# Multi-objective Model Predictive Control of Multivariable Systems

Thabo Koetje

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.

Supervisor:

Prof. M. Braae

Dept. of Electrical Engineering

University of Cape Town

Co-Supervisor:

Mr. M. Tsoeu

Dept. of Electrical Engineering

University of Cape Town

**May 2011**

# Declaration

I hereby declare that this thesis is my own unaided work. This thesis is being submitted for a degree of Master of Science in Engineering at the University of Cape Town. Except as stated below, neither the substance or any part of the thesis has been submitted in the past for any degree or examination in any other university.

Signature of the author: _____

Date:_____

# Abstract

Most engineering systems are multivariable in nature where more than one input controls more than one output. The challenges arise in controlling these types of systems due to interaction among inputs and outputs. In an attempt to optimise the performance of these processes, many performance objectives need to be considered simultaneously. In most cases, these objectives often conflict and hence a need for Multi-objective Optimization (MOO) analysis.

In this thesis, MOO design for Model Predictive Control (MPC) and Proportional Integral (PI) control are investigated for a multivariable process. The Pareto sets for both controllers are generated using Pareto Differential Evolution (PDE) and then compared using an n-Dimensional visualisation tool, *Level Diagrams* to evaluate which controller is best for the process. In addition, the MOO performance measures (or quality indicators) are further used to quantitatively compare the Pareto fronts generated out of these controllers. Finally, the solutions which provide a preferred performance are then selected and tested experimentally on the process.

# Acknowledgements

I would like to express my deepest gratitude to my supervisors, Mr. Tsoeu and Prof. Braae, for without them, this thesis would not have been possible. They did not only help me to refine my ideas, but they also willingly shared their own ideas and inspired me by asking the right questions. Clearly I feel very lucky to have work with two distinguished supervisors that work well together and are fun to be around.

I would like to thank Mr. Lebelo Serutla and his family for their constant unconditional support during my stay in Cape Town, without them being around here will have been impossible.

Thanks to iThemba LABS through NRF for their financial assistance.

To all the present students and friends in particular Makhamisa and Mohammed that I have shared moments and office space with, I say thank you guys for making the UCT such an enjoyable place!!!

Finally, to my family, all I can say is " Kea a le leboha bana beso, Molimo a le etse hantle"

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Most industrial control systems are Multi-Input Multi-Output (MIMO) or multivariable in nature. These types of systems have two or more outputs which are controlled by two or more inputs. In these systems, one input does not affect only one output response, but it affects all other output responses. In fact, when tracking a certain output due to its set-point changes, the other outputs' responses are also interactively affected. Due to the structural properties of these multivariable systems, issues which are not relevant in Single-Input Single-Output (SISO) systems such as interaction imposes a challenge in designing controllers for these types of systems [1]. The design complexity of these controllers can further increase if other performance measures, for example overshoots or oscillations minimisation, have to be integrated into the control design considerations.

Several methods have been used to control multivariable systems. One of the popular control methods, Model Predictive Control which is mostly applied

in industrial systems has also been found to perform well with MIMO systems [1, 2]. MPC rely on the model of the system to minimise a cost objective which is used to calculate the optimal inputs to be applied onto the system. Due to the structure of the MPC design, many parameters are involved that need to be considered to tune the controller performance. Numbers of works have been done to systematically tune the MPC for specific cases and these are reviewed in ref. [3]. However due to varying nature of MPC methods and control systems the tuning parameters of these methods can still be manipulated to obtain better performance. In ref. [4], it has been shown that different control design methods performance can considerably be improved if used in conjunction with Multi-Objective Optimisation (MOO). Some of the work that uses MOO with MPC can be found in ref. [5] where genetic algorithm is combined with multi-objective fuzzy decision making (MOFDM) in an attempt to find MPC tuning parameters that optimize the MIMO system performance. However, few works have been conducted on comparisons of MOO MPC with other control methods so as to evaluate its trade-off performance over other control methods. The work where MOO MPC was compared with the most popular control method, Proportional Integral (PI) control can be found in [6]. However, the comparison was done based only on theoretical design without any simulations, use of MOO visualisation methods and or practical implementations which are addressed in this work.

In MOO design methods, the idea is to try to deal with cases where many conflicting design objectives are to be satisfied simultaneously. This is achieved by designing cost objectives to represent the performance measures that are to be optimised. These design objectives are then either minimised or maximised to generate the optimal solutions. The outcome of these optimal

solutions depend entirely on the design objectives used [7]. As a result, care must be taken when designing these objectives to ensure that they accurately represent the performance measures to be evaluated.

The optimal solutions are the set in which all solutions that are obtained using the given cost objectives are non-dominated in nature that is, in this set there is no solution which is better than any other solutions. Improving any cost objective in this set, will result into at least one of the other objectives worsening. As a result, all solutions in this set can equally be acceptable as solutions for controller design. This set is known as Pareto-optimal set [8].

Generating the exact optimal set can be computationally expensive or infeasible [8]. However, various methods exist that have been used to generate the approximate of the Pareto set. The most recently used methods are Multi-objective Optimisation Evolutionary Algorithms (MOEAs). This is because they provide a good approximate of the Pareto set [8], that is, a set of solutions which are (hopefully) not too far away from the optimal solutions. Further, these algorithms as contrary to other algorithms that have been used like nonlinear programming, use the probabilistic instead of deterministic approach when progressing with the search to find global solutions. They also classified the solutions simply as populations and then apply the principle of the survival of the fittest. In this principle, the individuals which are weaker (i.e., the solutions which there exists one or more solutions which are better than them in all objective functions) are eliminated so that only stronger individuals can evolve towards better solutions.

In the Pareto set, since all points are equally acceptable as solutions, knowledge about this set can significantly help the Decision Maker (DM) in choosing the best compromise solution for a given design specification. As a result, tools are needed to aid the DM in selection for the preference. Visualisation

tools have been widely accepted as valuable tools to help the DM to analyze the Pareto set and select good solution [9]. Traditionally Pareto fronts are represented in 2-Dimension (2-D) or 3-D which are relatively easy to visualise but for dimensions higher than 3-D, it becomes difficult to visualise or extract useful information from such plots. Different methods for aiding n-Dimensional visualisation and decision making have been proposed. In ref. [10], Visually Interactive Decision-making and Design using Evolutionary Multi-objective Optimization (VIDEO) is presented but it can only be used for design objectives up to four (or 4-D). Parallel coordinates and scatter diagrams are popular methods in use for any dimensional space but their plots have been found to be complex [9] and hence difficult to interpret.

## 1.2 Scope

In this thesis, MOO is used to design MPC and Proportional Integral (PI) controllers to optimise the performance of a highly interactive multivariable system. Eight cost objectives are designed using the Integral Square Error (ISE) and Integral Square Control velocity (ISU) on which large values of their amplitudes are greatly penalised. From these cost objectives, the 8-D Pareto fronts are generated for both controllers using one of the MOEAs, Pareto Differential Evolution (PDE) algorithm.

The generated Pareto fronts for both controllers are visualised and compared using a visualisation tool for n-Dimensional Pareto front, i.e. *Level Diagrams* which is presented in ref. [9]. Furthermore, quantitative analysis for the Pareto fronts are carried out using quality indicators (unary hypervolume, binary hypervolume) presented in ref. [8] to further quantify which Pareto front gives a better coverage space.

Lastly, from the Pareto set, the solution which gives a better compromise solution closer to the preference from each controller are simulated and compared in real time. The experiments are then conducted in order to finally evaluate their performances on the physical process.

The thesis including Introduction is structured as follows: Chapter 2 gives an overview and design of different control methods that have been used. Chapter 3 then gives a detail of the multi-objective design methods and tools that are used for visualisation and analysis for MOO problems. Furthermore, Chapter 4 then takes a look at a multivariable system that is used. Analysis of the structure of the system and different cost objectives that are used are also discussed. This is followed by Chapter 5, which presents the Level Diagram results and their interpretation. Time simulation and experimental results are also presented and discussed. The conclusions that summarise the findings and future works are finally presented in Chapter 6.

# Chapter 2

# Control Methods



In this chapter, different controller methods used in this thesis are discussed. Firstly, the literature review behind the invention and application of MPC is presented. Then the formulation of the MPC controllers will be discussed. Finally, the classical PI controller designs will be discussed.

## 2.1 Model Predictive Control Review

The name Model Predictive Control (MPC) also referred to as Receding Horizon Control comes from the idea of employing an explicit model of the system to be controlled which is used to predict the future output behaviour. This prediction enables the capability of solving the optimal control problem on-line, where the tracking error (i.e. the difference between the predicted output and the set-point (desired reference)) is minimised over a future horizon, possibly subject to constraints on the manipulated inputs and outputs. The idea of Model Predictive Control can be traced back to the 1960s [11], but interest in the field started to be seen in the 1980s after the publications on IDCOM and Dynamic Matrix Control (DMC) and exposition of General

Predictive Control (GPC) [12]. Although at the first sight, the idea behind DMC and GPC looks similar but DMC was conceived as multivariable constrained control, while GPC is primarily suited for a single variable, and possibly adaptive control [2, 12]. MPC employ optimisation according to a receding horizon philosophy which provides the controller with the desired feedback characteristics.

MPC has some significant strength as compared with other methods. In ref. [13, 14] constraints have been shown as problem since they limit the system performance. However, the problem was addressed by ref. [13, 15] and MPC was found to handle the constraints systematically by imposing constraints on the predictions. This is of importance since constraints such as limited actuator power, force and slew rate, are virtually present in most control systems. Further, MPC has also been found to deal with multivariable (MIMO) system [2, 16] especially highly interactive MIMO systems [13]. These have enabled MPC to slowly become a popular control methodology. For example, based on the data described in the survey paper [17], the estimated numbers of MPC applications are between seven and ten thousand on a global scale. Most of these applications are found in the refining industries where slow process dynamic plants enable MPC implementations, however MPC has also found applications in fast process systems, for example, robotics, aerospace, automotive and pulp & paper industries [17].

On the other hand in order to control the process accurately, MPC need a very accurate model [14]. This is because in predictive control, the model is used solely to compute the process predictions which are in turn used to calculate the control actions. This is somehow disadvantageous as it is difficult to model all the physics, chemistry and internal behaviour of the processes [13].

MPC is classified into two main methods, Unconstrained and Constrained MPC.

## 2.2 Unconstrained MPC

This is a form of MPC in which the controller is formulated off-line and used at each time step to calculate the control action. The block diagram for this method is shown in Figure 2.1. This is the state feedback structure for the discrete model prediction control (DMPC). The matrices $A_p$, $B_p$, $C_p$ define the discrete state-space representation of the system model. The $r(k)$ is the set-point where we want to drive our system and the observer estimates the state variable $\widetilde{x}(k)$ using the output $y(k)$. The state variable vector $x(k) = \begin{bmatrix} \triangle x(k) & y(k) \end{bmatrix}^T$ and hence the controller matrix $K$ can be split into two $K = \begin{bmatrix} K_{mpc} & K_y \end{bmatrix}$, where $K_{mpc}$ corresponds to the feedback gain related to $\triangle x(k)$ and $K_y$ corresponds to the feedback gain related to $y(k)$. The $z^{-1}$ denotes the backward shift operator and the module $\frac{1}{1-z^{-1}}$ denotes the discrete-time integrator.

The control law is formulated using the state-space design which gives advantages of stability analysis, exponential data weighting, Linear Quadratic Regulator (LQR) equivalent, and easy extension from Single Input Single Output (SISO) to MIMO.

Consider the stable discrete state-space model

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.1}$$

where $k$ denotes the current sampling point, '$x$', '$u$', and '$y$' represent the

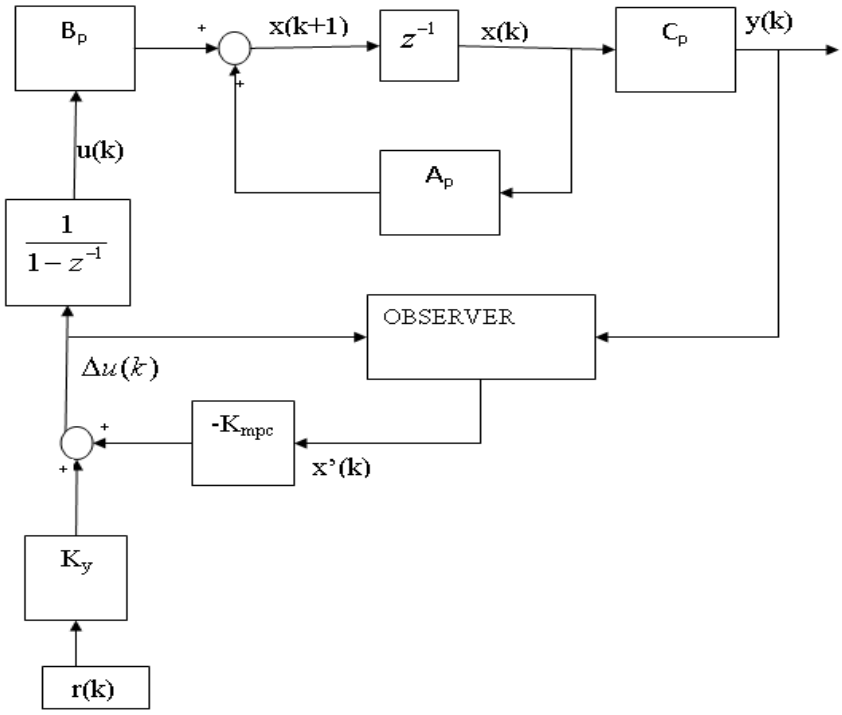


Figure 2.1: Unconstrained predictive control block diagram

state, the input and the model output, respectively, and A, B, C are system matrices.

The following traditional MPC cost function which penalizes the deviations of the controlled outputs from reference trajectory, i.e. $e(k+j|k)$ and control actions, i.e. $u(k+j|k)$ is minimised

$$J = \sum_{r=0,j=0,}^{N_p,N_p} Q(r,j)[e(k+j|k)]^2 + \sum_{r=0,j=0}^{N_c-1,N_c-1} R(r,j)[u(k+j|k)]^2 \qquad (2.2)$$

where $N_p$ and $N_c$ are the prediction and control horizon respectively, $e$ is the error, $u$ is the input. $Q$ and $R$ are the positive definite weighting matrices.

- The prediction horizon, $N_p$ corresponds to the future time interval used to compute predictions of the output. It dictates how far we want the future to be predicted.
- The control horizon, $N_c$ is a set of sampling intervals where the present and the future control actions are computed. After $N_c$, the controller will maintain the last control signal computed at $N_c$ until the end of the prediction horizon in order to calculate the output prediction.

### 2.2.1 Control law and Offset-free Tracking Control

In every control problem, the idea is to drive a system to a desire set-point. The above formulation, i.e. using the state-space in Equation 2.1 would not drive the system to its desired reference if subjected to disturbance or modelling errors. To eliminate these, the state-space matrices are augmented [18] to introduce an integral action. The integral action introduces extra

states to estimate the steady state error and attempt to eliminate the error. The complete velocity form method which considers the increments on both, the inputs and the states is used and has the following form:

$$\begin{aligned} \xi(k+1) &= \tilde{A}\xi(k) + \tilde{B}\triangle u(k) \\ y(k) &= \tilde{C}\xi(k) \end{aligned} \tag{2.3}$$

where

$$\tilde{A} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}, \tilde{B} = \begin{bmatrix} B \\ CB \end{bmatrix}, \tilde{C} = \begin{bmatrix} 0 & I \end{bmatrix}$$

$$\xi(k) = \begin{bmatrix} \triangle x(k) \\ y(k) \end{bmatrix}$$

$$\triangle x(k) = x(k) - x(k-1)$$

$$\triangle u(k) = u(k) - u(k-1)$$

and $A$,$B$, $C$ are matrices in Equation 2.1. The traditional cost function which penalizes the deviations of the controlled outputs $y(k+j|k)$ from reference trajectory $r(k+j|k)$ and the increments in control action(i.e $\triangle u$) is now defined as

$$J = \sum_{r=0,j=0}^{N_p,N_p} \bar{Q}(r,j)[e(k+j|k)]^2 + \sum_{r=0,j=0}^{Nc-1,N_c-1} \bar{R}(r,j)[\triangle u(k+j|k)]^2 \tag{2.4}$$

where

$$e(k+j|k) = r(k+j|k) - y(k+j|k)$$

$$\overline{R} = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_h \end{bmatrix}$$

$$\overline{Q} = \begin{bmatrix} \beta_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \beta_q \end{bmatrix}$$

The constants $\lambda_h$, $\beta_q$ are weighting coefficients on control action and error respectively, $h$ is the number of input and $q$ is the number of outputs.

In this cost function, the new state vector $\xi$ is directly penalised in order to achieve a stationary point in which the state increments are null and the output is equal to the set point [18].

In MPC, the future control velocity and states are captured over the whole range of $N_c$ and $N_p$ respectively. The future control velocity trajectory are denoted by $\triangle u(k|k)$ $\triangle u(k+1|k)$ $\triangle u(k+2|k)$ ... $\triangle u(k+N_c-1|k)$ and the future states are also denoted by $\xi(k+1|k)$ $\xi(k+2|k)$ $\xi(k+3|k)$ ... $\xi(k+N_p|k)$.

