FEEDBACK CONTROL FOR MAXIMIZING COMBUSTION EFFICIENCY OF A
COMBUSTION BURNER SYSTEM

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Marcus A. Horning

May, 2016
ABSTRACT

An observer-controller pair was designed to regulate the fuel flow rate and the flue-gas oxygen ratio of a combustion boiler. The structure of the observer was a proportional-integral state estimator. The designed controller was composed of a combination of two common controller structures: state-feedback with reference tracking and proportional-integral-derivative (PID).

A discrete-time, linear state-space model of the combustion system was developed such that the linear controller and observer could be designed. This required establishing separate models pertaining to the combustion process, actuators, and sensors. The complete model of the combustion system incorporated all three models. The combustion model, which related the flue-gas oxygen ratio to the fuel and oxygen flow rates, was obtained using the mathematical formulas corresponding to combustion of natural gas. The actuators were modeled using measured fuel and oxygen flow rate data for various actuator signals, and fitting the data to a parametric model. The established nonlinear models for the combustion process and actuators required linearization about a specified operating point. The sensors model was then obtained using the predictive error identification technique based on batch input-output data.

For the acquired model of the combustion system, a linear quadratic regulator was used to calculate the optimal state feedback gain. The classical controller gains were determined by tuning the gains and evaluating the simulation of the closed-loop response.
Computer-aided simulations provided evidence that the controller and state estimator could regulate the desired set point in the presence of moderate disturbances. The observer-controller pair was implemented and verified on an experimental boiler system by means of an embedded system. Even in the presence of a disturbance resulting from a 50% blockage of the surface area of the air intake duct, the closed-loop system was capable of regulating the desired set point for slow-varying reference signal changes.
DEDICATION

To my mom and dad.
ACKNOWLEDGEMENTS

I would like to thank so many people for their help and support throughout this project. My deepest gratitude is extended to the following individuals:

- my research advisor, Dr. Ida, for his support, guidance, advice, and encouragement over the past three years. It was truly a privilege to have the opportunity to work with him.
- my labmate and friend, Rikesh Shakya.
- lab technician, Max Fightmaster, for his constant support and assistance on the project.
- my committee members, Dr. Veillette and Dr. Lee, for their direction and recommendations.
- the R.W. Beckett Corporation for their knowledge, expertise, and tremendous support throughout the entirety of the project.
- my family and friends for their love and support. I couldn’t have done it without you all. I like especially like to thank my parents: Matt and Cindy, and my siblings: Lisa, Ed, María, and Nick.
TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................ xi

LIST OF FIGURES ....................................................................................................... xii

CHAPTER

I. INTRODUCTION .................................................................................................. 1

II. BACKGROUND .................................................................................................... 6

  2.1 Combustion Basics .......................................................................................... 6
  2.2 Current Control Strategies for Combustion ..................................................... 8
  2.3 Experimental Setup ....................................................................................... 11
  2.4 State-space Representation ........................................................................... 14
  2.5 Controllability and Observability .................................................................. 15
  2.6 Linearization .................................................................................................. 16
  2.7 Discretization ................................................................................................ 17
  2.8 System Identification ..................................................................................... 18
  2.9 Classical Control ........................................................................................... 19
  2.10 State Variable Feedback ............................................................................... 21
  2.11 Closed Loop State Observer ........................................................................ 22
  2.12 Separation Principle ..................................................................................... 23
  2.13 Linear Quadratic Regulator (LQR) .............................................................. 23

vii
III. CONTROLLER DESIGN ................................................................. 25

3.1 System Model ............................................................................. 25

3.2 Closed-Loop State Estimator ...................................................... 26

3.2.1 PI State Observer .................................................................... 28

3.2.2 Kalman Filter ......................................................................... 30

3.3 State Feedback ........................................................................... 33

3.3.1 Limitations of a Reference Tracking State-feedback Controller .... 33

3.3.2 State Feedback plus PID with Feedforward Control .............. 36

3.3.3 Feedforward Control Segment ............................................... 37

3.3.4 Complete Controller Design ..................................................... 38

3.4 Observer-Controller System ....................................................... 41

IV. CONTROLLER IMPLEMENTATION ............................................... 43

4.1 Development of Non-linear Model for Combustion Process .......... 43

4.2 Modeling of the Actuator Dynamics .......................................... 46

4.2.1 The Blower Motor Model ....................................................... 46

4.2.2 The Fuel Regulator Model ..................................................... 48

4.3 Linearization and Discretization of the System ............................. 52

4.4 System Identification of the Sensor Dynamics ............................. 53

4.4.1 Selection of the Model Identification Signals ........................... 54

4.4.2 Predictive Error System Identification ................................. 57

4.5 Complete Model ......................................................................... 59

4.6 Analysis of the Model ................................................................. 60

4.7 Controller Design ...................................................................... 60
4.7.1 Selection of Weighting Matrices for the Linear Quadratic Regulator .......... 60
4.7.2 Selection of the Optimal Feedback Gain ...................................................... 61
4.7.3 Calculation of the Feed-Forward Gain .......................................................... 67
4.7.4 Selection of the Classical Controller Gains ............................................... 67
4.8 Observer Design................................................................................................ 72
4.8.1 Selection of the Measurement Noise Matrix .............................................. 72
4.8.2 Selection of the Process Noise Matrix ....................................................... 73
4.8.3 Selection of the Observer Proportional Gains ........................................... 76
4.8.4 Selection of the Observer Integral Gains .................................................... 77
4.9 Closed-loop Stability ...................................................................................... 79

V. EXPERIMENTAL RESULTS AND VALIDATION .............................................. 80
5.1 Model Validation............................................................................................. 80
5.2 Observer Validation......................................................................................... 82
5.3 Closed-Loop Observer-Controller Validation ................................................. 90

VI. CONCLUSIONS AND FUTURE WORK ............................................................. 98

BIBLIOGRAPHY ........................................................................................................... 100

APPENDICES ................................................................................................................ 102

APPENDIX A: PICTURES OF THE BOILER SYSTEM ............................................. 103

APPENDIX B: MATLAB CODE FOR SIMULATION OF NON-LINEAR MODEL .......................................................... 105

APPENDIX C: MATLAB CODE FOR THE LINEARIZATION AND DISCRETIZATION OF THE NON-LINEAR MODEL ......................... 107

APPENDIX D: MATLAB CODE FOR THE SELECTION OF THE CONTROLLER GAINS .......................................................... 109
APPENDIX E: MATLAB CODE FOR SELECTION OF OBSERVER GAINS...... 117

APPENDIX F: MATLAB CODE FOR SIMULATION OF OBSERVER-CONTROLLER CLOSED-LOOP SYSTEM................................................................. 121

APPENDIX G: MPLAB CODE- PERIPHERALS.H FILE ............................................. 124

APPENDIX H: MPLAB CODE- PERIPHERALS.C FILE ......................................... 131

APPENDIX I: MPLAB CODE- MAIN.C FILE ........................................................... 136
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Experimental Setup Hardware</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Summarized Recursive Equations for Discrete-Time Kalman Filter</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Combustion Properties</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Gain Vector for Selection of Optimal Feedback Gain</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>Gain Vectors for the Selection of the PID Controller Gains</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Gain Vectors for Selection of the Observer Integral Gain</td>
<td>78</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1.1</td>
<td>Concentration of flue gases [P 1996]</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Control system block diagram</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Overview of a condensing boiler system [U 2013]</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Simulation diagram- LTI continuous-time state-space system</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Simulation diagram- linearized state-space system</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Block diagram- closed-loop system</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Simulation diagram- state feedback controller scheme</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Simulation diagram- combined observer and controller configuration</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>Simulation diagram- Luenberger observer</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Simulation diagram- PI observer</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Simulation diagram- state feedback controller with reference</td>
<td>34</td>
</tr>
<tr>
<td>3.4</td>
<td>Simulation diagram- state feedback with reference and process noise</td>
<td>35</td>
</tr>
<tr>
<td>3.5</td>
<td>Simulation diagram- state feedback plus PID with feedforward control for unperturbed system</td>
<td>37</td>
</tr>
<tr>
<td>3.6</td>
<td>Simulation diagram- state feedback control with reference tracking of the perturbed system</td>
<td>38</td>
</tr>
<tr>
<td>3.7</td>
<td>Simulation diagram- state feedback plus PID action with feedforward control for the perturbed system</td>
<td>39</td>
</tr>
<tr>
<td>3.8</td>
<td>Simulation diagram- controller-observer system</td>
<td>42</td>
</tr>
<tr>
<td>4.1</td>
<td>Blower motor steady-state measurements</td>
<td>47</td>
</tr>
</tbody>
</table>
4.2 Step response of the blower motor dynamics ................................................................. 48
4.3 Steady-state fuel rate measurements ............................................................................. 49
4.4 Fuel regulator model identification- slope parameter .................................................... 50
4.5 Fuel regulator model identification- Y-intercept parameter ........................................... 50
4.6 System identification data- reference signals ............................................................... 55
4.7 System identification data- perturbed control input signals ......................................... 56
4.8 System identification data- system state variables ....................................................... 56
4.9 System identification data- measured outputs ............................................................. 57
4.10 System validation data ............................................................................................... 58
4.11 Simulated system response- state variables for an open-loop response ................. 62
4.12 Simulated system response- control inputs to an open-loop response ...................... 63
4.13 Simulated system response- state variables for a state-feedback response ............. 65
4.14 Simulated system response- control input for a state-feedback response .............. 66
4.15 Simulated system response- closed-loop response for the complete system-state variables .................................................................................................................. 70
4.16 Simulated system response- closed-loop response for the complete system-control Inputs ..................................................................................................................... 71
4.17 System identification- determination of the measurement noise matrix ............... 73
4.18 System identification- determination of the process noise matrix ............................ 76
4.19 Simulated system response- open-loop response for selected observer gain vectors ......................................................................................................................... 78
5.1 System validation- comparison of the measured, linearized model, and non-linear model data .................................................................................................................. 81
5.2 System validation-model data for selected fuel regulator currents ......................... 82
5.3 Observer validation- reference signal changes-reference signals ......................... 83
5.4 Observer validation- reference signal changes- control input signals .............. 84
5.5 Observer validation- reference signal changes- measured outputs, estimated outputs, and estimated states ........................................................................... 86
5.6 Observer validation- reference signal changes- estimation errors .................. 87
5.7 Observer validation- induced disturbance- measured and estimated states ........ 89
5.8 Observer validation- induced disturbance- estimation errors ......................... 90
5.9 Observer-controller validation- references changes-reference signals .......... 91
5.10 Observer-controller validation- references changes- control input signals ...... 92
5.11 Observer-controller validation- references changes- measured outputs, estimated outputs, and estimated states ................................................................. 93
5.12 Observer-controller validation- references changes- output errors ............... 94
5.13 Observer-controller validation- induced disturbance- control input signals..... 95
5.14 Observer-controller validation- induced disturbance – measured states, estimated outputs, and estimated states ................................................................. 96
5.15 Observer-controller validation- induced disturbance – output errors .......... 97
A.1 Combustion Boiler System- Bosch wall-hung boiler ZBR21-3A .................. 104
A.2 (a) Modified air intake with Bosch air flow sensor (b) Fuel regulator with modulating gas valve ................................................................. 105
CHAPTER I
INTRODUCTION

During recent years, increased energy demands and higher energy costs have elevated the focus on energy consumption in the United States and throughout the world. In particular, the energy efficiency of natural gas furnaces and boiler systems has drawn considerable attention. According to the U.S. Energy Information Administration, about 40% of the energy delivered to the residential sector is related to space heating, while approximately half of that pertains to natural and propane gas fired furnaces [Co 2013]. Therefore, improving the energy efficiency of a furnace system would conserve a substantial amount of energy.

Combustion efficiency losses are a major cause of poor total efficiency of combustion boilers and furnaces. Combustion efficiency is related to how effective a burner converts fuel to heat. A major cause for energy loss that reduces combustion efficiency is when the energy contained by unburned fuel is not released. In theory, a stoichiometric mixture of fuel and air are required for a combustion efficiency of 100% since the perfect amount of oxygen is mixed to fully burn the entirety of the fuel. However, in reality this cannot be obtained since a stoichiometric mixture cannot occur, and also because imperfect mixing of fuel and air leads to unburned fuel. In practice, operating the air-fuel ratio with an excess of oxygen will result in the highest possible
combustion efficiency since it allows for more of the fuel to be burned. However, operating with too much excess oxygen will reduce the combustion efficiency since the surplus of oxygen is unnecessarily heated in the combustion process, thus consuming energy [U 2012].

The optimal amount of excess oxygen is determined by various factors including the fuel type, the load factor, and the physical structure of the combustion apparatus. Figure 1.1 illustrates an approximation of the overall efficiency as well as the exhaust gas concentrations for different fuel-air ratios.

![Figure 1.1: Concentration of flue gases [P 1996]](image-url)
According to the figure, the percentage of oxygen in the flue is related to the efficiency of the burner. Therefore, a feedback control structure can be formulated where the measured flue-gas oxygen ratio is used to calculate the control input signals which adjust the fuel and oxygen flow rates. Although it would be more desirable to have knowledge of all the components in the flue gas, it is satisfactory and less expensive to measure only the excess oxygen.

The residential furnace industry has not yet fully exploited the benefits of controlling the fuel-air ratio as a means of increasing the combustion efficiency. Currently, furnaces are set to operate with a fixed excess air ratio, thus resulting in a desired oxygen ratio in the flue gas. However, without actual feedback of the oxygen ratio there is no guarantee that the desired percentage is being obtained. Changes in the density of the combustion air, composition of the fuel, and actuator dynamics are a few of the causes that may alter the oxygen ratio in the flue gas. In recent years, nonlinear model predictive control, amid other control strategies, has been successfully implemented in control algorithms to increase the combustion efficiency [JS 2011]. All of these approaches used an in-situ gas analyzer located in the flue to provide feedback of the measured oxygen ratio in the flue gas. However, the analyzers are too expensive to be considered in a residential burner.

This thesis describes a less expense alternative for the design of a feedback controller to regulate the oxygen ratio in the flue gas and the flow rate of fuel. A state estimator is designed to produce an estimate of the oxygen ratio in the flue gas using noisy measurements obtained by flame temperature readings. Additionally, the state
estimator provides estimates of the fuel and oxygen flow rates. A general block diagram of the control system is illustrated in Figure 1.2.

![Control system block diagram](image)

**Figure 1.2: Control system block diagram**

The key contributions conveyed in this thesis are:

- The formation of a dynamic model representing the oxygen ratio in the flue gas based on mathematical formulas pertaining to the principles of the combustion process.

- The realization of models for the feedback sensors using predictive error system identification methods.

- The development and linearization of a state-space model of the complete system.

- A PI state observer capable of providing accurate state estimates with the presence of considerable and unknown process and measurement noise.

- A state feedback with reference tracking plus PID action controller.

- Experimental testing and verification of state observer and controller performance on a real-life combustion boiler system.

This thesis is composed of six chapters beginning with this Introduction chapter, which is followed by a Background chapter. The formulation of the control approach is
divided into three subsequent chapters: Design, Implementation, and Results and Verification. This thesis closes with a Conclusion chapter.

Chapter II provides background information on topics relating to basics of the combustion of natural gas, an overview of a Kalman filter, and control theory concepts relevant to the design of the controller. Chapter III focuses on developing the structures of the state estimator and controller while explaining the motivation behind their choice. In Chapter IV a state-space model is developed for the plant, the models for the sensors are identified, and gains for the PI state observer and controller are calculated. Also, a simulation of the closed-loop system is performed to verify the success of the observer-controller pair. The results of the obtained model and the implemented observer-controller pair are presented in Chapter V. It is shown experimentally that the controller meets the design requirements and the observer accurately estimates the states in the presence of a significant blockage in the air intake. Finally, in Chapter VI the work and results presented in the previous chapter are summarized and potential future work is discussed.
CHAPTER II
BACKGROUND

Background material covering the experimental setup and the basics of combustion systems are discussed in this chapter. Additionally, concepts relevant to the modeling of the combustion system and the design of the controller are also presented. An overview covering the combustion of natural gas, as well as the concepts of state-space representation, linearization, discretization, and system identification, were applied in the formation of the model of the combustion system. Control theory concepts including classical control methods, state feedback control, state estimation, and optimal gain selection using a linear quadratic regulator were implemented in the designing of the observer and controller.

2.1 Combustion Basics

Combustion is an exothermic chemical reaction between a fuel and an oxidant, which results in the production of heat and light. For residential burner systems, natural gas is a common fuel and air is the oxidant. Natural gas is composed of different concentrations of hydrogen, carbon, oxygen, nitrogen, sulfur, ash, and water. The chemical composition of natural gas can vary between samples as it is dependent on the geographical location where the gas was extracted from the Earth. Regardless of the sample, natural gas consists predominately of methane, $CH_4$, with lesser amounts of
ethane, \( C_2H_6 \). Since the composition of natural gas is unknown, it was estimated throughout this thesis that the fuel consisted of 87% methane and 13% ethane by molar volume. The composition of air is accurately approximated to be 23.2% oxygen and 76.8% nitrogen by volume.

