behaviour illustrated on the training loss curve. The implemented neural network is rather small in size, and reducing the model size by the removal of neurons at each iteration significantly affects the model architecture and computational characteristics. This would explain the large variations in the performance of training loss.

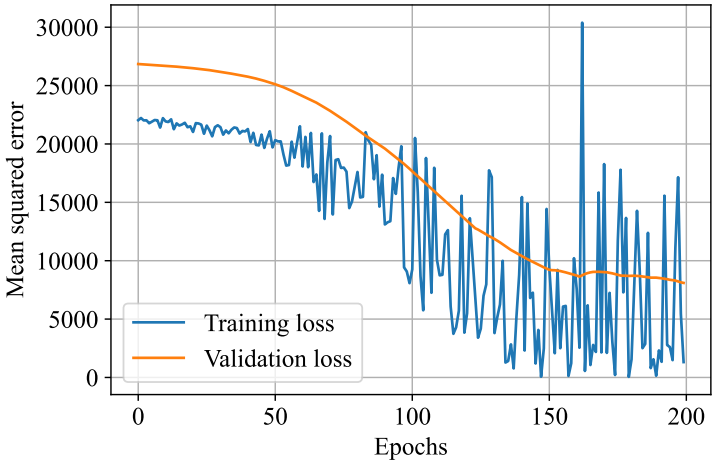


Figure 14: Implementation of *dropout* on the arbitrary neural network model.

The weight regularisation of parameters is not unique to neural networks, mention of ℓ_1 and ℓ_2 with respect to linear models has been made in Section 2.8.3 and 2.8.4. Similarly, the ℓ_1 and ℓ_2 norm of the neural network weights are added to the loss function at every iteration. Minimisation of the loss function (such as the mean squared error) reduces/shrinks the magnitude of the network weights.

2.14.4 Building neural networks in Python

For purposes of modelling the coffee roaster, neural networks will be developed using the TENSORFLOW and KERAS API. These libraries provide intuitive methods of building and regularising neural networks. Importing the appropriate libraries are done using the following call:

```
import tensorflow as tf
from tensorflow import keras
```

The model is initialised as a fully connected feedforward neural network by calling the SEQUENTIAL engine. This API allows for the convenient addition of neurons layer by

layer, each with specified constraints and activation. For the purposes of the coffee roaster, the first layer of the neural network consists of 4 neurons - for the time, LPG, rotation and blower inputs, respectively. Subsequently, a hidden layer is added to the network iteratively depending on the specification of the number of hidden layers. Upon the addition of each layer, a dropout rate is specified as well as the number of neurons (with appropriate activation and weight constraints such as the addition of the ℓ_1 or ℓ_1 norm or, as demonstrated below, the addition of the maximum norm constraint). The optimisation procedure chosen to iteratively determine the weights of the network was the "adam" optimisation procedure. Finally, a single neuron is specified at the output with linear activation since it is desired to model a single output temperature. The PYTHON function below builds and compiles a neural network model that can be trained. Since this function compiles the neural network model as a function of specification parameters, this function could be used in a cross-validation optimisation procedure.

```
def build_NN_model(n_hidden,n_neurons,learning_rate,drop_rate):

    model = keras.models.Sequential()
    model.add(keras.layers.Dense(4))
    for layer in range(n_hidden):
        keras.layers.Dropout(rate = drop_rate)
        model.add(keras.layers.Dense(n_neurons,activation='relu',
                                     kernel_constraint=keras.
                                     constraints.max_norm(3)))
    model.add(keras.layers.Dense(1))
    optimiser = keras.optimizers.Adam(learning_rate=learning_rate)
    model.compile(loss = 'mse',optimizer = optimiser)
    return model
```

3 Critical review on related published research

3.1 Introduction

In the field of coffee roaster modeling and control, significant progress has been made in modelling various roaster types, including drum and fluidised bed roasters. Most contributions in this area have focused on first-principles-based models, as seen in the works of Schwartzberg, Di Palma, and others (Schwartzberg, 2002; Di Palma *et al*, 2021; Hernández *et al*, 2007; Putranto & Chen, 2012). However, research on purely empirical models or hybrids of empirical and first-principles models is limited. The area of multivariable model-based control of coffee roasters is even less explored. Published research on model based coffee roaster control seems to be limited toaA model based PID design strategy which was implemented by Botha (Botha, 2018). This critical review aims to summarise and synthesize these efforts, highlighting key findings and identifying gaps where relevant. Understanding this broader context is crucial for grasping the current state of coffee roasting modelling and control technologies and for positioning the contributions of this dissertation within it.

3.2 Discussion and review

A notable contribution to the model based control of a coffee roaster was made by Botha (Botha, 2018). The dissertation titled, "A model-based control system design for a coffee roasting process" extends on the critical assessment of available first-principles models for a coffee roasting process completed by Vosloo (Vosloo, 2017). Vosloo validated the performance of three heat and mass transfer models (Hernández *et al*, 2007; Putranto & Chen, 2012; Schwartzberg, 2002) and concluded that the proposed models could be used in a model based control strategy. Notably, Vosloo concluded that the model proposed by Schwartzberg (Schwartzberg, 2002) tended to overestimate the roast profile of the coffee roaster, where an overestimation would be defined as a positive deviation between the predicted temperature and the experimental data (Vosloo, 2017).

Using Schwartzberg's model (Schwartzberg, 2002), Botha simulated the coffee roasting process, acknowledging its tendency to overestimate the temperature profile. Botha observed that the initial drying phase (preceding the turning point illustrated in Figure 2) was uncontrollable and identified a consistent turning point at 90 seconds. These observations were based on step tests conducted after the turning point, which may not accurately reflect earlier roast phases. This is further discussed by Bolt and de Vaal

(Bolt & Vaal, 2022), who emphasised the significant role of priming temperature and heat application in determining the turning point.

Botha designed a PID controller which manipulated the LPG flow rate to the roaster and controlled the derivative of the roast temperature profile to a target derivative set point that varies with time. While the strategy appeared to reach set point in the latter portions of the roast, the initial portion of the roast showed significant offset. This offset is critical as the integrating nature of the process means that offset in the temperature derivative during the initial portion of the roast will result in notable offset of the temperature profile from the intended measured temperature trajectory. An extract illustrating this offset in shown in Figure 15. This compounding effect illustrates the requirement for an alternative control approach. Botha concluded that the process would benefit from a multivariable control approach (utilising the additional inputs) and enhanced modelling methodologies such as fuzzy logic or neural networks (Botha, 2018). This would address the non-linear behaviour of the profile as proposed by this dissertation.

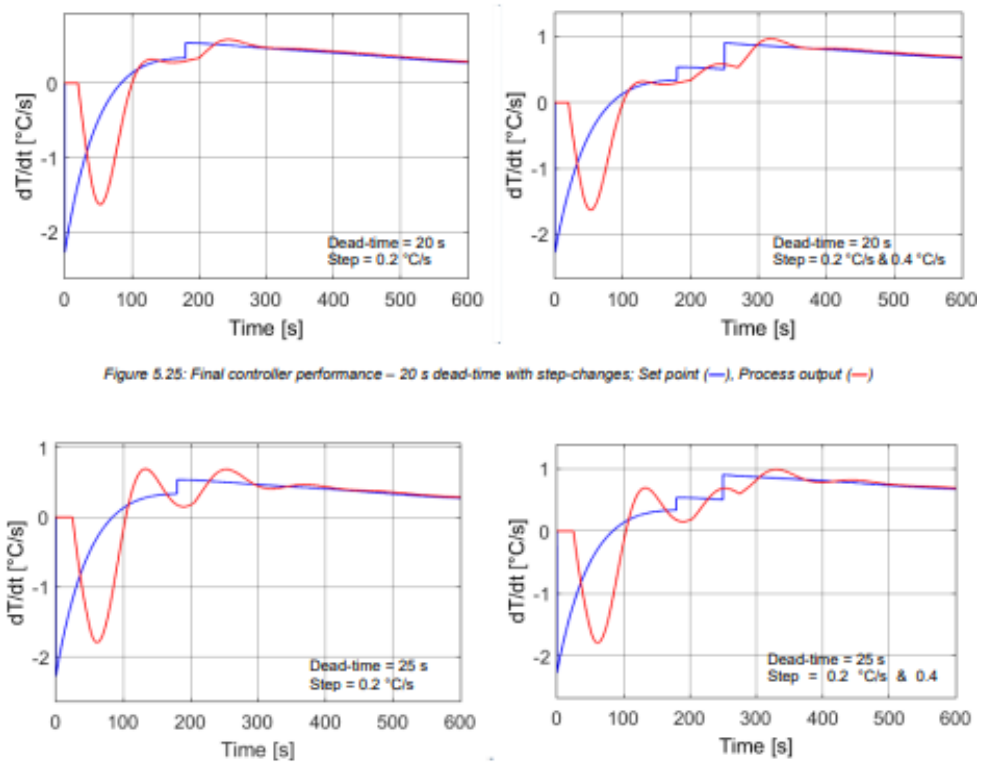


Figure 5.25: Final controller performance – 20 s dead-time with step-changes; Set point (—), Process output (—)

Figure 15: Extract of PID performance reported by Botha (Botha, 2018).

In the field of coffee roaster modelling, Iacono's research (Iacono, 2023) marks a significant advancement, particularly in empirical modeling. The thesis section under discussion compares the performance of a scalable drum roaster model, as proposed by Di Palma (Di Palma *et al*, 2021), with a Long Short-Term Memory (LSTM) neural network model.

Iacono's findings highlight the scalability of the first-principles model, which dynamically accommodates roasters of varying sizes—a notable stride in coffee roaster modeling.

The LSTM model, initially trained on datasets from 120 kg and 360 kg roasters, showed proficiency in model validations within the same size range but lacked generalisability across different sizes. To mitigate this limitation, Iacono refined the LSTM approach by training on a merged dataset from both roaster capacities, resulting in a performance improvement. Despite the challenges posed by limited data, Iacono concludes that coffee roaster modeling could be substantially improved with a hybrid approach which aligns with the methods proposed in this dissertation.

In summary, the reviewed literature on the modelling and control of a coffee roaster points towards an increased in interest the modelling of coffee roasters, often favouring first-principles methods. There is limited literature available which addresses a multi-variable control approach to the coffee roaster process. Botha's study highlights the complexity of the roast profile and the challenges it's non-linearity introduces. The limitations of a single-variable control strategy suggest the potential of non-linear modelling and multivariable control to better capture the dynamics of the coffee roaster process. This is a prospect that this dissertation aims to address. Iacono's research highlights the importance of scalability in the coffee roaster model performance and suggests the potential performance enhancements to be had through a hybrid modelling approach. The findings are valuable for this dissertation guiding it's approach in hybrid modelling.

4 Data preparation and modelling methodology

4.1 Data preparation

Data frames are available for the 6 kg, 15 kg and 30 kg roasters (where "6 kg" refers to the raw bean capacity of the roaster). The amount of data for each roaster varies significantly, approximately 880 000, 360 000 and 21 000 samples are available for the 6 kg, 15 kg and 30 kg roasters respectively (after filtering). Each sample consists of a vector of a measured bean temperature, measured environmental temperature, and a time point in addition to the LPG, rotation, and blower inputs (where the inputs are scaled as a percentage).

The data frames of each roaster are filtered by ensuring that the following conditions were met for every single roast/batch:

- The priming temperature of the roast is above 150 °C.
- The roast continues for more than 700 seconds.
- The final measured temperature of the beans is below 250 °C.

The criterion for the conditions was established based on consultation with the roaster manufacturers, but is largely based on the normal operating procedure. For example, consider that roasts that reach final temperatures of 230 °C typically result in scorching of the beans, while roasts that are terminated prematurely result in under-roasted beans.

The objective is to model the measured bean temperature as a function of the roaster inputs. Therefore, the data frames of each roaster are split into an array for the measured bean temperature, denoted as Y, and a matrix of inputs denoted as X. The inputs included in the matrix are the time, LPG, rotation, and blower inputs. Note that in the case of the physics-informed empirical modelling, the adapted Schwartzberg (Schwartzberg, 2002) model predictions are included in the input matrix.

The input matrix and output array are partitioned by an 80/20 split into a training and validation data set, respectively. The training data set will be used in the development of each model while the performance of the respective model will be assessed on the validation data set. The partitioned test and validation input matrix is scaled using standard scaling.

4.2 Modelling methodology

4.2.1 Schwartzberg model simulation

The set of differential equations describing the transfer of mass and heat within a rotational drum is described in Sections 2.6 and 2.7. The set of equations is encoded in PYTHON and solved using an *Euler* integration procedure. The simulation procedure is shown in Figure 16.

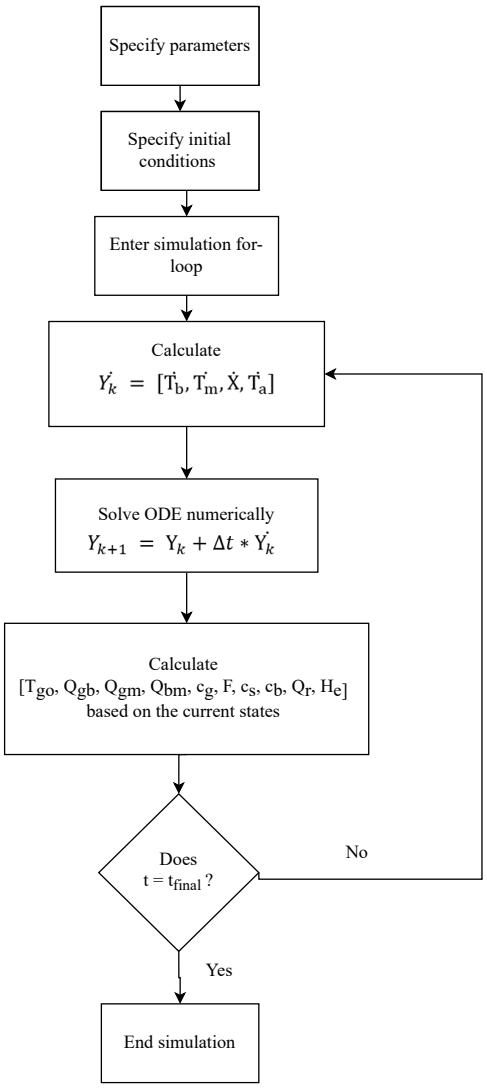


Figure 16: Schwartzberg model simulation procedure in PYTHON.

The volumetric flow rate of air passing through the roaster is measured at 660 m^3/h at a blower input of 100% for each roaster size. The 6 kg and the 15 kg roasters are both fitted with 25 kW burners while the 30 kg roaster is fitted with a 50 kW burner.

The Schwartzberg model accounts for both changes in the blower and burner inputs. The blower input is assumed to scale linearly between 0% and 100%. Meaning that at an input of 0%, the volumetric flow rate of air passing through the roaster is measured at 0 m^3/h and at 100% the flow rate is measured at 660 m^3/h. Instead of accounting for changes in the inlet temperature, T_{gi}, as a function of the input of LPG (burner), historically recorded inlet temperatures were sampled from the database at each instant. This is a limitation in the approach taken for the Schwartzberg simulation. Furthermore, the Schwartzberg model does not account for the rotational input changes.