Based on the state-space model in Equation 2.3, the future states variables can be calculated sequentially as

$$\begin{aligned}
\xi(k+1|k) &= \tilde{A}\xi(k|k) + \tilde{B}\triangle u(k|k) \\
\xi(k+2|k) &= \tilde{A}\xi(k+1|k) + \tilde{B}\triangle u(k+1|k) \\
&= \tilde{A}^2\xi(k|k) + \tilde{A}\tilde{B}\triangle u(k|k) + \tilde{B}\triangle u(k+1|k) \\
&\vdots \\
\xi(k+N_p|k) &= \tilde{A}^{N_p}\xi(k|k) + \tilde{A}^{N_p-1}\tilde{B}\triangle u(k|k) \\
&\quad + \ldots + \tilde{A}^{N_p-N_c}\tilde{B}\triangle u(k+N_c-1|k)
\end{aligned} \tag{2.5}$$

From the predicted state variables, by substitution, the predicted output variables are

$$\begin{aligned}
y(k+1|k) &= \tilde{C}\tilde{A}\xi(k|k) + \tilde{C}\tilde{B}\triangle u(k|k) \\
y(k+2|k) &= \tilde{C}\tilde{A}\xi(k+1|k) + \tilde{C}\tilde{B}\triangle u(k+1|k) \\
&= \tilde{C}\tilde{A}^2\xi(k|k) + \tilde{C}\tilde{A}\tilde{B}\triangle u(k|k) + \tilde{C}\tilde{B}\triangle u(k+1|k) \\
&\vdots \\
y(k+N_p|k) &= \tilde{C}\tilde{A}^{N_p}\xi(k|k) + \tilde{C}\tilde{A}^{N_p-1}\tilde{B}\triangle u(k|k) \\
&\quad + \ldots + \tilde{C}\tilde{A}^{N_p-N_c}\tilde{B}\triangle u(k+N_c-1|k)
\end{aligned} \tag{2.6}$$

This predicted output can then be written in a compact form as

$$Y = Fx(k) + \Phi\triangle U \tag{2.7}$$

where

$$F = \begin{bmatrix} \tilde{C}\tilde{A} \\ \tilde{C}\tilde{A}^2 \\ \tilde{C}\tilde{A}^3 \\ \vdots \\ \tilde{C}\tilde{A}^{N_P} \end{bmatrix}, \Phi = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & 0 & ... & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & 0 & ... & 0 \\ \tilde{C}\tilde{A}^2\tilde{B} & \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{N_P-1}\tilde{B} & \tilde{C}\tilde{A}^{N_P-2}\tilde{B} & \tilde{C}\tilde{A}^{N_P-3}\tilde{B} & ... & \tilde{C}\tilde{A}^{N_P-N_C}\tilde{B} \end{bmatrix}$$

and

$$Y = \begin{bmatrix} y(k+1|k) & y(k+2|k) & y(k+3|k) & ... & y(k+N_p|k) \end{bmatrix}^T$$

$$\triangle U = \begin{bmatrix} \triangle u(k|k) & \triangle u(k+1|k) & \triangle u(k+2|k) & ... & \triangle u(k+N_c-1|k) \end{bmatrix}^T$$

From Equation 2.4 , the optimization cost function can be simply be written as

$$J = (R_s - Y)^T \bar{Q}(R_s - Y) + \triangle U^T \bar{R} \triangle U \tag{2.8}$$

where

$$R_s^T = \overbrace{\begin{bmatrix} 1 & 1 & 1 & ... & 1 \end{bmatrix}}^{N_p} r(k)$$

$$\bar{R} = \lambda I_{(N_c \times N_c)}$$

$$\bar{Q} = \beta I_{(N_p \times N_p)}$$

where $\beta > 0$ and $\lambda \geq 0$.

To find optimal $\triangle U$ we substitute Equation 2.7 into Equation 2.8 and minimized $J$ (Discussed in Appendix A). Then the optimal control law is given as

$$\triangle U = (\Phi^T \bar{Q} \Phi + \bar{R})^{-1} \Phi^T \bar{Q} (R_s - Fx(k)) \tag{2.9}$$

with the assumption that $(\Phi^T \bar{Q} \Phi + \bar{R})^{-1}$exists. This is a Hessian matrix and in the MPC literature is known as the *System Matrix* which gives the sensitivity of the controller to model errors and tuning parameters variation. Since the matrix needs to be inverted to calculate the control action, it is properly conditioned by a careful selection of matrix $\bar{Q}$ and $\bar{R}$.

From Equation 2.9, $(\Phi^T \bar{Q} \Phi + \bar{R})^{-1} \Phi^T \bar{Q} R_s$ corresponds to set-point change, while $(\Phi\Phi + \bar{R})^{-1} \Phi^T \bar{Q} F$ corresponds to the state feedback control.

Equation 2.9 can then be re-written as

$$\triangle U = K_y r(k) - K_{mpc} x(k) \tag{2.10}$$

where

$$K_y = (\Phi^T \bar{Q} \Phi + \bar{R})^{-1} \Phi^T \bar{Q} \bar{R}_s$$

$$K_{mpc} = (\Phi^T \bar{Q} \Phi + \bar{R})^{-1} \Phi^T \bar{Q} F$$

### 2.2.2 Receding Horizon Control

When calculating the control law, the optimal control $\Delta U$ is a vector that contains all the future control increments, i.e.

$$\Delta U = [\Delta u(k|k), \Delta u(k+1|k), ..., \Delta u(k+N_c-1|k)]$$

With receding control idea, we only implement the first sample of this sequence, i.e.,$\triangle u(k|k)$ while ignoring the rest of the sequence. Since the prediction horizon remains of the same length as before, but slides along by one sampling interval at each step, this way of controlling a plant is often called a *Receding horizon* strategy [14].

## 2.3 Constrained MPC

In virtually all control systems, constraints do arise due to performance demands or the environment in which the system operates. These constraints are mostly found in the following forms [14]:

1. direct costs (e.g energy costs)
2. product quality
3. Saturation constraints
    - Valve adjustment
    - fixed pipe size
    - Road lane
4. Output constraints (e.g overshoots), etc

This form of MPC formulates the controller on-line at each step and calculates the control action.

### 2.3.1 Control, Control incremental and Output variable Constraits

Control variable constraints are mostly hard constraints in nature, for example, a DAQ card cannot be allowed to operate beyond a certain voltage range. In actual fact, we demand that

$$U_{min} \leq U(k+i|k) \leq U_{max} \tag{2.11}$$

where $U_{min}$ and $U$ is the minimum and maximum limits of control variables vectors respectively and $i = [0, 1, .., N_c - 1]$.

Similarly, the control incremental constraints are also hard constraints. They are of the form

$$\Delta U_{min} \leq \Delta U(k+i|k) \leq \Delta U_{max} \tag{2.12}$$

where $\Delta U_{min}$ and $\Delta U_{max}$ are the minimum and maximum limit of control incremental variables vectors respectively. In most cases, limits are defined in terms of $u(k)$ but since $u(k) = u(k-1) + \Delta u(k)$, the $\Delta u(k)$ can be used to dictate the direction of $u(k)$. For example, if we want $u(k)$ to only increase then we will define our constraints as

$$0 \leq \Delta U(k+i|k) \leq \Delta U_{max} \tag{2.13}$$

The output constraints are given as

$$Y_{min} \leq Y(k+j|k) \leq Y_{max} \tag{2.14}$$

where $Y_{min}$ and $Y_{max}$ are the minimum and maximum limits of output vectors respectively and $j = [1, 2, .., N_p]$.

The constraints on output are *"soft"* constraints in nature, because when the output constraints become active, they cause large changes in control and control incremental variables, which might violate their own constraints. Since these constraints are "*hard*" in nature, the problem of constraints conflict arises. This is resolved by adding a small slack variable $s_v$ in output constraints hence softening them [2].

$$Y_{min} - s_v \leq Y(k+j|k) \leq Y_{max} + s_v \tag{2.15}$$

### 2.3.2 Constrained Control formulation

The constrained control method includes constraints in its control law formulation hence the constraints have to be formulated as part of the controller design requirements. This is achieved by translating them into linear inequalities, and then relating them into the MPC problem. The idea is to combine the constraints with the original cost function $J$ used in Equation 2.8. In MPC literature, the $\Delta U$ parameter is mostly the parameter to be optimised, hence all constraints are expressed in as a set of linear equations based on $\Delta U$.

The constrained problem is decomposed into two problems to reflect the lower and upper limits, that is,

$$\Delta U_{min} \leq \Delta U \leq \Delta U_{max}$$

is expressed as

$$\begin{aligned} -\Delta U &\leq -\Delta U_{min} \\ \Delta U &\leq \Delta U_{max} \end{aligned} \tag{2.16}$$

Expressed in matrix form, it becomes

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U_{min} \\ \Delta U_{max} \end{bmatrix}$$

These constraints are imposed mostly in both input and output variables, and have to be expressed for the whole prediction. Hence, in the case of input variables, this is expressed as

$$\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c-1|k) \end{bmatrix} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} u(k-1) + \begin{bmatrix} I & 0 & 0 & ... & 0 \\ I & I & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ I & I & I & ... & I \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N_c-1|k) \end{bmatrix} \tag{2.17}$$

Expressing Equation 2.17 in matrix form as in Equation 2.16, we get

$$\begin{aligned} -\left(\gamma_1 u(k-1) + \gamma_2 \Delta U\right) &\leq -U_{min} \\ \left(\gamma_1 u(k-1) + \gamma_2 \Delta U\right) &\leq U_{max} \end{aligned} \tag{2.18}$$

The output constraints can also be expressed as,

$$Y_{min} \leq Fx(k) + \Phi \Delta U \leq Y_{max}$$

which can also be written in matrix form as

$$\begin{bmatrix} -(Fx(k) + \Phi\Delta U) \leq -Y_{min} \\ (Fx(k) + \Phi\Delta U) \leq Y_{max} \end{bmatrix} \tag{2.19}$$

Combining all constraints (i.e, Equations [2.16,4.4,2.19]) together, the Con trained MPC law will be designed as to find the parameter $\Delta U$ that minimise

$$J = \Delta U^T \left(\Phi^T \Phi + \lambda\right) \Delta U - 2\Delta U^T \Phi^T \left(R_{set} - Fx(k)\right) \tag{2.20}$$

subjected to inequality constraints

$$\begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}$$

where

$$\Omega_1 = \begin{bmatrix} -\gamma_2 \\ \gamma_2 \end{bmatrix}; \quad \mu_1 = \begin{bmatrix} -U_{min} + \gamma_1 u(k-1) \\ U_{max} - \gamma_1 u(k-1) \end{bmatrix}$$

$$\Omega_2 = \begin{bmatrix} -I \\ I \end{bmatrix}; \qquad \mu_2 = \begin{bmatrix} -\Delta U_{min} \\ \Delta U_{max} \end{bmatrix}$$

$$\Omega_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}; \qquad \mu_3 = \begin{bmatrix} -Y_{min} + Fx(k) \\ Y_{max} - Fx(k) \end{bmatrix}$$

Equation 2.20 is a quadratic programming problem which can be solved using numerical algorithms.

### 2.3.3 Quadratic Programming

Quadratic Programming (QP) is a special type of mathematical optimization problem. Since this is a field on its own, we will look at QP in overview and algorithms that are used to solve these problems.

In QP, we optimise (i.e., minimise or maximise) a quadratic function of several variables subject to linear constraints on these variables. We minimise

$$f(x) = \frac{1}{2}x^T Ex + x^T \Gamma \tag{2.21}$$

subjected to one or more constraints of the form

$$\psi x \leq b(inequality - constraints)$$
$$\Lambda x = d(equality - constraints)$$

where $x \in R^n$ space, $E$ is a symmetric $n \times n$ matrix and $\Gamma$ is $n \times 1$ column vector.

Heldrith Programming is one of the popular numerical algorithms that are used to solve the quadratic problems. This algorithm is used in this thesis because it avoids any matrix inversion when searching for optimal solution. It also gives a compromised, near-optimal solution if the situation of conflict in constraints arises [2] hence avoids giving an error message. This is a key strength of using this approach because in real-time applications, the plant is awaiting for the input to be applied at every sample instant, so if a case of constraints conflict arises and there is no solution which can satisfies all the active constraints, the algorithm should have the ability to automatically recover from this condition instead of giving error.

#### 2.3.3.1 Inequality Constraints

The inequality constraints are the constraints of the form

$$\psi x \leq b \tag{2.22}$$

where $\psi$ and $b$ are matrices of compatible size.

The inequality constraints may comprise active constraints and inactive constraints. In most cases, an inequality $\psi_i x \leq b_i$ is said to be active if $\psi_i x = b_i$ and inactive constraints if $\psi_i x < b_i$. Note that $\psi_i$ represent the $i^{th}$ row of $\psi$ matrix while $b_i$ is the $i^{th}$ element of $b$ column vector. In order to define the active and inactive constraints, the Kuhn-Tucker conditions should be met [19]. The necessary Kuhn-Tucker conditions are

$$\begin{aligned} Ex + \Gamma + \psi^T \lambda &= 0 \\ \psi x - b &\leq 0 \\ \lambda^T (\psi x - b) &= 0 \\ \lambda &\geq 0 \end{aligned} \tag{2.23}$$

where the vector $\lambda$ contains the Lagrange multipliers. At any given time instant, it is most probable that certain constraints are active. Denoting the index set of active constraints as $G_{act}$, then Equation (2.23) can be transformed to represent these active set of constraints and it can simply be written as

$$
\begin{aligned}
Ex + \Gamma + \textstyle\sum_{i \in G_{act}} \lambda_i \psi_i^T = 0 \\
\psi_i x - b_i = 0 \\
\psi_i x - b_i < 0 \\
\lambda_i \geq 0 \\
\lambda_i = 0 \\
i \in G_{act}
\end{aligned}
\tag{2.24}
$$

From Equation (2.24), note that $\psi_i x - b_i = 0$ means that this is an equality constraint, therefore active constraint. But in contrast, $\psi_i x - b_i < 0$ is inactive constraint which means that its constraints requirements are fulfilled. If the constraint is active, the corresponding Lagrange Multiplier is non-negative (i.e, $\lambda_i \geq 0$) whilst if the constraint is inactive the corresponding Lagrange Multiplier is zero (i.e, $\lambda_i = 0$).

#### 2.3.3.2 Equality Constraints

If the objective to be minimised is subjected to equality constraints, that is, $\psi x = b$ , then Lagrange Multiplier are introduced to solve the optimization function. The original objective function given in Equation (2.21) is re-written as

$$
f(x) = \frac{1}{2} x^T E x + x^T \Gamma + \lambda^T (\psi x - b) \tag{2.25}
$$

If $\psi x = b$, then Equation (2.25) is translated back into its original optimization function. Most importantly, it should be noted that satisfying all equality constraints that become active simultaneously, it is a bit difficult and some constraints might have to be violated.

## 2.4 Proportional Integral Control

PI control is a simplified version of Proportional Integral Derivative (PID) control method where the derivative term (i.e, D term) is removed. This method as contrary to MPC does not use a model of the process to calculate its control law; however it uses the error between the output and its reference to calculate the control action that is applied onto the system. Due to this simplicity in calculating the control law and its competitiveness in performance [20], it has remained one of the popular control methods in use in the industries.

### 2.4.1 PI Formulation

PI design is fully discussed in ref. [21]. In this thesis, the form used is slightly modified as shown in Equation 2.26 for easy parameter range determination and tuning (Explained later in Section 4.5).

$$K(s) = K_P + \frac{K_I}{s} \tag{2.26}$$

The MIMO PI is designed as

$$K(s) = \begin{bmatrix} K_1 & 0 & \dots & 0 \\ 0 & K_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_i \end{bmatrix} \tag{2.27}$$

where $K_i$ takes the form shown in Equation 2.26, which then gives the control law

$$\begin{bmatrix} \Delta u_1 \\ : \\ . \\ \Delta u_i \end{bmatrix} = \begin{bmatrix} K_1 & 0 & \cdots & 0 \\ 0 & K_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_i \end{bmatrix} \begin{bmatrix} \Delta e_1 \\ : \\ . \\ \Delta e_i \end{bmatrix} \tag{2.28}$$

$$i = [1, 2, 3, ...]$$

where $\Delta u_i$, $\Delta e_i$, $K_i$ are the control action velocity of output $i$, the change in error of the output $i$ due to reference $i$ and the controller on output $i$ respectively.

# Chapter 3

# Multiobjective Optimisation Design and Tools



When designing a control system, there are often a number of design objectives that need to be considered. These objectives most often conflict and no design exist which are considered the best with respect to all the objectives. These have led to a trade-off analysis between the objectives and hence a multi-objective optimisation (MOO) problem. MOO combined with control methods can considerably improve the performance of the process and also help to approximate conflicting design specifications [4].

## 3.1 Multiobjective design

Design is generally governed by multiple conflicting criteria, which require designers to look for good compromise designs by performing trade off studies involving the criteria [22]. These design specifications are often conflicting since there is no optimal solution that simultaneously satisfies all of them.

In general the multi-objective Optimisation problem is a problem of simultaneously minimising the $n$ objectives $J_n(\theta)$, that is

$$\underset{\theta \in \Omega}{min} \; J(\theta)$$

$$\theta = [\theta_1, \ldots, \theta_i] \in \Omega$$
$$J(\theta) = [J_1(\theta), \ldots, J_k(\theta)]$$

where $\theta$ is the input or decision vector, $\Omega$ is the decision space and $J(\theta)$ is the cost or objective vector. These generate a set of mutually optimal solutions $\Omega_P$ in which no point dominates any other. This set is known as the Pareto-optimal set [9]. In this set, there is no single solution which is better than the others, i.e., improving any objective functions in this set, will result into at least one of the other objectives worsening.

### 3.1.1 Pareto Optimal

A point $\theta* \in \Omega_P$ is defined as being Pareto-optimal if and only if there exists no other point $\theta \in \Omega_P$ such that

1. $J_n(\theta) \leq J_n(\theta*)$ for all $n$ and
2. $J_i(\theta) < J_i(\theta*)$ for at least one $i$

This is shown in Figure 3.1. $\phi_1$ and $\phi_2$ are cost objectives to be minimised. Note that a point lying in the interior of the attained set is sub-optimal, since both $\phi_1$ and $\phi_2$ can be reduced while a point lying on the boundary of the set, i.e., the Pareto optimal set, $\theta_P$, requires $\phi_1$ to be increase if $\phi_2$ is to be decreased or vice versa.