The contents of the flue gas are dependent on the intake air and fuel rates, and are determined by the stoichiometric equations related to the combustion of methane and ethane. The stoichiometric equation for the combustion of methane with air is given as

\[
CH_4 + 2O_2 + 2 \cdot 3.76 N_2 \xrightarrow{yields} CO_2 + 2H_2O + 2 \cdot 3.76 N_2 + \text{HEAT} \quad (2.1)
\]

where methane and oxygen are the reactants. The products of combustion are carbon dioxide, water, nitrogen and heat. The stoichiometric equation for the combustion of ethane with air is given as

\[
C_2H_6 + 3.5O_2 + 3.5 \cdot 3.76 N_2 \xrightarrow{yields} 2CO_2 + 3H_2O + 3.5 \cdot 3.76 N_2 + \text{HEAT} \quad (2.2)
\]

After combining Equation (2.1) with Equation (2.2) and also accounting for excess oxygen, the equation representing the chemical reaction for the mixture is given as

\[
0.87 \cdot CH_4 + 0.13 \cdot C_2H_6 + 2.195 \cdot (1 + x)O_2 + 8.2532 \cdot (1 + x) N_2 \xrightarrow{yields} 1.13 \cdot CO_2 + 2.13 \cdot H_2O + 2.195 \cdot x \cdot O_2 + 8.2532 \cdot (1 + x) N_2 \quad (2.3)
\]

The variable denoted by \( x \) represents the mole fractions of excess air, which is relative to the stoichiometric ratio of oxygen-to-fuel. The mole fraction of excess air is determined as
where $\phi_{O_2}$ and $\phi_{fuel}$ represent the pre-combustion mass flow rates of fuel and oxygen, respectively. The stoichiometric ratio of oxygen-to-fuel, denoted by $\varphi_{O_2-to-fuel}$, is the optimal oxygen-to-fuel ratio, which results in complete burning of the fuel. By inspection of Equation (2.3), the fuel mixture of 87% methane and 13% ethane results in a stoichiometric ratio of 2.195 moles oxygen to 1 mole fuel. The pre-combustion molar flow rates of oxygen and fuel are calculated as

\begin{equation}
\phi_{fuel} = \frac{\phi_f}{M_{fuel}} \tag{2.5a}
\end{equation}

\begin{equation}
\phi_{O_2} = \frac{\phi_{O_2}}{M_{O_2}}, \tag{2.5b}
\end{equation}

where $M_{fuel}$ and $M_{O_2}$ denote the molar masses of the fuel and oxygen, and $\phi_f$ and $\phi_{O_2}$ denote the pre-combustion mass flow rate of the fuel and oxygen.

### 2.2 Current Control Strategies for Combustion

In the residential sector, the furnaces installed in homes currently do not utilize any control strategy to optimize the combustion process. When the furnace is first installed, a technician sets the damper position to fix the air flow to the level which results in the optimal oxygen ratio in the flue gas. While this strategy is very inexpensive, it can be ineffective and a safety concern when disturbances affect the system. Fluctuations in the ambient temperature, composition in the fuel, and mechanical alterations in the valves and blower motor could result in low combustion efficiency and possibly the production in harmful gases. In addition, the fixed nature of the blower
motor and nonexistence of a variable gas valve precludes the flow rate of fuel from being regulated.

The boiler and furnace operations in the industrial sector utilize more complex and effective control strategies. In-situ ZrO$_2$ combustion gas oxygen analyzers, located in the flue, provide feedback of the measured oxygen content. Using this sensor feedback, existing approaches to optimizing the combustion efficiency employ different control methods [CCS 2006]. The first method uses soft-computing techniques such as fuzzy logic and neural networks. The second method uses model-based control strategies, where the analytic models are developed based on thermodynamics and chemistry. The third method uses a combination of soft-computing and model-based strategies.

Kusiak et al. [KZ 2006] developed a scheme which used a data-mining algorithm based on the functionality of a neural network. Boiler parameters, such as fan speed, feeders speed, pressure, steam temperature, load, among others were monitored and recorded. The boiler’s combustion efficiency was simultaneously calculated using predefined equations. Therefore, the system could learn the appropriate control setting based on the most current system parameters.

Zanoli et al. [ZBAB 2013] used classical control techniques by creating a PID controller to control the fuel and air rate which optimize combustion efficiency and regulates the furnace output temperature. A model of the system was constructed using a black-box approach. Separate controllers were designed to control the fuel and air rates. One of the drawbacks of using PID control for MIMO systems is that they do not provide a satisfactory response during transients. Therefore, Zanoli et al. coupled fuzzy logic techniques to improve response.
Grancharova et al. [GJJ 2007] developed a solution using modern control techniques. They designed a nonlinear model predictive controller (NMPC) with reference tracking. The NMPC solves the optimal solution of a finite horizon while having the capability to predict the state at future timeslots. The dynamic model of the combustion process was represented by a Gaussian process model. The Gaussian process model, or a nonparametric model based on a black-box approach, was developed by adding a Gaussian disturbance to the analytic model established by Cretnik et al. [CS 1988]. The Gaussian process model is appropriate because it provides uncertainty predictions for the stochastic process.

The industrial sector has the luxury of sophisticated computers and a range of sensors, which are not exploited in the residential sector due to cost restrictions. The unavailability of sensors to measure all the required furnace parameter makes the neural network method of control not a feasible option. Additionally, the model predictive control strategy can also be ruled out because the on-line computation that is required is too complex to be run on a low-cost embedded microcontroller. Classical control methods used alone are not acceptable because the system is multiple input-multiple output, whereas classical control methods are most effective for single input-single output systems. The designed method must be capable of running on an embedded computer and have a robust control strategy that can track a reference signal while using noisy measurements. The controller must regulate the outputs to a safe operating region within a prescribed time.
2.3 **Experimental Setup**

The experimental testing was conducted on a residential gas condensing wall-hung boiler manufactured by Bosch. In boiler systems, the heat from the hot gases produced in the combustion process is transferred to the water by the use of a heat exchanger. Unlike conventional boilers, a condensing boiler recovers energy in the steam that is present in the flue gas that would otherwise be lost. The concept of the condensing boiler system is depicted in Figure 2.1.

![Figure 2.1: Overview of a condensing boiler system [U 2013]](image)

The combustion boiler was modified such that the fuel and oxygen flow rates could be electronically controlled. The original fuel control was replaced with a servo pressure regulator and a modulating gas valve. A direct current applied to the coil on the modulating gas valve allows the fuel flow to be controlled. Due to the feasibility and cost constraints, it is not practical to measure the fuel flow rate.
Combustion air was supplied by fresh, outside air, which is drawn into the combustion chamber by an electric blower motor. A PWM signal commands the speed of the electric blower motor, thus allowing the flow rate of the combustion air to be controlled. An in-situ mass air flow sensor is used to measure the oxygen flow rate. Varying the intake fuel-oxygen ratio results in changes in the oxygen ratio in the flue gas. For data collection, a commercial combustion gas analyzer was used to measure the oxygen percentage in the flue gas. However, in the implemented controlled system, the oxygen ratio in the flue gas was measured by a sensor that determined the value from the temperature of the flame.

The combustion process of the boiler system was safely operated by means of burner controller designed by the R.W. Beckett Corporation. The burner controller is responsible for ignition as well as gas valve closure in the event of the loss of flame and/or failure of the blower motor. The sensors and hardware used in the experimental setup are listed in Table 2.1. Photographs of the experimental setup can be found in Appendix A.
Table 2.1: Experimental Setup Hardware

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Model/Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas Pressure Regulator</td>
<td>Honeywell VK8105M5039</td>
</tr>
<tr>
<td>Modulating-coil Gas Valve</td>
<td>Honeywell V7335A &quot;5000&quot; series</td>
</tr>
<tr>
<td>Air Blower Motor</td>
<td>NA</td>
</tr>
<tr>
<td>Air Flow Meter</td>
<td>Bosch HFM 5</td>
</tr>
<tr>
<td>Flue Gas Analyzer</td>
<td>Testo 330-2G</td>
</tr>
<tr>
<td>Flame Temperature Sensor/O₂ Sensor</td>
<td>CoorsTek</td>
</tr>
<tr>
<td>Burner Control Board</td>
<td>Beckett GeniSys 7590</td>
</tr>
<tr>
<td>Boiler System</td>
<td>Bosch Greenstar ZBR21-3A</td>
</tr>
</tbody>
</table>

A Microchip dsPIC30F2011 microcontroller was used for data acquisition. Voltages representing the flame temperature and air flow rate were measured with the ADC module on the microcontroller. These values were then processed to acquire the oxygen ratio in the flue gas and the oxygen flow rate. Selected data processed by the microcontroller is sent to a PC via serial RS-232 communication. The oxygen percentage in the flue gas which is measured by the flue gas analyzer is separately sent directly to the PC through a USB link, and is recorded using the provided Testo Easyheat 2.0 software.

In addition to data acquisition and sensor measurements, the microcontroller was used for the implementation of the observer and controller design. The microcontroller outputted the corresponding blower motor PWM signal (PWM) and gas valve current ($I_{fuel}$).
2.4 State-space Representation

In modern control theory, a state-space representation is a mathematical model relating the inputs and the outputs of a system. For a linear time-invariant continuous time system of order \( n \) that has \( m \) inputs and \( p \) outputs, the representation is depicted by

\[
\dot{x}(t) = Ax(t) + Bu(t) \tag{2.6a}
\]

\[
y(t) = Cx(t) + Du(t). \tag{2.6b}
\]

where the state vector \( x \in \mathbb{R}^n \), the output vector \( y \in \mathbb{R}^p \), the input vector \( u \in \mathbb{R}^m \), the system matrix \( A \in \mathbb{R}^{n \times n} \), the input matrix \( B \in \mathbb{R}^{n \times m} \), the output matrix \( C \in \mathbb{R}^{p \times n} \), and the feedforward matrix \( D \in \mathbb{R}^{p \times m} \). A block diagram of the state-space representation of a general linear, time-invariant system is shown in Figure 2.2.

![Figure 2.2: Simulation diagram- LTI continuous-time state-space system](image)

Similarly, a state-space representation for a linear time-invariant discrete time system is given as

\[
x_{k+1} = A_dx_k + B_d u_k \tag{2.7a}
\]

\[
y_k = C_dx_k + D_d u_k. \tag{2.7b}
\]
However, as a result of uncertainties to the system model, unknown disturbances, and measurement errors, a more realistic model includes noise parameters. The form of the state-space representation for a linear time-invariant discrete time system is

\[
x_{k+1} = A_dx_k + B_du_k + \varphi w_k \tag{2.8a}
\]
\[
z_k = C_dx_k + D_du_k + v_k \tag{2.8b}
\]

where \(w_k\) and \(v_k\) are process noise and measurement noise respectively [F 1986].

2.5 Controllability and Observability

Controllability and observability are two properties that determine if it possible to design a control strategy that will control and stabilize a system. A system is controllable if there a control signal that is able to force a system to any desired state from any initial condition. If the controllability matrix given as

\[
\Omega_c = [B \ AB \ A^2B \ \ldots \ \ A^nB] \tag{2.9}
\]

is full rank, then \(\{A,B\}\) is a controllable pair and the system is controllable. A system is observable if the states can be determined from output measurements and knowledge of the input signals. If the observability matrix given as

\[
\Omega_0 = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{2.10}
\]

is full column rank, then \(\{C,A\}\) is an observable pair and the system is observable. If the system is neither controllable nor observable, then additional actuators and/or sensors need to be included in the system [F 1986].
2.6 Linearization

The state equations for a nonlinear system are expressed as

\[ \dot{x}(t) = f(x(t), u(t)). \]  \hspace{1cm} (2.11)

The goal is to obtain a linear model for the non-linear system about a specified nominal operating point \((\bar{x}, \bar{u})\). When the system is driven by the nominal system input, \(\bar{u}(t)\), the nominal system trajectory, \(\bar{x}(t)\), will be satisfied by the differential equation given as

\[ \dot{\bar{x}}(t) = f(\bar{x}(t), \bar{u}(t)). \]  \hspace{1cm} (2.12)

If the perturbations from the nominal operating point, which are expressed as

\[ \Delta u(t) = u(t) - \bar{u}(t) \]  \hspace{1cm} (2.13a)
\[ \Delta x(t) = x(t) - \bar{x}(t) \]  \hspace{1cm} (2.13b)
\[ \Delta y(t) = y(t) - \bar{y}(t), \]  \hspace{1cm} (2.13c)

are small, then the system can be linearized using the Taylor series expansion about the nominal operating point for a fixed time [LV 1995]. When the perturbations are small the difference equation of the system is expressed as

\[ \dot{x}(t) + \Delta \dot{x}(t) = f(\bar{x}(t) + \Delta x(t), \bar{u}(t) + \Delta u(t)). \]  \hspace{1cm} (2.14)

Utilizing the Taylor series expansion, the right side of Equation (2.14) can be expanded, which produces

\[ \dot{x}(t) + \Delta \dot{x}(t) = f(\bar{x}(t), \bar{u}(t)) + \frac{\partial f}{\partial x} (\bar{x}(t), \bar{u}(t)) \Delta x(t) \]
\[ + \frac{\partial f}{\partial u} (\bar{x}(t), \bar{u}(t)) \Delta u(t) + \text{higher order terms} \]  \hspace{1cm} (2.15)

When the higher-order terms are canceled, the linear difference equation is given by

\[ \Delta \dot{x}(t) = \frac{\partial f}{\partial x} (\bar{x}(t), \bar{u}(t)) \Delta x(t) + \frac{\partial f}{\partial u} (\bar{x}(t), \bar{u}(t)) \Delta u(t). \]  \hspace{1cm} (2.16)
The partial derivatives are evaluated at the nominal operating point. Using the notation

\[ A = \frac{\partial f}{\partial x}(\bar{x}, \bar{u}) \quad \text{and} \quad B = \frac{\partial f}{\partial u}(\bar{x}, \bar{u}), \]

the linearized system is represented by

\[
\begin{align*}
\Delta \dot{x}(t) &= A \Delta x(t) + B \Delta u(t) \\
\Delta y(t) &= C \Delta x(t).
\end{align*}
\]

Inserting Equation (2.13a) and Equation (2.13c) expressing the perturbed input and perturbed output, the linearized system can be represented by

\[
\begin{align*}
\Delta \dot{x}(t) &= A \Delta x(t) + B (u(t) - \bar{u}(t)) \\
y(t) &= C \Delta x(t) + \bar{y}(t).
\end{align*}
\]

The simulation diagram for the linearized system is illustrated in Figure 2.3.

![Simulation diagram- linearized state-space system](image)

Figure 2.3: Simulation diagram- linearized state-space system

### 2.7 Discretization

A computer-based implementation of control systems requires discrete-time control inputs. Therefore, for controller design purposes, it is necessary to convert the plant model from a continuous-time model to a discrete time model. This process is known as discretization. Referring to the state-space representation given in Equation (2.6a) and Equation (2.6b), the discrete time state-space parameters can be determined from
\[ A_d = e^{AT}, \]  
(2.19a) 
\[ B_d = \int_0^T e^{At} B dt, \]  
(2.19b) 
\[ C_d = C, \]  
(2.19c) 
\[ D_d = D. \]  
(2.19d)

Although the discrete-time model is time invariant, the model is dependent on the chosen sampling time, \( T \) [K 1992].

2.8 \textbf{System Identification}

The purpose of system identification is to determine the dynamical model of an unknown system using measured input-output data, where the input and output of the system are denoted by \( u \) and \( y \) respectively. Although there are many different techniques used for system identification, the following approach describes the prediction error identification method. The objective of this method is to describe the discrete-time model as a predictor of the next output. The one-step ahead prediction is described by

\[ \hat{y}_m(k|k - 1) = f(Z^{k-1}) \]  
(2.20)

where \( f(Z^{k-1}) \) is function of the past observed data. The past observed data set, up to time \( N \), is denoted by \( Z^N = \{u(1), y(1), u(2), y(2), \ldots, u(N), y(N)\} \).

This predictor requires the explicit parameterization of the model in terms of a finite dimensional parameter vector, \( \theta \). Although, there are numerous parameterizations of the model, the general parameterization is described by

\[ \hat{y}(k|\theta) = f(Z^{k-1}, \theta). \]  
(2.21)

A specific example of a model parameterization is the discrete-time linear state-space model defined by
\[ x_{k+1} = A(\theta)x_k + B(\theta)u_k \] (2.22a)
\[ y_k = C(\theta)x_k + v_k \] (2.22a)

where \( \theta \) corresponds to the constants of the system matrices.

The estimate of \( \theta \) is determined based on the model parametrization and the measured data, \( Z^N \). This estimate is chosen such that the distance between the predicted outputs and the estimated outputs are minimized using a suitable norm, which can be described by

\[ \hat{\theta}_N = \arg \min V_N(\theta) \] (2.23a)
\[ V_N(\theta) = \sum_{k=1}^{N} \|y_k - f(Z^{k-1}, \theta)\|^2 \] (2.23a)

The calculation of the minimizing argument can be done using numerous methods, including the Gauss-Newton method [L 2002].

2.9 Classical Control

A classical closed-loop controller uses sensor feedback measurements to control the outputs of a dynamical system. The block diagram of a closed-loop system is shown in Figure 2.4.

![Figure 2.4: Block diagram- closed-loop system](image-url)
The transfer function blocks defined by \( C(s), G(s), \) and \( H(s) \) represent the controller, plant, and sensor respectively. The signals depicted by \( R(s), E(s), U(s), \) and \( Y(s) \) represent the reference or desired output, tracking error, control input which is sent to the plant, and output signals respectively.

A common closed-loop controller architecture is the PID (Proportional-Integral-Derivative) controller. A PID controller utilizes the error signal to compute a control signal. The error is operated on by three terms: proportional, integral, and derivative. The resulting input signal has the form given as

\[
u(t) = K_p e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t) \tag{2.24}\]

where \( K_p, K_I, \) and \( K_D \) are proportional, integral, and derivative gains respectively. The controller gains can be tuned independently until the desired system response is produced [N 2000].

Whenever a digital implementation of the PID controller is necessary, the discretization of the integral and derivative terms is required. The discrete-time integral term is approximated using trapezoidal rule for a sampling time of \( \Delta t \), and is given as

\[
\int_0^{k \cdot \Delta t} e(\tau)d\tau = y_k \approx y_{k-1} + \frac{e_k + e_{k-1}}{2} \Delta t. \tag{2.25}\]

The derivative term is approximated using backward finite differences, and is found as

\[
\frac{d}{dt} e(t_k) = \frac{e(t_k) - e(t_{k-1})}{\Delta t}. \tag{2.26}\]

Using the \( z \)-transform, the discrete-time transfer function of the PID controller given as

\[
C(z) = \frac{U(z)}{E(z)} \tag{2.27}\]
corresponds to a digital control signal of

\[ u_k = u_{k-1} + (K_p + K_i \frac{\Delta t}{2} + K_d) e_k + \left( -K_p + K_i \frac{\Delta t}{2} - 2K_d \right) e_k + K_d \Delta t. \quad (2.28) \]

### 2.10 State Variable Feedback

A common control scheme, known as state variable feedback (SVF), uses the state vector to relocate the eigenvalues of the closed loop matrix to any desired location, provided that the system is controllable. The location of the closed loop poles is decided by the selection of the state feedback gain matrix given as \( K \). A block diagram illustrating a state variable feedback control scheme is shown in Figure 2.5.