4.2.2 Empirical modelling methodology

Modelling will be performed on the filtered data set of each roaster size as discussed in Section 4.1. The modelling techniques that will be investigated include linear regression, decision tree regression (DTR), random forest regression (RFR), and artificial neural network regression (ANNR). Each of the modelling techniques will be developed in the following manner:

1. A PYTHON function will be written that expresses the modelling technique as a function of its parameters. This function will be called the builder function.
2. The builder function will compile a model using the appropriate library.
3. The builder function will return the mean squared error (MSE) between the predictions of the measured bean temperature and the true measured bean temperature in the validation data set. The builder function, therefore, represents a black-box-type objective function.
4. The black-box builder function will be passed as an argument to a Bayesian optimiser, which will return the optimal parameters that yield a minimum in the mean squared error (MSE) on the validation data set.
5. The search space for each parameter is user-defined and as an argument to the Bayesian optimiser function.

This process will be followed for all modelling techniques, excluding linear regression since the linear regression model is non-parametric. Parameters to be included in the search

space for decision tree regression (DTR), random forest regression (RFR), and artificial neural network regression (ANNR) are as follows:

- DTR: Tree depth (max_depth parameter)
- RFR: Tree depth (max_depth parameter) of each tree and the number of trees (n_estimators parameter) to be included in the ensemble.
- ANNR: the number of neurons per hidden layer, the number of hidden layers, the dropout rate for layers greater than the 5^{th} layer, and the learning rate used in the gradient descent optimiser. The max norm of the weights is fixed at a numeric value of 3, and the optimiser implemented is the Adam algorithm.

The compiled and optimised models will be compared to each other on the basis of the mean squared error (MSE), mean absolute percentage error (MAPE), and median absolute error (MedAE) on the respective validation data sets.

As stated above in point 5 of the empirical modelling methodology, the search space for each parameter is user defined and requires specification. The choice of the bounds on the search space was initially chosen at random. Subsequent to an optimisation procedure the bounds were increased if it was noted that the optimal value was found at the constraint. In most instances the logical lower bound of the parameters were set at an integer value of 1 such as the max_depth parameter, the n_estimators parameter, the number of neurons per hidden layer and the number of hidden layers. The drop out rate on the neural networks was bound between 0.1 and 0.9 (refer to Section 2.14.3). To determine the appropriate learning rate of a neural network, an initial network was trained at a random learning rate to determine a baseline in terms of network performance and time taken to train the model. A "fast" rate is determined at which degraded prediction performance is noted, and a "slow" rate is noted for very slow training cycles.

Table 2: Guideline on bounds chosen for the optimisation of empirical models.

Parameter	Lower Bound	Upper Bounds
Max depth of a DT	1.000	50.00
Max depth of a RFR	1.000	50.00
Number of trees within a RFR	1.000	250.00
NN droprate	0.100	0.900
Neurons per NN layer	1.000	100.00
Hidden layers per NN	1.000	50.00
Learning rate for NN backpropagation	4×10^{-4}	4×10^{-3}

5 Modelling

The theory and tools required to implement multiple modelling techniques, empirical as well as first principles in nature, have been presented. This section will present the results of the discussed modelling techniques and analyse the performance of each model on the validation data set for the 6 kg, 15 kg, and 30 kg rotating drum roaster.

5.1 Optimised Schwartzberg model

The optimised adapted Schwartzberg model, as presented in Section 2.6, introduced additional fitting parameters (k_3, k_4 and k_5) and included refitting of the gas to metal heat transfer coefficient (h_{gm}), the bean to metal heat transfer coefficient (h_{bm}) and the proportion of total bean surface area in contact with the metal surface (P_{bm}). The bounds on each of the parameters implemented during optimisation as well as the optimised parameters for each roaster size, are summarised in Tables 3, 4 and 5.

Table 3: Optimised adapted Schwartzberg parameters for the 6 kg roaster.

Parameter	Units	Lower bound	Upper bound	Optimised value
k_3	kW/m^2K	0.400	10.00	3.77
k_4	kJ/kgK	15.00	80.00	52.8
k_5	kg/kg	0.000	10.00	0.58
h_{gm}	kW/m^2K	0.000	10.00	0.00
h_{bm}	kW/m^2K	0.005	0.500	0.00
P_{bm}	m^2/m^2	0.100	1.000	0.39

Table 4: Optimised adapted Schwartzberg parameters for the 15 kg roaster.

Parameter	Units	Lower bound	Upper bound	Optimised value
k_3	kW/m^2K	0.400	10.00	0.78
k_4	kJ/kgK	15.00	80.00	20.0
k_5	kg/kg	0.000	10.00	0.94
h_{gm}	kW/m^2K	0.000	10.00	0.00
h_{bm}	kW/m^2K	0.000	3.000	0.07
P_{bm}	m^2/m^2	0.100	1.000	0.45

Table 5: Optimised adapted Schwartzberg parameters for the 30 kg roaster.

Parameter	Units	Lower bound	Upper bound	Optimised value
k_3	kW/m^{2}K	0.400	10.00	0.730
k_4	kJ/kgK	15.00	80.00	19.30
k_5	kg/kg	0.000	10.00	0.510
h_{gm}	kW/m^{2}K	0.000	10.00	1.220
h_{bm}	kW/m^{2}K	0.000	0.500	0.000
P_{bm}	m^2/m^2	0.100	1.000	0.480

Figure 17 compares the distribution of prediction error for the 6 kg, 15 kg and 30 kg adapted optimised Schwartzberg model. The 6 kg roaster model and the 15 kg roaster model achieved comparative performance. The mean squared error (MSE) on prediction between the historical measured temperature and the Schwartzberg model prediction was 103 °C^2, 174 °C^2 and 891 °C^2 for the 6 kg, 15 kg and 30 kg adapted optimised Schwartzberg models respectively. The optimised 6 kg roaster model achieved the best performance with a mean error on prediction of -1.65 °C and a standard deviation of 10.0 °C.

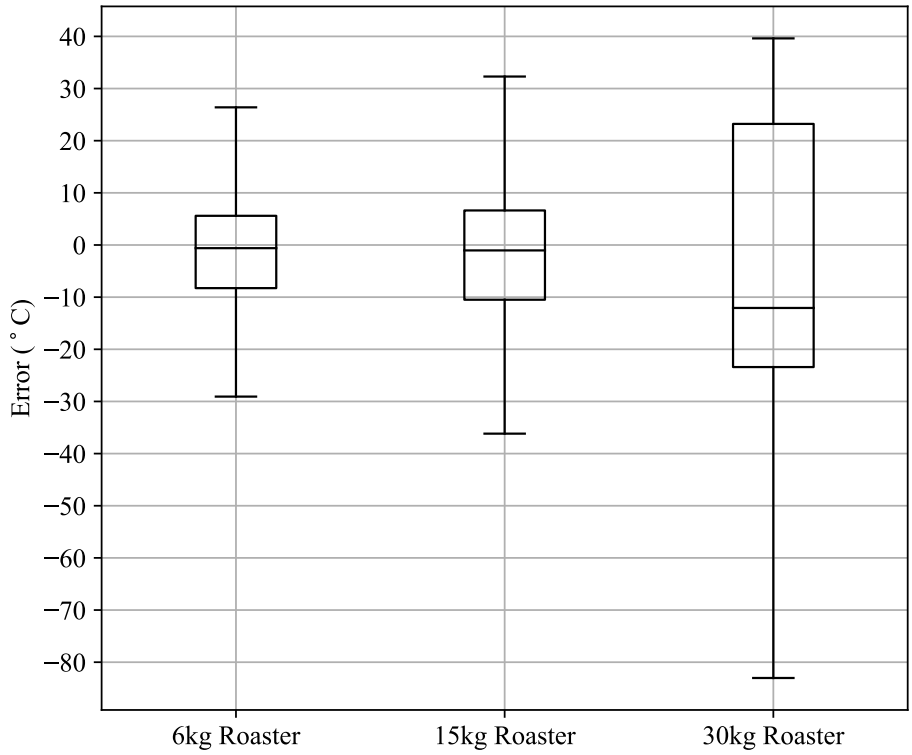


Figure 17: Box plot comparing the error on the adapted optimised Schwartzberg prediction for all roaster sizes.

The optimised parameter, k_3, is an empirical parameter used to adjust the air-to-bean heat transfer coefficient, h_e, to historical data. Similar optimal values were obtained for the 15 kg roaster and 30 kg roaster models. Values less than one indicate that a smaller heat transfer coefficient is fitted to the historical data when compared to the heat transfer coefficient constants provided by Schwartzberg (Schwartzberg, 2002). In the case of the optimised value for the 6 kg model, a much larger value was obtained.

The optimised parameter k_4 fits the overall bean heat capacity, c_b, to historical data. The overall heat capacity of the bean, described in Equation 16, is a function of the moisture content of the bean, X, and the temperature of the bean T_b. Similarly to the optimised parameter k_3, the optimal values for k_4 are comparable in magnitude for the 15 kg and 30 kg roaster models, and larger for the 6 kg roaster. The rate of change in bean temperature, $\dot{T}_b$, is directly proportional to the air-to-bean heat transfer coefficient and inversely proportional to the bean heat capacity. When comparing the relative magnitudes of the optimised k_3 and k_4, one might conclude that the overall heat transfer coefficient (such as a global heat transfer coefficient) for the 6 kg roaster is larger. For similar blower capacities, a smaller chamber diameter might provide reasoning for a larger heat transfer coefficient compared to the 15 kg and 30 kg roaster sizes.

The optimised parameter k_5 fits the rate of change in bean moisture content to historical data. It effectively refits the semi-empirical parameter k_1, provided in literature (Schwartzberg, 2002).

In the case of the 6 kg and 15 kg Schwartzberg model optimisation, the gas-to-metal heat transfer coefficient h_{gm} was optimised to a value equivalent to the lower limit. It was expected that the heat transfer to the roaster metal will be insignificant due to the priming process that occurs before roasting the beans. In all cases, the optimised bean-to-metal heat transfer coefficient h_{bm} was found to be near zero. In the case where the gas-to-metal and bean-to-metal heat transfer coefficients are found to be zero, one can conclude that the primary heat transfer mechanism roasting the beans is the convective heat transfer between the heated air and the beans. The proportion of bean surface area in contact with the metal surface, P_{bm}, is expected to be less than 50%, since only one side of each bean can be in contact with the metal surface should all the beans be in contact with the drum surface. Furthermore, each bean's position is dynamic as the beans are flailed around within the drum. Balancing the rotational speed affects the amount of contact the beans have with the metal surface and consequently influences the heat transfer mechanism responsible for roasting the beans.

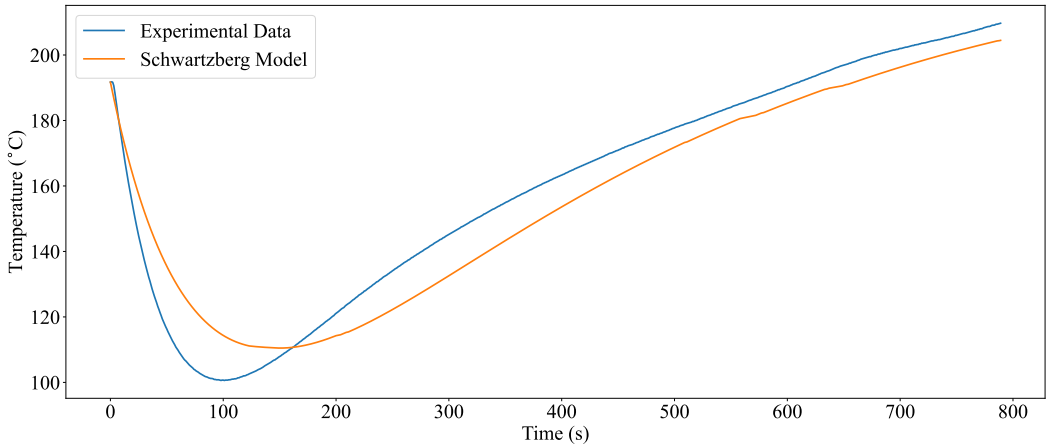


Figure 18: Example 6 kg Schwartzberg roaster model simulation.

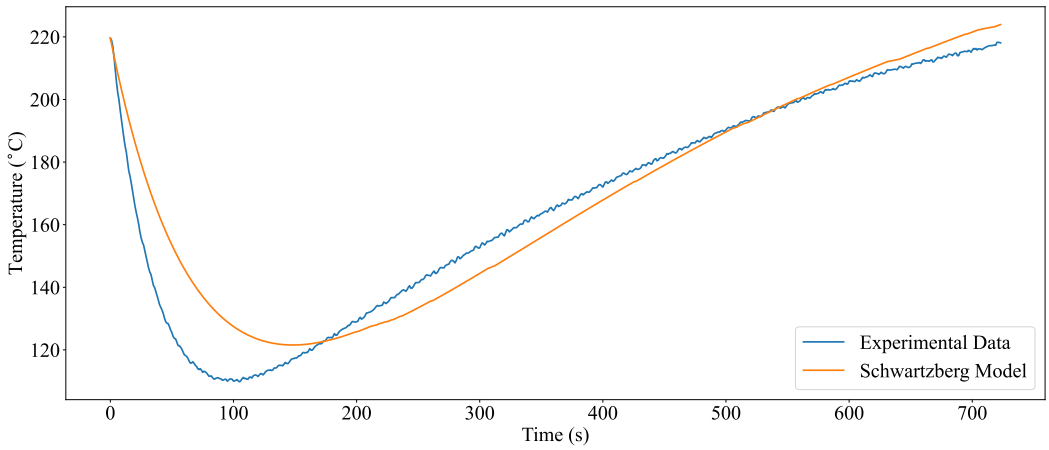


Figure 19: Example 15 kg Schwartzberg roaster model simulation.

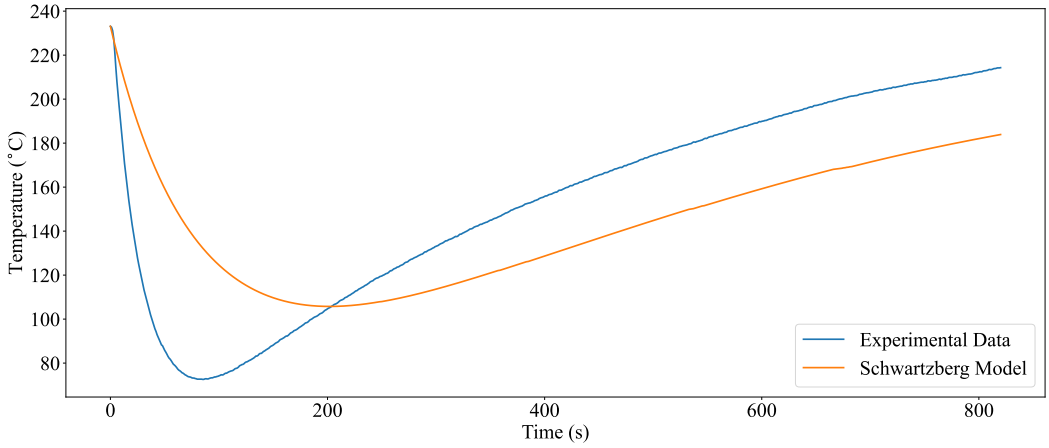


Figure 20: Example 30 kg Schwartzberg roaster model simulation.