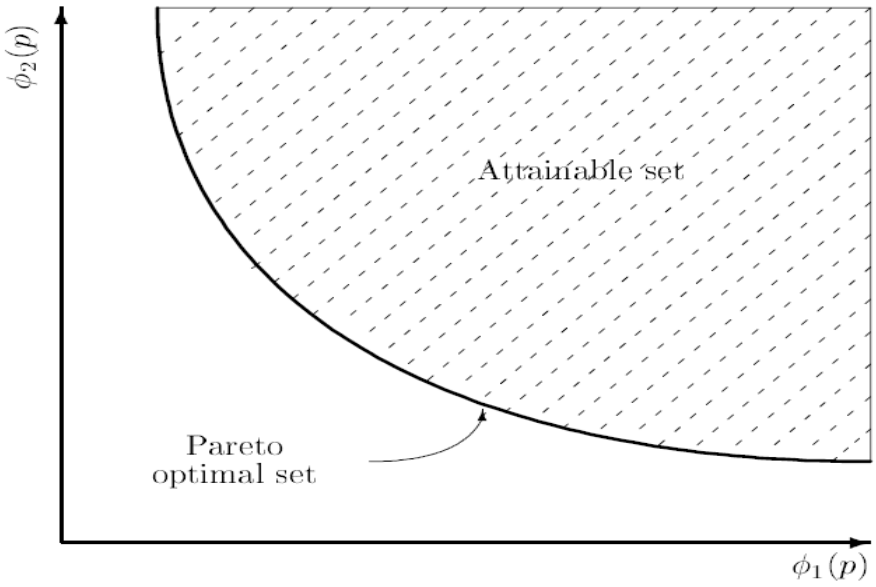


Figure 3.1: The Pareto-optimal set [4]

Therefore, the Pareto optimal point, $\theta_P$, is given by

$$\theta_P = \{\theta \in \Omega | \nexists \theta* \in \Omega : \theta* \prec \theta\}$$

### 3.1.2 Differential Evolution

MOO is a huge field with many different optimisation methods. Some of the very popular types of MOO methods are Evolutionary Algorithms (EA) [7]. These methods are preferred over other methods because they use probabilistic and not deterministic procedures for processing the search [4]. They also work well on the non-smooth objective functions hence they are less susceptible to the shape or continuity of the Pareto Front [7]. Another advantage of these methods is that they are more likely to find global optima, and not be stuck on local optima as gradient methods might do [23].

There are various competing EA which differ depending on the selection

of individuals for breeding, the crossover combination of genes by mating, genetic mutation and other factors such as population size. Most of the recent popular Multi-Objective Evolutionary Algorithm (MOEA) methods are the SPEA2 (Strength Pareto Evolutionary Algorithm), MOGA (Multiple Objective Genetic Algorithm), NGSA-II (Non-dominated Sorting Genetic Algorithm), NPGA-II (Niched Pareto Genetic Algorithm), Particle Swarm Optimization and Differential Evolution (DE) [6, 7].

In this thesis DE is used due to its desired properties. DE has been found to be competitive with other EA's [24] and it deals with real numbers rather than binary encoding as many EA's do [7]. Furthermore, there has been extensive research in the field of DE in the department. The Pareto Differential Evolution Algorithm (PDE) discussed in ref. [24] is used to generate the Pareto front.



## 3.2 Multiobjective Visualisation and Analysis

In multi-objective optimisation, the decision-maker (DM) will be stuck with the problem of having a Pareto optimal set of conflicting functions. As a result, tools are needed to aid DM to visualise, analyse and decide based on the preference which set better fits the problem.

Depending on how the computation and the decision-making processes are combined in the search for compromise solutions, there exist three main classes of multi-objective optimization methods: *a priori* articulation of preferences, *a posteriori* articulation of preferences, and progressive articulation of preferences [4].

- In *a priori* articulation of preferences, the DM makes the preferences

before optimization begins. Here the DM makes preference by combining individual objective functions into a single objective function.

- In *a posteriori* articulation of preferences, the DM makes the preferences after optimization. Here, the DM has the optimiser which determines the solution set before the DM makes any preferences. Then, the DM can choose the solution trade-off from the determined set.

- In a progressive articulation of preferences, the DM makes decisions during optimisation. At each step, the DM provides partial preference information to the optimiser, which in turn generates better alternatives based on the information received.

Preference articulation tries to implicitly define a utility function that discriminates between candidate solutions. Although it is very difficult to formalise such a utility function in every detail, approaches based on weighting coefficients, priorities and goal values have been widely used [4].

- Weighting coefficients are values that show the importance of the objectives and balance their involvement in the overall utility function.

- Priorities are integers which determine the order in which cost objectives are to be optimised based on their importance. Each cost objective is assigned a priority number.

- Goal values give an indication of the desired levels of performance in each objective dimension. These may represent the level of performance or the ideal performance levels to be matched as closely as possible. This is the most used method as it is easier to be interpreted and can relate more closely to the final solution of the problem.

### 3.2.1 Level Diagrams

In MOO design, it is mostly accepted that visualization tools are valuable and provide decision-makers with a meaningful method to analyze the Pareto set and select good solutions [9]. This method has been widely applied in control systems for analysis of Pareto fronts of different control design methods.

Traditionally Pareto fronts have been proposed to be represented in 2-D or 3-D plots which are easy to visualise. However when the Pareto front dimensions are increased to a dimension higher than 3-D, it becomes difficult and impossible to extract useful information's from the plots.

Different methods for aiding n-Dimensional visualisation have been proposed and the most commonly used are scatter diagrams and parallel coordinates [9]. Scatter diagrams arrange data in form of an n×n matrix where each dimension represent one row and column of the matrix. Parallel coordinates on the other hand try to represent an n-dimensional data on a two dimensional graph where each dimension is translated into an x-coordinate. However, plots complexity of both methods increases as dimensions of data increases [9] hence it becomes difficult to interpret.

*Level diagrams* are another new alternative method that enables easier visualization and analysis of a multi-dimensional Pareto set. This method is presented and discussed in ref. [9]. It can be used in *a priori* and progressive articulation of preferences to help the DM.

The *Level diagrams* tool try to approximate the Pareto front based on the proximity to the ideal point and then normalize each objective with respect to its minimum and maximum values, i.e.

$$J_i^M = \max_{\theta \in \Omega_{P*}} J_i(\theta)$$

$$J_i^m = \min_{\theta \in \Omega_{P*}} J_i(\theta)$$

$$i = 1, \ldots, s$$

$$\overline{J}_i(\theta) = \frac{J_i(\theta) - J_i^m}{J_i^M - J_i^m} \longrightarrow 0 \leq \overline{J}_i \leq 1 \tag{3.1}$$

Then a norm is evaluated to an approximate distance to the ideal point, and these norms are applied as shown in ref. [9].

1. 1-norm: $\left\|\overline{J}_i(\theta)\right\|_1 = \sum_{i=1}^{s} \left|\overline{J}_i(\theta)\right|$
2. Euclidean norm (2-norm): $\left\|\overline{J}_i(\theta)\right\|_2 = \sqrt{\sum_{i=1}^{s} \overline{J}_i(\theta)^2}$
3. Infinite norm ($\infty$-norm): $\left\|\overline{J}_i(\theta)\right\|_\infty = max\{\overline{J}_i(\theta)\}$

The values of each norm ranges as shown in Equation 3.2

$$\begin{aligned} 0 \leq \left\|\overline{J}_i(\theta)\right\|_1 &\leq s \\ 0 \leq \left\|\overline{J}_i(\theta)\right\|_2 &\leq \sqrt{s} \\ 0 \leq \left\|\overline{J}_i(\theta)\right\|_\infty &\leq 1 \end{aligned} \tag{3.2}$$

The Pareto front shape view is different for each norm. The Euclidean norms supply an accurate evaluation of the conventional geometrical distance to the ideal point, and then offer a better view of the '*real*' shape [9]. The $\infty$-norm shows the worst objective at a specific point. It is mostly used for trade-off analysis between different objectives.

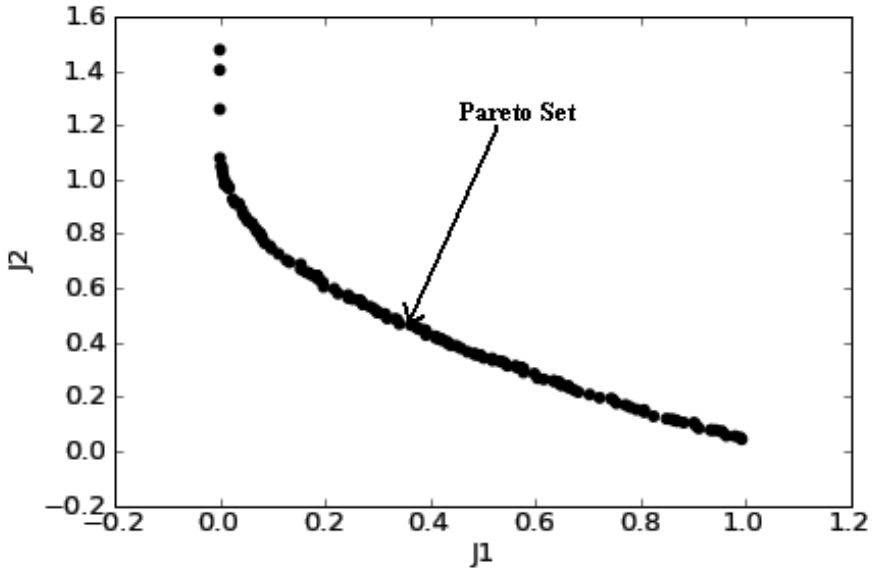


Figure 3.2: The Pareto front for 2 cost objectives

Generally, the ideal point is the point with a norm that is closest to zero. This point is mostly not a feasible solution because of the conflict between the objective functions.

To plot level diagrams, each objective $J_i$ and decision variables $\theta_i$ is drawn in its own graph where the Y axis correspond to the value of $\left\|\overline{J}_i(\theta)\right\|_x$ and X axis corresponds to the value of the objective or decision variable, in physical units. $x$ represent a norm used.

To show this an example representing a Pareto front in 2-D extracted from two objectives used in this thesis is shown in Figure 3.2.

By using 2-norm, the Pareto front is represented in *Level diagrams* as shown in Figure 3.3.

Since the plotted data points are represented in physical units, the DM can easily analyse the resulting plots. For example, it easier to see the unreachable range of a given cost objective ($J2 \approx [1.1, 1.3]$ from Figure 3.3).

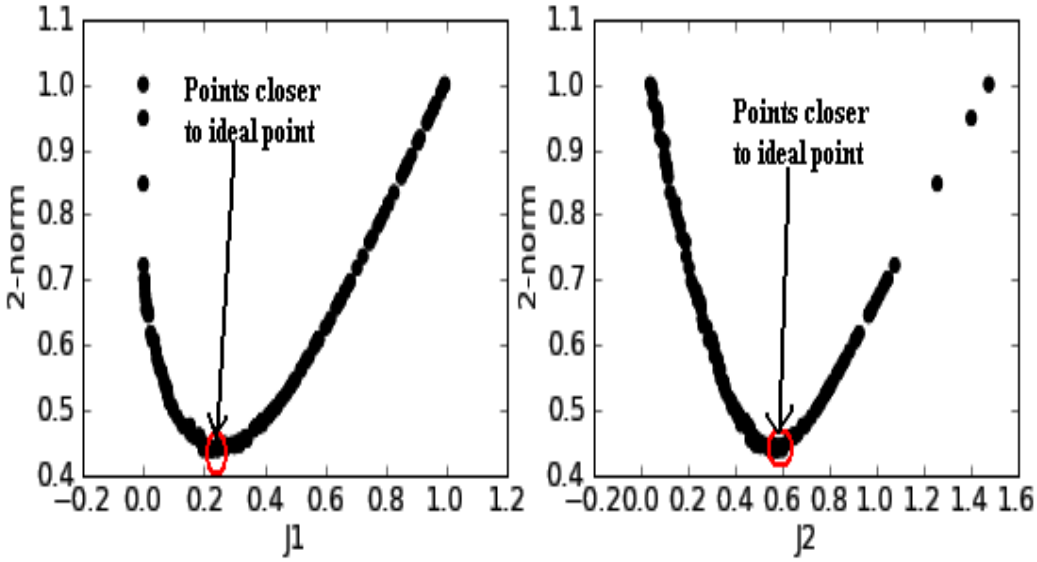


Figure 3.3: 2-Norm *Level Diagrams*

Furthermore, an analysis of trade-off between objectives can be observed and points situated closer to the ideal point are easily observed ($J1 \approx 0.22$, $J2 \approx 0.6$ from Figure 3.3). This information is of significance since in decision making criterion, the decision rule is to select a feasible solution such that the combined deviation between the selected solution and the ideal point is minimised.



## 3.3 Quality Measures

Evolutionary multi-objective optimization deals with computing the Pareto-optimal set for a given objective functions. When it comes to comparing the Pareto set of different control methods then quantitative measures are also of great significance as they give an exact value (or metric) by how much each Pareto set is better or dominates another.

When single objective function is used for comparison it is easier to come

up with a conclusion since the smaller (or larger) the value, the better the solution. However, if we compare two solutions in the presence of multiple optimization criteria, the concept of Pareto dominance can be used, although the possibility of two solutions being incomparable, i.e., neither dominates the other, complicates the situation [8]. Quality measures (or Quality indicators) as opposed to graphical plots have been used as a means to aid the DM to make better comparisons of the Pareto front generated from different multi-objective algorithms [8, 25].

In this thesis, quality indicators are applied to determine their usefulness and significance in comparing the Pareto fronts generated from different control design methods. These indicators are carefully selected since some of the indicators have direct real world analogies, while other don't relate to any practical realities and can be misleading especially when the Pareto front is not known exactly [7].

Quality measures are used to compare multi-objective optimizers quantitatively and their outcome is checked based on the three dominance relations $\rhd, \succ, \succ\succ$ which are discussed in Table 3.1.

For these relations to have an applicable meaning, they need to be interpreted. As a results, an interpretation function $E$ is introduced as shown in Figure 3.4.

This function maps vectors of real numbers to Booleans, for example, if we have a combination of quality indicators, $I$, and an interpretation function $E$, a comparison method is defined as

$$C_{I,E}(A, B) = E(I(A), I(B))$$

| relation | objective vectors | | approximation sets | |
|---|---|---|---|---|
| strictly dominates | $\mathbf{J}^1 \succ\succ \mathbf{J}^2$ | $\mathbf{J}^1$ is better than $\mathbf{J}^2$ in all objectives | $A \succ\succ B$ | every $\mathbf{J}^2 \in B$ is strictly dominated by at least one $\mathbf{J}^1 \in A$ |
| dominates | $\mathbf{J}^1 \succ \mathbf{J}^2$ | $\mathbf{J}^1$ is not worse than $\mathbf{J}^2$ in all objectives and better in at least one objective | $A \succ B$ | every $\mathbf{J}^2 \in B$ is dominated by at least one $\mathbf{J}^1 \in A$ |
| better | | | $A \rhd B$ | every $\mathbf{J}^2 \in B$ is weakly dominated by at least one $\mathbf{J}^1 \in A$ and $A \neq B$ |
| weakly dominates | $\mathbf{J}^1 \succeq \mathbf{J}^2$ | $\mathbf{J}^1$ is not worse than $\mathbf{J}^2$ in all objectives | $A \succeq B$ | every $\mathbf{J}^2 \in B$ is weakly dominated by at least one $\mathbf{J}^1 \in A$ |
| incomparable | $\mathbf{J}^1 \parallel \mathbf{J}^2$ | neither $\mathbf{J}^1$ weakly dominates $\mathbf{J}^2$ nor $\mathbf{J}^2$ weakly dominates $\mathbf{J}^1$ | $A \parallel B$ | neither $A$ weakly dominates $B$ nor $B$ weakly dominates $A$ |

Table 3.1: Relations Definition [8]

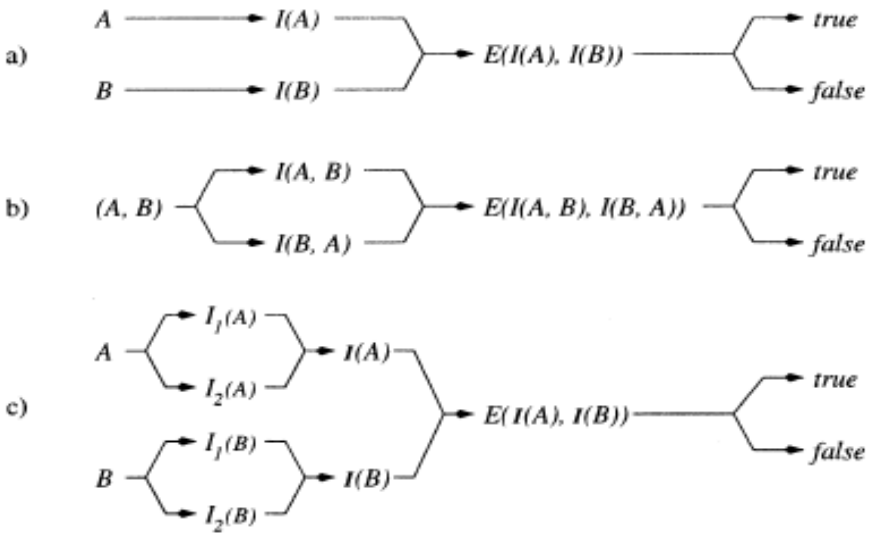


Figure 3.4: Illustration of the concept of a comparison method for (a) a single unary quality indicator, (b) a single binary quality indicator, and (c) a combination of the two unary quality indicators [8]

$$I = (I_1, I_2, \ldots, I_k)$$
$$E : \mathbb{R}^k \times \mathbb{R}^k \longrightarrow \{false, true\}$$

where $A$, $B \in \Omega$ are two approximation sets. In Figure 3.4, for cases (a) and (b) , the indicator $I$ is applied onto the approximation sets $A$ and $B$. This generates two real values which are passed through the interpretation function $E$. The interpretation function produces a Boolean outcome from the comparison. In case (c), each indicator $I_1$ and $I_2$ is applied separately on the approximation sets $A$ and $B$ and the resulting two indicator values are combined in a vector $I(A)$ & $I(B)$ respectively. The interpretation function $E$ is then applied on the two vector to decide the outcome of the comparison based on these two real numbers.

Two indicators that are used in this thesis are discussed in ref. [8] and applied in ref. [7, 26]. These are

- The unary hypervolume indicator, $I_H(A)$, is a measure of the percentage of the hypervolume within a bounded region that the approximation set $A$ bounds.
- The binary hypervolume indicator, $I_{H2}(A, B)$, is the percentage hypervolume of $A$ that is weakly dominated by $A$ but not by $B$. It is given as

$$I_{H2}(A, B) = I_H(A \cup B) - I_H(B)$$

# Chapter 4

# Control System Modelling and Design



Modelling is an important step when designing control systems. This chapter describes in detail the system modelling and analysis. Furthermore, it depicts the cost objectives used, and various controller tuning parameters selected.

## 4.1 System Modelling

For a control system design to be defined, an appropriate model or step test data should be obtained which relates the input variables to the output variables. Many methods exist to present but most of them are classified as either mathematical or physical modelling techniques. In this thesis, physical modelling was used to extract the system model using step test data. The system used was a thermal system, which consists of two inputs and two outputs. The step test plot is shown in Figure 4.1.