![Simulation diagram- state feedback controller scheme](image)

The form of the feedback control law is given as

\[ u(t) = r(t) - Kx(t). \quad (2.29) \]

When the reference signal, \( r(t) \), is zero, the control law drives the system to a zero state. This control scheme is known as a regulator. On the contrary, when the reference signal is non-zero, the control scheme is referred to as a tracker [LV 1995]. Using the control law given in Equation (2.29), the state and output equations of the closed loop system become
\[
\dot{x}(t) = Ax(t) + B[r(t) - Kx(t)] \\
= [A - BK]x(t) + Br(t) \quad (2.30a) \\
= \tilde{A}x(t) + \tilde{B}r(t) \\
y(t) = Cx(t) + D[r(t) - Kx(t)] \\
= [C - DK]x(t) + Dr(t) \quad (2.30b) \\
= \tilde{C}x(t) + \tilde{D}r(t)
\]

2.11 Closed Loop State Observer

In some cases, it is not physically possible to measure all of the system’s states. Therefore, a state observer can be designed utilizing the output of the system to estimate the actual state variables. The block diagram of the observer-controller configuration is illustrated in Figure 2.6.

![Figure 2.6: Simulation diagram- combined observer and controller configuration](image)

The state equations of the observer can be written as
\begin{align*}
\dot{x}(t) &= A\dot{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)] \\
&= A\dot{x}(t) + B[r(t) - K\hat{x}(t)] + L[Cx(t) - C\hat{x}(t)] \\
&= LCx(t) + (A - BK - LC)\hat{x}(t) + Br(t). 
\end{align*}

(2.31)

The observer gain, \( L \), should be chosen such that the eigenvalues of observer decay faster than the eigenvalues of the controller. Thus, the eigenvalues of \( (A - LC) \) must be further left in the s-plane than the eigenvalues of \( (A - BK) \) [LV 1995].

2.12 **Separation Principle**

The separation principle states that the observer and feedback controller can be designed independent of each other, and then be combined without adversely affecting the stability of the system. The controller gains are selected assuming exact state estimation of the system, and the effects of the observer do not need to be considered. Likewise, the observer is designed assuming an open-loop system, and their selection is not affected by the controller gains. Additionally, the observer estimations can be computed regardless if they are used in the state-feedback control or not [F 1986].

2.13 **Linear Quadratic Regulator (LQR)**

The LQR is an algorithm utilized to determine the optimal control input sequence, \( u_k \), for the linear system described by Equation (2.7a). The algorithm minimizes a performance index given as

\[
J = \frac{1}{2} \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k), \quad (Q \geq 0, R > 0).
\]

(2.32)

The weighting factors, given by the matrices \( Q \) and \( R \), are design parameters chosen by the engineer to satisfy design goals. The \( n \times n \) state weighting matrix is represented by \( Q \). Whereas, the \( p \times p \) control weighting matrix is represented by \( R \).
When selecting the performance characteristics of the system, there is a tradeoff between the system settling time and the magnitude of the control input. Selecting the appropriate weighting factors is more intuitive when $x_k^TQx_k$ and $u_k^TRu_k$ are understood as the energies of the states and controls respectively. Thus, increasing $Q$ with respect to $R$ will place a larger priority on speeding up the system response. However, this will result in more energy exerted by the control inputs. Alternatively, increasing $R$ with respect to $Q$ will decrease the control energy, but also slow down the response.

For the control law given by Equation (2.29), the feedback gain, $K$, is calculated solving

$$S = A^T(S - SB(R + B^TSB)^{-1}B^TS)A + Q \quad (2.33a)$$

$$K = (R + B^TSB)^{-1}B^TS A. \quad (2.33b)$$

Equation (2.33a) is called the discrete-time algebraic Matrix Riccati Equation [LV 1995]. The regulator can be modified to account for tracking of a constant reference signal, $r_k$.

The modified performance index, control input, and command input are given as

$$J = \frac{1}{2} \sum_{k=0}^{\infty} ((y_k - r_k)^TQ(y_k - r_k) + u_k^TRu_k), \quad (2.34a)$$

$$u_k = -Kx_k + (R + B^TSB)^{-1}B^Tv_k, \quad (2.34b)$$

and

$$v_k = [I - (A - BK)^T]^{-1}C^Tr_k. \quad (2.34c)$$
CHAPTER III
CONTROLLER DESIGN

In this chapter, methods for controlling a linearized system are developed. Additional noise and disturbances are included in the system in order to obtain a more robust controller strategy. Although a linearized state-space system will be developed in the subsequent chapter, this chapter includes concepts of the system that are beneficial in choosing the correct controller and observer. The state estimations of the system will be accomplished using a PI state observer. However, the concepts of a Kalman filter are utilized to obtain the proportional observer gains. Finally, a controller implementing state-feedback with a reference tracking plus PID control is developed to obtain zero steady-state error amongst model uncertainties and unknown disturbances.

3.1 System Model

Using the methods discussed in Section 2.6, a non-linear system can be linearized about a specified operating point. For the model that will developed in the following chapter, the perturbed input and system output are defined as

\[
\Delta u_k = u_k - \bar{u} \overset{\text{def}}{=} \begin{bmatrix} \Delta PW M \\ \Delta I_{fuel} \end{bmatrix}
\]

(3.1a)
\[ z_k = \Delta y_k + \bar{y} + v_k = C \Delta x_k + C \bar{x} + v_k = \begin{bmatrix} \bar{O}_2 \\ \bar{\Phi}_f \\ \bar{\Phi}_{O_2} \end{bmatrix} \]  

(3.1b)

where the nominal operating point is expressed by \((\bar{x}, \bar{u})\). The inputs of the system are the perturbed blower motor pulse width modulated signal, \(\Delta PWM\), and the perturbed fuel regulator electrical current, \(\Delta I_{fuel}\). The outputs of the system are the measured oxygen ratio in the flue, \(\bar{O}_2\), the measured flow rate of fuel, \(\bar{\Phi}_f\), and the measured flow rate of oxygen, \(\bar{\Phi}_{O_2}\).

3.2 Closed-Loop State Estimator

In order to properly control the system, the states of the system must be known with confidence. Therefore, it is vital that the designed state observer is accurate in the presence of noise and other disturbances. Thus, the Luenberger observer, mentioned in Section 2.11, is further examined with output distances and lumped uncertainties in the process. When the state representation is described by Equation (2.8a) and Equation (2.8a), the closed loop observer is illustrated in Figure 3.1.
With the additive noise, the equations for the closed loop observer are represented by

\[
\dot{x}_{k+1} = (A - LC)x_k + Bu_k + \dot{w}_k, \quad (3.2a)
\]

\[
\dot{z}_k = Cx_k + \dot{v}_k, \quad (3.2b)
\]

The effect of noise on the system can be observed by examining the estimation errors.

The output estimation error, state estimation error, process noise estimation error, and output estimation error are defined by

\[
\tilde{x}_k = \hat{x}_k - x_k, \quad (3.3a)
\]

\[
\tilde{z}_k = \hat{z}_k - z_k, \quad (3.3b)
\]

\[
\tilde{w}_k = \hat{w}_k - w_k, \quad (3.3c)
\]

\[
\tilde{v}_k = \hat{v}_k - v_k. \quad (3.3d)
\]
By substituting Equation (2.8a) and Equation (3.2a) into the definition for the output estimation error, given by Equation (3.3b), the output estimation error is redefined as

\[
\tilde{z}_k = C\tilde{x}_k + \tilde{v}_k. \tag{3.4}
\]

Whereas, by substituting Equation (2.8a) and Equation (3.2a) into the definition for the state estimation error, given by Equation (3.3b), the state estimation error is redefined as

\[
\tilde{x}_{k+1} = (A - LC)\tilde{x}_k + \tilde{w}_k - L\tilde{v}_k. \tag{3.5}
\]

With the addition of the process and measurement noise, the system is now driven. Therefore, the state estimation error will no longer asymptotically converge to zero. Since it is unlikely that the unknown noise will be zero, or yet alone converge, the Luenberger observer is not suitable for a noisy system.

3.2.1 PI State Observer

In [S 2007], Shelby proposed a PI state observer for estimating the states for a system with an unknown output disturbance. The form of the system is determined by

\[
\Delta x_{k+1} = A\Delta x_k + B\Delta u_k + w_k \tag{3.6a}
\]

\[
z_k = C\Delta x_k + \bar{y} + v_k. \tag{3.6b}
\]

The PI observer for the system determined by Equation (3.6a) and Equation (3.6b) is given by

\[
\Delta \hat{x}_{k+1} = A\Delta \hat{x}_k + B\Delta u_k + L_1(z_k - \hat{z}_k) + w_k \tag{3.7a}
\]

\[
\hat{v}_{k+1} = \hat{v}_k + L_2(z_k - \hat{z}_k) \tag{3.7b}
\]

\[
\hat{z}_k = C\Delta \hat{x}_k + \bar{y} + \hat{v}_k \tag{3.7c}
\]

This observer is an extension of the proportional observer, such that there is an additional feedback loop of the output estimation error. The output estimation error is integrated to
provide estimates of the output disturbance. The block diagram of the PI observer developed by Shelby is shown in Figure 3.2.

![Block Diagram of PI Observer](image)

Figure 3.2: Simulation diagram - PI observer

The PI observer will produce state, output, and output disturbance estimates that converge to the actual state, output, and output disturbance values provided that the four conditions are met:

- \((A, C)\) is observable
- \(A\) has no eigenvalues at \(z=1\)
- The sampling time is small enough to ensure that \(v_{k+1} \approx v_k\)
- \(w_k = 0\)

If the process noise is not zero, then state estimate will not approach the actual states. However, the output estimate is still guaranteed to converge to the actual output. The proportional gain, \(L_1\), was determined based on the concepts of the Kalman filter. The integral gain, \(L_2\), was selected using an iterative tuning method.
3.2.2 Kalman Filter

A Kalman filter is an optimal, linear observer which minimizes the mean square error of the estimated states. In terms of estimating the states in the presence of noise, the Kalman filter is more superior to the Luenberger observer. The form of the Kalman filter is similar to the closed loop observer represented by Equation (2.31), and is given as

\[
\hat{x}_{k+1}^- = A\hat{x}_k^+ + Bu_k \tag{3.8a}
\]

\[
\hat{x}_k^+ = \hat{x}_k^- + L_k[z_k - C\hat{x}_k^-]. \tag{3.8b}
\]

The gain for the Kalman filter, denoted by \(L_k\), is time varying, and is calculated on the basis of a probabilistic approach under the assumption of zero-mean Gaussian process and measurement noise. Additionally, the process and measurements noises are assumed to be uncorrelated with not only with each other but also with time, consequently their covariances are defined by

\[
E\{v_kv_k^T\} = R_k \tag{3.9a}
\]

\[
E\{w_kw_k^T\} = Q_k \tag{3.9b}
\]

The Kalman filter in the discrete-time form is a recursive algorithm which utilizes an update and a prediction phase. Prior to the state update phase, the predicted (a priori) state estimate and error covariance are denoted by \(\hat{x}_k^-\) and \(P_k^-\) respectively. Subsequent the state update phase, the (a posteriori) state estimate and the error covariance are denoted by \(\hat{x}_k^+\) and \(P_k^+\) respectively. The error covariances are defined by

\[
P_{k+1}^- = E\{\hat{x}_{k+1}^-,\hat{x}_{k+1}^-\} \tag{3.10a}
\]

\[
P_k^+ = E\{\hat{x}_k^+,\hat{x}_k^+\} \tag{3.10b}
\]

The optimal Kalman gain is computed by minimizing the trace of \(P_k^+\), which is equivalent to minimizing the estimation error. By substituting the definition for the state
estimation error, given in Equation (3.3a), into Equation (3.10a), the priori error covariance is redefined as

\[ P_{k+1}^- = E\{\hat{x}_{k+1}^- x_{k+1}^- T\} = E\{[\hat{x}_{k+1}^- - x_{k+1}^-] \cdot [\hat{x}_{k+1}^- - x_{k+1}^-]^T\} \] (3.11)

Using the expressions given by Equation (3.9) and Equation (2.8b), the expression for the a priori error covariance is expanded to

\[ P_{k+1}^- = E\{[(A\hat{x}_k^+ + Bu_k) - (Ax_k^+ + Bu_k + \varphi w_k)] \cdot [(A\hat{x}_k^+ + Bu_k) - (Ax_k^+ + Bu_k + \varphi w_k)]^T\} \]

\[ = E\{A\hat{x}_k^+ \hat{x}_k^T A^T\} - E\{\varphi w_k \hat{x}_k^T A^T\} - E\{A\hat{x}_k^+ w_k^T \varphi^T\} + E\{\varphi w_k w_k^T \varphi^T\}. \] (3.12)

Since the process noise, \( w_k \), and current posteriori state estimate error, \( \hat{x}_k^+ \), are uncorrelated it follows that \( E\{\varphi w_k \hat{x}_k^T A^T\} = E\{A\hat{x}_k^+ w_k^T \varphi^T\} = 0 \). Furthermore, using the definitions expressed by Equation (3.10b) and Equation (3.9b), Equation (3.12) can be simplified to

\[ P_{k+1}^- = AP_k^T A^T + \varphi Q_k \varphi^T \] (3.13)

The expression for \( P_k^- \) is developed by first deriving the expression for \( \hat{x}_k^+ \). By substituting the expression for \( z_k \), given by Equation (2.8b), into Equation (3.8b) yields

\[ \hat{x}_k^+ = \hat{x}_k^- + L_k [Cx_k + v_k - C\hat{x}_k^-]. \] (3.14)

Utilizing the expression for state estimation error, given by Equation (3.3), then Equation (3.14) reduces to

\[ \hat{x}_k^+ = (I - L_k C)\hat{x}_k^- + L_k v_k. \] (3.15)

By substituting Equation (3.15) into Equation (3.10a), the expression for the posteriori error covariance can be rewritten as
\[ P_{k}^{+} = E\{(I - L_{k}C_{k})\bar{x}_{k}^{-}\bar{x}_{k}^{-T}(I - L_{k}C_{k})^{T}\} + E\{L_{k}v_{k}v_{k}^{T}L_{k}^{T}\} + \]
\[ E\{(I - L_{k}C_{k})\bar{x}_{k}^{-}v_{k}^{T}L_{k}^{T}\} + E\{L_{k}v_{k}\bar{x}_{k}^{-T}(I - L_{k}C_{k})^{T}\}. \]

Since \( v_{k} \) and \( \bar{x}_{k}^{-} \) are uncorrelated, it follows that \( E\{\bar{x}_{k}^{-}v_{k}\} = E\{v_{k}\bar{x}_{k}^{-T}\} = 0. \)

Furthermore, using the definitions for measurement noise covariance and priori error covariance given in Equation (3.9a) and Equation (3.10a) respectively, the expression for the posteriori error covariance is reduced to
\[ P_{k}^{+} = [I - L_{k}C_{k}]P_{k}^{-}[I - L_{k}C_{k}]^{T} + L_{k}R_{k}L_{k}^{T}. \]

The optimal gain, \( L_{k} \), is obtained by minimizing the trace of the posteriori error covariance. Using the trace identities given by
\[
\frac{\partial}{\partial A} Tr(BAC) = B^{T}C^{T} \tag{3.18a}
\]
\[
\frac{\partial}{\partial A} Tr(ABA^{T}) = A(B + B^{T}) \tag{3.18b}
\]
\[
Tr(A + B) = Tr(A) + Tr(B) \tag{3.18c}
\]
the partial derivative of the trace of the posteriori error covariance with respect to the Kalman filter gain is
\[
\frac{\partial\{Tr(P_{k}^{+})\}}{\partial L_{k}} = -2P_{k}^{-}C_{k}^{T} - 2L_{k}C_{k}P_{k}^{-}C_{k}^{T} + 2L_{k}R_{k} = 0 \tag{3.19}
\]
Evaluating Equation (3.19) to determine the optimal gain yields
\[
L_{k} = P_{k}^{-}C_{k}^{T}[C_{k}P_{k}^{-}C_{k}^{T} + R_{k}]^{-1}. \tag{3.20}
\]
Using Equation (3.24), the expression representing the posteriori error covariance can be simplified [CJ 2004]. Table 3.1 summarized the simplified equation relevant to the discrete-time Kalman filter.
Table 3.1: Summarized Recursive Equations for Discrete-Time Kalman Filter

<table>
<thead>
<tr>
<th>Gain</th>
<th>$L_k = P_k^- C_k^T [C_k P_k^- C_k^T + R_k]^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update</td>
<td>$\hat{x}_k^+ = \hat{x}_k^- + L_k [z_k - C \hat{x}_k^-]$</td>
</tr>
<tr>
<td></td>
<td>$P_k^+ = [I - L_k C_k] P_k^-$</td>
</tr>
<tr>
<td>Prediction</td>
<td>$\hat{x}_{k+1}^- = A \hat{x}_k^+ + Bu_k$</td>
</tr>
<tr>
<td></td>
<td>$P_{k+1}^- = A P_k^+ A^T + \varphi Q_k \varphi^T$</td>
</tr>
</tbody>
</table>

3.3 State Feedback

A controller which regulates the inputs to track a reference signal while providing zero steady-state error of the system outputs is developed in the following section. Resembling the estimator, the controller must also function when the state representation of the system, which is described by Equation (2.8a) and Equation (2.8b), contains process and measurement noise.