5.2 Empirical modelling results

This section presents and discusses the performance of each empirical model and the physics-informed (PI) models, which include the Schwartzberg temperature simulation as an input. The models compared are Random Forest Regressor (RFR), Decision Tree Regressor (DTR), Artificial Neural Network (ANN), and Linear Regression. The performance metrics considered include the mean squared error (MSE), mean absolute percentage error (MAPE) and median absolute error (MedAE). In general, the RFR model achieved the best performance across all roaster sizes. The PI RFR models also showed improved performance compared to their non-PI counterparts.

5.2.1 The 6 kg roaster modelling

The performance of each model on the validation data set of the 6 kg roaster is presented in Table 6.

Table 6: Performance measures including measures of central tendency on prediction error of each 6 kg roaster model.

	Linear	PI Linear	ANN	PI ANN	DTR	PI DTR	RFR	PI RFR
MSE	206.5	53.09	8.463	4.737	8.707	5.881	8.029	4.175
MAPE	0.064	0.039	0.015	0.011	0.015	0.011	0.014	0.010
MedAE	5.968	4.349	1.614	1.268	1.522	0.916	1.429	0.915
mean	0.024	-0.023	0.021	0.062	0.000	-0.005	0.002	0.000
std	14.37	7.286	2.909	2.237	2.951	2.425	2.834	2.043
min	-55.85	-85.07	-24.64	-14.49	-39.00	-51.80	-25.61	-46.03
25%	-7.598	-4.124	-1.595	-1.166	-1.527	-0.909	-1.425	-0.889
50%	0.642	0.198	0.020	0.108	-0.000	0.000	0.005	0.022
75%	5.243	4.527	1.634	1.363	1.517	0.923	1.434	0.941
max	122.9	42.73	37.62	22.69	37.59	41.30	37.46	26.21

Analysis of the performance metrics for the empirical models (these are the models excluding the Schwartzberg predictions at each sampling instant) concludes that the RFR model performed best (possessing the smallest quantity) when considering the MSE, MAPE and MedAE. The ANN performed slightly better than the DTR on the validation set when the MSE of each model is compared. It appears that while the DTR achieved a mean of 0.000 °C compared to 0.021 °C for the ANN, the spread of error on the DTR is larger. The DTR achieved a slightly larger standard deviation in error, 2.951 °C compared to the 2.909 °C of the ANN. The larger MSE observed in the DTR is due to this slightly larger spread and the presence of outliers visible in the maximum and minimum error observed

for the DTR. A box plot of the error of each empirical model and physics-informed model on the validation set is displayed side by side in Figure 21.

An improvement (a reduction in the measures) was observed in the performance of all models with the inclusion of the Schwartzberg model predictions as an input. The inclusion showed a reduction of 49.70%, 30.24% and 31.09% in the average MSE, MAPE, and MedAE. The PI RFR achieved the best overall performance (in the case of the empirical models and the models including the Schwartzberg prediction). This can be observed graphically in Figure 21 as a reduction in the inter-quartile range (IQR) of each empirical model when compared to the respective first-principles informed model.

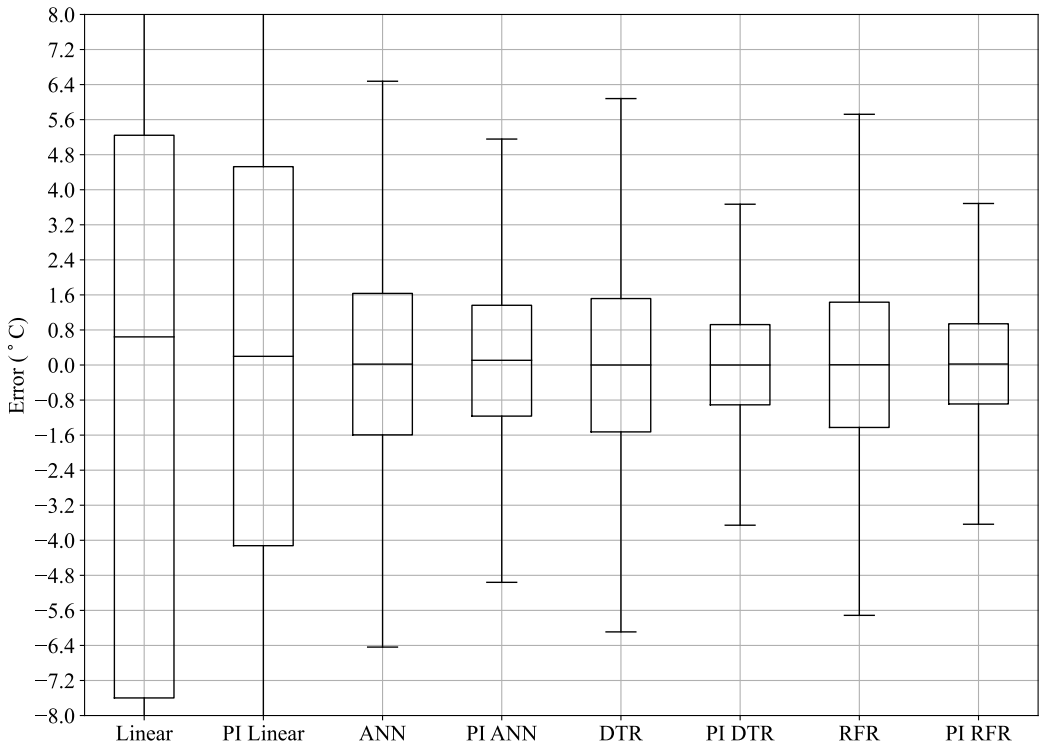


Figure 21: Box plot of the error on prediction of each 6 kg roaster model.

5.2.2 The 15 kg roaster modelling

The performance of each model on the validation data set of the 15 kg roaster is presented in Table 7.

Table 7: Performance measures including measures of central tendency on prediction error of each 15 kg roaster model.

	Linear	PI Linear	ANN	PI ANN	DTR	PI DTR	RFR	PI RFR
MSE	267.0	92.24	15.65	5.402	12.94	6.213	11.45	4.260
MAPE	0.074	0.048	0.018	0.011	0.015	0.010	0.014	0.009
MedAE	8.037	5.682	2.160	1.179	1.429	0.809	1.357	0.672
mean	0.032	0.016	0.251	0.043	0.017	0.006	0.016	0.002
std	16.34	9.604	3.985	2.324	3.598	2.495	3.383	2.066
min	-59.25	-36.46	-35.44	-18.23	-39.93	-19.20	-39.48	-18.02
25%	-9.607	-5.892	-1.655	-1.111	-1.275	-0.790	-1.212	-0.634
50%	-0.276	0.194	0.452	0.066	0.091	0.000	0.073	0.005
75%	7.007	5.535	2.552	1.237	1.591	0.845	1.518	0.707
max	107.4	53.71	27.63	15.95	40.30	19.60	29.16	15.46

Analysis of the performance metrics for the empirical models (these are the models excluding the Schwartzberg predictions at each sampling instant) concludes that the RFR model performed best when considering the MSE, MAPE, and MedAE. The performance of the DTR is comparable to the RFR, it appears that there is not a significant advantage in using an ensemble of decision trees over a single decision tree. The distribution of error in the DTR is comparable to that in the RFR. There is a significant additional computational cost in the development of an RFR (an ensemble of decision trees), which has yielded only a slight improvement in performance over the DTR (a single decision tree). The ANN, which consumed the most computational effort to develop, demonstrated disappointing performance compared to the performance of the DTR and RFR. The linear model provides the baseline of a high-bias model, improving model variance (such as in the case of the ANN, DTR and RFR) and shows significant improvement in performance and generalisation of the selected models to validation data. This was expected due to the non-linear behaviour of the temperature-time curve displayed in the coffee roaster.

An improvement (a reduction) was observed in the performance of all models with the inclusion of the Schwartzberg model predictions as an input. The inclusion showed a reduction of 61.43%, 35.77% and 42.15% in the average MSE, MAPE, and MedAE. The error IQR reduction can be observed in Figure 22. Considering the performance of all first-principles-informed empirical models, the first-principles-informed RFR achieved the

best performance on the validation data set. An example simulation utilising the first-principles informed RFR model is shown in Figure 23 with the associated inputs.

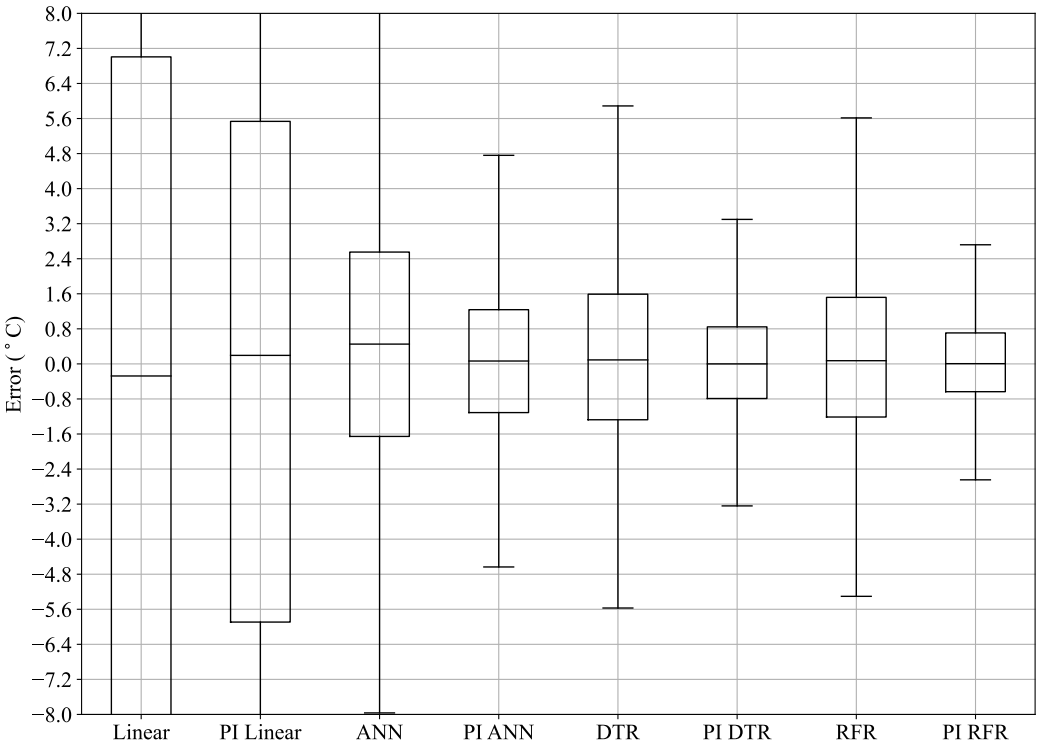


Figure 22: Box plot of the error on prediction of each 15 kg roaster model.

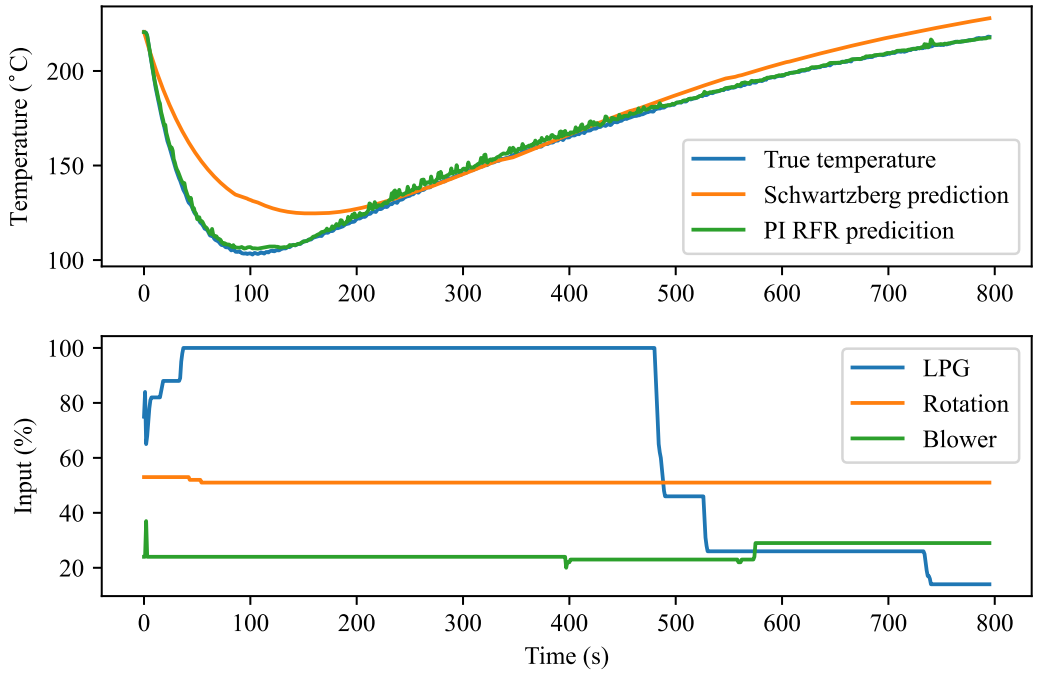


Figure 23: Example simulation of the 15 kg roaster with associated inputs.

5.2.3 The 30 kg roaster modelling

The performance of each model on the validation data set of the 30 kg roaster is presented in Table 8.

Table 8: Performance measures including measures of central tendency on prediction error of each 30 kg roaster model.

	Linear	PI Linear	ANN	PI ANN	DTR	PI DTR	RFR	PI RFR
MSE	357.0	258.1	1.472	1.858	1.788	1.731	1.207	0.833
MAPE	0.088	0.086	0.006	0.005	0.005	0.004	0.004	0.003
MedAE	7.384	8.944	0.634	0.448	0.300	0.200	0.308	0.251
mean	0.090	-0.001	0.061	0.091	0.030	0.005	0.046	0.020
std	18.89	16.07	1.212	1.360	1.337	1.316	1.098	0.913
min	-34.16	-47.21	-6.569	-21.05	-12.90	-21.80	-10.70	-10.23
25%	-8.949	-9.550	-0.628	-0.366	-0.300	-0.200	-0.259	-0.241
50%	0.772	3.550	0.040	0.071	0.000	0.000	0.032	0.015
75%	6.903	8.769	0.639	0.514	0.300	0.200	0.343	0.266
max	145.27	113.9	10.15	11.81	13.20	11.40	12.27	9.827

Analysis of the performance results of the empirical 30 kg roaster models on the validation data set concludes that the RFR achieved the lowest error scores when considering the MSE, MAPE and MedAE respectively. However, the DTR achieved a mean error of 0.030 °C compared to the mean error of 0.046 for the RFR. Although the DTR achieved a mean error slightly closer to zero, the spread of error in the DTR predictions is greater with a standard deviation of 1.337 °C compared to 1.098 °C in the case of the RFR. This yields larger magnitude performance metrics (MSE, MAPE and MedAE) when comparing the DTR and RFR. The ANN achieved a smaller MSE and larger MAPE and MedAE compared to the DTR. This can be explained by the larger mean error of 0.061 compared to the mean error of 0.030 °C for the DTR. The mean squared error penalises outliers as seen in the case of the DTR.