The system was modelled using first-order model where the input-output

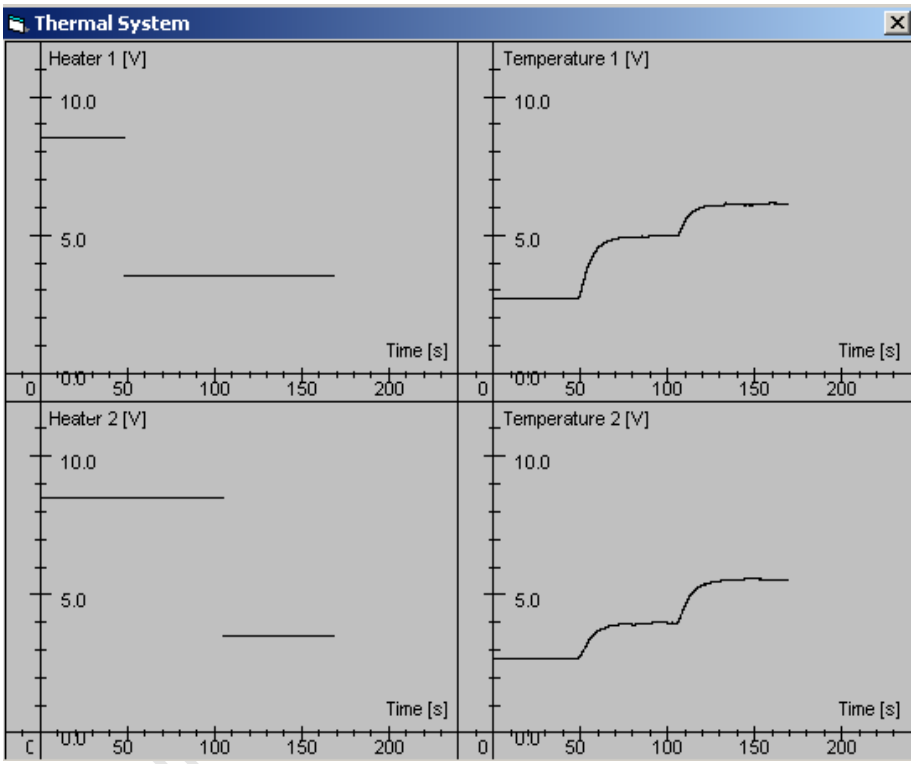


Figure 4.1: Step Test Results: Inputs (Heater1$\Longleftrightarrow$ Input 1, Heater 2$\Longleftrightarrow$ Input 2), Outputs (Temperature 1$\Longleftrightarrow$ Output 1, Temperature 2$\Longleftrightarrow$Output 2

relation is given by

$$G(s) = \frac{Y(s)}{U(s)} = \frac{A}{T_p s + 1} \tag{4.1}$$

where $A$ is the system gain which is given by

$$A = \frac{\triangle Y}{\triangle U}$$

and $T_p$ is the time-constant of the system.

For a multivariable system the model is given by

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & \dots & g_{1j}(s) \\ g_{21}(s) & g_{22}(s) & \dots & g_{2j}(s) \\ \vdots & \vdots & \ddots & \vdots \\ g_{i1}(s) & g_{i2}(s) & \dots & g_{ij}(s) \end{bmatrix} \tag{4.2}$$

where $g_{ij}$ takes the form of Equation 4.1, $i$ is the number of outputs and $j$ is the number of inputs.

From Equation 4.2 and step test data in Figure 4.1, the transfer matrix model was obtained as (Check Appendix D)

$$G(s) = \frac{Y(s)}{U(s)} = \begin{bmatrix} \frac{0.37 \pm 0.01}{(8.0 \pm 0.4)s+1} & \frac{0.2317 \pm 0.002}{(7.6 \pm 0.4)s+1} \\ \frac{0.19 \pm 0.001}{(8.0 \pm 0.4)s+1} & \frac{0.3483 \pm 0.01}{(8.0 \pm 0.4)s+1} \end{bmatrix} V/V \tag{4.3}$$

where $U$(s) is the input measured in volt (V), $Y(s)$ is the output also measured in volts (V) and $G(s)$ is given by (V/V).

The signal limits on both inputs and outputs that can be tolerated or achieved are plotted as shown in Figure 4.2.

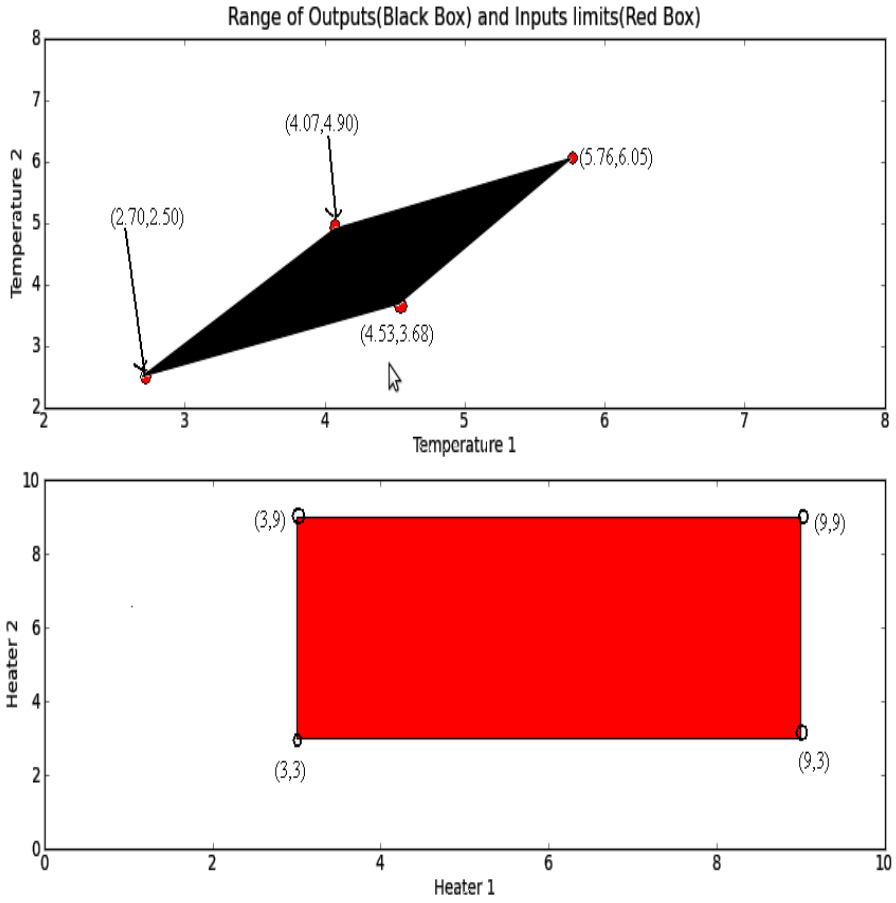


Figure 4.2: Inputs and outputs limits

## 4.2 System Analysis

### 4.2.1 Stability

For a linear system, the system stability must be defined. A linear system is stable if the poles of the transfer matrix are only in the open left-half-plane (OLHP) of the complex plant. This is discussed in depth in ref. [27].

The system poles and zeros were determined using Smith-McMillan form described in ref. [27]. These were found as

- Poles: $[-0.1250, -0.1316, -0.1250]$
- Zeros: $[-0.1353]$

Since all the poles are only on OLHP, i.e. they are all negative, therefore the system is stable. There is only one zero on the OLHP, hence no problem of unstable zeros which introduces difficulties in controlling the system.

### 4.2.2 Diagonal Dominance

For a multivariable system, one of the important properties is that the gain matrix has to be diagonally dominant. This particular condition of a gain matrix is known as diagonal dominance [1]. The dominance is analysed either by rows or columns using Nyquist-Gershgorin bands or circles. For the $G(s)$ given in Equation 4.3, the row Gershgorin bands were found as shown in Figure 4.3.

Since the Gershgorin bands in Figure 4.3 excludes the origin, $G(s)$ is diagonally dominant.

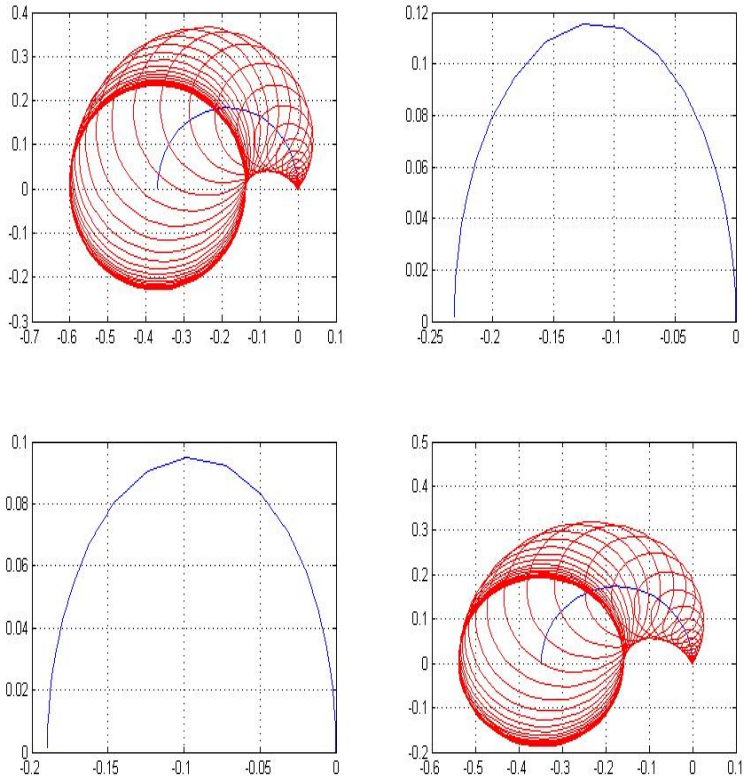


Figure 4.3: Nyquist - Gershgorin plot

### 4.2.3 Constraints Form

The constraints were implemented based only on the input saturation. This is because the inputs on the process were limited to within the range of [3-9] volts. This was to avoid damaging the Thermal heater since any voltage outside the range would damage the heater. Based on the formulation and parameters given in Equation 4.4 for constrained MPC, the constraints to include the limits were defined as

$$\begin{aligned} -\gamma_2 \Delta U &\leq -3 + \gamma_1 u(k-1) \\ \gamma_2 \Delta U &\leq 9 - \gamma_1 u(k-1) \end{aligned} \tag{4.4}$$

where$\gamma_1$ and $\gamma_2$ are the matrices in Equation 4.4 and $u(k-1)$ is the previous control action when at sample instant $k$.

This is written in a constrained standard form as

$$\Omega_1 \Delta U \leq \begin{bmatrix} -3 \\ 9 \end{bmatrix} + \eta \tag{4.5}$$

where

$$\psi = \begin{bmatrix} -\gamma_2 \\ \gamma_2 \end{bmatrix}$$

$$\eta = \begin{bmatrix} \gamma_1 u(k-1) \\ -\gamma_1 u(k-1) \end{bmatrix}$$

## 4.3 State-Space Representation

State-space models are a popular mathematical representation of MIMO systems [28]. They are mostly given in discrete form as shown in Equation

2.1, Section 2.2. Using the matrix transfer function model in Equation 4.3 sampled at 0.5 seconds, the state space equation was obtained as shown in Equation 4.6 ( check Appendix C):

$$x(k+1) = \begin{bmatrix} 0.9394 & 0 & 0 \\ 0 & 0.9363 & 0 \\ 0 & 0 & 0.9394 \end{bmatrix} x(k) + \begin{bmatrix} 0.1212 & 0 \\ 0 & 0.0605 \\ 0 & 0.1212 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} -0.1850 & -0.2439 & 0 \\ -0.0950 & 0 & -0.1742 \end{bmatrix} x(k) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} u(k) \tag{4.6}$$

### 4.3.1 Augmented Model

In every control problem, the idea is to drive a system to a desire set-point in the presence of disturbances. The above formulation, i.e. using the state-space in Equation 2.1 would not drive the system to its desired reference if subjected to disturbances. To overcome these, the system is augmented as shown in Equation 2.3, Section 2.2. This gives

$$\xi(k+1) = \begin{bmatrix} 0.9394 & 0 & 0 & 0 & 0 \\ 0 & 0.9363 & 0 & 0 & 0 \\ 0 & 0 & 0.9394 & 0 & 0 \\ -0.1738 & -0.2283 & 0 & 1 & 0 \\ -0.0892 & 0 & -0.1636 & 0 & 1 \end{bmatrix} \xi(k) + \begin{bmatrix} 0.1212 & 0 \\ 0 & 0.0605 \\ 0 & 0.1212 \\ -0.0224 & -0.0148 \\ -0.0115 & -0.0211 \end{bmatrix} \triangle u(k)$$

$$y(k) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \xi(k) \tag{4.7}$$

The augmentation increases the dimensions of the state-space matrices depending on how many outputs the system has. This is because each output must track its given set-point hence an integrator has to be added for each output. This can be a problem as the number of output increases because the dimensions of the matrices also increase. The larger the matrix dimensionality, the longer it takes to obtain results from the resulting matrix computations. This is a disadvantage if the sampling time is very small since all the control action computations have to be completed within this time. In this thesis, a sampling time of 0.5 seconds was large enough to complete the computation within this time.

### 4.3.2 Controllability and Observability

Controllability and observability are important concepts in the design of state-space control systems, since the design of a controller and observer need a system to be controllable and observable respectively. This is also important in the design of MPC controllers since all the states are to be ob-

served. The Equation 4.6 was found to be both controllable and observable.

## 4.4 Optimisation Costs

When optimisation is performed cost objectives to be used need to be carefully determined. This is because the Pareto front depends entirely on the cost objectives used [7]. As a result, cost functions selected should be as applicable as possible to control engineering. In this thesis, 8 different cost objectives are used which are based on the Integral Square Error (ISE) and Integral Square Control velocity (ISU). These are defined and described in this section and summarized as shown in Table 4.1.

### 4.4.1 Setpoint Tracking Costs

#### Error Costs

Two error costs are proposed which define the error between the set-point, $r$ and the plant output, $y$. These are given as $ISE_{e_1r_1}$ and $ISE_{e_2r_2}$ where

$$e = r - y$$

Since this is a MIMO system, we have different combination of errors, i.e. $e_{ij}$ which is error on output $i$ due to change in set-point $j$. From Equation 4.8, the $t_{i1}$ is the initial time when the output 1 reference is changed while $t_{f1}$ is the final time just before the reference of output 2 is also changed. Similarly, the $t_{i2}$ is the initial time when the output 1 reference is changed while $t_{f1}$ is the final time for the time simulation. Note that the integral errors or control velocities are calculated over different time spans because

| Cost functions | Notation | Mathematical description |
|---|---|---|
| $J_1$ | $ISE_{e_1r_1}$ | $\int_{t_{i1}}^{t_{f1}} (e_{11})^2 \, dt$ |
| $J_2$ | $ISE_{e_1r_2}$ | $\int_{t_{i2}}^{t_{f2}} (e_{12})^2 \, dt$ |
| $J_3$ | $ISE_{e_2r_1}$ | $\int_{t_{i1}}^{t_{f1}} (e_{21})^2 \, dt$ |
| $J_4$ | $ISE_{e_2r_2}$ | $\int_{t_{i2}}^{t_{f2}} (e_{22})^2 \, dt$ |
| $J_5$ | $ISU_{\triangle u_1r_1}$ | $\int_{t_{i1}}^{t_{f1}} (\triangle u_{11})^2 \, dt$ |
| $J_6$ | $ISU_{\triangle u_1r_2}$ | $\int_{t_{i2}}^{t_{f2}} (\triangle u_{12})^2 \, dt$ |
| $J_7$ | $ISU_{\triangle u_2r_1}$ | $\int_{t_{i1}}^{t_{f1}} (\triangle u_{21})^2 \, dt$ |
| $J_8$ | $ISU_{\triangle u_2r_2}$ | $\int_{t_{i2}}^{t_{f2}} (\triangle u_{22})^2 \, dt$ |

Table 4.1: Cost functions design Table

the output references are altered at different times. This ensures that the interactions among output responses can be clearly extracted.

These errors are given as

$$\begin{aligned} ISE_{e_1r_1} &= \int_{t_{i1}}^{t_{f1}} (e_{11})^2 \, dt \\ ISE_{e_2r_2} &= \int_{t_{i2}}^{t_{f2}} (e_{22})^2 \, dt \end{aligned} \tag{4.8}$$

**Controller Velocity Costs**

Another two controller cost functions are defined i.e. $ISU_{\triangle u_1 r_1}$ and $ISU_{\triangle u_2 r_2}$. This gives the amount of controller action that is needed to drive a system output to a steady-state.

$$ISU_{\triangle u_1 r_1} = \int_{t_{i1}}^{t_{f1}} (\triangle u_{11})^2 \, dt$$

$$ISU_{\triangle u_2 r_2} = \int_{t_{i2}}^{t_{f2}} (\triangle u_{22})^2 \, dt$$

where $\Delta u_{ij}$ is controller velocity used to track the output $i$ due to a change in set-point $j$.

### 4.4.2 Interaction Costs

**Interaction Error Costs**

Further two interaction error costs based on $ISE$ are defined. Similarly as in Equation 4.8, these are given as

$$ISE_{e_2r_1} = \int_{t_{i1}}^{t_{f1}} (e_{21})^2 \, dt$$
$$ISE_{e_1r_2} = \int_{t_{i2}}^{t_{f2}} (e_{12})^2 \, dt \tag{4.9}$$

These give the amount of error on each output due to a set-point change on another output.

**Interaction Controller Velocity Costs**

Lastly, interaction based on $ISU$ are defined as

$$ISU_{\triangle u_2r_1} = \int_{t_{i1}}^{t_{f1}} (\triangle u_{21})^2 \, dt$$
$$ISU_{\triangle u_1r_2} = \int_{t_{i2}}^{t_{f2}} (\triangle u_{12})^2 \, dt \tag{4.10}$$

These gives the amount of controller action needed to drive the output $i$ back to its set-point when set-point $j$ is changed.

## 4.5 Controllers Tuning

It's important in control engineering to achieve a specified given performance and stability with minimal overshoots. For a given controller to achieve these, the controller parameters must be properly tuned. Different methods exist for different types of controllers. Traditionally PI Controllers are tuned using Ziegler-Nichols method. Recently MOO has been used for tuning PID controller parameters [6]. Likewise, MPC have no formally defined method of tuning since many parameters are involved in tuning the controller, but different methods have been proposed in literature [3]. Similarly, most re-

cently MOO has been used as a way to find best achievable parameters for given cost objectives as shown in ref. [6].