3.3.1 Limitations of a Reference Tracking State-feedback Controller

The concept of the regulator, mentioned in Section 2.10, is extended to provide tracking of a reference signal. A state-feedback controller with reference tracking is illustrated in Figure 3.3. The feedforward parameter, denoted by $N$, scales the input signal such that the output tracks the reference signal with zero-state error when noise is not existent.
Figure 3.3: Simulation diagram- state feedback controller with reference

The control law for the corresponding controller is given as

\[ u = -Kx + Nr. \]  \hspace{1cm} (3.21)

The feedforward term can be determined by evaluating the system at steady state. The system is first converted from the reference tracking system to the familiar regulator, where the steady state behavior is more explicitly known. The input, state, and output are therefore redefined as

\[ \tilde{u} = u_k - u_{ss}, \]  \hspace{1cm} (3.22a)

\[ \tilde{x} = x_k - x_{ss}, \]  \hspace{1cm} (3.22b)

\[ \tilde{y} = y_k - r \]  \hspace{1cm} (3.22c)

Substituting Equation (3.22a) and Equation (3.22b) into the state equation defined in Equation (2.7) yields the state equation redefined as

\[ \tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k = Ax_k - Ax_{ss} + Bu_k - Bu_{ss} \]  \hspace{1cm} (3.23)

For the regulator system, as \( k \to \infty \) then \( \tilde{x}, \tilde{y} \to 0 \). Therefore, for the reference tracking system, as \( k \to \infty \) then \( y(k) \to r \) such that Equation (3.22c) is satisfied. The control input for the reference tracking system can be determined by substituting Equation (3.22a) and Equation (3.22b) into the regulator control law given by Equation
Thus, the control law for the state feedback with reference tracking controller is denoted by
\[ u_k = -Kx_k + u_{ss} + Kx_{ss}. \]  
(3.24)

As \( k \to \infty \), \( x_{k+1} = x_k = x_{ss} \). Using the state equations given by Equation (2.7) and the control input defined by Equation (3.24), the steady state of the system is given as
\[ x_{ss} = -(A - BK - I)^{-1}B(u_{ss} + Kx_{ss}). \]  
(3.25)

In order for \( Cx_{ss} = r \) and control law given in Equation (3.21) to hold, it follows that
\[ N^{-1} = -C(A - BK - I)^{-1}B. \]  
(3.26)

The controller will achieve zero steady state error for a noiseless system, however the controller must be robust to noise and other uncertainties. The effect of what happens to the steady error when process noise is added to the system is examined. Figure 3.4 illustrates the controller with process noise, denoted by \( w \), supplemented to the input node of the system.

\[ x_{k+1} = (A - Bk)x_k + B(Nr + B^{-1}w_k). \]  
(3.27)
At steady-state $x_{k+1} = x_k = x_{ss}$ and $w_k = w_{ss}$, therefore, the values of the state and output as $k \to \infty$ are given as

$$x_{ss} = -(A - Bk - I)^{-1}B(Nr + \varphi w_{ss})$$

(3.28a)

$$y_{ss} = -C(A - Bk - I)^{-1}B(Nr + \varphi w_{ss}).$$

(3.28b)

By substituting in Equation (3.26) into Equation (3.28b), the expression for the steady state output is reduced to

$$y_{ss} = r + N^{-1}B^{-1}\varphi w_{ss}.$$  

(3.29)

There is no guarantee that the noise will converge to zero, much less converge to any value. Clearly, this particular controller is insufficient at providing zero steady state error in the presence of noise.

3.3.2 State Feedback plus PID with Feedforward Control

In order to eliminate the steady state error and speed up the system response, a PID control segment was added to the system. However, this addition by itself would cause the system to initially operate at low fuel air and fuel rates, which is a potential safety concern. In order to alleviate the risk, a feedforward term is also supplemented to the state feedback control. The feedforward term is selected to use the correct open loop gain. Therefore, the PID control will only be utilized when there is noise added to the system model. Figure 3.5 illustrates the state-feedback plus PID with feedforward control for a general unperturbed system.
The transfer function for $M(z)$ is the desired system transfer function, such that under ideal conditions the error is equivalent to zero. The control block denoted by $D(z)$ is composed of multiple classical controllers. The number of controllers is equivalent to the number of inputs to the system $G(z)$.

3.3.3 Feedforward Control Segment

The feedforward segment is augmented with a state feedback controller to ensure reference tracking for the linearized model. The perturbed control law for the linearized form of the state feedback with reference tracking system illustrated in Figure 3.4.

Without the PID action included, the feedback of the perturbed system is given as

$$ \Delta u_k = K \Delta x_k + Nr. \quad (3.30) $$

By applying the definition of the perturbed input in Equation (2.13a) and substituting the equality into Equation (3.30), the unperturbed control input has the form

$$ u_k = K \Delta x_k + Nr + \bar{u}_k. \quad (3.31) $$

A simulation diagram of state-feedback control with reference tracking of the linearized system is illustrated in Figure 3.6.
3.3.4 Complete Controller Design

Multiple states are added to the system when PID action is supplemented to the system. The transfer function of $M(z)$, introduced in section 3.3.2, is modeled by the state-space representation of the state feedback with reference tracking system appended to the sensor feedback matrix, $C_f$. The control block, $D(z)$, is substituted by the parametric state-space representation of the PID controllers. The simulation diagram of the complete controller design is illustrated in Figure 3.7.
Figure 3.7: Simulation diagram - state feedback plus PID action with feedforward control for the perturbed system.
The complete system is represented by the states given as

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k + w_{1_k}$$  \hspace{1cm} (3.32a) \\
$$\Delta \dot{x}_{k+1} = A\Delta \dot{x}_k + B\Delta \dot{u}_k$$  \hspace{1cm} (3.32b) \\
$$\eta_{k+1} = A_c \eta_k + B_c e_k.$$  \hspace{1cm} (3.32c)

The inputs of the state equations are defined by

$$\Delta u_k = C_c \eta_k + N r_k - K \Delta x_k + \bar{u}_k - \bar{u}_k$$  \hspace{1cm} (3.33a) \\
$$\Delta \dot{u}_k = N r_k - K \Delta \dot{x}_k + \bar{u}_k - \bar{u}_k$$  \hspace{1cm} (3.33b) \\
e_k = C_s \dot{h}_k - C_s h_k.$$  \hspace{1cm} (3.33c)

The representation of the closed loop system is obtained by substituting the appropriate input definition expressed by Equation (3.33a) through Equation (3.33a) into the state equations expressed by Equation (3.32a) through Equation (3.32a). The form of the state equation representing the closed loop system is given as

$$\bar{x}_{k+1} = \bar{A}\bar{x}_k + R r_k + \varphi w_k + V v_k + \gamma$$  \hspace{1cm} (3.34)

where $R = \begin{bmatrix} BN \\ BN \\ 0 \end{bmatrix}$, $V = \begin{bmatrix} 0 \\ B_s \\ 0 \end{bmatrix}$, $\varphi = \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$, $\gamma = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Additionally, the vector $r$ defines the reference input vector, and the state vector is represented by

$$\bar{x}_k = \begin{bmatrix} \Delta x_k \\ \Delta \dot{x}_k \\ \eta_k \end{bmatrix}.$$  \hspace{1cm} (3.35)

The dynamics of the system are described by the system matrix of the closed loop system which is given as

$$\bar{A} = \begin{bmatrix} A - BK & 0 & BC_c \\ 0 & A - BK & 0 \\ -B_c C_f C & B_c C & A_c \end{bmatrix}.$$  \hspace{1cm} (3.36)
3.4 **Observer-Controller System**

The simulation diagram of the closed-loop observer-controller system is illustrated in Figure 3.8.
Figure 3.8: Simulation diagram- controller-observer system
4.1 Development of Non-linear Model for Combustion Process

A theoretically-developed model of the combustion process for a residential furnace is formed based upon the principle of conservation of mass. The dynamic equation for the change of the volumetric ratio of oxygen in the combustion chamber is influenced by the mass flow of oxygen entering and the mass flow of oxygen leaving the control volume. The control volume is defined as the volume encompassing the path beginning at the point that combustion occurs and ending where oxygen percentage is analyzed. The established model is simplified and assumes:

- The mixing of the flue gas is uniform throughout the control volume.
- The combustion process and mixing of the flue gas occurs instantaneously.
- The densities of the flue gas and oxygen are constant.
- The total mass of the flue gas is constant.

Using the principle of conservation of mass, the mass ratio of oxygen in the flue gas compared to the total mass of all the gases in the flue is described by the dynamic expression given as

\[
\dot{m}_{oxygen} = \frac{d}{dt} \left( \frac{1}{m_{flue}} \cdot m_{O_2,cv} \right) = \frac{m_{O_2, in} - m_{O_2, out}}{m_{flue}}
\]  

(4.1)
where \( m_{flue} \) is the total mass of the constituents in the flue gas, \( m_{O_{2, CV}} \) is the mass of oxygen in the control volume, \( \dot{m}_{O_{2, in}} \) is mass flow rate of oxygen entering the control volume, and \( \dot{m}_{O_{2, out}} \) is the mass flow rate of oxygen exiting the control volume. Using the relationship that the volume of a substance is related to its mass divided by the density of the substance, Equation (4.1) can be modified to express the volumetric ratio of oxygen in the flue, \( \dot{O}_2 \). Thus, the dynamic expression for the volumetric oxygen ratio in the flue is given as

\[
\dot{O}_2 = \frac{d}{dt} \left( \frac{1}{m_{flue}} \cdot m_{O_{2, CV}} \right) \cdot \rho_{flue} \frac{\rho_{O_2}}{\rho_{O_2}} \frac{m_{O_{2, in}} - \dot{m}_{O_{2, out}} \cdot \rho_{flue}}{(m_{flue} \cdot \rho_{O_2})}
\]

where the densities of the flue gas and oxygen are represented by \( \rho_{flue} \) and \( \rho_{O_2} \) respectively.

By inspection of the equation for chemical reaction of combustion process given by Equation (2.3), the mass flow rate of the oxygen entering the control volume is represented by

\[
\dot{m}_{O_{2, in}} = \frac{M_{O_2}}{M_{fuel}} \cdot \Phi_{fuel} \cdot \Phi_{o_2-to-fuel} \cdot \chi.
\]

By substituting the expression for the molar fraction of excess oxygen, defined by Equation (2.4) in Section 2.1, into Equation (4.3), the mass flow rate of oxygen entering the control volume can be redefined as

\[
\dot{m}_{O_{2, in}} = \frac{M_{O_2}}{M_{fuel}} \cdot \Phi_{fuel} \cdot \Phi_{o_2-to-fuel} \left( \frac{\Phi_{O_2} \cdot M_{fuel}}{\Phi_{O_2-to-fuel} \cdot \Phi_{fuel} \cdot M_{O_2}} - 1 \right).
\]
The expression for the mass flow rate of oxygen is formed on the basis that the total flow leaving the control volume is the sum of the air and fuel flow rates. Thus, the mass flow rate of oxygen exiting the control volume is given as

\[ \dot{m}_{O_2,\text{out}} = \left( \Phi_{\text{fuel}} + \frac{1}{\varphi_{O_2-\text{to-air}}} \cdot \Phi_{O_2} \right) \cdot m_{p,\text{oxygen}}. \]  \hspace{1cm} (4.5)

The variable \( m_{p,\text{oxygen}} \) represents the mass percentage of oxygen in the flue, and is related to the volumetric flow by

\[ m_{p,\text{oxygen}} = \frac{\rho_{O_2}}{\rho_{\text{flue}}} \cdot O_2. \]  \hspace{1cm} (4.6)

The complete equation relating the change in the volumetric oxygen ratio over time with fuel flow rate, air flow rate, and the volumetric oxygen ratio is given as

\[
\dot{O}_2 = \frac{1}{V \cdot \rho_{O_2}} \left( \Phi_{O_2} - \frac{M_{O_2}}{M_{\text{fuel}}} \varphi_{O_2-\text{to-fuel}} \cdot \Phi_{\text{fuel}} \right) - \frac{1}{\rho_{\text{flue}} \cdot V} \left( \Phi_{\text{fuel}} \right)
+ \frac{1}{\varphi_{O_2-\text{to-air}}} \cdot \Phi_{O_2} \cdot O_2
\]  \hspace{1cm} (4.7)

where the combustion properties of the system under test are defined in Table 4.1.

<table>
<thead>
<tr>
<th>Table 4.1: Combustion Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molar Mass of ( O_2 ) ( M_{O_2} ) 31.999 ([\text{grams}])</td>
</tr>
<tr>
<td>Molar Mass of the fuel ( M_{\text{fuel}} ) 17.82 ([\text{grams}])</td>
</tr>
<tr>
<td>Oxygen-to-air ratio ( \varphi_{O_2-\text{to-air}} ) 0.232</td>
</tr>
<tr>
<td>Stoichiometric air-to-fuel ratio ( \varphi_{O_2-\text{to-fuel}} ) 2.195</td>
</tr>
<tr>
<td>Density of the flue gas ( \rho_{\text{flue}} ) 1200 ([\frac{g}{m^3}])</td>
</tr>
<tr>
<td>Density of oxygen ( \rho_{O_2} ) 1331 ([\frac{g}{m^3}])</td>
</tr>
<tr>
<td>Volume of chamber ( V ) 0.15 ([m^3])</td>
</tr>
</tbody>
</table>
4.2 **Modeling of the Actuator Dynamics**

The unknown characteristics, such as linearity and transient response, of the oxygen and fuel flow sensors makes the system identification method mentioned in 2.8 unfeasible. This is because the response time of the sensors, which measure the oxygen and fuel flows, are slow relative to the response time of the actuators. Thus, it is difficult to accurately determine the dynamics of the blower motor and fuel regulator without reliable sensor measurements. Although the dynamics of the two systems are not insignificant, their models can be estimated with little consequences. However, this conjecture is only justifiable if reference command inputs of the system are constant or slow varying.

Based on their measured responses, both systems are modeled after a first-order, time-invariant system represented by

\[
\frac{dX(t)}{dt} = -\frac{1}{\tau}(X(t) - f(t)).
\]  

(4.8)

This simple model is suitable because neither system exhibited an overshoot in their responses. The symbol denoted by \(\tau\) represents the system time constant, whereas \(f(t)\) depicts the forcing function. The two models will relate the oxygen and fuel flow rates to the control inputs.

4.2.1 **The Blower Motor Model**

The forcing function for the oxygen flow can be determined from the steady-state relationship between the blower motor control input, designated by PWM, and the flow rate of oxygen. The flow rate of oxygen was not dependent on the gas regulator control...
input. The steady state flow rate of oxygen was recorded for a range of appropriate PWM inputs, and the relationship is illustrated in Figure 4.1.

![Steady-State Oxygen Flow Rate for Blower Motor Control Inputs](image)

**Figure 4.1: Blower motor steady-state measurements**

The steady-state oxygen flow rate is represented by a second-order equation given by

$$\Phi_{O_2} = -0.001069755249 \cdot PWM^2 + 0.10940425590 \cdot PWM - 0.10408973987.$$  \hspace{1cm} (4.9)

The time constant of the blower motor was approximated from the step response of the oxygen flowrate when the PWM control input was varied from 0 to 45%. The airflow rate is measured with sensor with a response time of approximately but no slower than 15ms. Figure 4.2 illustrates the step response.
By inspection of the response, the time constant is 0.5 seconds. Therefore, the first-order model of the oxygen flow rate is given by

$$\dot{\phi}_{O_2} = -2 \cdot \left( \phi_{O_2} - (-0.001069755249 \cdot PWM^2 + 0.10940425590 \cdot PWM - 0.10408973987) \right)$$

(4.10)

4.2.2 The Fuel Regulator Model

The measurement of the gas flow rate is more challenging because an inexpensive gas flow sensor that accurately measures mass flow rate with a fast response is not obtainable. Therefore, the value of the flow rate was approximated by means of the stoichiometric equation for combustion of methane and the measurements of air and oxygen percentage in the flue. The theoretical fuel flow rate is given by

$$\phi_f = \frac{\left( \frac{\rho_{flue}}{\rho_{O_2}} - \frac{V_p}{\rho_{O_2-air}} \right) \phi_{O_2}}{V_p + \frac{\rho_{flue}}{\rho_{O_2}} \cdot \frac{M_{O_2}}{M_{fuel}} \cdot \rho_{O_2-air}}$$

(4.11)
However, due to the slow response of the oxygen ratio sensor, Equation (4.11) only holds when a steady-state is achieved. It also assumes that the mixing of fuel and combustion air is perfect.

The forcing function for the fuel flow rate state equation is dependent on the gas regulator current and the oxygen flow rate. Figure 4.3 illustrates the correlation between the fuel flow rate and the oxygen flow rate when the gas regulator current [mA] is fixed at numerous values.

The equation for the fuel flow rate can be fitted with the linear parametric model given as

\[ \Phi_f = a(I_{fuel}) \cdot \Phi_{O_2} + b(I_{fuel}) \]  

(4.12)

Figure 4.3: Steady-state fuel rate measurements
where $a(I_{fuel})$ and $b(I_{fuel})$ are the linear coefficients in terms of the gas regulator current. Figure 4.4 and Figure 4.5 depict the relationships between the regulator current and the coefficients $a(I_{fuel})$ and $b(I_{fuel})$.