In the case of the 30 kg roaster modelling, the inclusion of the Schwartzberg temperature predictions showed a reduction in the MSE, MAPE and MedAE performance measures except in the cases of the calculated MSE of the ANN and PI ANN as well as the calculated MedAE of the Linear and PI linear models. While the PI RFR achieved the best MSE and MAPE the DTR performed best when considering the MedAE. As in the analysis of the empirical models, the calculated MedAE is smaller in the case of the DTR due to a mean error closer to zero when compared to the mean error of the PI RFR. The larger calculated MSE and MAPE of the DTR can be explained by the larger variance in the error of the PI DTR compared to the PI RFR. In all cases excluding the ANN and

PI ANN, a reduction in the variance of the calculated error on the validation data set was observed. This explains the observed increase in MSE of the ANN with the inclusion of the Schwartzberg predictions. The effect of the Schwartzberg inclusion on the central tendency statistics is visualised in Figure 24.

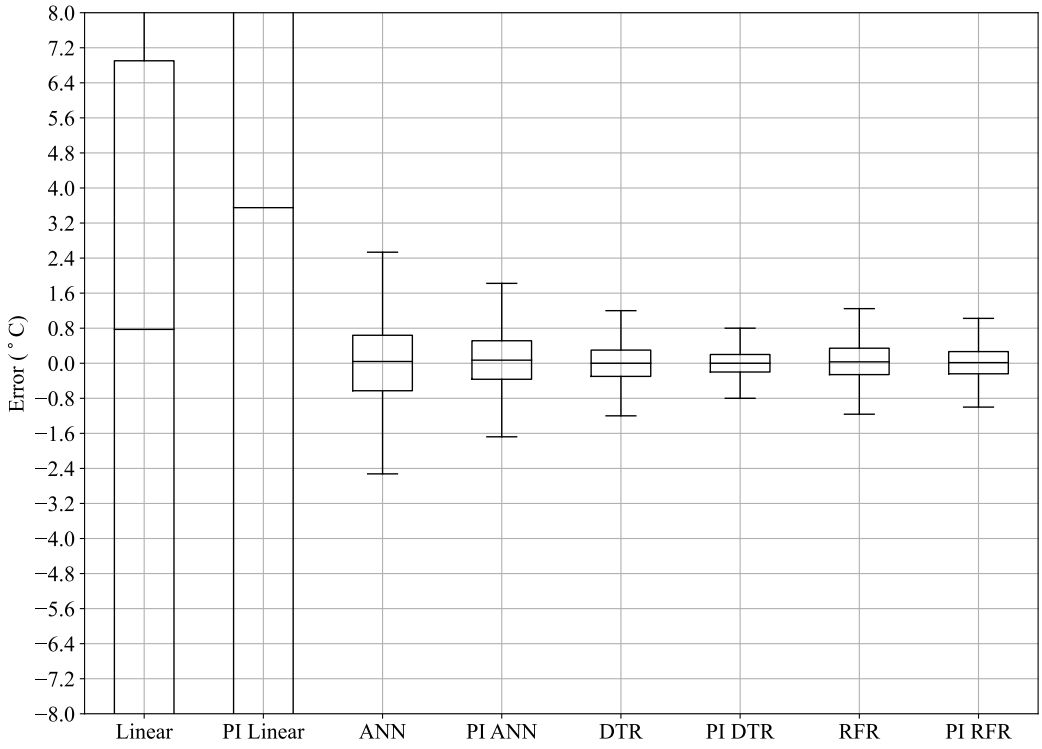


Figure 24: Box plot of the error on prediction of each 30 kg roaster model.

5.3 Modelling conclusions

In conclusion, the optimised adapted Schwartzberg model, with additional fitting parameters and refitting of heat transfer coefficients, has been presented and analysed for various roaster sizes. The 6kg and 15kg roaster models showed similar performance, with the 6kg model achieving the best overall performance. The primary heat transfer mechanism responsible for roasting the beans was identified as the convective heat transfer between the heated air and the beans, with both the gas-to-metal and bean-to-metal heat transfer coefficients found to be near zero. The optimised empirical parameters, k_3, k_4, and k_5, were used to adapt the model to historical data, providing insights into the differences between the roaster sizes and their heat transfer characteristics.

These results provide a better understanding of the heat transfer mechanisms involved in the coffee roasting process and the behaviour of the roaster models for different sizes. This information can be helpful for roasting equipment manufacturers and coffee professionals

in designing and optimising the roasting process, ultimately leading to improved control and consistency in the final roasted coffee product.

Databases of varying sizes, belonging to different-sized roasters, have been used in two distinct ways. First, empirical models of each roaster size have been developed and compared. Second, physics-informed (PI) models incorporating the Schwartzberg temperature predictions as inputs were developed and compared. In both cases, the RFR model provided the best overall performance when considering MSE, MAPE, and MedAE. The same can be said for the enhanced modelling techniques; the PI RFR achieved the best overall performance.

The linear model and the enhanced linear model did not compete well with the performance of more complex models such as the random forest, decision tree, and neural network. This was partly expected due to the non-linear nature of the temperature curve. However, this demonstrates the trade-off between model bias and variance. A high-bias model, such as the linear model, makes strong assumptions about the relationship between the outputs and inputs, which can limit the model's generalisation to the validation data sets.

As fewer assumptions about the model structure are made, such as in the case of the DTR, RFR, and ANN, the variance of the model structure increases. The complexity of the calculation of the model parameters scales with variance. To reduce overfitting and improve generalisation, techniques such as early stopping for the ANN and pruning for decision trees and random forests need to be implemented. As demonstrated, the development of the DTR, RFR, and ANN can be optimised by minimising the error metrics of each model on the validation set as a function of the model's parameters in question. The final optimised structure of each model is summarised in Appendix A.1.

The computation time required to train each of the high-variance models is scaled as a function of the number of optimisation parameters. The DTR and RFR, which have fewer optimisation parameters, were developed in significantly less time than the ANN. This was the case for all roaster databases. Although the performance of the ANN was comparable to that of the DTR and RFR in each application, the longer computation time required to develop the ANN made it a less viable option.

The Schwartzberg model accounts for both changes in the blower and burner inputs. Instead of accounting for changes in the inlet temperature, T_{gi}, as a function of the LPG (burner) input, the historically recorded inlet temperatures were sampled from the database at each instant. This is a limitation of the approach used for the Schwartzberg simulation. **Further research** could focus on extending the Schwartzberg model to include an energy balance on the inlet air so that the percentage of LPG input can be

directly accounted for. This will be especially useful for the application of the MPC, so that at least two of the roaster inputs can be used to make predictions over the prediction horizon using the Schwartzberg model. Subsequently, the result of this prediction could be used as input to an empirical model to improve predictive capabilities.

In conclusion, the RFR model emerged as the most suitable choice for modelling coffee roasters among the models considered. It provided the best performance in terms of the MSE, MAPE, and MedAE metrics while requiring less computation time compared to the ANN. The inclusion of the Schwartzberg temperature predictions as inputs in the PI models led to improvements in the performance of all models, further demonstrating the benefits of incorporating physics-based information in empirical models. Future research could explore the use of other physics-informed modelling techniques or investigate alternative machine learning approaches to further improve the modelling and control of coffee roasters.

6 Control system design

In this section, the focus is on designing a multivariable control system for a coffee roaster. The model predictive control (MPC) algorithm is developed using the Random Forest Regressor (RFR) model, chosen for its superior performance (refer to Section 5.2). The control system aims to maintain the coffee roaster's measured bean temperature at a setpoint based on a roast profile. The control system is designed to adjust three available roaster inputs (LPG, rotation, and blower) simultaneously to control the measured bean temperature. The system has to account for disturbance variables, such as LPG quality and bean moisture content, which are not measurable.

The control system block diagram is presented to illustrate the relationship between the MPC algorithm and the coffee roaster. The algorithm implementation, controller tuning, and simulation are discussed in detail. The roaster model controller (MPC) is implemented using a step-by-step algorithm to control the roasting process. The optimisation is performed using Bayesian optimisation, chosen for its suitability for computationally expensive objective functions. The optimisation routine and framework are described, focusing on the initialisation of the optimisation and the execution of the control law.

Finally, controller tuning and simulation are discussed. The tuning process involves adjusting the controller parameters, such as prediction and control horizons, to improve performance. The performance is assessed by comparing the predicted temperature response of the roaster to the target/setpoint temperature profile. The results of the tuning

procedure demonstrate the effectiveness of the roaster MPC in maintaining the desired roasting conditions.

6.1 Control strategy

The objective of the control system is to maintain the measured bean temperature of the coffee roaster at the temperature setpoint. The setpoint is a function of time and is based on a previous roast, commonly referred to as the roast profile. The final control system should be able to adjust the three available roaster inputs simultaneously to leverage changes in the measured bean temperature. The available manipulated variables are the LPG, rotation, and blower inputs. Disturbance variables such as variance in LPG quality, as well as bean moisture content, are present but are not measurable. The roaster is typically housed indoors, and the variance in ambient temperature is not expected to have a large effect on the insulated drum. Modelling techniques between the roaster inputs and the measured bean temperature have been presented and will be applied in the final control strategy using model predictive control. The sections to follow will expand on how the final controller was tuned and developed.

6.2 Control system block diagram

The control system block diagram shown in Figure 25 summarises the model predictive control algorithm as it relates to the coffee roaster. The roaster MPC will use a model, $f(x)$, which is a function of the roaster inputs to make predictions of the measured bean temperature over the prediction horizon, P. The control horizon, M, will be chosen to be smaller or equal to the prediction horizon. Reduction of the control horizon will reduce the degrees of freedom during the optimisation procedure. The control and prediction horizon are specified as the number of samples taken in the future horizon. The sampling rate on the coffee control system is 1 second. This requires that the execution of the control law is made in less than 1 second.

The optimisation routine minimises the objective function, J, as defined in Equation 7. The solution to the optimisation problem is the set of inputs which minimises the error between the projected measured temperature and the set point temperature, where the set point temperature is defined by a saved roaster profile. The control moves (inputs) passed to the roaster at each sampling interval is only the first control signal for each input i.e. the first element in each row of the solution matrix. This ensures flexibility in the algorithm as well as inherent disturbance rejection; the control actions can adapt to unmeasured disturbances in the system.

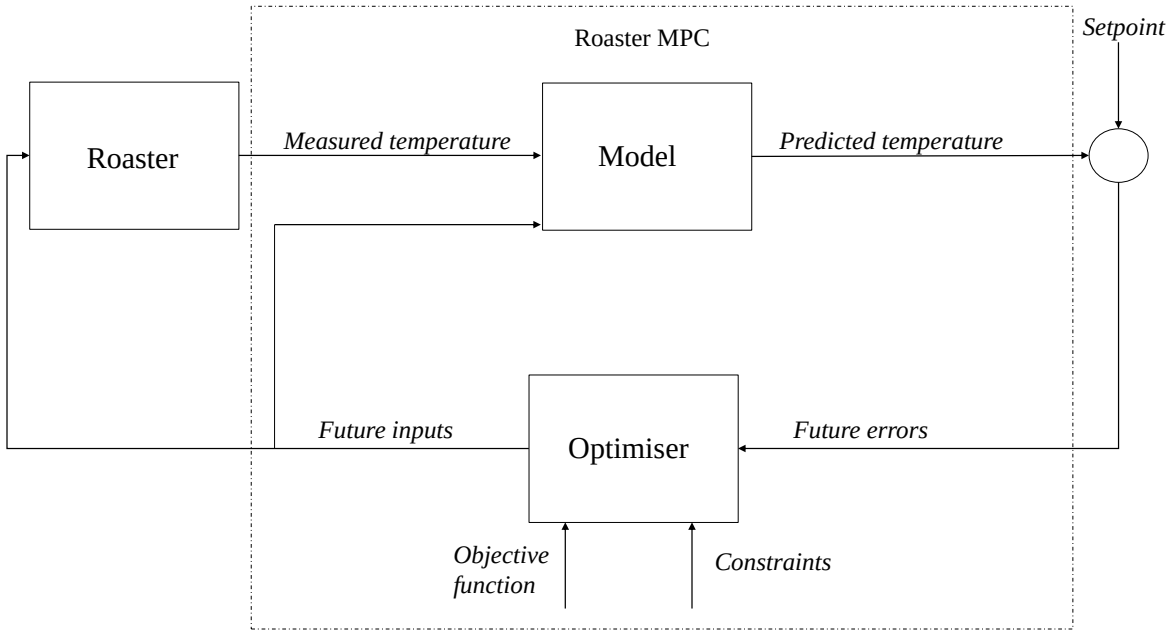


Figure 25: Roaster model predictive control block diagram.

6.3 The control system algorithm

The roaster model controller (roaster MPC), discussed above, will be implemented using the algorithm as stated below:

1. The roaster operator loads a roaster profile from the HMI, which contains a historical set of inputs and a known measured temperature profile. The measured temperature profile will be used as a set point, which varies as a function of time.

2. The roaster will be primed to the initial temperature of the roaster profile using the LPG and blower inputs. This is handled by an established automation procedure.

3. The roaster operator specifies maximum and minimum bounds on each of the roaster inputs, in this way, the roaster operator specifies constraints on the inputs.

4. At the initiation of the roast (after the priming of the roaster and the onset of control action by the roaster MPC) the LPG, rotation and blower inputs are initialised to be the same as during the priming phase to ensure bumpless transfer between the two phases. A bumpless transfer is commonly implemented in advanced process control applications when there is a transfer between the control action being calculated by the base-layer control (typically PID control) and the advanced layer (of which model predictive control is an example). This prevents a “bump” in the process inputs and subsequent process outputs (the controlled variable/s).

5. Once the priming is complete and the roaster operator releases the beans into the drum, the roast begins. The roaster MPC samples the measured bean temperature at each sampling interval. The difference between the predicted temperature and the measured temperature is passed as a feedback signal to the controller. The error signal is added to the model prediction at each iteration within the optimisation.

6. At each sample, a prediction is made P samples into the future based on the set of control inputs. The control inputs are initialised by assuming that the current input value remains unchanged P inputs into the future, i.e. if the current value of the LPG is 100%, the optimisation routine at each interval is initialised assuming that the LPG input remains unchanged throughout the control horizon. The same procedure is followed for the blower and rotation inputs.

7. The solution to the optimisation problem shown in Equation 7 is the set of inputs over the control horizon, M where $M \leq P$, which minimises the objective function over the prediction horizon. The objective function consists of the sum of the squared error between the predicted measured temperature and the set point temperature profile, as shown in Section 2.4.1. The inputs are constrained between a lower and an upper bound, while the absolute change in each input is restricted for all $j \leq P$.