### 4.5.1 MPC Tuning

MPC is classified as one of the methods which is difficult to tune due to many parameters in consideration [3]. The controller designer had to set the prediction horizon ($N_p$), control horizon ($N_C$), weights on the outputs($\overline{Q}$), and weights on the change in inputs($\overline{R}$). Different tuning methods that have been discovered in the MPC literature are discussed in ref. [3] for which MOO is not used as a way for tuning.

MOO tuning of MPC was discussed in ref. [4, 6] on which cost objectives and evolutionary algorithms are used for tuning the MPC. In this thesis, PDE optimizer was used to find the best achievable performance and tuning parameters for MPC for the given cost functions described in Section 4.4. To tune MPC, five parameter were assumed, that is, prediction horizon ($N_p$), control horizon ($N_C$), move and weighting coefficients for inputs and outputs ($\lambda_1$, $\lambda_2$, $\beta_1$, $\beta_2$) (more detail in Section 5.2).

### 4.5.2 PI Tuning

PI is one of the traditional methods. It has widely been applied in most industries due to its simplicity and competitiveness. Ziegler-Nichols is one of the popular methods used for tuning PI parameters [21] but its generated parameters can still be greatly improved. MOO has been applied to design control systems based on PI/PID [6]. The MOO approach was applied in ref. [6, 29] for tuning PI parameters. In this thesis, four tuning parameters were assumed ($K_{p1}$, $K_{p2}$, $K_{I1}$, $K_{I2}$) for the PI controller.

# Chapter 5

# Simulations and Results



This chapter presents the *Level Diagrams* of the different (MPC, PI) controllers. It further outlines the use of performance measures to compare the Pareto front generated from these controllers. Then simulation time responses of the produced controllers are compared and tested experimentally on the physical system.

## 5.1 PDE Optimizer

Using the design objectives discussed in Table 4.1, the PDE optimizer was run for 50 generations with a crossover rate and mutation rate set to 0.15. The population size for the optimizer was set to be 200 so that enough points were initially generated for the optimizer. The optimizer was developed using Python based on the pseudo-code presented in ref. [24] and some algorithms ported from [30]. The accuracy of the Python Optimizer was verified by comparing the plots generated with the plots given in ref. [24] when the same parameters are used for optimisation (see Appendix B).

## 5.2 Controller Tuning Parameters

### 5.2.1 MPC Parameters

As described above in section 4.5, for the MPC controllers five parameters to be tuned are control horizon, $N_c$, prediction horizon, $N_p$, move and weighting suppression coefficients for each output $(\lambda_1, \lambda_2, \beta_1, \beta_2)$. As discussed in [3], the parameter, $N_p$ does not need to be tuned using an optimizer. This is because $N_p$ can be selected off-line as long as an initial value large enough (greater than largest settling time, which is 12.6 seconds) to cover the system dynamics is chosen. As a result, $N_p$ was set to be 30 samples. Other tuning parameters were tuned using an optimizer and the range on which they are bounded had to be specified. $N_c$ was bounded on the range $[1 - 30]$. This was based on the fact that the value should be greater than 1 [3] but should also be less than or equal to $N_p$. The $(\lambda_1, \lambda_2, \beta_1, \beta_2)$ were each bounded on the range$[0 - 1]$. These weighting coefficients are used to condition the system matrix [2] hence any range can be used.

### 5.2.2 PI Parameters

For the PI controller, four tuning parameters for the optimizer were assumed $(K_{P1}, K_{P2}, K_{I1}, K_{I2})$. Each parameter was bounded on the range$[-2 - 0]$ . The range was selected based on the fact that the model gains of the system were negative hence negative gains were required. Large gains were practically not implementable on the physical system hence lower gains were opted.

## 5.3 Level Diagrams Results

The *Level Diagrams* for different controllers, i.e unconstrained MPC, constrained MPC and PI is shown in Figure 5.1. The 2-norm is used. This is because it provides an accurate evaluation of the conventional geometrical distance to the ideal point and offers a better view of the *'real'* shape of the Pareto front in n-D space [9].

From the plots in Figure 5.1, it can be seen that all points of the unconstrained MPC and constrained MPC are clustered together giving the same Pareto shape. In fact, these controllers seem to perform relatively the same way for all objectives, hence generating similar Pareto fronts. This result could be due to the fact that the constraints that arose during the process were only based on the inputs. The input constraints arise due to the limitation on the amount of voltage that the thermal heaters can tolerate. As a result, the voltages were forced to be limited between 3 volts and 9 volts to avoid damaging the heater system. In addition, since no constraints were imposed on the outputs of the system, there were few constraints that needed to be dealt with hence no performance deviation from both unconstrained and constrained MPC. Finally, dealing with limits on the inputs is easier as long as the set-points set can be achieved within the specified input control range. This is because if the control action calculated at any instant exceeds the limits, it can be saturated by applying the minimum or maximum of the given limits depending on the value of the control action obtained. Therefore, if the control input is below minimum limit, then the minimum limit is used or if it is above maximum limit, then the maximum limit is used.

As a results, the comparison with PI was further detailed using either one of the MPC controllers (unconstrained MPC used (now labelled MPC)).

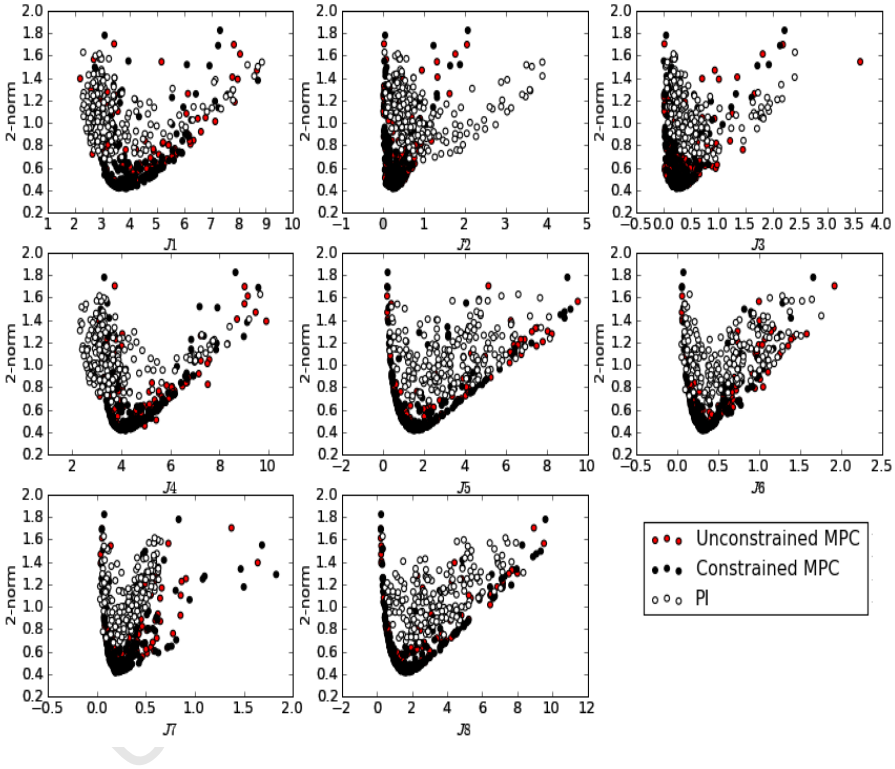


Figure 5.1: *Level Diagrams* of Controllers

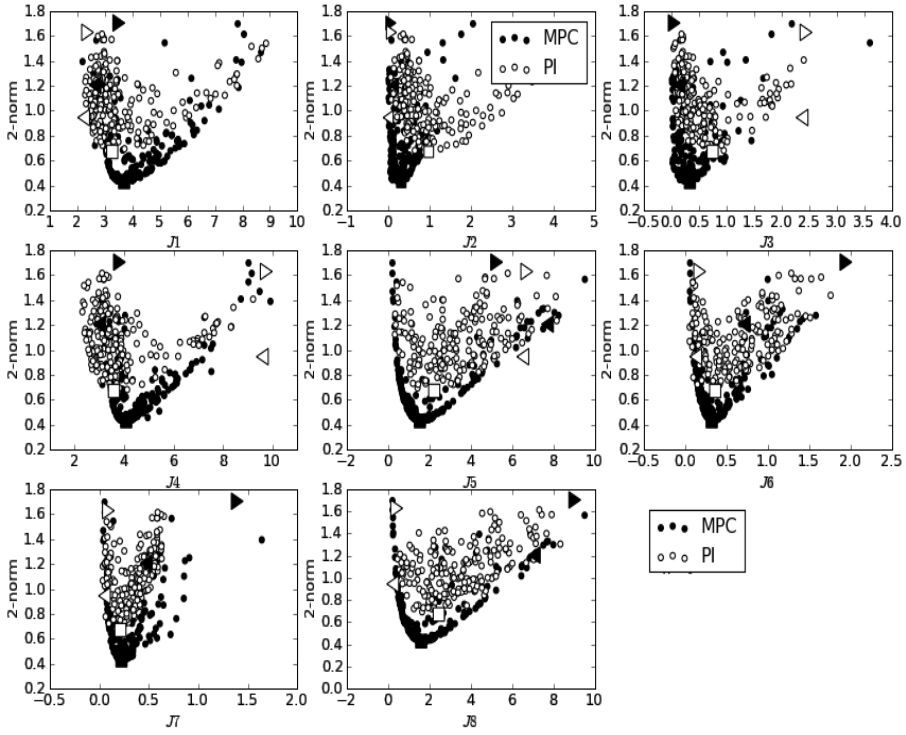


Figure 5.2: *Level Diagrams* of the cost objectives

Unconstrained MPC was selected due to its easier and straight forward design as compared to constrained MPC.

### 5.3.1 MPC versus PI

The *Level Diagrams* plots for the 8-D Pareto front of MPC and PI control methods are shown in Figure 5.2 and for the controller tuning parameters in Figure 5.3 and 5.4.

From the plots, note the points marked with square bracket, left-faced triangle and right-faced triangle represent the minimum, medium, maximum norms respectively.

From the *Level Diagrams* plots, number of observations can be drawn. In

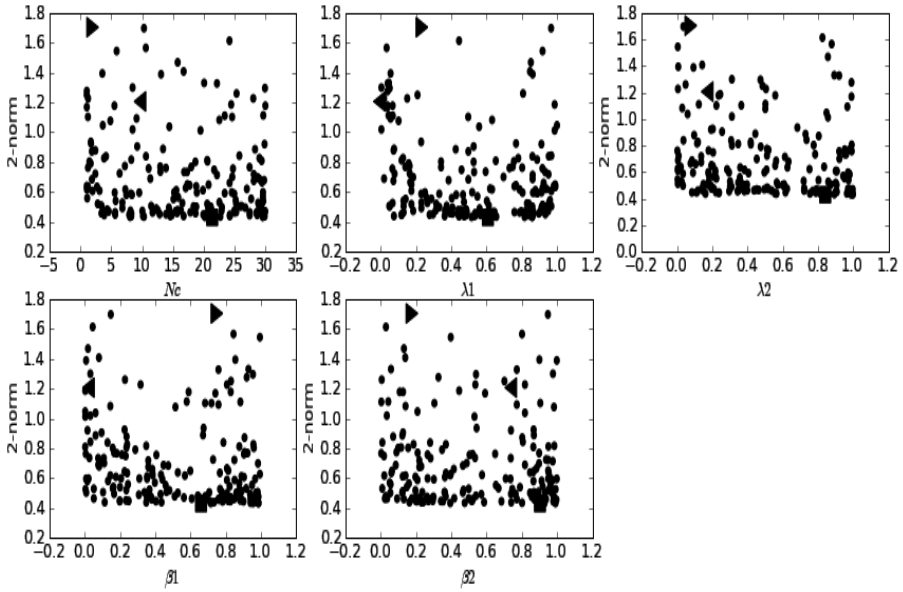


Figure 5.3: *Level Diagrams* of the MPC tuning parameters

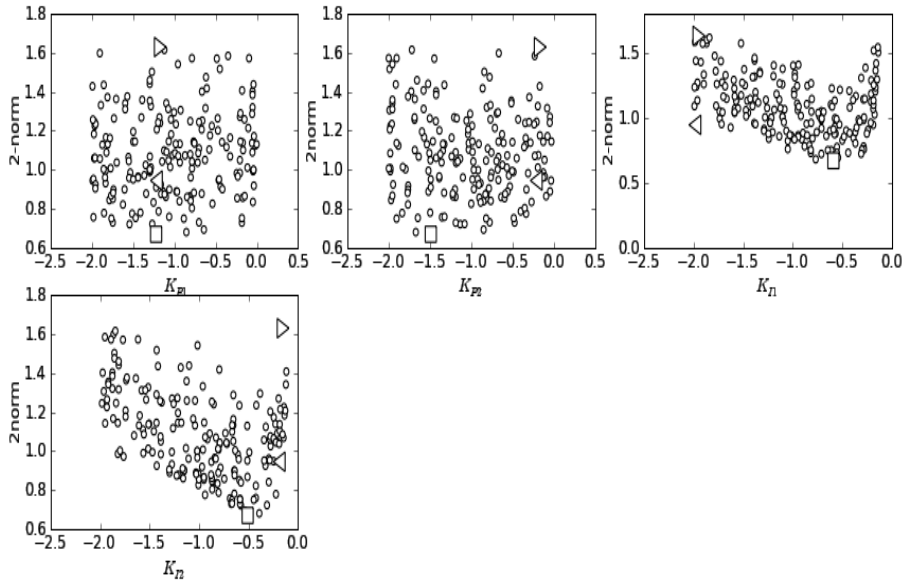


Figure 5.4: *Level Diagrams* of the PI tuning parameters

Figure 5.2, by observing all the plots, MPC gives more points with lower norm than PI. This shows that MPC have more points concentrated towards an ideal point hence generally a better method to use for controller design.

However, on the $J_1$ and $J_4$ plots most points for PI seem to be more concentrated towards the lower values of the cost objectives (i.e $J_1 \approx J_4 \approx 2$) than MPC. If minimising these two cost objectives is of higher priority for the DM then PI gives more chances of getting a minimum values for both cost objectives. These imply that PI gives a better set-point tracking on both outputs. However care must be taken as the points could be far worse in other objectives. Looking at $J_1$ plot it can be seen that most points for MPC are concentrated below 1 with a maximum value less than 2 while looking at PI all points are distributed over the range [0-4]. This shows that MPC gives a better interaction rejection on output 1 than PI.

From $J_3$ plot, both controllers' points seem to be clustered around the same values hence gives no significant difference in output 2 interaction rejection. In general, PI gives better set-point tracking performances but worse in interaction rejection performances.

For the controller effort plots, i.e $J_5$ and $J_8$, all points for both controller seem to be widely spread for all objective values. This means that in terms of the amount of the controller effort applied, both controllers perform similarly. However for MPC, it can be observed that the norm of the objectives decrease with an increase in objective values until a certain value (i.e $J_5 \approx 1.7$, $J_6 \approx 0.3$,$J_7 \approx 0.25$,$J_5 \approx 1.9$) and then start increasing.

Figure 5.3 shows the plots for MPC tuning parameters. For all the tuning parameters, the points seem to be distributed over the whole range of values with no range preferably better than any other. This shows the difficulty of

tuning MPC parameters. This is different for PI tuning parameters (Figure 5.4), as it can be seen from $K_{I1}$ and $K_{I2}$ plots that lower norms can be achieved with lower absolute values of $K_{I1}$and $K_{I2}$.

### 5.3.2 Performance Measures Results

Quality indicators as opposed to visualisation tools can be used to compare the Pareto fronts in a qualitative manner. Two of the mostly applied indicators $I_H$ and $I_{H2}$ discussed in section 3.3 are applied to compare the Pareto fronts generated from MPC and PI controllers. These performance indicator coverage region is hard to visualise in an n-D space (n>3). This is because of the computer graphical space (for now) which can clearly visualize dimensions up to 3-D. As a result, to enable an easier indicators space region view, an illustrative example using a 2-D Pareto front from two cost functions for both MPC and PI controllers is shown in Figure 5.5.

From Figure 5.5, *REGION A* is in a set of points (or objectives) which can be achieved only by a PI controller. MPC controller cannot achieve any of the given objective in this region because its Pareto set lie above the region. As a result, if the design objective is to achieve a performance in this region, then the control designer will be forced to use a PI control method. This region lie inside the hypervolume of PI, ($REGION\ A \in I_H(PI)$) but outside the hypervolume of MPC, ($REGION\ A \notin I_H(MPC)$). This set of points in this region is denoted by $I_{H2}(PI, MPC)$.

*REGION C* gives a set of points (or objectives) which can be achieved only by MPC controller. Similarly, the region lie inside the hypervolume of MPC, ($REGION\ A \in I_H(MPC)$) but outside the hypervolume of PI, ($REGION\ A \notin I_H(PI)$). This is denoted by $I_{H2}(MPC, PI)$. The points in *REGION B*,

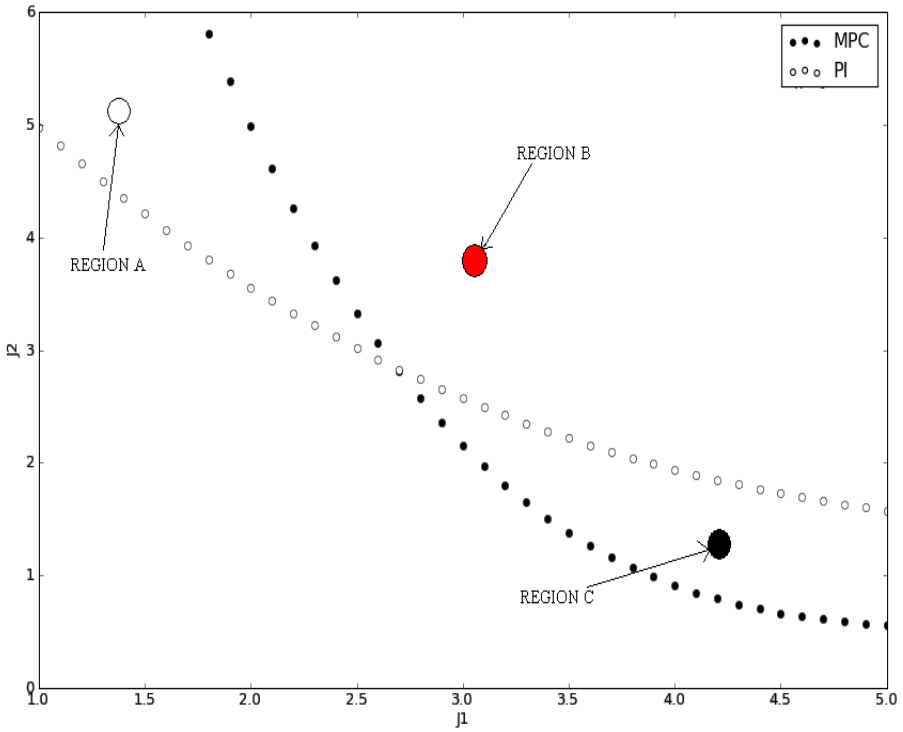


Figure 5.5: Performance Measures in a 2-D space

| Controller | $I_H$ | $I_{H2}$ |
|---|---|---|
| MPC | 0.53737 | 0.1098 |
| PI | 0.51923 | 0.09166 |

Table 5.1: The Performance Measure Results for MPC and PI

can be achieve by both control methods since they lie inside both controllers hypervolume, i.e, $REGION\ B \in I_H(MPC)$ and $REGION\ B \in I_H(PI)$.