![Graph](image1.png)

**Figure 4.4:** Fuel regulator model identification- slope parameter

![Graph](image2.png)

**Figure 4.5:** Fuel regulator model identification- Y-intercept parameter
The expressions for $a(I_{fuel})$ and $b(I_{fuel})$ are given as

$$a(I_{fuel}) = -0.00029056416 \cdot I_{fuel} + 0.053341179827$$  \hspace{1cm} (4.13a)$$

and

$$b(I_{fuel}) = -0.00029056416 \cdot I_{fuel} + 0.053341179827.$$  \hspace{1cm} (4.13b)$$

After substituting Equation (4.13a) and Equation (4.13b) into Equation (4.9), the expression for the fuel flow rate is given as

$$\Phi_{fuel} = \left( -0.00029056416 \cdot I_{fuel} + 0.053341179827 \right) \cdot \Phi_{O_2} +$$

$$\left( -0.00029056416 \cdot I_{fuel} + 0.053341179827 \right).$$  \hspace{1cm} (4.14)$$

Due to limitations in acquiring real-time fuel flow measuring, the time constant characterizing the step response of the fuel flow rate must be estimated. After experimental testing, it is reasonable to approximate the time constant to be in the range of 0.1 to 1.5 seconds. However, a slower response of the fuel flow rate presents a more hazardous situation because the oxygen ratio produced at the flame would vary outside the safe operating range of 0.03 to 0.06. Therefore, the time constant was chosen as 1.5 seconds. Using the approximated time constant of the fuel actuator and the steady-state expression for fuel given in Equation (4.14), the fuel actuator system can be represented as

$$\dot{\Phi}_{fuel} = -\frac{2}{3} \cdot \left( \Phi_{O_2} - \left( -0.00029056416 \cdot I_{fuel} + 0.053341179827 \right) \cdot \Phi_{O_2} + \left( -0.00029056416 \cdot I_{fuel} + 0.053341179827 \right) \right).$$  \hspace{1cm} (4.15)$$
4.3 Linearization and Discretization of the System

Using the concepts presented in Section 2.6, a linearized state-space representation is developed for the expressions describing the combustion process and actuator dynamics. The steady state operating point was determined by simulating the response of the nonlinear system to the desired steady-state input. The three differential equations which completely describe the nonlinear system are represented by Equation (4.7), Equation (4.10), and Equation (4.14). Thus, the steady state operating point is given by

\[ U_{ss} = (\overline{PWM}, \overline{I}_{fuel}) = (24[\%], 120[\text{mA}]) \]  

(4.16a)

\[ X_{ss} = (\overline{O}_2, \overline{\Phi}_f, \overline{\Phi}_{O_2}) = \left( 0.0432, 0.3791 \frac{g}{s}, 1.9054 \frac{g}{s} \right) \]  

(4.16b)

\[ Y_{ss} = (\overline{O}_2, \overline{\Phi}_f, \overline{\Phi}_{O_2}) = \left( 0.0432, 0.3791 \frac{g}{s}, 1.9054 \frac{g}{s} \right) \]  

(4.16c)

where \( \overline{O}_2, \overline{\Phi}_f, \) and \( \overline{\Phi}_{O_2} \) are the steady-state solutions when the control inputs of \( \overline{PWM} \) and \( \overline{I}_{fuel} \) were selected. The system matrix and input matrix when calculated about the operating point are given as

\[
A = \begin{bmatrix}
\frac{\partial \dot{O}_2}{\partial O_2} & \frac{\partial \dot{O}_2}{\partial \phi_f} & \frac{\partial \dot{O}_2}{\partial \phi_{O_2}} \\
\frac{\partial \phi_f}{\partial O_2} & \frac{\partial \phi_f}{\partial \phi_f} & \frac{\partial \phi_f}{\partial \phi_{O_2}} \\
\frac{\partial \phi_{O_2}}{\partial O_2} & \frac{\partial \phi_{O_2}}{\partial \phi_f} & \frac{\partial \phi_{O_2}}{\partial \phi_{O_2}}
\end{bmatrix}
\]  

(4.17)

and
The discretized system was found using the concepts presented in Section 2.7. The “c2d()” function in MatLab was used to compute the discretized state equations for a sampling time of 50ms, and the complete state-space equation is given as

\[
\begin{bmatrix}
\Delta O_2 \\
\Delta \Phi_f \\
\Delta \Phi_{O_2}
\end{bmatrix}_{k+1} = \begin{bmatrix}
\Delta O_2 \\
\Delta \Phi_f \\
\Delta \Phi_{O_2}
\end{bmatrix}_k + \begin{bmatrix}
\Delta P_{PWM} \\
\Delta I_{fuel}
\end{bmatrix}_k
\]

Where the system matrix and input matrix are

\[
A = \begin{bmatrix}
0.976416 & -0.0097089 & 0.0018623 \\
0 & 0.967216 & 0.0015549 \\
0 & 0 & 0.904837
\end{bmatrix}
\]

and

\[
B = \begin{bmatrix}
0.0000055274 & -0.00000038716 \\
0.000046152 & 0.000077693 \\
0.0055248 & 0
\end{bmatrix}.
\]

4.4 System Identification of the Sensor Dynamics

A state-space parametric model was formulated to represent the relationship between the values of the state variables defined in Equation (4.19) to the perturbed measured values of oxygen ratio in the flue, \(\Delta \bar{O}_2\), and the oxygen flow rate, \(\Delta \bar{\Phi}_{O_2}\).

The dynamics of the fuel flow rate sensor need not be identified in this section because they are indirectly measured using the equation for the theoretical fuel rate computed by Equation (4.11). Although, this method of measurement provides an instantaneous
reading of the fuel flow, its value is only accurate when the system is in a steady-state. Therefore, the fuel flow rate measurement is not used within 30 seconds of a reference change and/or a disturbance being sensed.

Section 2.8 provides an introductory review on the prediction error system identification method that was used to identify the sensors model. Before identification of the system, the input signals were selected to identify the system. The reference signals were programmed into a microcontroller to automatically alter the control input signals applied to the physical system. The state variables, which were used as the system identification input signals, were calculated in MatLab, and the output data was collected.

Based on the input-output data, the MatLab function “ssest()” was utilized to estimate a state-space model with free-parameterization. The function automatically obtained the best discrete-time model for the selected order of the system. The order of the system was chosen based on the comparison of the identified models to the validation data.

4.4.1 Selection of the Model Identification Signals

The input signal must be selected such that the system is continuously excited, thus ensuring that none of the state variables reach a steady state. The model defined by Equation (3.1a) and Equation (3.1b) is a linearized model perturbed from a nominal operating point defined by Equation (4.16a) and Equation (4.16a). Therefore, the input data for this model is the amount by which the actual input data was perturbed from $X_{ss}$. Likewise, the output data for this model is the amount by which the actual output data was perturbed from $Y_{ss}$. 

54
The reference signal and corresponding control input signal used to obtain the simulated state variables are shown in Figure 4.6 and Figure 4.7 respectively. The identification input and output data is shown in Figure 4.8 and Figure 4.9.

Figure 4.6: System identification data- reference signals
Figure 4.7: System identification data- perturbed control input signals

Figure 4.8: System identification data- system state variables
4.4.2 Predictive Error System Identification

The MatLab System Identification Toolbox was used to determine the parametric state-state model that most accuracy represented the system defined by Equation (4.20a) and Equation (4.20b). The identified parametric model was of the form

\[ x_{k+1} = Ax_k + B \begin{bmatrix} \Delta O_2 \\ \Delta \Phi_f \\ \Delta \Phi_{O_2} \end{bmatrix} \]  

(4.20a)

\[
\begin{bmatrix}
\Delta \bar{O}_2 \\
\Delta \bar{\Phi}_{O_2}
\end{bmatrix} = Cx_k
\]

(4.20b)

The order of the system was determined by comparison of the identified models. Various models were obtained for the system order ranging from 1 to 10. By comparing the
validation data to the respective identified models, it was determined that a second-order system was the best fit. The resulting state-space model for a sampling time of $T_s=50\text{ms}$ and a system order of $n=2$ is given by

$$
A = \begin{bmatrix} 0.99143 & -0.0045205 \\ -0.033727 & 0.90756 \end{bmatrix},$

$$
B = \begin{bmatrix} -0.043998 & 0.0013765 & -0.0030346 \\ -0.21419 & -0.063966 & -0.051297 \end{bmatrix},$

and $C = \begin{bmatrix} -0.15202 & -0.018944 \\ -0.56067 & -1.3305 \end{bmatrix}.$

The comparison of the validation data to the identified linearized state-space model is illustrated in Figure 4.10.
4.5 Complete Model

The complete model of the entire system is obtained by augmenting the state-space system representing the sensor dynamics given by Equation (4.20a) and Equation (4.20b) with the state-space system given by Equation (4.19). The state-space model is defined by

\[
\begin{bmatrix}
\Delta O_2 \\
\Delta \Phi_f \\
\Delta \Phi_{o2} \\
x_1 \\
x_2 
\end{bmatrix}_{k+1} =
\begin{bmatrix}
\Delta O_2 \\
\Delta \Phi_f \\
\Delta \Phi_{o2} \\
x_1 \\
x_2
\end{bmatrix}_k +
\begin{bmatrix}
\Delta PWM \\
\Delta I_{fuel}
\end{bmatrix}_k
\]

(4.21)

and

\[
\begin{bmatrix}
\Delta \bar{O}_2 \\
\Delta \bar{\Phi}_f \\
\Delta \bar{\Phi}_{o2}
\end{bmatrix}_k =
\begin{bmatrix}
\Delta O_2 \\
\Delta \Phi_f \\
\Delta \Phi_{o2}
\end{bmatrix}_k
\]

(4.22)

The state, input, and output matrices are given as

\[
A =
\begin{bmatrix}
0.97641 & -0.0097089 & 0.0018623 & 0 & 0 \\
0 & 0.96722 & 0.0015549 & 0 & 0 \\
0 & 0 & 0.904837 & 0 & 0 \\
-0.043998 & 0.0013765 & -0.0030346 & 0.99143 & -0.0045205 \\
-0.21419 & -0.063966 & -0.0030346 & -0.033727 & 0.90756
\end{bmatrix},
\]

\[
B =
\begin{bmatrix}
0.0000055274 & -0.00000038716 \\
0.000046152 & 0.000077693 \\
0.0055248 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix},
\]

and

\[
C =
\begin{bmatrix}
0 & 0 & -0.15202 & -0.018944 \\
0 & 1 & 0 & 0 \\
0 & 0 & -0.56067 & -1.3305
\end{bmatrix}.
\]
4.6 Analysis of the Model

Controllability and observability are essential properties of the model. If the model is not controllable then additional control input(s) needs to be added to the system. Likewise, if the system is not observable then additional sensor(s) needs to be added to the system. Using the concepts presented in Section 2.5, controllability and observability can be determined from the $A$, $B$, and $C$ matrices in Equation (4.21) and Equation (4.22). Thus, controllability of the model was confirmed because

$$\text{rank}([B \ AB \ A^2B \ A^3B \ A^4B]) = 5$$

Likewise, the observability of the model was confirmed because

$$\text{rank} \begin{pmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \end{pmatrix} = 5.$$

4.7 Controller Design

The feed-forward gain, $N$, state feedback gain, $K$, and classical controller gains were selected in order to meet the time response requirements of the closed-loop system. The feed-forward segment of the controller was designed using the techniques described in Section 3.3.3. A linear quadratic regulator was utilized to calculate the values of the feedback gains. Introductory background material related to the linear quadratic regulator is provided in Section 2.13. The classical controller gains were determined using a tuning algorithm.

4.7.1 Selection of Weighting Matrices for the Linear Quadratic Regulator

The linear quadratic regulator will minimize a cost function that is defined by the matrices $Q$ and $R$. The $Q$ matrix defines the weighting on the states, whereas the $R$ matrix
defines the weight on the control input. The choice of Q and R is directed by the
designer, and is carefully chosen based on system requirements such as desired system
response and limitations in the control input. The following iterative method is used to
choose the weighting of the LQR:

1. **Choose the initial weighting for the Q and R matrices.** For a diagonal Q
   matrix, increasing an element will put a larger priority on the corresponding state.
   Thus, the state will vary less from the steady-state value. Likewise, for a diagonal
   R matrix, increasing an element will lessen the energy exerted by the
   corresponding control input.
2. **Given A,B,Q, and R, use the MatLab function dLQR() to compute the
   optimal feedback gain, denoted K.** The MatLab function solves a discrete-time
   algebraic riccati equation presented in Section 2.13 by Equation (2.33a), and
   subsequently computes the optimal feedback gain given by Equation (2.33a).
3. **Simulate the closed-loop system.** Inspection of the system verifies if the system
design requirements were achieved.
4. **If the design requirements were satisfied, end with the calculated optimal
   feedback gain found in Step 2. Otherwise, adjust the Q and R matrices and
   repeat Steps 2-3 until the design requirements are satisfied.**

4.7.2 **Selection of the Optimal Feedback Gain**

Before selecting the optimal feedback gain, it is important to observe the open-
loop response to help determine the attributes of the desired closed-loop response. The
open-loop response to a reference signal of

\[
r = \begin{bmatrix} 0.03 \\ 0.35 \end{bmatrix}
\]
was simulated in MatLab. The responses of the states are shown in Figure 4.11, whereas the responses of the control inputs are shown in Figure 4.12. The value denoted as $O_{2_{\text{flame}}}$ is defined as the oxygen ratio as the volume of the chamber approaches zero. This is equivalent to the oxygen ratio produced at the flame, and is computed using Equation (4.7) where $\dot{O}_2 = 0$.

![State Variables for an Open-Loop Response to a Commanded Reference Signal](image)

Figure 4.11: Simulated system response- state variables for an open-loop response
Figure 4.12: Simulated system response- control inputs to an open-loop response

Using the iterative method presented in Section 4.7.1, the Q and R matrices were selected to satisfy the following design requirements when a Gaussian noise is considered:

- The $O_{2\text{flame}}$ must not operate outside the operation region of 0.03~0.06 when considering all possible combinations of desired fuel flow rate in the region $[0.35~0.42\, \text{g/s}]$ and desired oxygen ratio in the region $[0.03~0.06]$.
- The $O_2$ must settle to the desired reference (within 5%) in under 5 seconds.
- The fuel flow rate must settle to the desired reference (within 5%) in under 10 seconds.
- The flame must persist after a reference signal change.
- The $I_{\text{fuel}}$ control input must not vary by more than 15% of its steady-state value for any specific reference signal.
The iterative method was completed for selected combinations of Q and R matrices, which are given in Table 4.2. The Q and R matrices are defined as

\[
Q = \begin{bmatrix}
q_1 & 0 & 0 & 0 & 0 \\
0 & q_2 & 0 & 0 & 0 \\
0 & 0 & q_3 & 0 & 0 \\
0 & 0 & 0 & q_4 & 0 \\
0 & 0 & 0 & 0 & q_5
\end{bmatrix}
\]

and

\[
R = \begin{bmatrix}
r_1 & 0 \\
0 & r_2
\end{bmatrix}.
\]

Table 4.2: Gain Vector for Selection of Optimal Feedback Gain

<table>
<thead>
<tr>
<th>Gain Vector</th>
<th>q_1</th>
<th>q_2</th>
<th>q_3</th>
<th>q_4</th>
<th>q_5</th>
<th>r_1</th>
<th>r_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>K_1</td>
<td>1</td>
<td>100000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10000</td>
<td>50</td>
</tr>
<tr>
<td>K_2</td>
<td>1</td>
<td>100000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>0.1</td>
</tr>
<tr>
<td>K_3</td>
<td>1</td>
<td>5000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0.1</td>
</tr>
<tr>
<td>K_4</td>
<td>1</td>
<td>20000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The resulting state-feedback responses are shown in Figure 4.13, while the corresponding control input signals are illustrated in Figure 4.14.
Figure 4.13: Simulated system response- state variables for a state-feedback response
After completion of the iterative method, the weightings of the linear quadratic regulator were chosen as

\[
R = \begin{bmatrix} 1000 & 0 \\ 0 & 0.1 \end{bmatrix}
\]

and

\[
Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 100000 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\]

The resulting gain vector is calculated using Equation (2.33a) and Equation (2.33a), and is given as

\[
K = K_2 = \begin{bmatrix} 0.000001824944 & 0.04181643 & 0.0006444849 & 0 & 0 \\ -0.001452467 & 635.3475 & 6.026146 & 0 & 0 \\ 66 & & & & \end{bmatrix}
\]
4.7.3 Calculation of the Feed-Forward Gain

The feed-forward gain is necessary for a reference tracking system. The value of the feedforward gain can be computed using Equation (3.26) presented in Section 3.3.

Given the values of A, B, C, and K, the feedforward gain is calculated to be

\[ N = \begin{bmatrix} 206.15 & 82.075 \\ -153.35 & 1356.1 \end{bmatrix} \]

4.7.4 Selection of the Classical Controller Gains

The values of the PID controller gains were selected using a trial-and-error approach, where the simulation of the complete closed-loop was inspected to verify that the appropriate gains were selected. After the proper gains were selected, the state-space representation of the controller transfer function matrix was found. The following iterative method is used to determine the value of the integral gain.

1. Choose the structure of the controller transfer function.

The form of the controller transfer function matrix is given as

\[ U_c(z) = C(z) = \begin{bmatrix} C_1(z) & 0 \\ 0 & C_2(z) \end{bmatrix}, \]

where the error is determined by \( E(z) \equiv Z\{e_k\} = Z\{C_f(\dot{y}_k - z_k)\} \). Whereas, the controller output is defined as \( U_c(z) = Z\{-Nr_k - \bar{u} + K\Delta x_k + u_k\} \).

Therefore, the two control inputs could be independently controlled based on the two measured outputs.

2. Choose the initial value of the controller gains corresponding to \( C_1(z) \).

The controller gains corresponding to \( C_2(z) \) are set to zero, such that the controller output related to \( C_2(z) \) does not alter the selection of the controller gains of \( C_1(z) \).

Both controllers are of the form given by Equation (2.27).
3. **Simulate the closed-loop system.**

   A constant offset was added to the measurement signal to mimic a DC noise. This noise is a result of blocking the air intake pipe.

4. **Check if the design requirements were satisfied.** If the response of the closed-loop system meets the design requirements related to $C_1(z)$, use the selected controller gains and proceed to step 5. Otherwise, alter the value of the gain and repeat steps 3 and 4.

5. **Choose the initial value of the controller gains corresponding to $C_2(z)$.**

   The controller gains for $C_1(z)$ are fixed, and should not be altered when finding $C_2(z)$.

6. **Simulate the closed-loop system.**

7. **Check if the design requirements were satisfied.** If the response of the closed-loop system meets all the design requirements, end with the selected controller gains. Otherwise, alter the value of the gains and repeat steps 6 and 7.

The iterative method was completed to satisfy the following design requirements when a non-zero mean disturbance was considered:

- The PWM control input must not exceed 30%. Additionally, after a disturbance, the time-rate change of the PWM control input must not surpass $\frac{3\%}{10 \text{ sec}}$.
- The $I_{fuel}$ control input must not exceed 150mA.
- The $O_2$ must settle to the desired reference (within 5%) in under 45 seconds after the disturbance is induced.
The fuel flow rate must settle to the desired reference (within 5%) in under 30 seconds after the disturbance is induced.

The iterative method was completed for the combination of PID gains given in Table 4.3.