6.4 Optimisation routine

The objective function of the optimisation problem in Equation 7 was simplified by removing the penalisation (weights) on input changes. This resulted in an objective function that was the sum of squared errors between the setpoint roaster temperature and the predicted roaster temperature over the prediction horizon. The motivation for simplifying the optimisation problem was to reduce the computational load in light of the fact that the optimisation was already nonconvex.

To solve the nonconvex optimisation problem mentioned in Section 2.4.1, Sequential Quadratic Programming (SQP) was proposed as a possible solution. SQP is a well-known optimisation algorithm that is often used to solve nonlinear optimisation problems with constraints. The algorithm works by solving a quadratic programming subproblem at each iteration, which is a convex optimisation problem that can be efficiently solved. However, the algorithm requires that the Hessian matrix of the objective function to be positive definite, which is not always the case for nonconvex problems (Camacho & Alba, 2013). A quadratic approximation of the objective function surface at each iteration is computationally expensive and will likely force suboptimal results to be obtained if

calculation time is a constraint. Alternatively, the size of the optimisation problem needs to be reduced to reduce the degrees of freedom.

Bayesian optimisation was chosen for real-time optimisation of the nonconvex objective function due to its suitability for computationally expensive objective functions and the author's familiarity with the technique during roaster model development. The method is effective for a wide range of applications, including tuning machine-learning models, hyperparameter optimisation, and black-box optimisation (Frazier, 2018b; Greenhill *et al*, 2020).

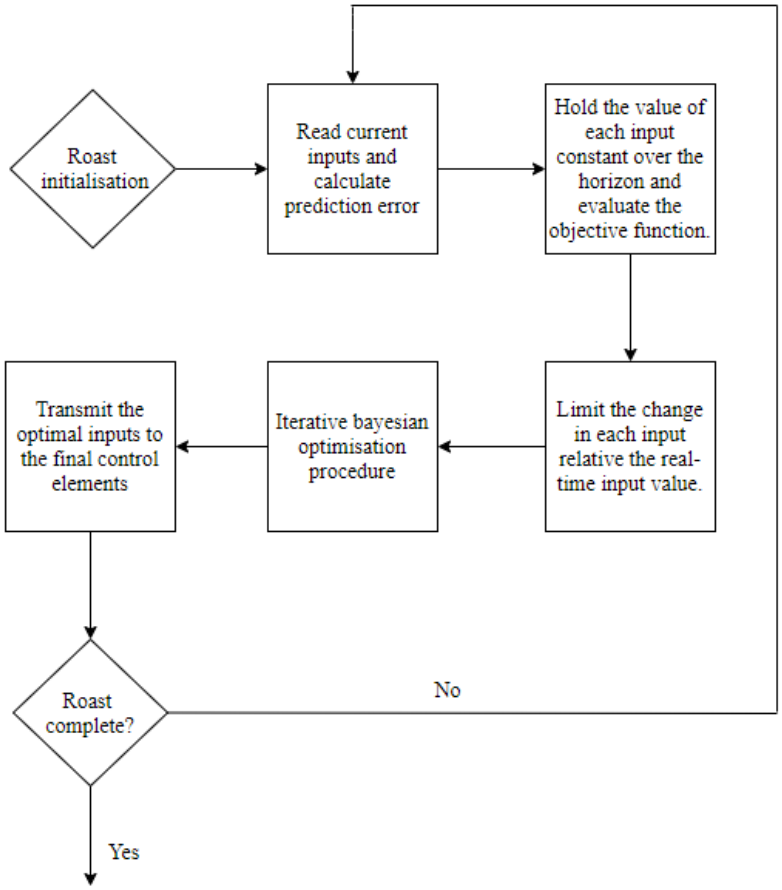


Figure 26: Optimisation routine framework.

To initialise the optimisation during each roast, an array of inputs equivalent to the prediction horizon length is populated, and the current values of the three inputs are held constant over the horizon. An initial objective function is calculated, and each input's value is limited to a specified change relative to the real-time input value. This differs from the strategy used by the current control system (see Section 2.3), where the change is limited relative to historical profile/input values. The input search space is implemented in PYTHON by creating a function that builds a PYTHON dictionary based

on the current state of each input, and the search space for each input is limited based on global upper and lower bounds defined by the operator.

Since Bayesian optimisation is used, the number of initial points to be sampled randomly from the objective function before approximating the objective function needs to be specified. Since the size of the optimisation problem varies depending on the selection of the prediction and control horizon, the total number of function evaluations and initial samples were scaled as a function of the prediction horizon. The total number of function calls was specified to be equivalent to the prediction horizon specification, so the number of samples scales directly with the optimisation degrees of freedom.

Once the solution to the optimisation problem is calculated, the optimal input is transmitted to the final control elements.

6.5 Controller tuning and simulation

This section describes the functional implementation of various components of the roaster model predictive controller. As discussed in Sections 4.2 and 5, the roaster model was developed with and without the Schwartzberg model prediction.

Section 6.3 highlighted that the control system consists of three functional elements, namely the roaster model, the roaster, and the optimiser. For controller tuning and simulation purposes, it is assumed that the roaster model is equivalent to the actual roaster, which means that there is no error in the roaster model prediction.

The process of tuning a model predictive controller (MPC) involves adjusting the controller parameters to improve its performance. In the case of the roaster MPC, the performance is assessed by comparing the predicted temperature response of the roaster to the target/setpoint temperature profile. This will be achieved by calculating the mean squared error (MSE) between the predicted temperature profile and the target. The MSE penalises outliers to a greater extent than the mean absolute error. More importantly, it is a differentiable function, which means that it can be used in gradient-based optimisation procedures (Bermejo & Cabestany, 2001).

It is worth noting that the selection of prediction and control horizons is a crucial step in MPC tuning, as it directly affects the aggressiveness of the control actions. Typically, shorter prediction horizons and longer control horizons lead to more aggressive control, while longer prediction horizons and shorter control horizons result in more conservative control (Seborg *et al*, 2011: 384). Over and above the effect of the tuning on the control

actions themselves, tuning has to be chosen such that the optimisation executes in less than one second. This is a requirement specified by the roaster system manufacturers.

By adjusting the horizons and other controller parameters, the MSE can be minimised, leading to improved controller performance. It is worth noting that the tuning process is iterative and may require multiple rounds of adjustment and testing to achieve improved performance.

The MPC was tuned on a single reference temperature profile shown in Figure 27 below. The results of the tuning procedure are discussed in Section 7. The average LPG, rotation and blower inputs were 63.9%, 60.0% and 21.9% respectively.

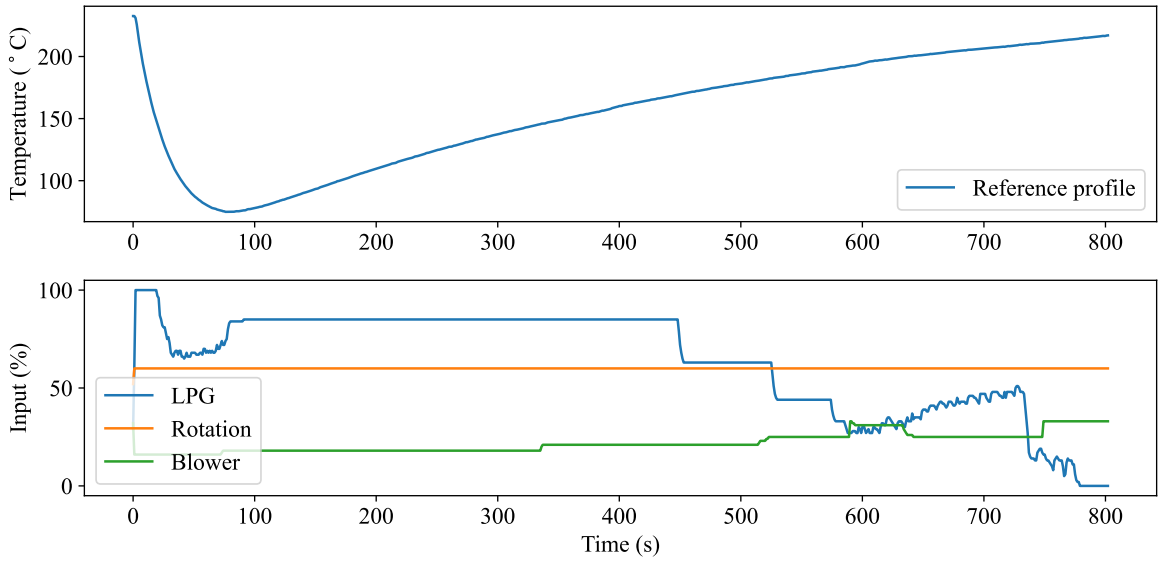


Figure 27: Reference profile used for tuning of the roaster MPC.

7 Controller implementation

7.1 Controller tuning

The MPC tuning process involved adjusting the prediction horizon (P) and the control horizon (M) while limiting the change in inputs relative to the real-time input value. To evaluate controller performance, the mean squared error (MSE) between the predicted temperature profile and the target was used. The results are presented in Appendix A.2 as a summary in tabular form and as 3-dimensional surface plots in this section. The colour bar on the right side of the figures maps the calculated MSE values to a colour scale, making it easy to visualise the performance over different combinations of prediction and control horizons.

In the first round of tuning efforts, the change in LPG, blower, and rotation input was limited to an absolute change of 2%, 1%, and 1%, respectively. The best result achieved was a mean squared error of 7.32 $°C^2$ for a prediction horizon (P) and a control horizon (M) of 8 and 2 sampling intervals, respectively, as shown in Table A.12. Although Figure 28 suggests that shorter control horizons are favoured, this is not a reliable assessment based on the saddle shape of the surface plot shown in Figure 28. Heuristically, shorter control horizons achieving superior performance would suggest that conservative controller actions are being favoured. In subsequent tuning simulations, the maximum change in the LPG, blower, and rotation inputs was increased to determine whether the controller was favouring more significant controller moves.

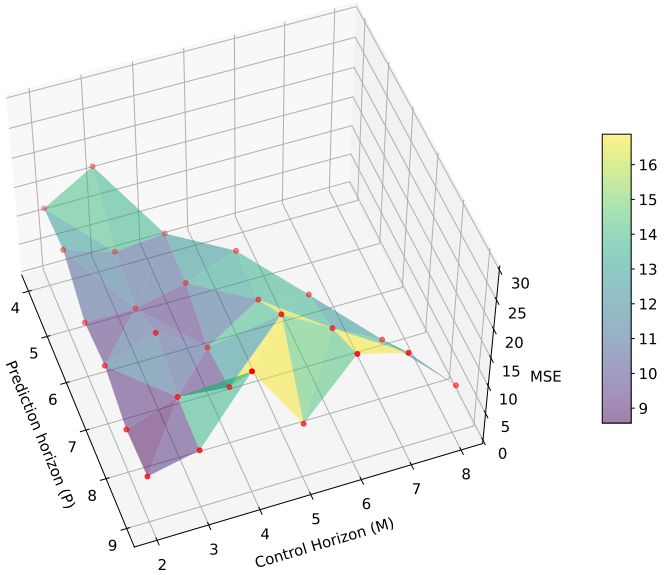


Figure 28: MSE surface plot for a maximum change of LPG, blower, and rotation inputs of 2%, 1%, and 1% respectively (elevation of 50° and azimuth of 340°).

In the second round of tuning efforts, the change in the LPG, blower, and rotation input was limited to an absolute change of 3%, 2%, and 2%, respectively. The best result achieved was a mean squared error of 5.59 °C for a prediction horizon (P) and a control horizon (M) of 5 and 3 sampling intervals, respectively, as shown in Table A.13. Increasing the allowable input changes improved the best mean squared error that was achieved and resulted in a reduction in the average usage of LPG and blower input. The shape of the 3-dimensional surface plot flattened when comparing Figures 28 and 29, indicating improved robustness.

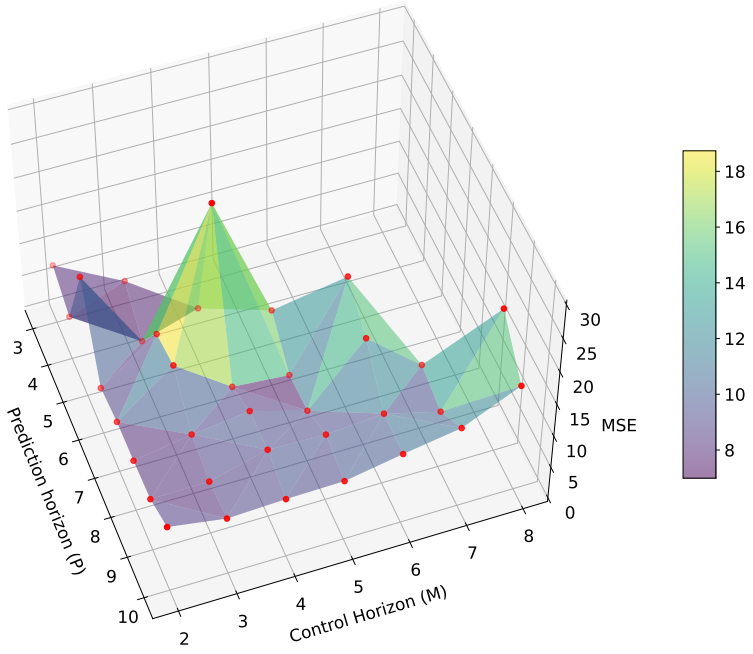


Figure 29: MSE surface plot for a maximum change of LPG, blower, and rotation inputs of 3%, 2%, and 2% respectively (elevation of 50° and azimuth of 340°).

As a result of the improved performance when increasing the allowable change in the inputs at each iteration, tuning simulations were performed at a maximum absolute change in the LPG, blower, and rotation inputs of 5%, 3% and 3% as well as of 7%, 5%, and 5% respectively. The recorded data are tabulated in Tables A.14 and A.15 and visualised in Figures 30 and 31.

Further tuning simulations were performed with a maximum absolute change in the LPG, blower, and rotation inputs of 5%, 3%, and 3%, as well as 7%, 5%, and 5%, respectively. The recorded data are tabulated in Tables A.14 and A.15 and visualised in Figures 30 and 31. A further flattening of the surface plot is observed in Figure 30. The mean squared error (MSE) and interquartile range (IQR) of the MSE for each of the tuning configurations were used to quantify this flattening.