The space region covered by these indicators is mostly given in percentage. This gives the number (in percentage) of the points that are found in that region as per total number of points that are uniformly distributed for the entire space coverage.

Using the design objectives given in Table 4.1, the results for the 8-D Pareto front for each control method are tabulated as shown in Table 5.1.

Using dominance relations discussed in Table 3.1 in section 2.5, a number of observations can be drawn from Table 5.1. MPC coverage space is better than PI since $I_H(MPC) \succ I_H(PI)$. Thus it shows that MPC dominate a bigger portion of the objective space, and therefore gives more chances of obtaining different performances in a given space.

Looking at the $I_{H2}$ indicator both controllers give a value which is closer to 0.1 i.e. $I_{H2}(MPC, PI) \geq I_{H2}(PI, MPC)$. This shows that both MPC and PI can considerably produce similar portions that each controller is not able to produce indivisually. From all these, it can be concluded that MPC Pareto front gives a better space coverage than PI front (MPC $\triangleright$ PI). However, it is difficult to decide which controller is actually better than the other from the indicators values because both controllers do not seem to entirely dominate the other.



## 5.4 Simulation and Experimental Time Responses

Time responses are also significant since they show the behaviour of the system in time domain. These enable the control designer to observe some important control measures like overshoots, settling time, oscillations on the output, and saturation on the input. These control measures cannot be observed from the *Level Diagrams* plots.

### 5.4.1 MPC Time Responses

Three points for MPC controller that give the minimum (min), maximum (max) and medium (mid) norms (noted with black square, left-faced triangle, right-faced triangle respectively) were selected from the *Level Diagrams*

| Norm | Norm Values | Input Parameters | Cost Objectives Values |
|---|---|---|---|
| Min | 0.428 | $N_C = 2, \lambda_1 = 0.6073$<br>$\lambda_2 = 0.841, \beta_1 = 0.659$<br>$\beta_2 = 0.9026$ | $J_1 = 3.74, J_2 = 0.028$<br>$J_3 = 0.02, J_4 = 3.96$<br>$J_5 = 2.73, J_6 = 0.73$<br>$J_7 = 0.45, J_8 = 2.93$ |
| Max | 1.704 | $N_C = 1, \lambda_1 = 0.221$<br>$\lambda_2 = 0.067, \beta_1 = 0.738$<br>$\beta_2 = 0.165$ | $J_1 = 3.20, J_2 = 0.064$<br>$J_3 = 0.054, J_4 = 3.27$<br>$J_5 = 5.159, J_6 = 1.26$<br>$J_7 = 1.048, J_8 = 8.944$ |
| Mid | 1.206 | $N_C = 9, \lambda_1 = 0.0028$<br>$\lambda_2 = 0.183, \beta_1 = 0.031$<br>$\beta_2 = 0.748$ | $J_1 = 2.607, J_2 = 0.221$<br>$J_3 = 0.241, J_4 = 2.997$<br>$J_5 = 7.814, J_6 = 0.63$<br>$J_7 = 0.4381, J_8 = 7.8$ |

Table 5.2: Different MPC Norm values and corresponding cost and input parameters

plots. Their cost objectives and input parameters values are tabulated as shown in Table 5.2. The given time response plots are shown in Figure 5.6.

From Table 5.2 and Figure 5.6, the performances of these controllers seem to be competitive with each other. Generally, all the responses do not have overshoots in their output responses and have settling time ( $t_{\pm 2\%}$) below 11.5 seconds. Moreover, the mid norm gives a good set-point tracking responses as compared to both max and min norms with min norm worse than both. However, by looking at the interaction rejection responses (or cost values), the minimum norm is far better than both mid and max norms with mid worse than max. Looking more closely on the controller action responses, mid and max seem to be having larger values of controller actions as compared to min and their controller responses saturate faster and remain in saturation longer than the min norm. In fact, there seem to be a trade-off between

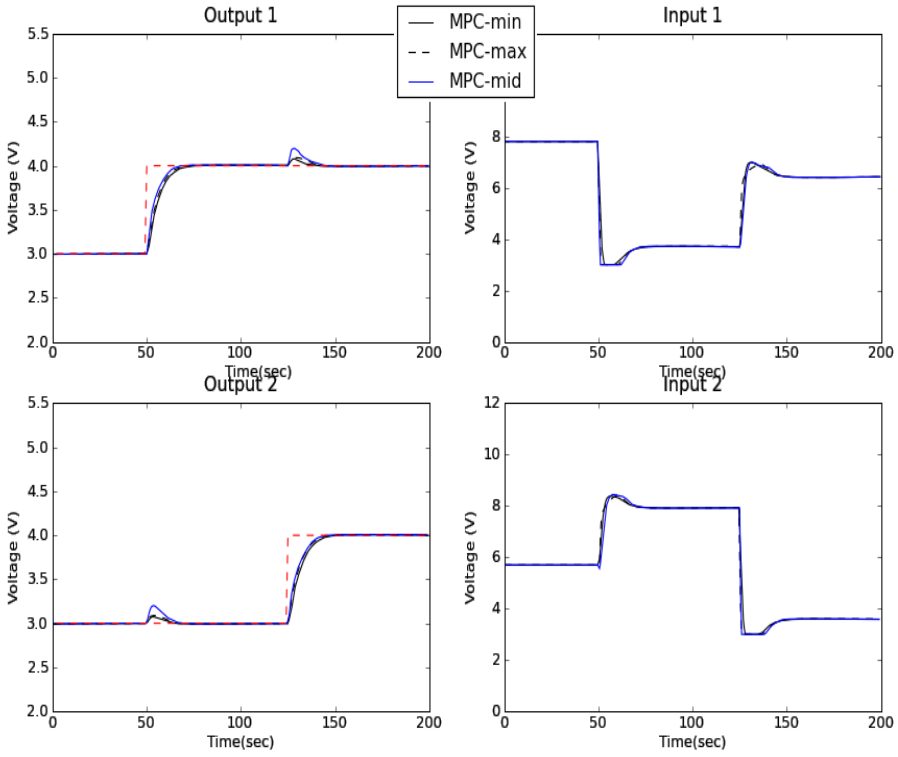


Figure 5.6: Different MPC Norms Time Responses

the set-point tracking, interaction rejection and amount of controller action applied. Summarising these, it can be concluded that the better the set-point tracking, the worse the interaction rejection and the more the controller effort needed.

These results might be dictated by the input parameters given to the controller especially the penalty weights on the inputs. This is because the more the penalty weight on a certain input, the less the movement in the control action of that input, hence a slower response on the corresponding output.

In conclusion, by observing the responses and the cost objective values, on average the min norm seem to be giving the better performance as compared to the other norms.

### 5.4.2 PI Time Responses

Similar to MPC, three points that gives min, max and mid responses (noted with white square, left-faced triangle, right-faced triangle respectively) were picked and their cost objectives and input parameters values are tabulated as shown in Table 5.3 with the corresponding time responses shown in Figure 5.7.

The time responses for the PI look more interesting as the number of observations can be singled out. Looking at the set-point tracking for output 1, the max norm gives lower value for this cost objective (i.e, $J_1$ ) than both min and mid norms, but the corresponding response is oscillatory. If a lower value for this objective is of priority, it could be tempting to use tuning parameters that offer this cost value, but as it seems, it is a worse controller choice due to oscillations on the output response. However, the interaction rejection encountered on the other output (i.e. output 2) while tracking

| Norm | Norm Values | Input Parameters | Cost Objectives Values |
|---|---|---|---|
| Min | 0.669 | $K_{P1} = -1.221$<br>$K_{P2} = -1.489$<br>$K_{I1} = -0.587$<br>$K_{I2} = -0.509$ | $J_1 = 3.26, J_2 = 0.98$<br>$J_3 = 0.75, J_4 = 3.59$<br>$J_5 = 2.23, J_6 = 0.36$<br>$J_7 = 0.21, J_8 = 2.45$ |
| Max | 1.628 | $K_{P1} = -1.195$<br>$K_{P2} = -0.186$<br>$K_{I1} = -1.968$<br>$K_{I2} = -0.171$ | $J_1 = 2.30, J_2 = 0.038$<br>$J_3 = 2.41, J_4 = 9.66$<br>$J_5 = 6.62, J_6 = 0.15$<br>$J_7 = 0.071, J_8 = 0.32$ |
| Mid | 0.945 | $K_{P1} = -1.993$<br>$K_{P2} = -0.030$<br>$K_{I1} = -0.185$<br>$K_{I2} = -0.876$ | $J_1 = 5.95, J_2 = 2.50$<br>$J_3 = 0.19, J_4 = 3.42$<br>$J_5 = 2.43, J_6 = 0.13$<br>$J_7 = 0.14, J_8 = 1.95$ |

Table 5.3: Different PI Norm values and corresponding cost and input parameters

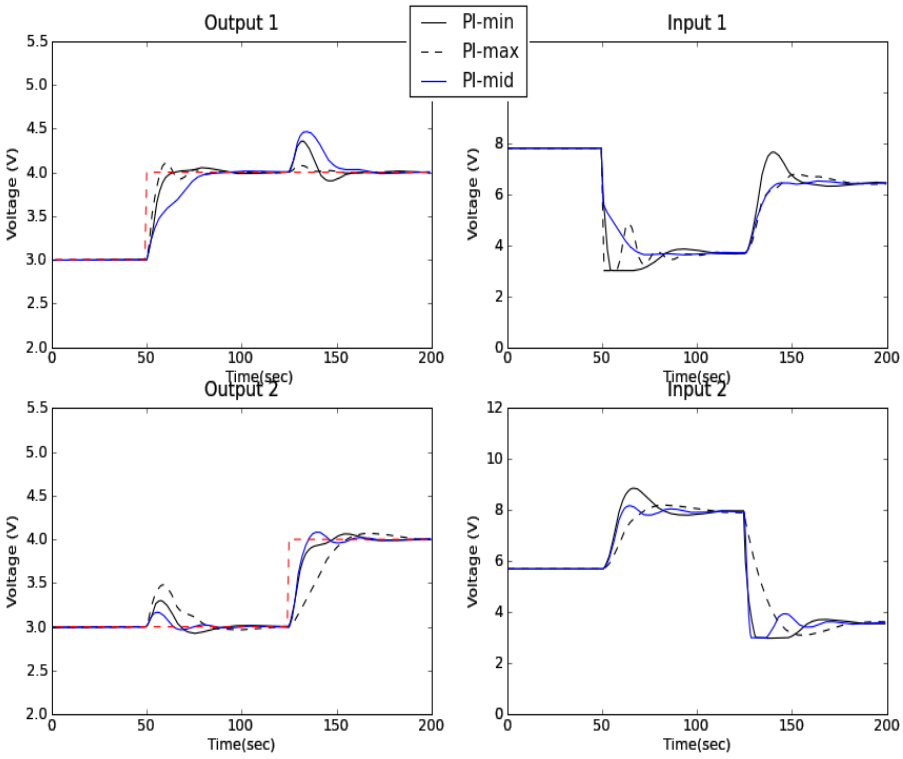


Figure 5.7: Different PI Norms Time Responses

output 1 seems worse than other norm even though it's a bit oscillatory.

The set-point tracking for output 2 seem to be a different scenario. The max norm gives a far worse response than min and mid norms. In fact, the cost value for this objective (or $J_4$) is almost double in value as compared with corresponding values of other norms. This might be due to small gains ($K_{P2}$ and $K_{I2}$) that are associated with this output. However, when it comes to interaction rejection output 1 when tracking output 2, this seems to give a far much better response, with a cost value which is almost near zero. In essence, the best set-point tracking on one output is achieved at the expense of worse interaction rejection on the other output. This trade-off can even be observed on the other norms' responses.

By observing the overshoots encountered by all norms, it can be seen that both max and mid norms give responses with considerable amount of overshoot when tracking the corresponding set-points while min norm gives a minimal overshoots. Furthermore, on average the controller effort used on both set-point tracking and interaction rejection is better for min norm as compared with other norms.

These observations found results due to the fact that an attempt to drive one output to its set-point in a short time span (or small settling time), makes the controller effort of the corresponding input harsh. Since the system is fully interactive, the other output is considerably affected hence results into bigger interaction response.

### 5.4.3 MPC versus PI

For controllers comparison, the minimum norms from both controllers were used since on average, they provide a better performance and a best compro-

| Controller | Norm Values | Input Parameters | Cost Objectives Values |
|---|---|---|---|
| MPC | 0.428 | $N_C = 2, \lambda_1 = 0.6073$<br>$\lambda_2 = 0.841, \beta_1 = 0.659$<br>$\beta_2 = 0.9026$ | $J_1 = 3.74, J_2 = 0.028$<br>$J_3 = 0.02, J_4 = 3.96$<br>$J_5 = 2.73, J_6 = 0.73$<br>$J_7 = 0.45, J_8 = 2.93$ |
| PI | 0.669 | $K = -1.221$<br>$K_{P2} = -1.489$<br>$K_{I1} = -0.587$<br>$K_{I2} = -0.509$ | $J_1 = 3.26, J_2 = 0.98$<br>$J_3 = 0.75, J_4 = 3.59$<br>$J_5 = 2.23, J_6 = 0.36$<br>$J_7 = 0.21, J_8 = 2.45$ |

Table 5.4: Tuning Parameters and Cost Values for MPC and PI

mise solution that is closest to the ideal point (i.e. point with norm closest to zero) in the objective space. The tuning parameters for these points are noted with square brackets on Figure 5.3 and 5.4. Their norm values, tuning parameters and cost objective values are also tabulated in Table 5.4. The time responses are compared as shown in Figure 5.8.

The experimental results sampled at 0.5 seconds for the same parameters given in Table 5.4 were conducted and plotted as shown in Figure 5.9.

From the results in Figure 5.8 and 5.9, MPC gives a better response in interaction elimination than PI controller. The PI gives slower interaction responses which are also oscillatory. However, when it comes to set-point tracking, PI controller seem to perform somehow better than MPC. These might be due to the fact that the PI design (contrary to the MPC design) does not include knowledge of the interaction model and hence when the interaction occurs it appears to controller as a disturbance. However, its

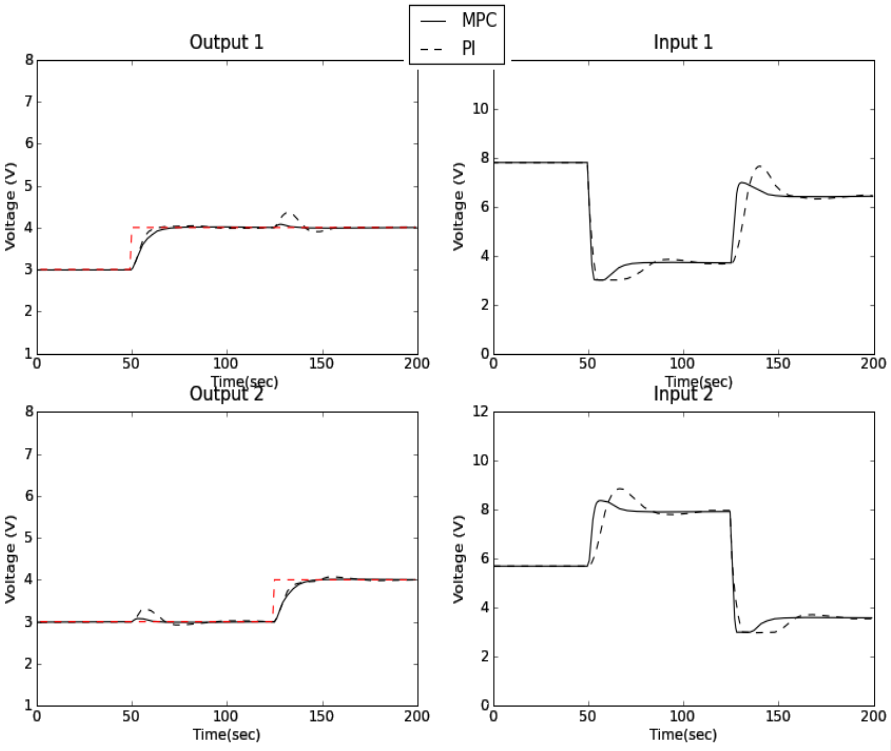


Figure 5.8: Simulation results for system response

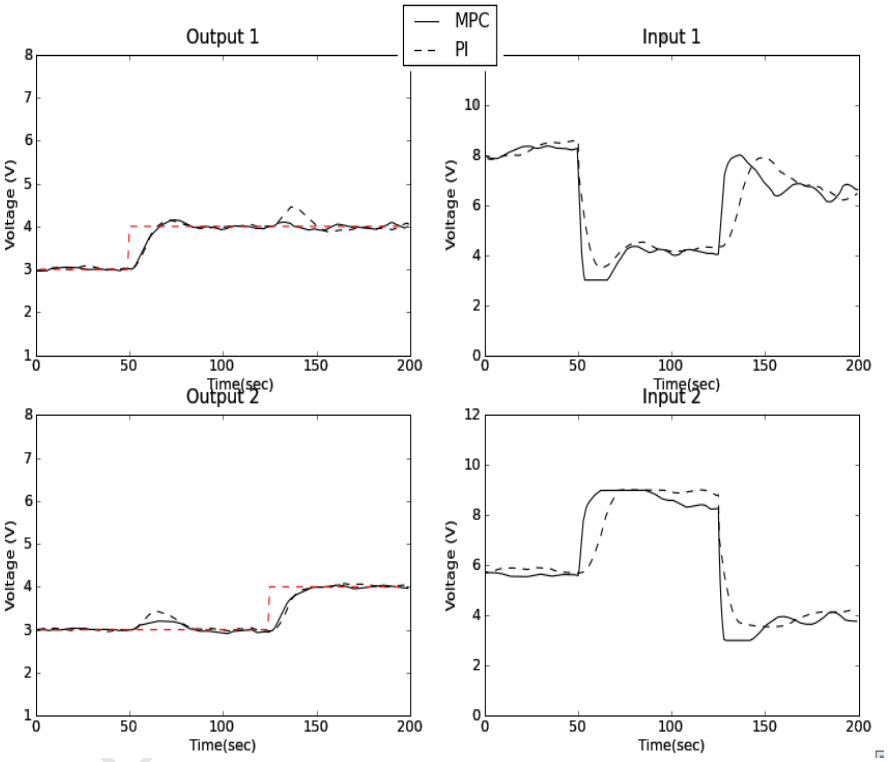


Figure 5.9: Experimental results for the controllers

competitive performance for set-point tracking might be due to the high gains introduced in the gain parameters. In general, these plots are also found to conform to the analysis and conclusions derived from the *Level Diagrams* plots.