Table 4.3: Gain Vectors for the Selection of the PID Controller Gains

<table>
<thead>
<tr>
<th>Gain Vector</th>
<th>Controller 1</th>
<th>Controller 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_p$</td>
<td>$K_i$</td>
</tr>
<tr>
<td>$k_1$</td>
<td>150</td>
<td>30</td>
</tr>
<tr>
<td>$k_2$</td>
<td>100</td>
<td>15</td>
</tr>
<tr>
<td>$k_3$</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>$k_4$</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

The closed-loop response of the system was simulated in MatLab for the selected gain vectors. The reference signal was chosen as $r = \begin{bmatrix} 0.03 \\ 0.39 \end{bmatrix}$, and an input disturbance of

$$w = \begin{bmatrix} -0.0000179558056 \\ -0.000250234268 \\ -0.0191370078 \\ 0 \\ 0 \end{bmatrix}$$

was added at 60 seconds to simulate a 50% blockage of the air intake pipe. The closed-loop response is depicted in Figure 4.15 and Figure 4.16.
Figure 4.15: Simulated system response- closed-loop response for the complete system- state variables
Figure 4.16: Simulated system response- closed-loop response for the complete system- control Inputs

After completing the iterative method, the gain vector $k_1$ was chosen based on the closed-loop responses. The resulting controller transfer function is given as

$$C(z) = \frac{250.8z^2 - 349.3z + 100}{z^2 - z}\begin{bmatrix}0 \\ z^2 - z \\ 0.025z^2 + 0.025z \end{bmatrix}.$$  

Using the “ss()” function in Matlab, the state-space representation of $C(z)$ is established as

$$A_c = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$B_c = \begin{bmatrix} 16 \\ 0 \\ 0 \end{bmatrix},$$

$$71$$
\[ C_c = \begin{bmatrix} -6.1563 & 6.25 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \]
\[ D_c = \begin{bmatrix} 250.75 & 0 \\ 0 & 0.025 \end{bmatrix}. \]

4.8 Observer Design

Referring to Section 3.2.2, the Kalman filter gain can be computed using Equation (3.20). Since the system is time-invariant, the Kalman filter gain will quickly converge to a steady-state value. Although it is suboptimal, a pre-computed constant gain will be used. This will greatly reduce the computational burden without much loss in optimality. Before the steady-state Kalman filter gain can be computed, the measurement and process noises, which are defined by Equation (3.9a) and Equation (3.9a), need to be chosen.

4.8.1 Selection of the Measurement Noise Matrix

The measurement noise is related to the standard deviation of the measurement error. The standard deviation of the measured values can be easily computed from a collection of measurement data. Therefore, the measurement noise is defined by a diagonal matrix whose elements are the variances of the respective measurements. The variances are determined from measurement readings of the open-loop system response with a reference value of

\[ r = \begin{bmatrix} 0.045 \\ 0.39 \end{bmatrix}. \]

Although the process noise cannot be completely removed from the measurement data, the measurements were collected during the steady-state in order to minimize the effect of the process noise. In addition, the covariances were calculated with their respective moving-averages removed from their measurement value. The span of the
moving-average was selected as 50 seconds. The measurements and their moving-averaged values are shown in Figure 4.17.

![Graph showing measurement and moving-average measurement values](image)

Figure 4.17: System identification - determination of the measurement noise matrix

Utilizing the MatLab var() function, the measurement noise matrix is calculated as

$$R = \begin{bmatrix}
0.0000021483475385 & 0 & 0 \\
0 & 0.000093974214002 & 0 \\
0 & 0 & 0.00040198563389
\end{bmatrix}.$$  

4.8.2 Selection of the Process Noise Matrix

Unlike the measurement noise, the standard deviations of the error for the process noise are more difficult to obtain. However, the process noise matrix can be reconstructed using the model simulations of open-loop response depicting the “true” states and “true” outputs represented in Equations (2.8a) and (2.8a). With knowledge of the behavior of the system along with the measured output of the open-loop response, which was shown
previously in Figure 4.17, it was estimated that the peak-to-peak variations in the steady-
state values should be approximately 0.002, 0.020 grams/second, and 0.020 grams/second
respectively.

Although the system was coupled, a diagonal matrix was used to define the
process error. The following iterative method was used to approximate the process noise
matrix:

1. **Choose the initial value for the element of the process noise matrix related to
   the covariance of the oxygen flow rate.** This element is manipulated first
   because the oxygen flow rate is the only state that is independent of the other two
   states. For a diagonal matrix, increasing an element implies a larger variance on
   the corresponding state. The variance of the oxygen flow rate is affected by, but
   no limited to, the non-uniformity of the air flow, obstructions in the air flow path,
   and deviations in the air density.

2. **Simulate the open-loop response.** The outputs of the system for a reference
   signal of \( r = \begin{bmatrix} 0.045 \\ 0.39 \end{bmatrix} \) are measured.

3. **If the steady-state peak-to-peak variation of the oxygen flow is close to the
   anticipated value, use the covariance value and proceed to Step 4.** Otherwise,
   alter the element of the process noise correlated to the covariance of the oxygen
   flow rate and repeat from step 2.

4. **Choose the initial value for the element of the process noise matrix related to
   the covariance of the fuel flow rate.** This element is altered second because the
   fuel flow rate is only dependent on the oxygen flow rate, whose variance has
already been fixed. The variance of the fuel flow rate is affected by, but no
limited to, the non-uniformity of the fuel flow and deviations in the fuel density.

5. **Simulate the open-loop response conducted in step 2.**

6. **If the steady-state peak-to-peak variation of the fuel flow is within the limits
   of the anticipated value, use the covariance value and proceed to Step 7.**
   Otherwise, only alter the element of the process noise correlated to the covariance
   of the fuel flow rate and repeat from step 5.

7. **Choose the initial value for the element of the process noise matrix related to
   the covariance of the oxygen percentage in the flue.** This element was altered
   last because the oxygen percentage in the flue is dependent on both the fuel and
   air flow rates. The variance of the fuel flow rate is affected by, but no limited to,
   the differences in composition of the fuel as well as the non-uniform mixing of
   the air and fuel.

8. **If the steady-state peak-to-peak variation of the percentage of the oxygen in
   the flue is within the limits of the anticipated value, use the three covariance
   values that have been determined and finish the algorithm.** Otherwise, only
   alter the element of the process noise correlated to the covariance of the fuel flow
   rate and repeat from step 7.

Using the iterative method above, the process noise matrix was found to be

\[
Q = \begin{bmatrix}
5 \times 10^{-14} & 0 & 0 \\
0 & 8 \times 10^{-9} & 0 \\
0 & 0 & 1 \times 10^{-5}
\end{bmatrix}.
\]

Matlab was used to simulate the open-loop response shown in Figure 4.18.
4.8.3 Selection of the Observer Proportional Gains

The observer proportional gains are calculated using the Kalman filter equations summarized in Table 3.1. By simulating the Kalman filter equations with the appropriate
system equation and noise matrices found earlier, the steady-state observer proportional gain was found to converge to

\[
L_1 = \begin{bmatrix}
0.05093895 & -0.000311007 & 0.0009227998 \\
0.0249819 & 0.01246835 & 0.0008645406 \\
1.498755 & 0.03968705 & 0.03508152 \\
0.06956443 & -0.001118251 & 0.0006512646 \\
-0.6304353 & -0.02213788 & -0.0142931
\end{bmatrix}.
\]

4.8.4 Selection of the Observer Integral Gains

The selections of the observer integral gains are determined using the following iterative method:

1. **Choose the structure of the observer integral gain matrix.**

For the PI observer system described by Equation (3.7a), Equation (3.7b), and Equation (3.7c), the observer integral gain matrix is defined as

\[
L_2 = \begin{bmatrix}
l_1 & 0 & 0 \\
0 & l_2 & 0 \\
0 & 0 & l_3
\end{bmatrix}.
\]

2. **Choose the initial values of the gain components.**

3. **Simulate the response of the open-loop system.**

A non-zero mean process noise was added to the system.

4. **Check if the design requirements were satisfied.**

If the design requirements were satisfied, end with the selected observer integral gain. Otherwise, alter the values of the gain matrix and repeat steps 3 and 4.

The iterative method was completed for selected combinations of integral observer gains, which are given in Table 4.4. The open-loop responses for the selected gain vectors are illustrated in Figure 4.19.
Table 4.4: Gain Vectors for Selection of the Observer Integral Gain

<table>
<thead>
<tr>
<th>Gain Vector</th>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{2,1}$</td>
<td>0.1</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>$L_{2,2}$</td>
<td>0.01</td>
<td>0.02</td>
<td>0.1</td>
</tr>
<tr>
<td>$L_{2,3}$</td>
<td>0.005</td>
<td>0.03</td>
<td>0.2</td>
</tr>
<tr>
<td>$L_{2,4}$</td>
<td>0.001</td>
<td>0.007</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Figure 4.19: Simulated system response- open-loop response for selected observer gain vectors
Using the iterative method, the observer integral gain matrix was selected as

\[
L_z = \begin{bmatrix}
0.001 & 0 & 0 \\
0 & 0.007 & 0 \\
0 & 0 & 0.20 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}.
\]

4.9 Closed-loop Stability

As presented in Section 3.3.4, the dynamics of the closed loop system is described by the system matrix given by Equation (3.36). Using the \( A, A_s, B, B_s, C, \) and \( C_s \) system matrices and the state feedback gain matrix, \( K \), the closed loop system matrix is described by eigenvalues given as

\[
eig(\tilde{A}) = \begin{bmatrix}
0.0009076629 \\
0.8835039 + 0.001787835i \\
0.8835039 - 0.001787835i \\
0.9343845 \\
0.9733967 \\
0.9983409 + 0.006392248i \\
0.9983409 - 0.006392248i \\
0.9999631 \\
0.9764155 \\
0.9935054 \\
0.8910956 \\
0.9035694 \\
0.9077556
\end{bmatrix}.
\]

The stability of the system is proven since all of the eigenvalues lie within the unit circle.
CHAPTER V
EXPERIMENTAL RESULTS AND VALIDATION

In the previous chapter, the controller and observer were designed and simulated using MATLAB. The controller was shown to have met the design requirements of the system. This chapter explores the experimental results, and attempts to validate that the system still meets the requirements with a real-world application. The realization of the developed model, the observer, and the combined observer-controller system were verified.

The observer was implemented in a PIC Microcontroller, and was tested on an open-loop system for changes in reference signal in addition to disturbances. Likewise, the combined controller-observer was also implemented on a PIC microcontroller, and it was tested against disturbances and reference signal changes. The ability of the observer-controller to maintain the reference signal in the presence of moderate disturbances was examined. The coding of the microcontroller can be found in the Appendices G,H, and I.

5.1 Model Validation

Experimental data was collected to demonstrate the success of the developed nonlinear and linearized models. The oxygen ratio in the flue was measured with the Testo analyzer for various combinations of fuel regulator current and blower motor PWM. In Figure 5.1, the measured data is compared with the oxygen ratio corresponding
to the linear and nonlinear models. Figure 5.2 illustrates the comparison of the measured and the linear model data representing the oxygen ratio for various blower motor PWMs along curves of constant fuel regulator currents. For the operating range which was previously selected, the fuel regulator current and blower motor PWM can vary between the range [112mA,127mA] and [19%,30%]. This results in a maximum error between the measured oxygen ratio and the value obtained by the linear model of 0.007.

![Steady-State Input-Output Model Data](image)

Figure 5.1: System validation- comparison of the measured, linearized model, and non-linear model data
5.2 Observer Validation

The observer illustrated in Figure 3.2 was tested against the open-loop system for a changing reference signal and induced disturbances. The ability of the observer to estimate the states effectively was proven by validating the convergence of the estimated states to the measured outputs. Additionally, it was verified that the observer could accurately estimate the unobservable states.

First, the observer was tested for changes of the reference signal only. The reference signal was arbitrarily defined and was programmed to change every 45 seconds. The reference signal and the resulting inputs associated with this test are illustrated in Figure 5.3 and Figure 5.4, respectively.
Figure 5.3: Observer validation- reference signal changes-reference signals
Figure 5.4: Observer validation- reference signal changes- control input signals

Figure 5.5 illustrates the measured outputs, estimated outputs, and the estimated values of selected states. The ability of the observer to accurately estimate the unobservable states can be deduced by inspection of the figure. Due to measurement delays, it is not possible to determine the actual values of the states. Therefore, the states defined as $O_2$, $\Phi_f$, and $\Phi_{O_2}$ are considered unobservable. Nonetheless, it is shown that these states can be accurately estimated because their estimates value anticipate the measured outputs. This inference is justified because the simulated states follow more closely to the estimated states than the measured outputs.

The convergence of the estimated states to the measured outputs is proven in Figure 5.5. The output estimation error is defined as the difference between the estimated
output and the measured output. Figure 5.6 illustrates the output estimation errors of the oxygen ratio, fuel flow rate, and oxygen flow rate. It is shown that the estimated outputs converge to within approximately 0.003, 0.006 $\frac{g}{s}$, and 0.02 $\frac{g}{s}$ of the measured oxygen ratio, fuel flow rate, and oxygen flow rate respectively. However, given the nature of the PI observer, these values will gradually converge to zero. By increasing the appropriate observer integral gains, the convergence would speed up. However, the chosen gains are shown to be adequate since the steady-state values of the estimated states closely match the measured outputs.
Figure 5.5: Observer validation- reference signal changes- measured outputs, estimated outputs, and estimated states
In the second test case, the observer was tested against a constant reference signal with a disturbance. The disturbance was induced by physically blocking the air intake pipe by approximately 50% of the surface area of the orifice. This blockage resulted in a
drop of the oxygen flow rate from $2.16 \, \frac{g}{s}$ to $1.86 \, \frac{g}{s}$. The control input signals for a reference signal of

$$r = \begin{bmatrix} 0.055 \\ 0.38 \end{bmatrix}$$

resulted in a constant control signal of

$$\begin{bmatrix} \text{PWM} \\ I_{\text{fuel}} \end{bmatrix} = \begin{bmatrix} 27.4\% \\ 120.5mA \end{bmatrix}.$$  

The measured outputs, estimated outputs, and selected estimated states are shown in Figure 5.7. It is shown that the observer is capable of closely estimating the states in the presence of a large non-zero mean disturbance. In addition, the estimation errors, which are illustrated in Figure 5.8, show that the estimated outputs converge to the measured outputs.
Figure 5.7: Observer validation - induced disturbance - measured and estimated states
5.3 Closed-Loop Observer-Controller Validation

The closed-loop controller and state observer were implemented in a microcontroller. Figure 3.8 illustrates the simulation diagram representing the observer-controller system. The closed-loop system was tested against a changing reference signal.
in addition to a non-zero mean disturbance, and the ability of controller to regulate the system to the desired reference signal was verified.

First, the observer-controller was tested for changes of the reference signal without a non-zero mean disturbance. The reference signal and the resulting inputs associated with this test are illustrated in Figure 5.9 and Figure 5.10.

Figure 5.9: Observer-controller validation- references changes-reference signals
The estimated and the measured states are shown in Figure 5.11. The output error is defined as the difference between the measured outputs and the desired reference signals. The output errors, illustrated in Figure 5.12, show that the measured oxygen ratio settles to within 5% of the steady-state value in approximately 32 seconds. Likewise, the measured fuel flow rate settles to within 5% of the steady-state value in approximately 26 seconds.
Figure 5.11: Observer-controller validation—references changes—measured outputs, estimated outputs, and estimated states
Finally, the observer-controller was tested against a constant reference signal with a disturbance. The disturbance was a result of a 50% blockage of the surface area of the air intake aperture. The reference signal was selected as

\[ r = \begin{bmatrix} 0.03 \\ 0.38 \end{bmatrix} \]

and the resulting control input signals are illustrated in Figure 5.13.
Figure 5.13: Observer-controller validation- induced disturbance- control input signals

The measured and estimated states are shown in Figure 5.14. The output errors, shown in Figure 5.15, verify that the controller-observer system can meet all the requirements set in Section 4.7.2 and Section 4.7.3. The measured oxygen ratio settles to within 5% of the steady-state value in approximately 37 seconds. Likewise, the measured fuel flow rate settles to within 5% of the steady-state value in approximately 16 seconds.
Figure 5.14: Observer-controller validation - induced disturbance – measured states, estimated outputs, and estimated states
Figure 5.15: Observer-controller validation- induced disturbance – output errors
A non-linear state-space model was developed to represent the combustion process. The model was generic to suffice different combustion parameters including the oxygen-to-air ratio, the stoichiometric air-to-fuel ratio, the densities of the flue gas and oxygen, and the volume of the chamber. The dynamics of the actuator were integrated with this model to completely represent the combustion boiler system. A generic linearized model of the combustion boiler system was then obtained. System identification methods can be used to determine the specific actuator and sensor models for any boiler system. Additionally, a generic observer-controller system was developed to regulate the oxygen ratio and fuel flow rate of most boiler systems. However, the controller and observer gains would need to be recalculated for boiler systems of different models.

It was demonstrated that the observer-controller was able to regulate the oxygen ratio and the fuel flow rate against reference signal changes and an induced non-zero mean disturbance. Furthermore, all the established design requirements were satisfied. The observer was proven to accurately estimate the unobservable states for non-zero mean disturbance and changes in the reference signal. Likewise, for an induced non-zero mean disturbance, the observer-controller was able to regulate the oxygen ratio and the
fuel flow rate to the desired values in approximately 37 seconds and 16 seconds respectively. It was proven in this thesis that the developed state-feedback plus PID with feedforward control was capable of controlling the non-linear system.