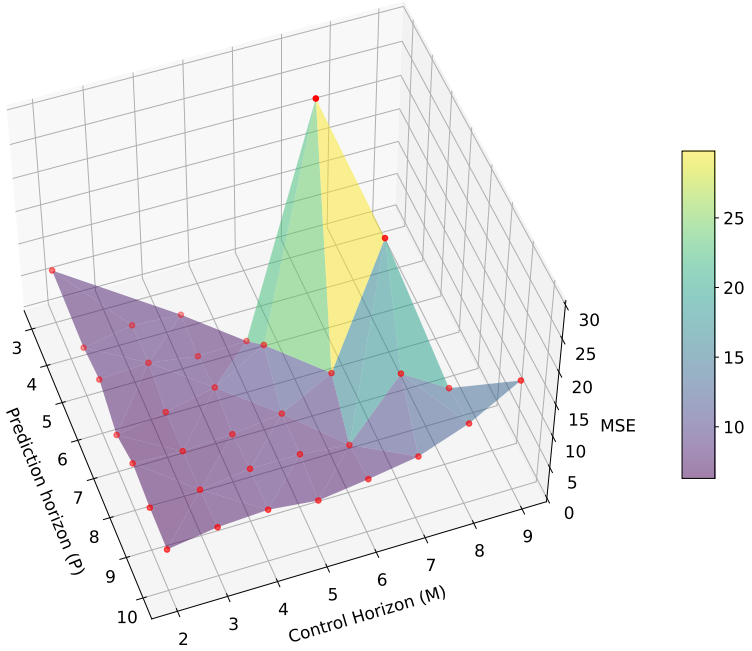


Figure 30: MSE surface plot for a maximum change of LPG, blower, and rotation inputs of 5%, 3%, and 3% respectively (elevation of 50° and azimuth of 340°).

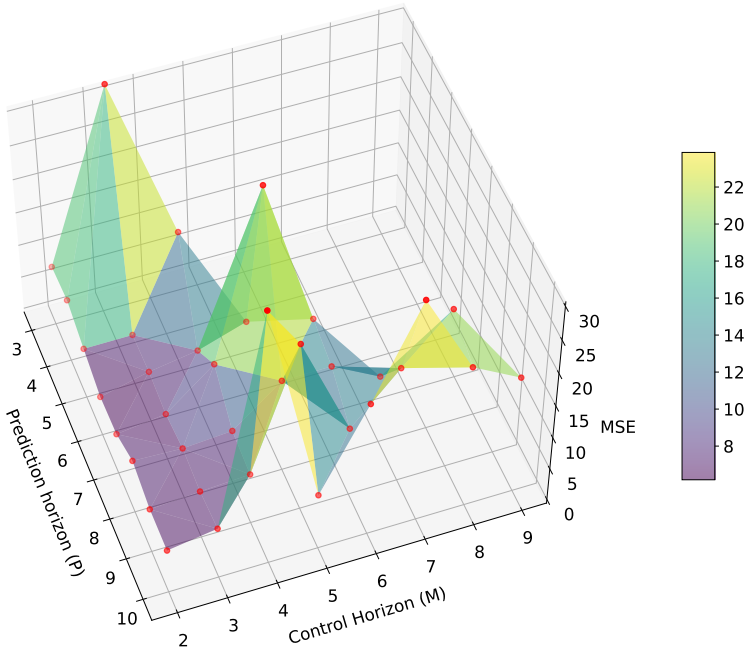


Figure 31: MSE surface plot for a maximum change of LPG, blower, and rotation inputs of 7%, 5%, and 5% respectively (elevation of 50° and azimuth of 340°).

7.2 Controller tuning discussion

The box plot of the MSE for each tuning configuration is shown in Figure 32. For a maximum input change of 2%, 1%, and 1%, a mean MSE of 12.28 °C^2 and an IQR of 4.810 °C^2 was achieved. When increasing the maximum input change to 3%, 2%, and 2%, the mean MSE was reduced to 10.92 °C^2 and the IQR was reduced to 3.830 °C^2. A marginal reduction in the mean MSE to 10.34 °C^2 was observed when increasing the maximum input change to 5%, 3%, and 3%, while the IQR showed a significant reduction to 1.79 °C^2. This iterative reduction in the MSE IQR is visible in Figure 32 and quantifies the flattening of the surface plots. A further increase in the maximum input change to 7%, 5%, and 5% resulted in a performance degradation. A mean MSE of 14.13 °C^2 and an IQR of 10.74 °C^2 is achieved. Notice that the erratic behaviour is illustrated for larger control horizons in Figure 31, leading to the conclusion that the higher maximum input change significantly increased the complexity of the optimisation problem.

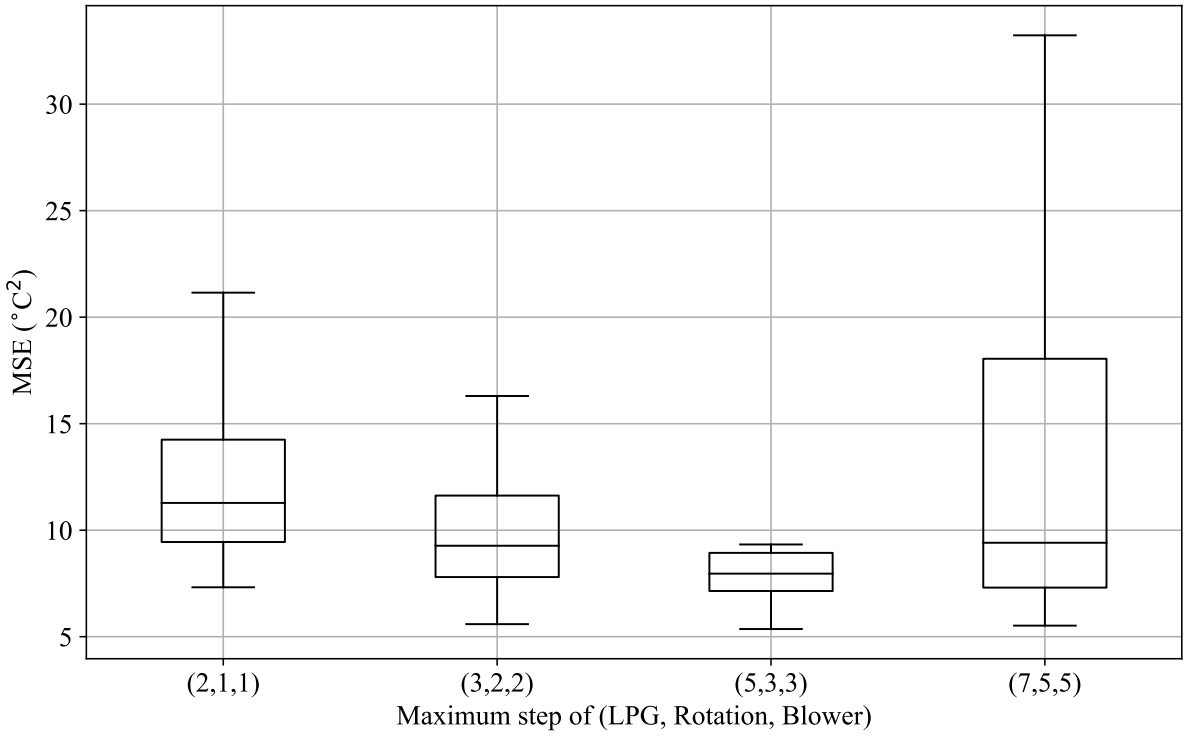


Figure 32: A box plot of the mean squared error (MSE) observed when varying the maximum input change of the LPG, Rotation and blower inputs (outliers removed).

Overall, the results indicate that increasing the allowable input changes during the tuning process improved controller performance, as seen through the reduction in mean squared error (refer to Section 6.5) and input usage. The relationship between prediction and control horizons was found to be complex, with shorter control horizons sometimes providing superior performance. The flattening of the performance surface with increasing input changes suggests that the controller is becoming more robust and further adjustments do not significantly improve the performance of the controller. **Further experimentation** could explore the relationship between the controller's performance and other tuning parameters, such as the weighting factors in the cost function.

The optimal tuning configuration is selected based not only on the controller's performance in relation to the reference temperature profile, but also on utility usage. To illustrate this, box plots were developed to display the distribution of average utility usage (input usage) per simulation, calculated by the MPC. A red line is also plotted in each figure to represent the average input usage of the reference profile, as demonstrated in Figure 27. Furthermore, the distribution of the average input usage per roast from the historical database is included for comparison. These comparisons can be seen in Figures 33, 34, and 35.

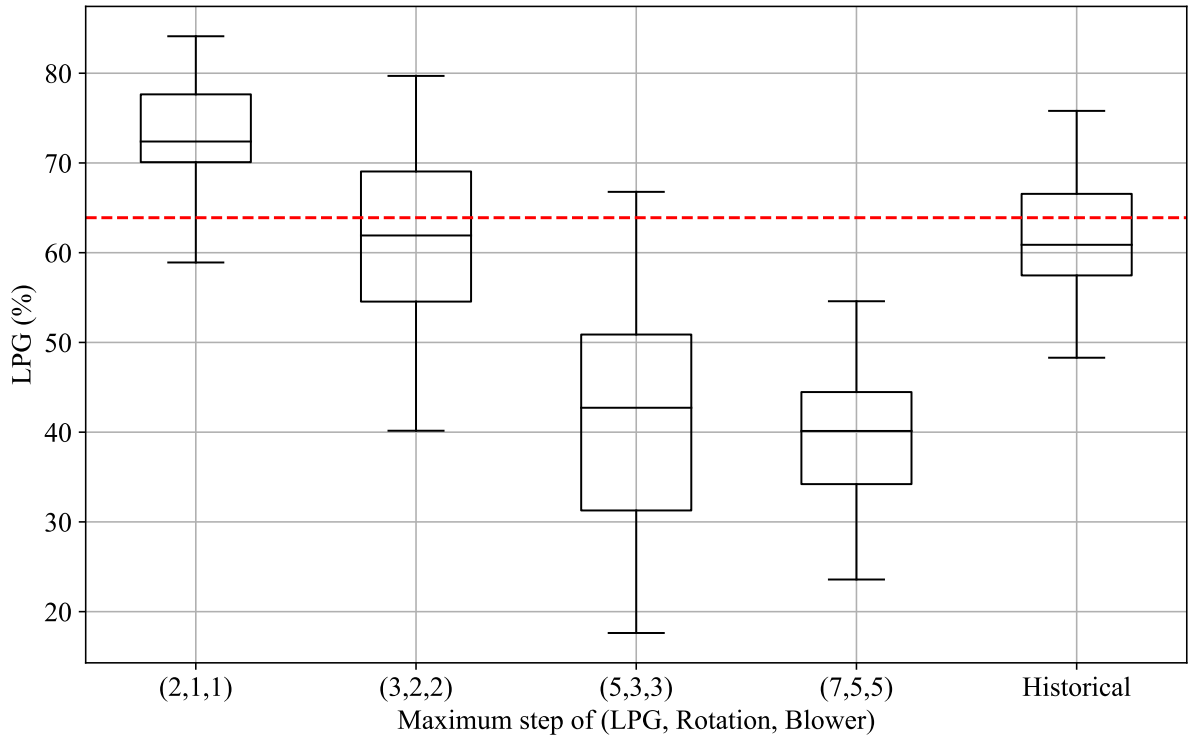


Figure 33: A boxplot comparing LPG usage when varying the maximum input change of the LPG, rotation and blower inputs.

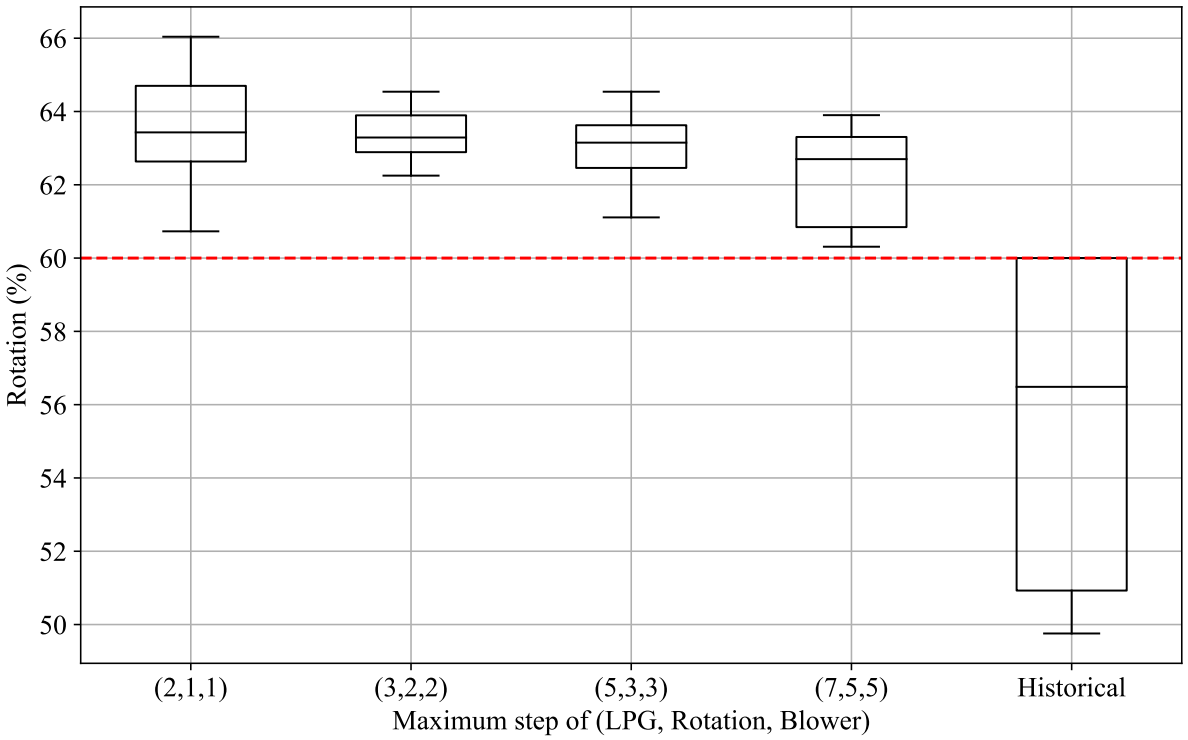


Figure 34: A boxplot comparing rotation usage when varying the maximum input change of the LPG, rotation and blower inputs.