# Chapter 6

# Conclusions



## 6.1 General Conclusions

The MOO designs for MPC and PI controllers have been outlined for a multi-variable process. The optimum performances of these controllers were generated and comparisons were conducted using the *Level Diagrams* visualization tool.

It was found that both controllers are competitive when compared to each other. MPC was found to be better in interaction rejection than PI but however when it comes to set-point tracking both controllers' performance were relatively competitive with PI better in most scenarios.

Further, *Level Diagrams* were found to be useful tool for visualization and interpretation of the higher dimensions of the Pareto fronts which seem to have been difficult with other methods. However, care must be taken in the analysis using this tool since some points which are better in some objectives might be the worst in other objectives. *Level Diagrams* further help and guide the Decision Maker in selection of the tuning parameters for

a given controller that gives a compromise or better average performance. The parameters of this controller were used for controller designs which were implemented or tested experimentally on the system.

Quantitative analysis of the controllers' Pareto fronts were also investigated using quality measures (unary and binary hypervolume). Although these indicators do not give a clear vision of which objective is better as compared to the corresponding one on another controller, they do help in giving a view of which controller would dominate another in terms of the coverage space in the objective space.

Furthermore, time response plots have also helped in giving an inside into performance of the controllers in time domain. This has enabled a more detailed analysis and comparison of the controllers since some control performance measures (e.g. oscillations, overshoots, saturation of controller actions, settling time of the responses) that were not observed in *Level Diagrams* plots were further quantified.

Finally, testing the controllers in a physical system was also a challenge since bigger gains on the controller and observer had to be avoided. This is because a small noise that comes into the system might be greatly amplified by these higher gains hence resulting in oscillatory responses.



## 6.2 Future Work

Investigation is further needed in the accuracy of the approximation of the Pareto fronts generated from the Evolutionary algorithms so as to improve the accuracy of comparisons and design for different control methods using these tools.

Increase in the number of cost objectives to represent more performance measures (for example damping, overshoots, etc) should also be considered. However care must be taken because with the increase in cost objectives, the complexity of the Pareto fronts analysis and interpretation becomes more difficult. As a result, the analyses of the higher dimensional Pareto fronts still remain a challenge. Hence visualization tools for n-D Pareto front that are easier and straight forward to interpret their plots need to be invented. Moreover, because of the increases in complexity of the Pareto front analysis when dealing with many cost objectives, deep investigation into the reduction of the number of cost objectives that can be used without losing significant information might be a breakthrough.

Furthermore introducing the controller gain values as cost objectives should also be investigated, especially for PI controllers since their larger gain parameters can greatly affect the performance of the controller both negatively and positively. This is because larger gains seem to improve the set-point tracking performance of the controller but at the expense of oscillatory response.

# Bibliography


[1] P. Albertos and A. Sala, *Multivariable Control System: An Engineering Approach.* Springer, 2004.

[2] L. Wang, *Model predictive Control System Design and Implemetation using MATLAB.* Springer-Verlag London Limited, 2008.

[3] J. L. Garriga and M. Soroush, "Model predictive control tuning methods: A review," *Indust. Chem. Eng. Res.*, vol. 49, pp. 3505–3515, 2010.

[4] G. P. Liu, J. B. Yang, and J. F. Whidborne, *Multiobjective Optimisation and Control.* Research Studies Press Ltd, 2003.

[5] J. V. der Lee, W. Svrcek, and B. Young, "A tuning algorithm for model predictive controller based on genetic algorithm and fuzzy decision making," *ISA Transactions*, vol. 47, pp. 53–59, 2008.

[6] G. Adrian, "Mpc and pid control based on multi-objective optimization," *American Control Conference*, June 2008.

[7] D. Moore, "Optimal controller comparison using pareto fronts," Master's thesis, University of Cape Town, Dec. 2009.

[8] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evol. Comp.*, vol. 2, Apr. 2003.

[9] X. Blasco, J. Herrero, J. Sanchis, and M. Martinez, "A new graphical visualization of n-dimensional pareto fro decision-making in multiobjective optimization," *Information Sciences*, vol. 178, pp. 39068–3924, 2008.

[10] J. B. Kollat and P. Reed, "A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (video)," *Enviromental Modelling & Software*, vol. 22, pp. 1691–1704, 2007.

[11] C. Garcia, D. Prett, and M. Morari, "Model predictive control: Theory and practice- a survey," *Automatica*, 1989.

[12] A. Bemporad and M. Morari, "Robust model predictive control: A survey," *Automatic Control Laboratory, Swiss Federal Institute of Technology(ETH), Physikstrasse*, vol. 3, 1999.

[13] J. Rossiter, *Model-Based Predictive Control: A Practical Approach.* CRC Press, 2003.

[14] J. M. Maciejowski, *Predictive Control with Constraints.* Prentice Hall, 2001.

[15] C. E.F. and C. Bordons, *Model Predictive Control.* Springer, 2004.

[16] A. haj Ali, A. Ajbar, E. Ali, and K. Alhumaizi, "Robust model-based control of a tubular reverse-osmosis desalination unit," *Desalination*, vol. 255, pp. 129–136, 2010.

[17] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Journal of Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[18] A. Gonzalez and E. Adam, "Conditions for offset elimination in state space receding horizon controllers: A tutorial analysis," *Chem. Eng. Proc.*, vol. 47, pp. 2184–2194, 2008.

[19] W. Y. Yang, W. Cao, T.-S. Chung, and J. Morris, *Applied Numerical Method in MATLAB*. John Wiley & Sons, Inc., 2005.

[20] S. F. G., "Robust and adaptive control of an unknown plant: A benchmark of new format," *Automatica*, vol. 30, no. 4, pp. 567–575, 1994.

[21] M. S. Fadali, *Digital Control Engineering: Analysis and Design*. Elsevier Inc., 2009.

[22] D. E. Grierson, "Pareto multi-criteria decision making," *Advanced Engineering Informatics*, vol. 22, pp. 371–384, 2008.

[23] J. Andersson, "A survey of multiobjective optimization in engineering design," tech. rep., Linköping University, Sweden, 2000.

[24] H. A. Abbas and R. A. Sarker, "The pareto differential evolution algorithm," *Internation Journal on Artificial Intelligence Tools*, vol. 11, no. 4, pp. 531–552, 2002.

[25] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," tech. rep., Computer Engineering (TIK), ETZ Zurich, 2007.

[26] R. Berghamme, T. Friedrich, and F. Neumann, "Set-based multiobjective optimization, indicators, and deteriorative cycles," in *Proceed-*

*ings of the 12th annual conference on Genetic and evolutionary computation*, 2010.

[27] J. Maciejowski, *Multivariable Feedback Design*. Addison-Wesley-Publishers Ltd., 1989.

[28] M. Braae, "Design of multivariable systems," 1994.

[29] A. Popov, A. Farag, and H. Werner, "Tuning of a pid controller using a multi-objective optimisation technique applied to a neutralization plant," in *Proc. of the 44th IEEE Conference on Decision and Control*, Dec. 2005.

[30] *(2009, Oct.)[Online].www.dacya.ucm.es/Jam/downloads/DESolver.py.*

# Appendix A

# Cost Optimization



In model predictive control, the objective is to find the best control parameter vector such that the error between the set-point and the predicted output is minimized.

The cost function $J$ that is to be optimised is given as

$$J = (R_s - Y)^T \bar{Q}(R_s - Y) + \triangle U^T \bar{R} \triangle U \tag{A.1}$$

$$R_s^T = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \end{bmatrix} r(k)$$

$$\bar{R} = \lambda I_{(N_c \times N_c)}$$

$$\bar{Q} = \beta I_{(N_p \times N_p)}$$

where $Y$ is the predicted output vector, $r(k)$ represent the set-point, $\overline{Q}$ & $\overline{R}$ are the weighting coefficients for the output error and control action velocities respectively, with $\beta > 0$ and $\lambda \geq 0$. $N_c$ and $N_P$ represent the control horizon

and prediction horizon respectively.

The predicted output $Y$ is expressed in compact form as:

$$Y = Fx(k) + \Phi \triangle u \tag{A.2}$$

where

$$F = \begin{bmatrix} \tilde{C}\tilde{A} \\ \tilde{C}\tilde{A}^2 \\ \tilde{C}\tilde{A}^3 \\ \vdots \\ \tilde{C}\tilde{A}^{N_P} \end{bmatrix}, \Phi = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & 0 & ... & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & 0 & ... & 0 \\ \tilde{C}\tilde{A}^2\tilde{B} & \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{N_P-1}\tilde{B} & \tilde{C}\tilde{A}^{N_P-2}\tilde{B} & \tilde{C}\tilde{A}^{N_P-3}\tilde{B} & ... & \tilde{C}\tilde{A}^{N_P-N_C}\tilde{B} \end{bmatrix}$$

and

$$Y = \begin{bmatrix} y(k+1|k) & y(k+2|k) & y(k+3|k) & ... & y(k+N_p|k) \end{bmatrix}^T$$

$$\triangle U = \begin{bmatrix} \triangle u(k) & \triangle u(k+1) & \triangle u(k+2) & ... & \triangle u(k+N_c-1) \end{bmatrix}^T$$

and $\tilde{A}$, $\tilde{B}$, & $\tilde{C}$ are the augmented matrix given by:

$$\tilde{A} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}, \tilde{B} = \begin{bmatrix} B \\ CB \end{bmatrix}, \tilde{C} = \begin{bmatrix} 0 & I \end{bmatrix}$$

To find the optimal change in control actions, $\triangle U$, $J$ in Equation A.1 is rewritten as:

$$J = (R_s - Fx(k))^T (R_s - Fx(k)) - 2\triangle U^T \Phi^T (R_s - Fx(k)) + \Delta U^T (\Phi^T \Phi + \overline{R}) \triangle U \tag{A.3}$$

Differentiating the cost function $J$ in terms of $\triangle U$, we get

$$\frac{\partial J}{\partial \triangle U} = -2\Phi^T (R_s - Fx(k)) + 2(\Phi^T \Phi + \overline{R}) \triangle U \tag{A.4}$$

For optimal condition:

$$\frac{\partial J}{\partial \triangle U} = 0$$

from which we get the optimal solution for control actions as

$$\triangle U = \left(\Phi^T \Phi + \overline{R}\right)^{-1} \Phi^T (R_s - Fx(k)) \tag{A.5}$$

# Appendix B

# The Pareto Differential Evolution (PDE) algorithm



Pareto Differential Evolution is one of the popular Evolutionary Algorithms (EA) that is used to generate the Pareto front approximation. Its Pseudocode is described in Section B.1.

## B.1 Pseudocode

*let $G$ denote a generation, $P$ a population of size $M$, and $\overrightarrow{x}^{j}_{G=k}$ the $j^{th}$ individual of dimension $N$ in population $P$ in generation $k$, and $CR$ denotes the crossover probability*

***input*** *$N$, $M \geq 4$,$\alpha$,$CR \in [0,1]$, and initial bounds: lower ($x_i$), upper ($x_i$) , $i = 1, \ldots, N$*

*initialize $P_{G=0} = \{\overrightarrow{x}^{1}_{G=0}, \ldots, \overrightarrow{x}^{M}_{G=0}\}$ as*

**for each** *individual $j \in P_{G=0}$:*

$\overrightarrow{x}^{j}_{i,G=0} = Guassian(0.5, 0.15), i = 1, \ldots, N$

*Repair* $\overrightarrow{x}^{j}_{G=k}$ *if any variable is outside its boundaries*

**end for each**

*evaluate* $P_G = 0$

$k = 1$**return**

**while** *the stopping criterion is not satisfied* **do**

*remove all dominated solutions from* $P_{G=k-1}$,

**if** *the number of non-dominated solutions in* $P_{G=k-1} > \alpha$,

*then apply the neighborhood rule*

**end if**

**for** $j = 0$ *to number of non-dominated solutions in* $P_{G=k-1}$

$\overrightarrow{x}^{j}_{G=k} \leftarrow \overrightarrow{x}^{j}_{G=k-1}$

**end for**

**while** $j \leq M$

*randomly select* $r_1, r_2, r_3 \in (1, \ldots, \alpha)$, *from the non-dominated solutions of* $P_{G=k-1}$, *where* $r_1, \neq r_2, \neq r_3$

*randomly select* $i_{rand} \in (1, \ldots, N)$

**forall** $i \leq N$

$$\overrightarrow{x}^{j}_{i,G=k} = \begin{cases} \overrightarrow{x}^{r_3}_{i,G=k-1} + Guassian(0,1) \times \left(\overrightarrow{x}^{r1}_{i,G=k-1} - \overrightarrow{x}^{r2}_{i,G=k-1}\right) \\ \qquad if\,(random\,[0,1) < CR \wedge i = i_{rand} \\ \overrightarrow{x}^{j}_{i,G=k-1} \qquad\qquad otherwise \end{cases}$$

**end for all**

*Repair* $\overrightarrow{x}^j_{G=k}$ *if any variable is outside its boundaries*

**if** $\overrightarrow{x}^j$ *dominates* $\overrightarrow{x}^{r3}_{G=k-1}$ *then*

$\overrightarrow{x}^j_{G=k} \leftarrow \overrightarrow{x}^j$

$j = j + 1$

**end if**

**end while**

$k = k + 1$

**end while**

**return** *the set of non-dominated solutions*

## B.2 Test Problems

The Python code of the Pseudocode was generated and tested using the two benchmark problems used in ref. [24]. Both test problems used have two objective functions to be optimised and thirty input variables. The optimiser was run for a maximum of 200 generation with an initial population set to 400. The crossover rate was set to 0.15. The first benchmark problem (Test Problem 1 (Equation B.1)) gives a convex Pareto-front as shown in Figure B.1 while the second problem (Test Problem 2 (Equation B.2)) gives a discontinuous Pareto-front shown in Figure B.2.

$$f_1(x) = x_1$$

$$f_2(x) = g \times (1 - \sqrt{\frac{f_1}{g}}) \qquad (B.1)$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)}$$

$$x_i \in [0, 1], i = 1, \ldots, 30$$

$$f_1(x) = x_1$$

$$f_2(x) = g * (1 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g}) sin(10\pi f_1)) \qquad (B.2)$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^{n} x_i)}{(n-1)}$$

$$x_i \in [0, 1], i = 1, \ldots, 30$$

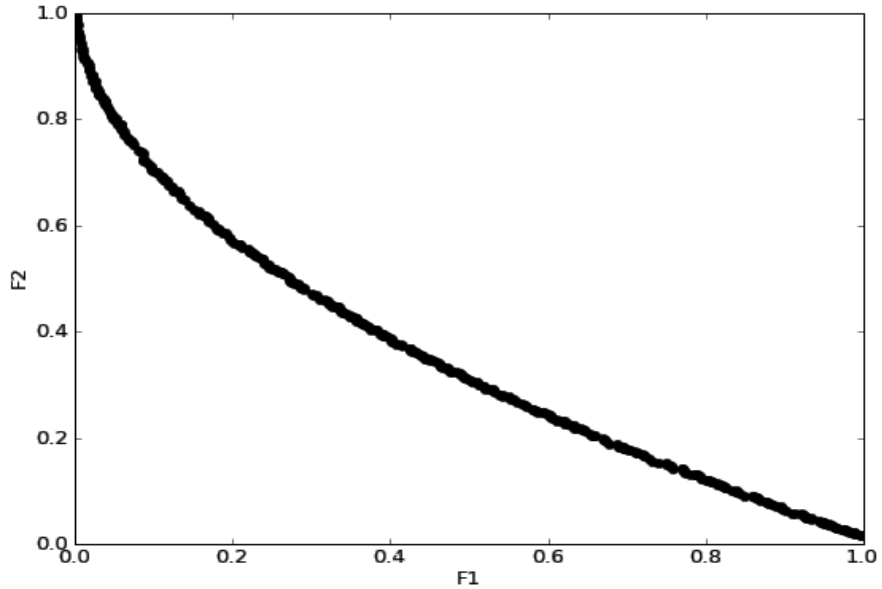


Figure B.1: The Convex Pareto-front (Test Problem 1)

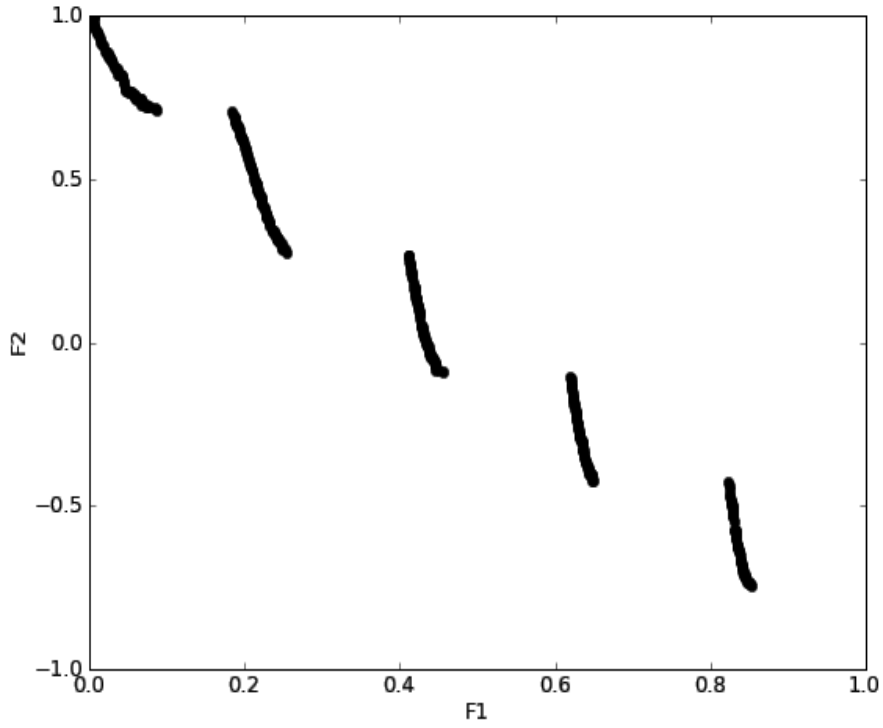


Figure B.2: The Discontinuous Pareto-front (Test Problem 2)