Suggested future work includes developing a non-linear controller for the non-linear model that was formulated. The implementation of a non-linear controller would effectively expand the range of operation. Likewise, the development of a non-linear observer could be applied.
BIBLIOGRAPHY


APPENDIX A

PICTURES OF THE BOILER SYSTEM

Figure A.1: Combustion Boiler System - Bosch wall-hung boiler ZBR21-3A
Figure A.2: (a) Modified air intake with Bosch air flow sensor (b) Fuel regulator with modulating gas valve
APPENDIX B

MATLAB CODE FOR SIMULATION OF NON-LINEAR MODEL

%nonlinear model
%Marcus Horning

clc
close all
clear all

%system parameters
MO2=31.999;
Mfuel=17.82;
r_O2_to_air=0.232;
r_O2_to_fuel=2.195;
p_flue=1200;
p_O2=1331;
V=0.015;
tao1=1.5; %time constant fuel
tao2=0.5; %time constant O2
PWM=24;
I_fuel=120;

%simulation parameters
Ts=0.001;
t=0:Ts:25;
N=length(t);

%initial conditions
O2(1)=.21;
If(1)=0;
Io2(1)=0;

for n=1:N-1
    Uo2(n)= -0.001069755249*PWM^2 +0.10940425590*PWM - 0.10408973987;
    Uf(n)=(-0.000029056416*I_fuel + 0.05341179827)*Io2(n)+(0.002425215793*I_fuel - 0.006952195693);
    Ifd(n)=1/tao1*(-If(n)+Uf(n));
    Io2d(n)=1/tao2*(-Io2(n)+Uo2(n));
\[ O_{2d}(n) = \frac{1}{(V \cdot p_{O2})} \left( I_{O2}(n) - \frac{M_{O2}}{M_{fuel}} \cdot r_{O2 \to fuel} \cdot I_f(n) \right) - \frac{1}{(p_{flue} \cdot V)} \left( I_f(n) + \frac{1}{r_{O2 \to air}} \cdot I_{O2}(n) \right) \cdot O_{2}(n) \]

\[ O_{2}(n+1) = O_{2}(n) + Ts \cdot O_{2d}(n) \]

\[ O_{2 \ instantaneous}(n) = \frac{1}{(V \cdot p_{O2})} \left( I_{O2}(n) - \frac{M_{O2}}{M_{fuel}} \cdot r_{O2 \to fuel} \cdot I_f(n) \right) / \left( 1 \cdot (p_{flue} \cdot V) \cdot (I_f(n) + I_{O2}(n) / (r_{O2 \to air})) \right) \]

\[ I_f(n+1) = I_f(n) + Ts \cdot I_{fd}(n) \]

\[ I_{O2}(n+1) = I_{O2}(n) + Ts \cdot I_{2d}(n) \]

end

figure(1);
hold on;
plot(t, O2);
plot(t(1:length(t)-1), O2_instantaneous);
hold off;

figure(2);
hold on;
plot(t, I_f);
plot(t, I_{O2});
hold off;
APPENDIX C
MATLAB CODE FOR THE LINEARIZATION AND DISCRETIZATION OF THE NON-LINEAR MODEL

% Marcus Horning
% Linearization and Discretization of the Non-linear Model

run systemID
clc
clearvars -EXCEPT As Bs Cs
close all

% |Vpd|   | a1   b1     b2      | |Vp|     |0      0 |
% |Ifd|   | 0 -1/tao1   g3/tao1 | |If|   + |0 g2/tao1| |u1|
% |Io2d|= |0    0    -1/tao2    | |Io2|    |g1/ta02 0|  |u2|

% u2=I
% u1==PWM

% System parameters
M_O2=31.999;
M_fuel=17.82;
r_O2_to_air=0.232;
r_O2_to_fuel=2.195;
p_flue=1200;
p_O2=1331;
V=0.015;
tao1=1.5;  % time constant fuel
tao2=0.5;  % time constant O2

% Steady State parameters
Vpss=0.0432;
Ifss=0.37906796;
Io2ss=1.9054334;
PWMin=24;
Ifuelss=120;
xss=[Vpss Ifss Io2ss ]';
\begin{align*}
a_1 &= -\frac{1}{(p_{\text{flue}}V)}(\text{Ifss}+1/(r_{\text{O}_2 \text{ to air}})\text{Io}_2\text{ss})
\end{align*}

\begin{align*}
b_1 &= -\frac{1}{(Vp_{\text{O}_2})}\text{MO}_2/\text{Mfuel}r_{\text{O}_2 \text{ to fuel}}V\text{pss}/(p_{\text{flue}}V)
\end{align*}

\begin{align*}
b_2 &= \frac{1}{(Vp_{\text{O}_2})}V\text{pss}/(p_{\text{flue}}V)
\end{align*}

\begin{align*}
g_1 &= -0.001069755249*\text{PWMss}*2 + 0.109404255903
\end{align*}

\begin{align*}
g_2 &= -0.0000290564161*\text{Io}_2\text{ss}+0.002425215793
\end{align*}

\begin{align*}
g_3 &= -0.0000290564161*\text{Ifuelss}+0.05334117982
\end{align*}

\begin{align*}
A &= [a_1 \ b_1 \ b_2 \ 0 \ -1/\tau_01 \ g_3/\tau_01 \ 0 \ 0 \ -1/\tau_02 \ ]
\end{align*}

\begin{align*}
B &= [0 \ 0; \ g_2/\tau_01; \ g_1/\tau_02 \ 0; \ ]
\end{align*}

\begin{align*}
C &= [1 \ 0 \ 0; \ 0 \ 0 \ 1]
\end{align*}

\begin{align*}
D &= \text{zeros}(2,2)
\end{align*}

\begin{align*}
\text{Ts}_2 &= 0.05;
\text{t}_1 &= 0:\text{Ts}_2:10; \quad \% \text{open loop}
\text{Nl} &= \text{length}(\text{t}_1);
\text{SYSC} &= \text{ss}(A,B,C,D);
[\text{SYSD},G] &= \text{c2d}(\text{SYSC},\text{Ts}_2,\text{'zoh'});
[\text{Ad},\text{Bd},\text{Cd},\text{Dd}] &= \text{ssdata}(\text{SYSD});
\end{align*}

\begin{align*}
\text{Nx} &= 2; \quad \% \text{sub-system order}
\end{align*}

\begin{align*}
\text{Ad} &= [\text{Ad} \ \text{zeros}(3,\text{Nx}); \ \text{Bs} \ \text{As}] ;
\text{Bd} &= [\text{Bd}; \ \text{zeros}(\text{Nx},2)];
\text{Cd} &= [[\text{zeros}(1,3) \ \text{Cs}(1,:)]; \ 0 \ 0 \ \text{zeros}(1,\text{Nx}); \ [\text{zeros}(1,3) \ \text{Cs}(2,:)]];\]
\text{Cf} &= [1 \ 0 \ 0; \ 0 \ 0 \ 1]; \quad \% \text{sensor feedback matrix}
\end{align*}

\begin{align*}
\% \text{realization properties}
\text{co} &= \text{ctrb}(\text{Ad},\text{Bd});
\text{ob} &= \text{obsv}(\text{Ad},\text{Cd});
\% \ \text{An} &= [\text{Ad} - \text{Bd}*\text{G} \ \text{zeros}(5,5) \ \text{Bd}*\text{Cc}; \ \text{zeros}(5,5) \ \text{Ad} - \text{Bd}*\text{G} \ \text{zeros}(5,3)];
\% \ -\text{Bc}^*\text{Cf}^*\text{Cd} \ \text{Bc}^*\text{Cd}^2 \ \text{Ac}];
\end{align*}
APPENDIX D

MATLAB CODE FOR THE SELECTION OF THE CONTROLLER GAINS

%CONTROLLER DESIGN
%Marcus Horning

run Linearized_model_estimated_4_30_2
close all
c1c
clearvars -EXCEPT Ad Bd Cd Cf xss Nx Ts2 PWMss Ifuelss K

%Select Type of Response
Response=1;
  % Open-loop=1
  % Feedback=2
  % Closed-loop=3

% Set simulation time
if Response==1| Response==2
  t1=0:Ts2:10; %open loop
  Nl=length(t1);
else
  t1=0:Ts2:200; %open loop
  Nl=length(t1);
end

%Measurement and Process Noise Matrices
R=[0.0000000019020417687 0 0 ;0 0.00000091146242844 0; 0 0.0003931759603 ];
Q=[0.00000000000000005 0 0 zeros(1,Nx) ;0 0.0000000008 0 zeros(1,Nx) ;
  0 0 0.00001 zeros(1,Nx); zeros(1,3+Nx); zeros(1,3+Nx) ];
v=sqrt(R)*randn(3,Nl); %measurement noise
w=sqrt(Q)*(randn(3+Nx,Nl)); %process noise
%PID parameters :Controller C1
Kp=[150 100 50 10]; %set proportional gain
Ki=[30 15 5 1]; %set integral gain
Kd=[5 10 20 10]; %set derivative gain
Ts2=0.05;
a=(Kp+Ki*Ts2/2+Kd/Ts2);
b=(-Kp+Ki*Ts2/2-2*Kd/Ts2);
c=Kd/Ts2;

%PID parameters :Controller C2
Kp=[0 0 0 0]; %set proportional gain
Ki=[1 1.5 2 3]; %set integral gain
Kd=[0 0 0 0]; %set derivative gain
a2=(Kp+Ki*Ts2/2+Kd/Ts2);
b2=(-Kp+Ki*Ts2/2-2*Kd/Ts2);
c2=Kd/Ts2;

%Determining Feedback Gain
Q1=[1 0 0 zeros(1,Nx) ; 0 100000000 0 zeros(1,Nx) ;
    0 0 1 zeros(1,Nx); zeros(Nx,Nx+3)];
R1=[10000 0 ; 0 50 ];
Q2=[1 0 0 zeros(1,Nx) ; 0 100000 0 zeros(1,Nx) ;
    0 0 1 zeros(1,Nx); zeros(Nx,Nx+3)];
R2=[1000 0 ; 0 0.1 ];
Q3=[1 0 0 zeros(1,Nx) ; 0 500000 0 zeros(1,Nx) ;
    0 0 1 zeros(1,Nx); zeros(Nx,Nx+3)];
R3=[5000 0 ; 0 0.1 ];
Q4=[1 0 0 zeros(1,Nx) ; 0 20000 0 zeros(1,Nx) ;
    0 0 1 zeros(1,Nx); zeros(Nx,Nx+3)];
R4=[1000 0 ; 0 0.1 ];
S=zeros(3+Nx,2);
E=eye(3+Nx);
[X,L,K1]=dare(Ad,Bd,Q1,R1,S,E);
[X,L,K2]=dare(Ad,Bd,Q2,R2,S,E);
[X,L,K3]=dare(Ad,Bd,Q3,R3,S,E);
[X,L,K4]=dare(Ad,Bd,Q4,R4,S,E);

%Select Feedback Gain
if Response==2
    K(:, :, 1)=K1;
    K(:, :, 2)=K2;
    K(:, :, 3)=K3;
    K(:, :, 4)=K4;
else
    K=K2; %preferred Gain
end

%Determining Feedforward Gain
Cd2=[Cd(1,:) ; 0 1 0 zeros(1,Nx) ]; % volume and fuel flow (set outputs)
if Response==1
    N=inv(-Cd2*inv(Ad-Bd*zeros(2,3+Nx)-eye(3+Nx))*Bd);
```MATLAB
elseif Response==2
    N(:,:,1)=inv(-Cd2*inv(Ad-Bd*K1-eye(3+Nx))*Bd);
    N(:,:,2)=inv(-Cd2*inv(Ad-Bd*K2-eye(3+Nx))*Bd);
    N(:,:,3)=inv(-Cd2*inv(Ad-Bd*K3-eye(3+Nx))*Bd);
    N(:,:,4)=inv(-Cd2*inv(Ad-Bd*K4-eye(3+Nx))*Bd);
else
    N=inv(-Cd2*inv(Ad-Bd*K-eye(3+Nx))*Bd);
end

%Set reference signal
r=[0.03; 0.35];
r=[r(1)-xss(1) r(2)-xss(2)];

%set initial conditions
xt=zeros(3+Nx,Nl-1,length(a)); %truth
uc=zeros(2,Nl-1,length(a));
xs=zeros(3+Nx,Nl-1,length(a));
z=zeros(3,Nl-1,length(a));
error=zeros(2,Nl-1,length(a));

for i=1:length(a)
    for n=1:Nl-1
        %Integral action
        if (Response==3)
            us(:,n+1,i)=K*xs(:,n,i)+N*r';
            xs(:,n+1,i)=Ad*xs(:,n,i)+Bd*us(:,n,i);
            error(:,n+2,i)=Cf*Cd*xs(:,n,i)+[xss(1);xss(2)]-
                           [z(1,n,i)+xss(1);(z(2,n,i)+xss(2))];
            uc(:,n+1,i)=[uc(1,n,i)+a(i)*error(1,n+2,i)+b(i)*error(1,n+1,i)+c(i)*error(1,n,i);
                         uc(2,n,i)+a2(i)*error(2,n+2,i)+b2(i)*error(2,n+1,i)+c2(i)*error(2,n,i)];
        end
        %Input Selection
        if Response==1
            u(:,n+1,i)=N*r';
        elseif Response==2
            u(:,n+1,i)=-K(:,:,i)*xt(:,n,i)+N(:,:,i)*r';
        else
            u(:,n+1,i)=-K*xt(:,n,i)+N*r'+uc(:,n+1,i);
        end

    %Adding non-zero mean disturbance
    if n>(60/0.05)
        d=[0.015 0.2 ]';
    else
        d=[0 0 ]';
    end

    if Response==2
```
N2 = inv([-1 0 0 zeros(1,Nx); 0 0 1 zeros(1,Nx)] * inv(Ad-Bd*K(:,:,i)-eye(3+Nx))*Bd);
else
N2 = inv([-1 0 0 zeros(1,Nx); 0 0 1 zeros(1,Nx)] * inv(Ad-Bd*K-eye(3+Nx))*Bd);
end

% Simulated True States and Measurements
xt(:,n+1,i) = Ad*xt(:,n,i) + Bd*u(:,n+1,i) + w(:,n) + Bd*N2*d;
% truth
z(:,n+1,i) = Cd*xt(:,n+1,i) + v(:,n);

% Instantaneous O2
X1ss(1,n,i) = 1/(1-Ad(1,1))*(Ad(1,2)*xt(2,n,i) + Ad(1,3)*xt(3,n,i) + Bd(1,1)*u(1,n,i) + Bd(1,2)*u(2,n,i));

% Select Preferred PID gains
a = a(1);
b = b(1);
c = c(1);
a2 = a2(1);
b2 = b2(1);
c2 = c2(1);

% PID transfer function and State-space representation
H_num1 = [a b c];
H_den1 = [1 -1 0];
H_num2 = [a2 b2 c2];
H_den2 = [1 -1 0];
H11 = tf(H_num1, H_den1, 0.05);
H12 = 0;
H21 = 0;
H22 = tf(H_num2, H_den2, 0.05);
H = [H11 H12; H21 H22];

[Ac, Bc, Cc, Dc] = ssdata(ss(H, 'min'));

xt(1,:,:)=xt(1,:,:)+xss(1);
xt(2,:,:)=xt(2,:,:)+xss(2);
xt(3,:,:)=xt(3,:,:)+xss(3);
z(1,:,:)=z(1,:,:)+xss(1);
z(2,:,:)=z(2,:,:)+xss(2);
z(3,:,:)=z(3,:,:)+xss(3);
u(1,:) = u(1,:) + PWMss;
u(2,:) = u(2,:) + Ifuelss;
X1ss=X1ss+xss(1);

% Closed Loop Response
if Response==3
    figure(1);
    subplot(3,1,1,'YTick', [0.01 0.02 0.03 0.04 0.05 0.06]);
    title('Closed-loop Response ');
    ylabel('O_{2}');
    legend(' k_{1}', 'k_{2}', 'k_{3}', 'k_{4}');
    hold off;
    subplot(3,1,2);
    hold on;
    plot(t1(1:Nl),z(1,:), 'g');
    plot(t1(1:Nl),xt(1,:,1));
    plot(t1(1:Nl),xt(1,:,2));
    plot(t1(1:Nl),xt(1,:,3));
    plot(t1(1:Nl),xt(1,:,4));
    ylabel('$\Phi_{f} (g/s)$');
    legend(' k_{1}', 'k_{2}', 'k_{3}', 'k_{4}');
    hold off;
    subplot(3,1,3);
    hold on;
    plot(t1(1:Nl),z(3,:), 'g');
    plot(t1(1:Nl),xt(3,:,1));
    plot(t1(1:Nl),xt(3,:,2));
    plot(t1(1:Nl),xt(3,:,3));
    plot(t1(1:Nl),xt(3,:,4));
    ylabel('$\Phi_{O_{2}} (g/s)$');
    legend(' k_{1}', 'k_{2}', 'k_{3}', 'k_{4}');
    xlabel('Time, in seconds');
    hold off;
figure(2);
    subplot(2,1,1);
    hold on;
    plot(t1(1:Nl),u(1,:,1));
    plot(t1(1:Nl),u(1,:,2));
    plot(t1(1:Nl),u(1,:,3));
    plot(t1(1:Nl),u(1,:,4));
    hold off;
    title('Control Inputs for an Open-Loop Response to a Commanded Reference Signal');
    ylabel('PWM (%)');
    legend(' k_{1}', 'k_{2}', 'k_{3}', 'k_{4}');
    subplot(2,1,2);
    hold on;
    plot(t1(1:Nl),u(2,:,1));
    plot(t1(1:Nl),u(2,:,2));
    plot(t1(1:Nl),u(2,:,3));
    plot(t1(1:Nl),u(2,:,4));
    hold off;
ylabel('I_{fuel} (mA)');
xlabel('Time (seconds)');
legend('k_{1}','k_{2}','k_{3}','k_{4}');
end

if Response==2

% Feedback Response (states)
figure(2);
subplot(4,1,1);
hold on;
plot(t(1:Nl),x(:,1));
plot(t(1:Nl),x(:,2));
plot(t(1:Nl),x(:,3));
plot(t(1:Nl),x(:,4));
ylim([0.015 0.05]);
title('State Variables for a State-Feedback Response to a Commanded Reference Signal');
ylabel('O_{2}');
legend('K_{1}','K_{2}','K_{3}','K_{4}');
hold off;
subplot(4,1,2);
hold on;
plot(t(1:Nl-1),X1ss(:,1));
plot(t(1:Nl-1),X1ss(:,2));
plot(t(1:Nl-1),X1ss(:,3));
plot(t(1:Nl-1),X1ss(:,4));
ylim([0.015 0.05]);
ylabel('O_{2}_flame');
legend('K_{1}','K_{2}','K_{3}','K_{4}');
hold off;
subplot(4,1,3);
hold on;
plot(t(1:Nl),x(:,1));
plot(t(1:Nl),x(:,2));
plot(t(1:Nl),x(:,3));
plot(t(1:Nl),x(:,4));
ylabel('$\Phi_{f} \,(g/s)$');
legend('K_{1}','K_{2}','K_{3}','K_{4}');
hold off;
subplot(4,1,4);
hold on;
plot(t(1:Nl),x(:,1));
plot(t(1:Nl),x(:,2));
plot(t(1:Nl),x(:,3));
plot(t(1:Nl),x(:,4));
ylabel('$\Phi_{O_{2}} \,(g/s)$');
legend('K_{1}','K_{2}','K_{3}','K_{4}');
hold off;