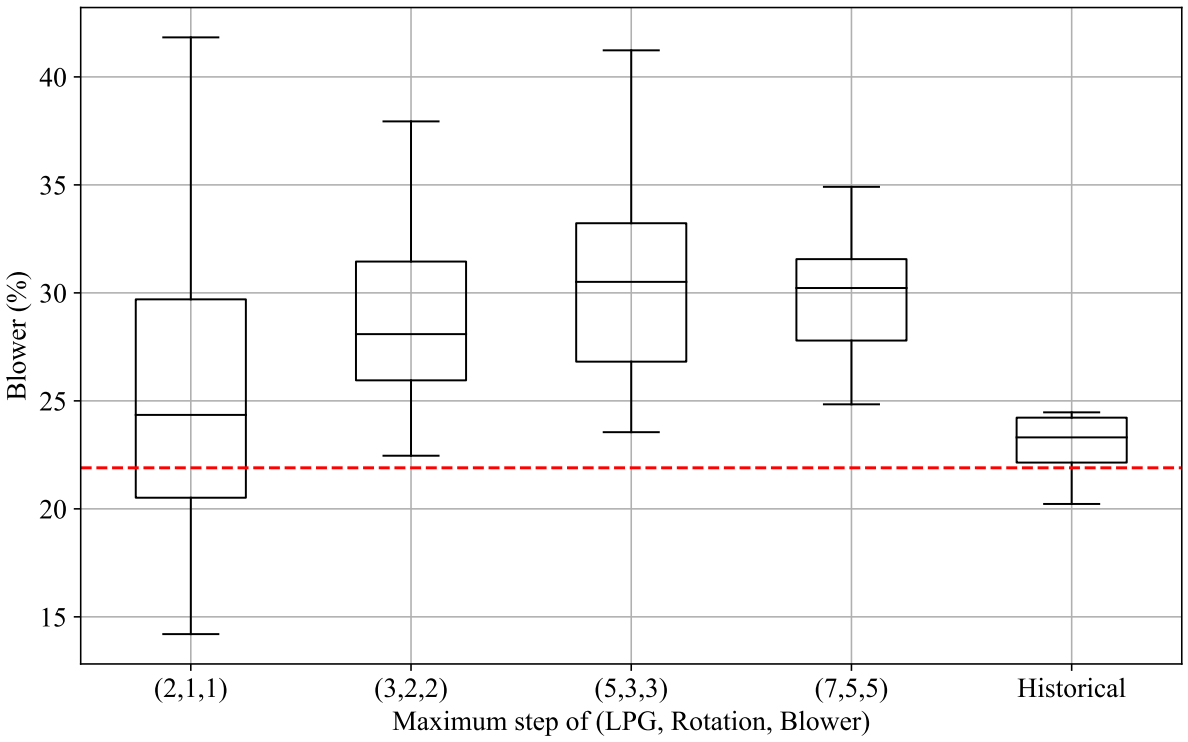


Figure 35: A boxplot comparing blower usage when varying the maximum input change of the LPG, rotation and blower inputs.

Taking Figure 33 into account, recall that the optimal MPC configuration achieved the best performance, specifically a maximum absolute change in the LPG, blower, and rotation inputs of 5%, 3%, and 3%. This (5,3,3) configuration resulted in a standard deviation in the average LPG usage of 12.7% as opposed to the standard deviation in average LPG usage of 7.33% for the historical LPG usage. Additionally, the (5,3,3) configuration achieved an average of 41.9% in the average usage of LPG, which is considerably lower than the average of 62.3% in the average historical use of LPG. These results suggest that average LPG usage can be reduced while maintaining a large input range (using the full extent of the input).

Regarding Figure 34, a historical average rotational input of 55.5% and a standard deviation of 4.71% were observed. The (5,3,3) configuration reached an average of 62.8% and a standard deviation of 1.42% in the average rotational input.

Moving on to Figure 35, a historical average blower input of 24.1% and a standard deviation of 3.91% were noted. The (5,3,3) configuration achieved an average of 30.7% and a standard deviation of 4.76% in the average blower input.

As expected, a reduction in average LPG usage led to an increase in average blower input to ensure that the same amount of heat is transferred to the beans. Interpreting why the average rotational input also increased is more challenging. The effect of the rotational input is non-linear; at slow rotational speeds, the beans spend a considerable amount of time in contact with the drum, which may result in scorching. Conversely, at exceptionally high rotational speeds, the centrifugal force exerted on the beans becomes equivalent to the bean mass, leading to scorching as well. Within normal operating ranges, as the speed of rotation increases, the beans are flailed through the air at a faster rate. In this situation, convective heat transfer plays a larger role in roasting the beans as a result of the reduced contact with the drum.

7.3 Controller tuning conclusions

In conclusion, the results of the model predictive controller's tuning, as demonstrated in Figures 33, 34, and 35, highlight the significance of meticulously selecting the MPC configuration to balance optimising utility usage and maintaining the desired bean roasting quality. The optimal (5,3,3) configuration reveals the potential for more efficient and consistent control of the coffee roasting process. The relationship between prediction and control horizons proved to be intricate, with shorter control horizons occasionally providing superior performance. The flattening of the performance surface with increasing input changes suggests that the controller is becoming more robust and further adjustments do not significantly improve the performance of the controller.
Further research could investigate the impact of these changes on the final quality and taste of the roasted beans, as well as examine the robustness of the controller under varying operating conditions and potential disturbances. Additionally, future experimentation could explore the relationship between the controller's performance and other tuning parameters, such as the weighting factors in the cost function.

To determine the value add of the advanced modelling and control methods discussed, traditionally one would compare the performance of such an implementation against a baseline. Typically, a baseline in such a instance would be the performance of a PID controller, where the performance would be assessed based on the ability of the controller to maintain setpoint as well as consider utility usage to achieve the desired setpoint. Unfortunately, PID controller performance is not historised within the roaster software solution and this is why performance of the model predictive controller is quantified in terms of the historical utilities usage.

It is worth noting that while the model predictive controller was able to achieve improvements in terms of utilities usage, the optimal utilities were sometimes calculated at values outside the normal operating ranges of the historical usage. It can be concluded that an improvement in utilities usage has been achieved but this should be confirmed in series of real time implementations. This is primarily due to the tendency of empirical models to degrade in performance under extrapolated conditions albeit that significant effort is taken in the modelling stages to improve the generalisation of the developed models. It is worth noting that this consideration is not only taken in the application of empirical models within a MPC framework but also in the extrapolation of linear dynamic models to unknwon operating regions. This is an ongoing field of study and the robustness of empirical models within a MPC is a new and exciting field of study.

8 Conclusions and recommendations

This study aimed to design and implement a model predictive controller for a coffee roasting process that would optimise the roasting quality of the coffee beans while minimising energy consumption by manipulating all available process inputs, namely the LPG, rotation, and blower inputs. The controller's performance was evaluated through a series of simulations that involved adjusting the prediction and control horizons while limiting the change in inputs relative to the real-time input value.

The results of the tuning process showed that increasing the allowable input changes during the tuning process improved the controller's performance, as seen through the reduction in mean squared error and input usage. The relationship between prediction and control horizons was found to be complex, with shorter control horizons sometimes providing superior performance. The flattening of the performance surface with increasing input changes suggests that the controller is becoming more robust and further adjustments do not significantly improve the performance of the controller.

The optimal (5,3,3) configuration achieved the best performance, specifically a maximum absolute change in the input of the LPG, blower and rotation of 5%, 3%, and 3%. This configuration resulted in a standard deviation in the average LPG usage of 12.7% as opposed to the standard deviation in the average LPG usage of 7.33% for historical LPG usage. Additionally, the (5,3,3) configuration achieved an average of 41.9% in the average usage of LPG, which is considerably lower than the average of 62.3% in the average historical use of LPG. These results suggest that average LPG usage can be reduced while maintaining a large input range (using the full extent of the input).

The impact of the intelligent modelling and control system on the reduction of raw material waste, the improvement of the quality of the final product, and the overall efficiency of the roasting process was evaluated, showing significant improvements in all three areas. The proposed system enables operators to perform roast simulations and reduce raw material waste when developing roast profiles, providing a valuable contribution to the coffee roasting industry, particularly for the local South African coffee roaster manufacturer.

The final paragraphs of the section concluding the controller tuning, Section 7.3, make important comments regarding the limitations of simulation based study. Specifically, it highlights the requirement to assess the performance of the developed models in real time implementations in addition to the assessing the performance gain to be had from implementing the developed models within a MPC framework. While simulation studies are an important step in the design and development of process control implementations, questions around the generalisation of the developed models and the MPC to the en-

tire operating range of the roaster can only be answered through a series of real-world applications.

The results of this study demonstrate the potential for nonlinear MPC for temperature control of rotating drum roasters. However, there are several routes for future work and improvements:

- Nonlinear MPC utilising hybrid modelling: the performance enhancements to be had by combining empirical and first-principles modelling have been demonstrated. The adapted Schwartzberg model should be extended to include an energy balance on the heated air inlet so that the LPG input can be directly manipulated within the prediction horizon. Future research could explore the use of other physics-informed modelling techniques or investigate alternative machine learning approaches to further improve the modelling and control of coffee roasters.
- Nonlinear optimisation: the optimiser used to optimise the nonconvex MPC objective function is slow, since it is non-gradient based. Faster gradient-based optimisation techniques, such as sequential quadratic programming (SQP), can be investigated as a solution to the optimisation of the nonconvex optimisation problem. Additionally, penalisation (weights) of the input movement can be included in the model predictive control objective function.
- Real-world application: Significant effort should be taken to implement the developed models and the MPC in a real-world setting to determine whether the developed models generalise and the improvements in utilities usage can be obtained as reported.

By exploring these recommendations, the potential of MPC for rotary drum roasting temperature control can be further realised, leading to improved energy efficiency, product quality, and overall process optimisation. This research serves as a stepping stone for the development of more advanced control strategies that can be applied to various industrial drying processes.

References

Anderson, TW and Rubin, H (1949), "Estimation of the parameters of a single equation in a complete system of stochastic equations", *The Annals of Mathematical Statistics*, 20: 46–63.

Bermejo, S and Cabestany, J (2001), "Oriented principal component analysis for large margin classifiers", *Neural Networks*, 14 (10): 1447–1461.

Bishop, CM (2006), *Pattern Recognition and Machine Learning*, 2nd ed., Springer, 233 Spring Street, New York, NY 10013, USA, ISBN: 978-0387-31073-2.

Bolt, CE and Vaal, PL de (2022), "A Practical Guide to Coffee Roaster Modelling", in: *Computer Aided Chemical Engineering*, vol. 51, Elsevier: pp. 145–150.

Borase, RP, Maghade, D, Sondkar, S and Pawar, S (2021), "A review of PID control, tuning methods and applications", *International Journal of Dynamics and Control*, 9 (2): 818–827.

Botha, CM (2018), "A model-based control system design for a coffee roasting process", dissertation, North-West University, Potchefstroom Campus.

Breiman, L (1997), *Arcing the edge*, tech. rep., Technical Report 486, Statistics Department, University of California at Berkeley: pp. 1–14.

Breiman, L (1996), "Bagging predictors", *Machine learning*, 24: 123–140.

Breiman, L (2001), "Random Forests", *Machine learning*, 45: 5–32.

Breiman, L, Friedman, JH, Olshen, RA and Stone, CJ (1984), "Classification and regression trees Belmont", *CA: Wadsworth International Group*,

Brownlee, J (2018), *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*, Machine Learning Mastery, URL: https://books.google.co.za/books?id=T1-nDwAAQBAJ.

Camacho, EF and Alba, CB (2013), *Model Predictive Control*, Springer London, ISBN: 978-1-85233-694-3, DOI: https://doi.org/10.1007/978-0-85729-398-5.

Chatterjee, M (2022), *Top 20 Applications of Deep Learning in 2022 Across Industries*, URL: https://www.mygreatlearning.com/blog/deep-learning-applications/#cars (visited on 05/22/2022).

Cutler, CR and Ramaker, BL (1980), "Dynamic matrix control?? A computer control algorithm", paper presented at *Joint Automatic Control Conference*: p. 72.

Di Palma, F, Iancono, F, Toffanin, C, Ziccardi, A and Magni, L (2021), "Scalable model for industrial coffee roasting chamber", *Procedia Computer Science*, 180: 122–131.

Dobbelaere, MR, Plehiers, PP, Van de Vijver, R, Stevens, CV and Van Geem, KM (2021), "Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats", *Engineering*, 7 (9): 1201–1211.

Drucker, H (1997), "Improving regressors using boosting techniques", paper presented at *ICML*, vol. 97, Citeseer: pp. 107–115.

Frazier, PI (2018a), "A tutorial on Bayesian optimization", *arXiv preprint arXiv:1807.02811*,

Frazier, PI (2018b), "Bayesian optimization", in: *Recent Advances in Optimization and Modeling of Contemporary Problems*, Informs: pp. 255–278.

Freund, Y and Schapire, RE (1997), "A Decision-Theoretic Generalisation of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences*, 55: 119–139.

Garcia, CE and Morari, M (1982), "Internal model control. A unifying review and some new results", *Industrial & Engineering Chemistry Process Design and Development*, 21 (2): 308–323.

Géron, A (2019), *Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow: Concepts, Tools and Techniques to Build Intelligent Systems*, 2nd ed., O'Reilly, Sebastopol, CA 95472, ISBN: 978-1-492-03264-9.

Glorot, X and Bengio, Y (2010), "Understanding the difficulty of training deep feedforward neural networks", paper presented at *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings: pp. 249–256.

Greenhill, S, Rana, S, Gupta, S, Vellanki, P and Venkatesh, S (2020), "Bayesian optimization for adaptive experimental design: a review", *IEEE Access*, 8: 13937–13948.

Hajek, A *et al* (2019), "Interpretations of Probability", *Stanford Encyclopedia of Philosophy*, URL: https://plato.stanford.edu/ENTRIES/probability-interpret/.

Hastie, T, Tibshirani, R and Friedman, J (2001), *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA.

Hastie, T, Tibshirani, R, Friedman, J, Hastie, T, Tibshirani, R and Friedman, J (2009), "Linear methods for regression", *The elements of statistical learning: Data mining, inference, and prediction*, 43–99.

Hernández, J, Heyd, B, Irles, C, Valdovinos, B and Trystram, G (2007), "Analysis of the heat and mass transfer during coffee batch roasting", *Journal of Food Engineering*, 78 (4): 1141–1148.

Hochreiter, S, Bengio, Y, Frasconi, P, Schmidhuber, J, *et al* (2001), "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies", in: *A Field Guide to Dynamical Recurrent Neural Networks*, Kremer, SC and Kolen, JF (Eds.), IEEE Press, ISBN: 0-7803-5369-2.

Horn, RA and Garcia, SR (2020), "Block Matrices in Linear Algebra", *PRIMUS*, 30: 285–306.

Iacono, F (2023), "LSTM neural networks for industrial and biomedical applications", phdthesis, Università degli studi di Pavia.

Kellher, JD (2019), *Deep Learning*, 2nd ed., The MIT Press, 1 Broadway, Cambridge, Massachusetts, ISBN: 978-0262-53755-1.

Keras (2022), *Layer activation functions*, URL: https://keras.io/api/layers/activations/ (visited on 06/26/2022).

Manning-Smith, T and Mustain, A (2019), *How Video Games Help Fuel The Insatiable Demand For Artificial Intelligence*, URL: https://www.forbes.com/sites/sap/2019/02/14/how-video-games-help-fuel-the-insatiable-demand-for-artificial-intelligence/?sh=261db6ba6f1b (visited on 05/22/2022).

Marlin, TE (2000), *Process Control: Designing Processes and Control Systems for Dynamic Performance*, McGraw-Hill Education, ISBN: 978-0070-39362-2.

McCulloch, WS and Pitts, W (1943), "A logical calculus of the ideas immanent in nervous activity", *The Bulletin of Mathematical Biophysics*, 5 (4): 115–133.

Mitchell, T (1997), *Machine Learning*, 1st ed., McGraw-Hill Education, United States of America.

Pomerleau, A, Desbiens, A, Hodouin, D, *et al* (1996), "Development and evaluation of an auto-tuning and adaptive PID controller", *Automatica*, 32 (1): 71–82.

Putranto, A and Chen, XD (2012), "Roasting of barley and coffee modeled using the lumped-reaction engineering approach (L-REA)", *Drying Technology*, 30 (5): 475–483.