# Appendix C

# The Augmented State-space m-file



```
num11=-0.37;
den11=[8.0 1];
num21=-0.19;
den21=[8.0 1];
num12=-0.23167;
den12=[7.6 1];
num22=-0.3483;
den22=[8.0 1];
% Calculate Transfer Functions
sys11=tf(num11,den11);
sys21=tf(num21,den21);
sys12=tf(num12,den12);
```

```
sys22=tf(num22,den22);

%Total System Transfer Function

SysT=[sys11 sys12;sys21 sys22]

%Convert to state-Space

Sys1=ss(SysT); [Ac,Bc,Cc,Dc]=ssdata(Sys1);

[Ap,Bp,Cp,Dp]=c2dm(Ac,Bc,Cc,Dc,0.5)

%Augument the State-space Matrices

[m1,n1]=size(Cp);

% m1 = # of outputs

[n1,n_in]=size(Bp);

% n1 = # of outputs

A_e=eye(n1+m1,n1+m1);

A_e(1:n1,1:n1)=Ap;

A_e(n1+1:n1+m1,1:n1)=Cp*Ap;

B_e=zeros(n1+m1,n_in);

B_e(1:n1,:)=Bp;

B_e(n1+1:n1+m1,:)=Cp*Bp;

C_e=(zeros(m1,n1+m1));

C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

# Appendix D

# System Modelling



The thermal system model was approximated using the first-order model where the input-output relation is given by

$$G(s) = \frac{Y(s)}{U(s)} = \frac{A}{T_p s + 1} \tag{D.1}$$

where $A$ is the system gain which is given by

$$A = \frac{\triangle Y}{\triangle U}$$

$T_p$ is the time-constant of the system.

The thermal plant is a multivariable system with two outputs (temperature sensors) and two inputs (heaters) hence its transfer function model is given by

$$G(s) = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \tag{D.2}$$

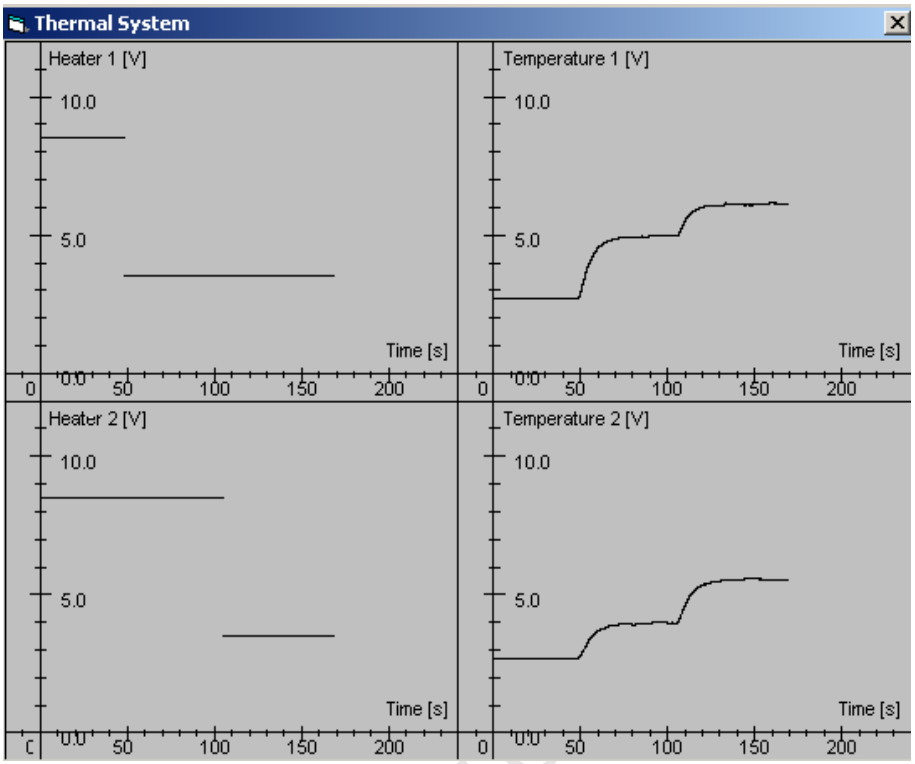


Figure D.1: Step test 1

where $g_{11}$,$g_{12}$,$g_{21}$and $g_{22}$ takes the form in Equation D.1.

Step tests were conducted on the thermal system as shown in Figure D.1 and D.2.

From Figure D.1, the model was obtained as

$$G(s) = \begin{bmatrix} \frac{0.37}{8.0s+1} & \frac{0.2317}{7.6s+1} \\ \frac{0.19}{8.0s+1} & \frac{0.3483}{8.0s+1} \end{bmatrix} \frac{[V]}{[V]} \tag{D.3}$$

From Figure D.2, the model was obtained as

$$G(s) = \begin{bmatrix} \frac{0.365}{8.0s+1} & \frac{0.2337}{7.6s+1} \\ \frac{0.198}{7.6s+1} & \frac{0.338}{8.0s+1} \end{bmatrix} \frac{[V]}{[V]} \tag{D.4}$$

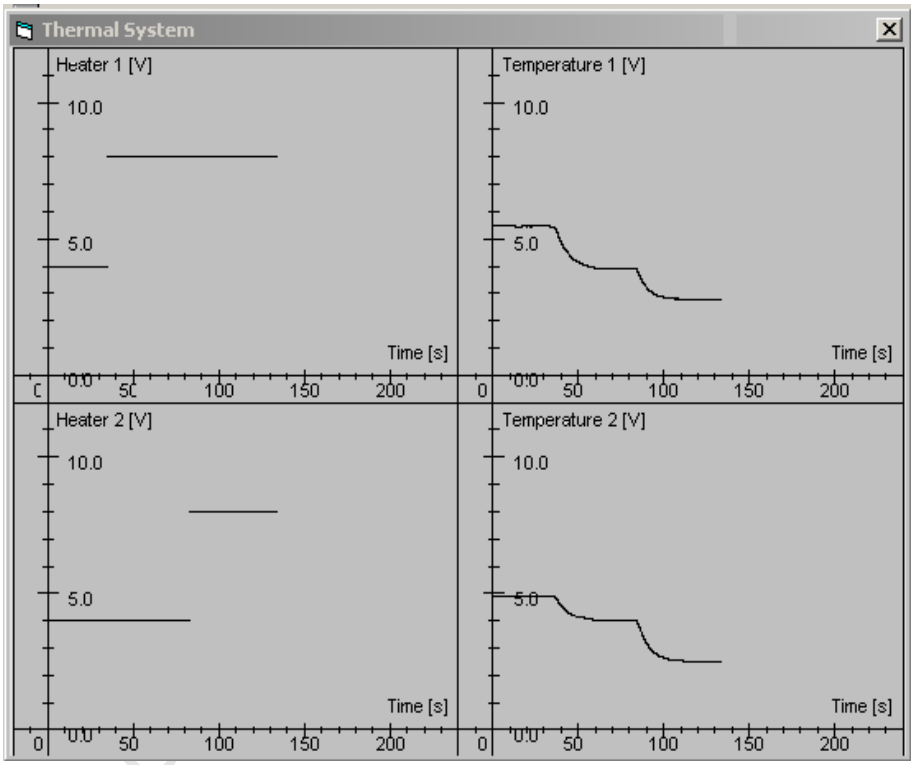


Figure D.2: Step test 2

Both model gains of the models seem to vary insignificantly by an error within 0.01. As a results, the model in Equation D.5 was used and is given in Equation D.5 with its error tolerance.

$$G(s) = \begin{bmatrix} \frac{0.37\pm0.01}{(8.0\pm0.4)s+1} & \frac{0.2317\pm0.002}{(7.6\pm0.4)s+1} \\ \frac{0.19\pm0.001}{(8.0\pm0.4)s+1} & \frac{0.3483\pm0.01}{(8.0\pm0.4)s+1} \end{bmatrix} \frac{[V]}{[V]} \tag{D.5}$$

Step responses of the model were taken to validate whether the model gives the same responses as the plant step tests. The graph of both responses were compared as shown in Figure D.3, D.4, D.5, and D.6 to ensure that they match. Response 1 and Response 2 represent the responses of output 1 due to step change in input 1 and input 2 respectively while Response 3 and Response 4 represent the responses of output 2 due to step change in input 1 and input 2 respectively. The responses were found to approximately match hence a model in Equation D.4 gives a good approximate of the thermal system.

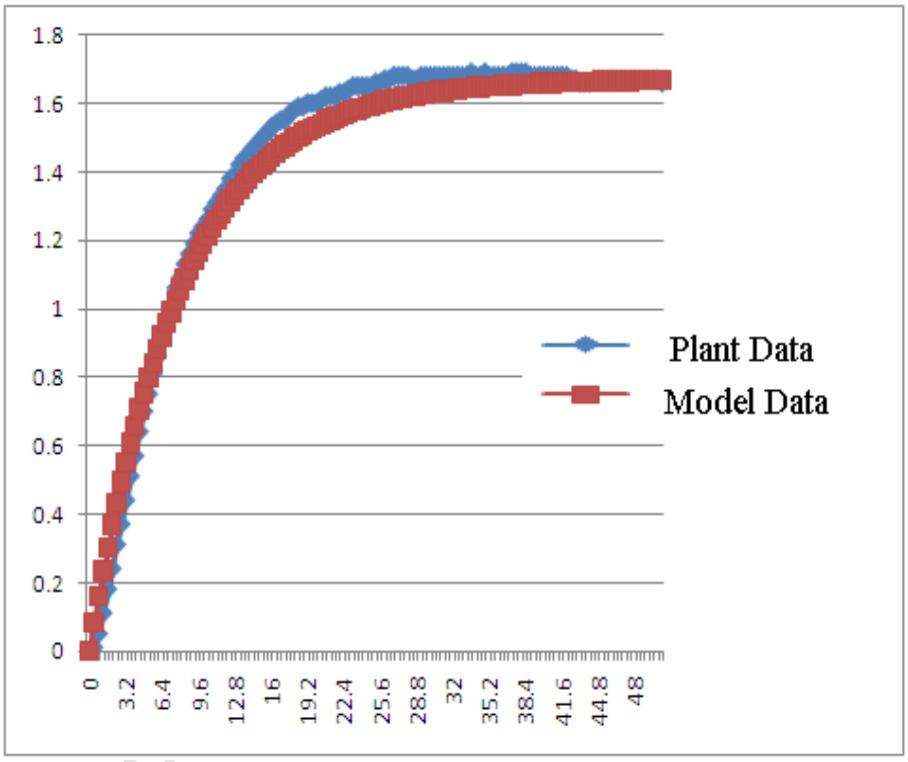


Figure D.3: Response 1

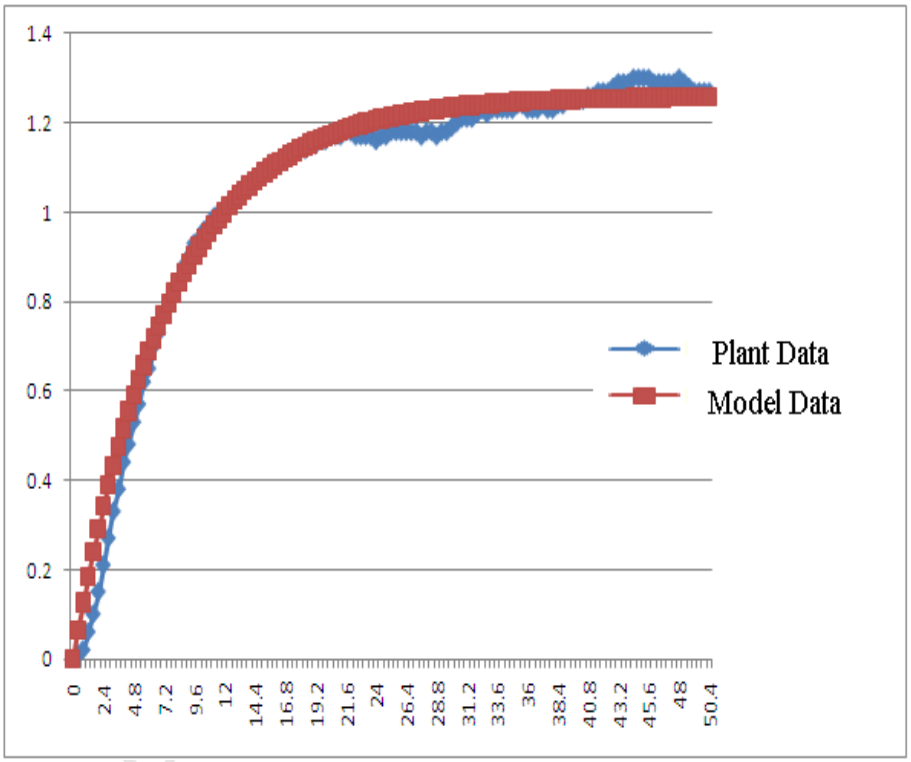


Figure D.4: Response 2

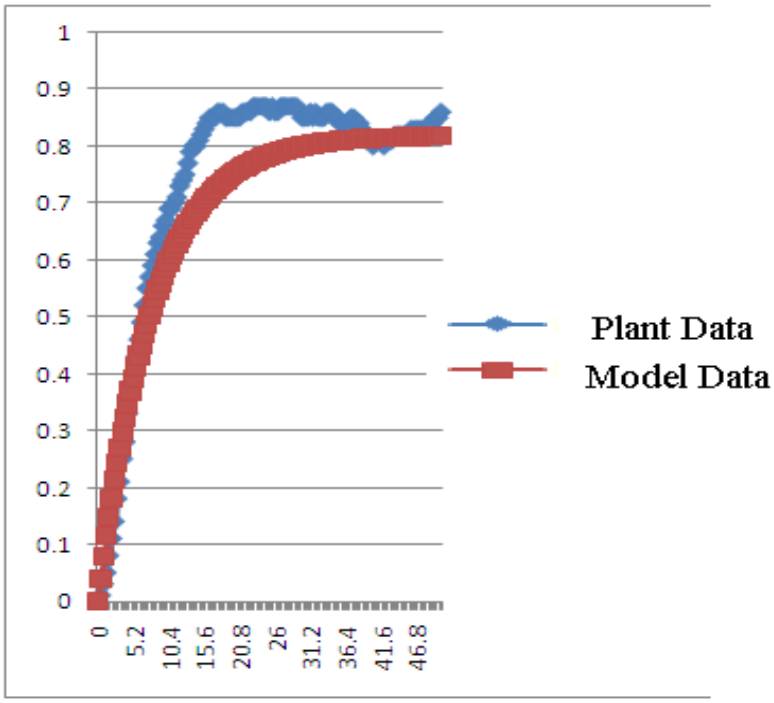


Figure D.5: Response 3

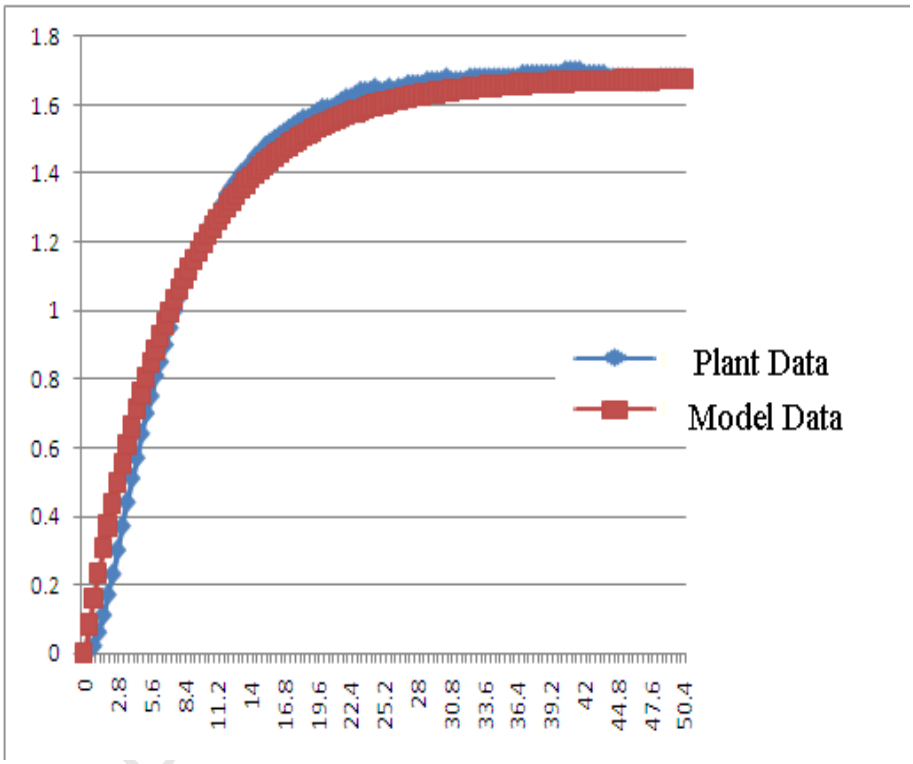


Figure D.6: Response 4

# Appendix E

# Paper Submission

Some of the work in this thesis were published on a this paper.

- T. Koetje, M. Braae, M. Mohohlo, "Multi-objective Performance Evaluation of Controllers for a Thermal Process", **CISSE**, USA, December 2010 (In Press)

# Appendix F

# Resources Used

This is included in a compact disk (CD) that is submitted together with this thesis.

The main files are *LevelGen.py* and *ControlCompare.py* which generate the Level Diagrams results and compare controllers respectively. The other files are C/C++ files that were used for experimental testing. Text files of the data extracted from step tests of the thermal process are also included.