Rao, S (2014), *The Coffee Roaster's Companion*, ISBN: 978-1-4951-1819-7.

Richalet, J, Rault, A, Testud, J and Papon, J (1978), "Model predictive heuristic control: Applications to industrial processes", *Automatica*, 14 (5): 413–428.

Rosenblatt, F (1958), "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological Review*, 65 (6): 386.

Schwartzberg, HG (2002), "Modeling Bean Heating during Batch Roasting of Coffee Beans", in: *Engineering and Food for the 21st Century*, Welti-Chanes, J and Aguilera, JM (Eds.), 1st ed., CRC Press, United States of America, chap. 52: pp. 863–882.

Scikit-learn (2022a), *sklearn.ensemble.AdaBoostRegressor*, URL: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html#sklearn.ensemble.AdaBoostRegressor` (visited on 04/13/2022).

Scikit-learn (2022b), *sklearn.ensemble.AdaBoostRegressor*, URL: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html#sklearn.ensemble.AdaBoostRegressor` (visited on 04/13/2022).

Scikit-learn (2022c), *sklearn.ensemble.BaggingRegressor*, URL: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegressor` (visited on 04/13/2022).

Scikit-learn Decision Tree Regressor (2022), *sklearn.tree.DecisionTreeRegressor*, URL: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html` (visited on 04/03/2022).

Scikit-learn Linear Models (2022), *1.1 Linear Models*, URL: `https://scikit-learn.org/stable/modules/linear_model.html` (visited on 03/14/2022).

Seborg, DE (1987), "The prospects for advanced process control", *IFAC Proceedings Volumes*, 20 (5): 281–289.

Seborg, DE, Mellichamp, DA, Edgar, TF and Doyle III, FJ (2011), *Process dynamics and control*, 3rd ed., John Wiley & Sons, ISBN: 978-0-470-64610-6.

Simon, D (2006), *Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches*, Wiley, ISBN: 978-0-47170-858-2, DOI: `https://doi.org/10.1002/0470045345`.

Srivastava, N, Hinton, G, Krizhevsky, A, Sutskever, I and Salakhutdinov, R (2014), "Dropout: a simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, 15 (1): 1929–1958.

Torun, HM, Swaminathan, M, Davis, AK and Bellaredj, MLF (2018), "A global Bayesian optimization algorithm and its application to integrated system design", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26 (4): 792–802.

Vosloo, J (2017), "Heat and mass transfer model for a coffee roasting process", phdthesis, North-West University, Potchefstroom Campus.

Yang, N, Liu, C, Liu, X, Degn, TK, Munchow, M and Fisk, I (2016), "Determination of volatile marker compounds of common coffee roast defects", *Food Chemistry*, 211: 206–214.

Yao, X, Panaye, A, Doucet, JP, Zhang, R, Chen, H, Liu, M, Hu, Z and Fan, BT (2004), "Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression", *Journal of Chemical Information and Computer Sciences*, 44 (4): 1257–1266.

Zeger, SL, Thomas, D, Dominici, F, Samet, JM, Schwartz, J, Dockery, D and Cohen, A (2000), "Exposure measurement error in time-series studies of air pollution: concepts and consequences." *Environmental Health Perspectives*, 108 (5): 419–426.

A Appendices

A.1 Empirical model optimisation results

This appendix section summarises the final structure of each model developed for the 6 kg, 15 kg and 30 kg roasters. The structure of each model type was defined in Section 4.2. The optimised structure of each model was determined through Bayesian optimisation as presented in Section 2.5.1. Since linear models are a linear sum of the weighted inputs, and were not of significance from a performance perspective, the structure of the linear models are not reported i.e. the weights of the individual inputs.

A.1.1 Decision tree regressor (DTR)

Table A.9: Optimised decision tree structure for all roaster sizes.

Model	Max depth of the tree
6 kg DTR	16.00
6 kg PI DTR	21.00
15 kg DTR	17.00
15 kg PI DTR	18.00
30 kg DTR	17.00
30 kg PI DTR	19.00

A.1.2 Random forest regressor (RFR)

Table A.10: Optimised random forest structure for all roaster sizes.

Model	Max depth	Number of trees
6 kg RFR	18.00	199.0
6 kg PI RFR	24.00	150.0
15 kg RFR	17.00	217.0
15 kg PI RFR	21.00	135.0
30 kg RFR	18.00	203.0
30 kg PI RFR	22.00	162.0

A.1.3 Neural network regressor (NN)

Table A.11: Optimised neural network structure for all roaster sizes.

Model	Droprate	Neurons/layer	Hidden layers	Learning rate
6 kg NN	0.8310	54.00	22.00	3.898×10^{-4}
6 kg PI NN	0.6798	98.00	23.00	5.126×10^{-4}
15 kg NN	0.7382	63.00	35.00	7.633×10^{-4}
15 kg PI NN	0.6171	88.00	42.00	9.344×10^{-4}
30 kg NN	0.5965	81.00	41.00	1.916×10^{-3}
30 kg PI NN	0.3200	99.00	5.000	1.355×10^{-3}

A.2 Roaster model predictive controller tuning

Table A.12: Roaster model predictive controller tuning for a maximum percentage change in LPG, rotation and blower inputs of 2%, 1% and 1% respectively.

P	M	MSE	LPG	Rotation	Blower	Execution
4.0	2.0	13.19	77.20	65.97	26.30	0.20
4.0	3.0	18.11	74.83	65.83	29.46	0.22
5.0	2.0	13.88	80.60	61.52	41.83	0.26
5.0	3.0	11.12	70.93	62.42	41.57	0.30
5.0	4.0	11.90	74.33	63.56	29.88	0.31
6.0	2.0	9.24	71.76	62.12	24.72	0.51
6.0	3.0	9.27	71.59	64.89	20.06	0.58
6.0	4.0	11.28	75.95	63.25	21.77	0.59
6.0	5.0	14.44	84.12	63.66	16.85	0.61
7.0	2.0	10.05	72.39	64.51	47.57	0.60
7.0	3.0	13.30	80.18	63.43	23.91	0.70
7.0	4.0	8.12	29.93	62.32	24.35	0.71
7.0	5.0	14.06	82.03	65.64	29.52	0.73
7.0	6.0	12.44	78.18	63.50	20.58	0.74
8.0	2.0	7.32	67.55	60.73	35.26	0.77
8.0	3.0	10.53	73.20	66.04	23.57	0.82
8.0	4.0	9.62	71.42	61.42	25.38	0.84
8.0	5.0	19.85	62.96	63.26	20.45	0.86
8.0	6.0	14.88	83.28	65.91	17.22	0.87
8.0	7.0	10.30	71.94	64.00	21.19	0.90
9.0	2.0	7.87	68.50	65.19	23.25	1.00
9.0	3.0	9.88	70.60	61.48	34.91	1.04
9.0	4.0	21.15	67.75	62.90	14.20	1.08
9.0	5.0	9.19	73.12	63.84	24.51	1.10
9.0	6.0	18.92	69.58	62.85	16.63	1.10
9.0	7.0	16.51	78.09	63.36	17.40	1.12
9.0	8.0	8.07	58.91	62.85	47.45	1.14

Table A.13: Roaster model predictive controller tuning for a maximum percentage change in LPG, rotation and blower inputs of 3%, 2% and 2% respectively.

P	M	MSE	LPG	Rotation	Blower	Exec
3.0	2.0	8.45	40.16	64.35	32.85	0.15
4.0	2.0	6.11	55.11	63.37	39.98	0.22
4.0	3.0	9.27	50.18	63.03	27.84	0.22
5.0	2.0	18.08	55.71	63.13	24.83	0.28
5.0	3.0	5.59	55.82	62.92	25.22	0.30
5.0	4.0	8.31	61.92	63.90	29.55	0.33
6.0	2.0	6.68	54.97	63.22	33.45	0.36
6.0	3.0	12.65	79.70	63.25	26.33	0.38
6.0	4.0	29.83	68.98	59.22	30.71	0.40
6.0	5.0	11.35	32.61	61.07	26.18	0.40
7.0	2.0	7.42	66.01	63.73	30.43	0.45
7.0	3.0	13.74	67.21	62.60	25.72	0.48
7.0	4.0	7.80	42.28	63.49	26.75	0.49
7.0	5.0	7.09	41.39	63.09	28.42	0.50
7.0	6.0	19.87	56.38	63.29	22.46	0.53
8.0	2.0	7.47	63.32	64.33	37.94	0.56
8.0	3.0	9.00	61.41	62.83	27.31	0.60
8.0	4.0	10.06	72.55	62.86	28.21	0.61
8.0	5.0	7.51	44.19	63.89	26.79	0.61
8.0	6.0	16.30	73.17	62.25	27.36	0.64
8.0	7.0	9.59	60.20	64.16	37.37	0.65
9.0	2.0	7.71	69.65	62.73	32.32	0.72
9.0	3.0	7.80	67.40	64.54	23.74	0.87
9.0	4.0	10.18	66.63	64.07	36.22	0.84
9.0	5.0	9.92	67.17	63.51	28.09	0.83
9.0	6.0	10.60	65.30	63.38	25.26	0.80
9.0	7.0	8.29	46.64	64.00	29.19	0.78
9.0	8.0	21.69	72.81	63.54	23.14	0.80
10.0	2.0	9.77	71.42	63.46	42.38	0.80
10.0	3.0	8.34	69.12	62.95	24.20	0.83
10.0	4.0	8.69	71.05	62.65	26.39	0.86
10.0	5.0	8.85	58.71	64.35	29.06	0.87
10.0	6.0	10.46	71.67	63.09	32.19	0.89
10.0	7.0	11.90	40.91	60.72	29.52	0.89
10.0	8.0	15.87	54.13	64.02	22.99	0.91

Table A.14: Roaster model predictive controller tuning for a maximum percentage change in LPG, rotation and blower inputs of 5%, 3% and 3% respectively.

P	M	MSE	LPG	Rotation	Blower	Exec
3.0	2.0	7.70	30.61	63.36	24.36	0.15
3.0	2.0	7.70	30.61	63.36	24.36	0.15
5.0	2.0	7.13	61.82	62.97	36.91	0.28
5.0	3.0	8.52	25.38	62.33	30.03	0.30
5.0	4.0	8.06	50.72	63.12	27.80	0.31
6.0	2.0	8.08	66.78	64.32	41.23	0.37
6.0	3.0	8.54	23.71	63.23	25.69	0.38
6.0	4.0	7.39	35.63	63.41	27.43	0.39
6.0	5.0	7.64	39.44	61.97	26.20	0.41
7.0	2.0	5.36	57.74	63.65	28.02	0.46
7.0	3.0	6.78	39.04	63.03	36.34	0.47
7.0	4.0	8.48	20.07	62.83	30.10	0.48
7.0	5.0	12.97	30.70	64.06	23.55	0.51
7.0	6.0	46.47	17.62	57.04	31.75	0.52
8.0	2.0	7.08	60.80	61.11	31.14	0.57
8.0	3.0	6.75	46.70	63.78	31.52	0.59
8.0	4.0	7.19	31.09	63.15	33.51	0.59
8.0	5.0	8.22	51.04	64.32	34.03	0.62
8.0	6.0	12.39	44.50	63.69	35.71	0.64
8.0	7.0	30.54	34.23	60.49	32.58	0.65
9.0	2.0	6.38	64.86	63.29	30.51	0.69
9.0	3.0	6.92	49.35	62.90	38.41	0.71
9.0	4.0	7.96	40.72	60.64	25.05	0.72
9.0	5.0	7.97	33.66	62.59	30.45	0.73
9.0	6.0	7.16	52.68	63.38	32.94	0.77
9.0	7.0	16.16	52.01	62.30	31.48	0.78
9.0	8.0	11.68	31.46	63.56	28.65	0.79
10.0	2.0	6.16	50.17	64.52	29.05	0.81
10.0	3.0	7.38	27.86	62.93	37.17	0.83
10.0	4.0	7.82	53.51	63.60	40.31	0.87
10.0	5.0	6.94	43.78	62.85	30.66	0.87
10.0	6.0	8.03	43.81	64.54	24.23	0.89
10.0	7.0	9.33	45.74	63.74	25.40	0.91
10.0	8.0	12.28	38.71	61.87	25.46	0.93
10.0	9.0	16.74	42.72	61.49	30.87	0.95

Table A.15: Roaster model predictive controller tuning for a maximum percentage change in LPG, rotation and blower inputs of 7%, 5% and 5% respectively.

P	M	MSE	LPG	Rotation	Blower	Exec	
0	3.0	2.0	8.47	40.45	62.80	31.27	0.16
1	4.0	2.0	8.97	34.37	62.99	33.41	0.21
2	4.0	3.0	38.78	33.73	55.65	27.28	0.23
3	5.0	2.0	7.17	31.38	63.56	28.39	0.28
4	5.0	3.0	7.21	39.05	63.30	32.73	0.30
5	5.0	4.0	20.97	46.78	60.77	26.56	0.32
6	6.0	2.0	5.52	53.49	63.49	28.09	0.37
7	6.0	3.0	7.32	39.22	63.48	32.30	0.38
8	6.0	4.0	8.51	41.22	62.09	27.03	0.40
9	6.0	5.0	10.89	51.45	63.12	30.76	0.42
10	7.0	2.0	5.70	34.67	63.90	26.58	0.45
11	7.0	3.0	6.67	35.70	63.05	27.65	0.48
12	7.0	4.0	12.34	33.28	61.85	28.12	0.50
13	7.0	5.0	36.74	44.37	57.00	31.43	0.51
14	7.0	6.0	15.07	24.10	61.39	30.37	0.52
15	8.0	2.0	7.71	45.73	63.14	32.25	0.57
16	8.0	3.0	7.40	43.58	63.24	32.53	0.61
17	8.0	4.0	7.93	45.95	63.63	27.84	0.62
18	8.0	5.0	13.60	46.75	62.60	24.84	0.63
19	8.0	6.0	13.65	39.21	61.84	30.20	0.65
20	8.0	7.0	9.85	29.71	61.69	25.72	0.67
21	9.0	2.0	6.32	61.45	61.89	29.07	0.71
22	9.0	3.0	6.85	42.38	63.36	31.72	0.72
23	9.0	4.0	7.26	26.38	63.71	30.63	0.73
24	9.0	5.0	25.21	41.93	60.31	28.53	0.76
25	9.0	6.0	9.99	54.59	62.87	31.51	0.78
26	9.0	7.0	17.21	29.64	61.22	26.12	0.80
27	9.0	8.0	23.97	43.89	60.68	31.90	0.81
28	10.0	2.0	6.22	36.28	62.87	29.77	0.83
29	10.0	3.0	7.36	42.25	63.78	29.03	0.86
30	10.0	4.0	38.08	38.36	56.21	30.25	0.88
31	10.0	5.0	8.01	24.58	63.32	31.45	0.89
32	10.0	6.0	20.02	40.26	60.43	33.75	0.92
33	10.0	7.0	33.23	23.58	56.14	31.07	0.92
34	10.0	8.0	21.17	40.01	60.70	34.91	0.95
35	10.0	9.0	17.39	44.77	60.87	25.46	0.98