% Open loop Response (inputs)
figure (3);
subplot(2,1,1);
hold on;
plot(t(1:Nl),u(:,1));
plot(t1(1:Nl),u(1,:,2));
plot(t1(1:Nl),u(1,:,3));
plot(t1(1:Nl),u(1,:,4));
title('Control Inputs for State-Feedback Response to a Commanded Reference Signal');
ylabel('PWM (%)');
legend(' K_{1}', 'K_{2}', 'K_{3}', 'K_{4}');
hold off;

subplot(2,1,2);
hold on;
plot(t1(1:Nl),u(2,:,1));
plot(t1(1:Nl),u(2,:,2));
plot(t1(1:Nl),u(2,:,3));
plot(t1(1:Nl),u(2,:,4));
ylabel('I_{fuel} (mA)');
hold off;
xlabel('Time (seconds)');
legend(' K_{1}', 'K_{2}', 'K_{3}', 'K_{4}');
end

if Response==1
    figure(4);
    subplot(5,1,1);
    hold on;
    plot(t1(1:Nl),xt(1,:,1));
    plot(t1(1:Nl-1),X1ss(1,:,1));
ylim([0.015 0.05]);
title('State Variables for an Open-Loop Response to a Commanded Reference Signal');
ylabel('O_{2}');
legend(' True State', 'O_{2}_{flame}');
hold off;
    subplot(5,1,2);
    plot(t1(1:Nl),xt(2,:,1));
ylabel('\Phi_{f} (g/s)');
    subplot(5,1,3);
    plot(t1(1:Nl),xt(3,:,1));
ylabel('\Phi_{O_{2}} (g/s)');
    subplot(5,1,4);
    plot(t1(1:Nl),xt(4,:,1));
ylabel('x_{1}');
    subplot(5,1,5);
    plot(t1(1:Nl),xt(5,:,1));
ylabel('x_{2}');
xlabel('Time (seconds)');
end

%Open loop Response (inputs)
figure (5);
subplot(2,1,1);
plot(t1(1:Nl),u(1,:,1));
title('Control Inputs for an Open-Loop Response to a Commanded Reference Signal');
ylabel('PWM (\%)');
subplot(2,1,2);
plot(t1(1:Nl),u(2,:,1));
ylabel('I_{fuel} (mA)');
xlabel('Tīme (seconds)');
end
APPENDIX E
MATLAB CODE FOR SELECTION OF OBSERVER GAINS

% Observer DESIGN
% Marcus Horning

run Selection_Controller_Gains

close all
clc
clearvars -EXCEPT Ad Bd Cd Cf xss N1 Nx Ts2 PWMss Ifuelss K Ac BcCc Dc a b c a2 b2 c2
% Set simulation parameters
T=0:Ts2:120;
Nl=length(T);

% Measurement and Process noise
% measurement noise covariance
R=[0.00000019020417687 0 0; 0 0.0000091146242844 0; 0 0 0.00039317159603 ];
% process noise covariance
Q=[0.00000000000005 0 0 zeros(1,Nx) ; 0 0.000000008 0 zeros(1,Nx) ; 0 0.00001 zeros(1,3+Nx) zeros(1,3+Nx) ];
v=sqrt(R)*randn(3,Nl);
w=sqrt(Q)*(randn(3+Nx,Nl));

% Set feedforward gain
Cd2=[Cd(1,:) ; 0 0 zeros(1,Nx) ];
N=inv(-Cd2*inv(Ad-Bd*zeros(2,3+Nx)-eye(3+Nx))*Bd);
N2=inv([-1 0 zeros(1,Nx); 0 0 1 zeros(1,Nx)]*inv(Ad-Bd*zeros(2,3+Nx)-eye(3+Nx))*Bd);

r=[0.03; 0.35];

r=[r(1)-xss(1) r(2)-xss(2)];
% Select observer integral gain
L2(:,:,1)=[0.1 0 0 ;0 0.05 0 ;0 0 0.15; zeros(1,3); zeros(1,3) ];
L2(:,:,2)=[0.01 0 0 ;0 0.02 0 ;0 0 0.1; zeros(1,3); zeros(1,3) ];
L2(:,:,3)=[0.005 0 0 ;0 0.03 0 ;0 0 0.2; zeros(1,3); zeros(1,3) ];
L2(:,:,4)=[0.001 0 0 ;0 0.007 0 ;0 0 0.2; zeros(1,3); zeros(1,3) ];

% set initial conditions
Pm=0.5*ones(3+Nx,Nl-1,4);
xm=zeros(3+Nx,Nl-1,4);
xt=zeros(3+Nx,Nl-1);

%truth
neta_Kal=zeros(3,Nl-1,4);
z=zeros(3,Nl-1);

for i=1:4
  for n=1:Nl-1
    L_temp(:,(3)*n-(2):(3)*n,i)=Pm(:,((3+Nx)*n-(2+Nx)):(3+Nx)*n,i)*Cd'*inv(Cd*Pm(:,((3+Nx)*n-(2+Nx)):(3+Nx)*n,i)*Cd'+R);
    L1(:,:,i)=L_temp(:,(3)*n-(2):(3)*n,i);
    Pp(:,((3+Nx)*n-(Nx+2)):(3+Nx)*n,i)=(eye(3+Nx)-L1(:,:,i)*Cd)*Pm(:,((3+Nx)*n-(Nx+2)):(3+Nx)*n,i);
    Pm(:,((3+Nx)*(n+1)-(Nx+2)):(3+Nx)*(n+1),i)=Ad*Pp(:,((3+Nx)*n-(Nx+2)):(3+Nx)*n,i)*Ad'+Q;
  end
end

for i=1:4
  for n=1:Nl-1

    %update phase
    neta_Kal(:,n+1,i)=neta_Kal(:,n,i)+Ts2*(z(:,n)-Cd*xm(:,n,i));
    xp(:,n,i)=xm(:,n,i)+L1(:,:,i)*(z(:,n)-Cd*xm(:,n,i))+L2(:,:,i)*neta_Kal(:,n,i);
    u(:,n+1)=N*r';
  end
end

% Add a non-zero mean disturbance
if n>(60/0.05)
  d=[0.015 0.2 ]';
else
  d=[0 0 ]';
end

%Calculate truths and measurement values
xt(:,n+1)=Ad*xt(:,n)+Bd*u(:,n+1)+w(:,n)+Bd*N2*d;  %truth
z(:,n+1)=Cd*xt(:,n+1)+v(:,n);

%Prediction phase
xm(:,n+1,i)=Ad*xp(:,n,i)+Bd*u(:,n);
end

% Select preferred observer gains
L1=L1(:,:,1);
L2=L2(:,:,1);

xp(1,:,:)=xp(1,:,:)+xss(1);
xp(2,:,:)=xp(2,:,:)+xss(2);
xp(3,:,:)=xp(3,:,:)+xss(3);
xt(1,:)=xt(1,:)+xss(1);
xt(2,:)=xt(2,:)+xss(2);
xt(3,:)=xt(3,:)+xss(3);
z(1,:)=z(1,:)+xss(1);
z(2,:)=z(2,:)+xss(2);
z(3,:)=z(3,:)+xss(3);
u(1,:)=u(1,:)+PWMss;
u(2,:)=u(2,:)+Ifuelss;

% Observer Gain Selection
figure(4);
subplot(5,1,1);
hold on;
plot(t1(1:Nl),z(1,:),'g');
plot(t1(1:Nl),xt(1,:),'+r');
plot(t1(1:Nl-1),xp(1,:,1));
plot(t1(1:Nl-1),xp(1,:,2));
plot(t1(1:Nl-1),xp(1,:,3));
plot(t1(1:Nl-1),xp(1,:,4));
ylabel('\hat{O}_2','Interpreter','Latex');
hold off;

subplot(5,1,2);
hold on;
plot(t1(1:Nl),z(2,:),'g');
plot(t1(1:Nl),xt(2,:),'+r');
plot(t1(1:Nl-1),xp(2,:,1));
plot(t1(1:Nl-1),xp(2,:,2));
plot(t1(1:Nl-1),xp(2,:,3));
plot(t1(1:Nl-1),xp(2,:,4));
ylabel('\hat{\Phi}_f','Interpreter','Latex');
hold off;

subplot(5,1,3);
hold on;
plot(t1(1:Nl),z(3,:),'g');
plot(t1(1:Nl),xt(3,:),'+r');
plot(t1(1:Nl-1),xp(3,:,1));
plot(t1(1:Nl-1),xp(3,:,2));
plot(t1(1:Nl-1),xp(3,:,3));
plot(t1(1:Nl-1),xp(3,:,4));
legend('Measured State','True State','L_{2,1}','L_{2,2}','L_{2,3}','L_{2,4}');
ylabel('\hat{\Phi}_{O_2}','Interpreter','Latex');
hold off;

ylabel('\hat{\Phi}','Interpreter','Latex');
hold off;
subplot(5,1,4);
hold on;
plot(t1(1:Nl),xt(4,:), 'r')
plot(t1(1:Nl-1),xp(4,:,1));
plot(t1(1:Nl-1),xp(4,:,2));
plot(t1(1:Nl-1),xp(4,:,3));
plot(t1(1:Nl-1),xp(4,:,4));
ylabel('$\hat{\mathbf{X}}_{1}$', 'Interpreter', 'Latex');
hold off;
subplot(5,1,5);
hold on;
plot(t1(1:Nl),xt(5,:), 'r')
plot(t1(1:Nl-1),xp(5,:,1));
plot(t1(1:Nl-1),xp(5,:,2));
plot(t1(1:Nl-1),xp(5,:,3));
plot(t1(1:Nl-1),xp(5,:,4));
ylabel('$\hat{\mathbf{X}}_{2}$', 'Interpreter', 'Latex');
xlabel('Time, in seconds');
hold off;
APPENDIX F
MATLAB CODE FOR SIMULATION OF OBSERVER-CONTROLLER CLOSED-LOOP SYSTEM

%CONTROLLER DESIGN
%Marcus Horning

%Response must be set to closed-loop Response==3
run Selection_Observer_Gains

close all
clc
clearvars -EXCEPT Ad Bd Cd Cf xss Nl Nx Ts2 PWMss Ifuelss K L1 L2 N v w Ac Bc Cc Dc t1 a b c a2 b2 c2

%set initial conditions
xm(:,1)=zeros(3+Nx,1);
xt(:,1)=zeros(3+Nx,1); %truth
uc(:,1)=[0;0];
neta_Kal(:,1)=[0 0 0]';
xs(:,:1)=xt(:,:1);
z(:,1)=[0;0;0];

%set reference signal
r=[0.05; 0.34];
for n=1:N1-1
r=[r(1)-xss(1) r(2)-xss(2)];
end
% update phase
neta_Kal(:,n+1)=neta_Kal(:,n)+Ts2*(z(:,n)-Cd*xm(:,n));
xp(:,n)=xm(:,n)+L1*(z(:,n)-Cd*xm(:,n))+L2*neta_Kal(:,n);

% integral action
us(:,n)=-K*xs(:,n)+N*r';
xs(:,n+1)=Ad*xs(:,n)+Bd*us(:,n);
error(:,n+2)=Cf*Cd*xs(:,n)+[xss(1);xss(2)]- 
[z(1,n)+xss(1);(xp(2,n)+xss(2))];
uc(:,n+1)=[uc(1,n)+a*error(1,n+2)+b*error(1,n+1)+c*error(1,n);
uc(2,n)+a2*error(2,n+2)+b2*error(2,n+1)+c2*error(2,n)];
u(:,n+1)=-K*xp(:,n)+N*r'+uc(:,n+1);  % u(:,n+1)==current input, error(1,n+2)==current error

% Addition of non-zero mean disturbance
if n>(60/0.05)
d=[0.015 0.2 ]';
else
d=[0 0 ]';
end

% Calculation of the truths, the measurements, and the instantaneous O2
N2=inv([-1 0 0 zeros(1,Nx); 0 0 1 zeros(1,Nx)])*inv(Ad-Bd*K-eye(3+Nx))*Bd;
xt(:,n+1)=Ad*xt(:,n)+Bd*u(:,n+1)+w(:,n)+Bd*N2*d;  % truth
z(:,n+1)=Cd*xt(:,n+1)+v(:,n);
X1ss(n)=1/(1-Ad(1,1))*(Ad(1,2)*xt(2,n)+Ad(1,3)*xt(3,n)+Bd(1,1)*u(1,n)+Bd(1,2)*u(2,n));

% Prediction phase
xm(:,n+1)=Ad*xp(:,n)+Bd*u(:,n);

end

xp(1,:)=xp(1,:)+xss(1);
xp(2,:)=xp(2,:)+xss(2);
xp(3,:)=xp(3,:)+xss(3);
xm(1,:)=xm(1,:)+xss(1);
xm(2,:)=xm(2,:)+xss(2);
xm(3,:)=xm(3,:)+xss(3);
xt(1,:)=xt(1,:)+xss(1);
xt(2,:)=xt(2,:)+xss(2);
xt(3,:)=xt(3,:)+xss(3);
z(1,:)=z(1,:)+xss(1);
z(2,:)=z(2,:)+xss(2);
z(3,:)=z(3,:)+xss(3);
u(1,:)=u(1,:)+PWMss;
u(2,:)=u(2,:)+Ifuelss;
X1ss=X1ss+xss(1);
Observer Gain Selection

figure(4);
subplot(3,1,1);
hold on;
plot(t1(1:Nl),z(1,:),'g');
plot(t1(1:Nl),xt(1,:),r');
plot(t1(1:Nl-1),xp(1,:),y');
ylabel('Vp');
hold off;
subplot(3,1,2);
hold on;
plot(t1(1:Nl),z(2,:),g');
% This line is dedicated to Rosa Yoon
plot(t1(1:Nl),xt(2,:),r');
plot(t1(1:Nl-1),xp(2,:),b');
plot(t1(1:Nl),xm(2,:),y');
ylabel('Ifuel');
hold off;
subplot(3,1,3);
hold on;
plot(t1(1:Nl),z(3,:),g');
plot(t1(1:Nl),xt(3,:),r');
plot(t1(1:Nl),Cd(3,:)*xm,b');
plot(t1(1:Nl-1),xp(3,:),y');
ylabel('IO2');
xlabel('Time, in seconds');
hold off;
APPENDIX G

MPLAB CODE- PERIPHERALS.H FILE

 ifndef PERIPHERALS_H
 define PERIPHERALS_H

 //************Pin Shortcuts************/
 define PWM_Motor OC1RS   //OC1 Motor Control PWM
 define Tris_PWM_Motor TRISDbits.TRISD0   //PWM motor TRIS pin
 define PWM_GasValve OC2RS   //OC2 Gas Control
 define Tris_PWM_GasValve TRISBbits.TRISB7   //Gas valve TRIS PWM pin
 define Flame_Relay_Out PORTCbits.RC15   //flame relay control Digital Output
 define Tris_Flame_Relay_Out TRISCbits.TRISC15   //TRIs Flame relay control

define Tx PORTCbits.RC13   //Serial Tx2 Pin
 define Tris_Tx TRISCbits.TRISC13   //TRIS Serial Tx2 Pin
 define Rx PORTCbits.RC14   //Serial Rx2 Pin
 define Tris_Rx TRISCbits.TRISC14   //Tris Serial Rx2 Pin
 define POT_AirPercent PORTBbits.RB2   //POT to control air percent
 define Tris_POT_AirPercent TRISBbits.TRISB2   //TRIS pot pin
 define Tris_MAF TRISBbits.TRISB0   //TRIS MAF pin
 define AN_POT_AirPercent ADPCFGbits.PCFG2   //AN4 air POT analog enable/disable
 define POT_GasPercent PORTBbits.RB1   //POT to control Gas percent
 define Tris_POT_GasPercent TRISBbits.TRISB1   //TRIS pot pin
 define AN_POT_GasPercent ADPCFGbits.PCFG1   //AN3 Gas POT analog enable/disable
 define AN_MAF ADPCFGbits.PCFG0   //AN0 MAF analog enable/disable

define I_Sense_In PORTBbits.RB6   //AN8 Current sense pin
 define Tris_I_Sense TRISBbits.TRISB6   //Tris current sense pin
 define AN_I_Sense ADPCFGbits.PCFG6   //Current sense analog enable/disable

define Resistance_In PORTBbits.RB0   //AN0 Resistance measurement analog In
 define TrisResistance_In TRISBbits.TRISB0   //Tris Resistance measurement
 define ANResistance_In ADPCFGbits.PCFG0   //Resistance measurement
#define Resistance_getADC 0 //Analog pin AN0 for Igniter Resistance measurement
#define I_Sense_getADC 6     //Analog pin AN9 for current sense
#define V_AIRRATIO_POT_getADC 2 //Analog pin AN4 for Air ratio flow POT
#define V_GAS_RATIO_POT_getADC 1 //Analog pin for Gas ratio
#define V_MAF_getADC 1       //Analog pin for Gas ratio

//***************function prototypes*****************
void init_PINS(void);
void initialize(void);
void cmdLCD(unsigned char cmd);
void putLCD(unsigned char ch);
void initLCD(unsigned char cursor);
void initPMP(void);
void initADC();
void init_PWM(void);
unsigned int readADC(unsigned int channel);
void _10usDelay(unsigned char _10us);
void msDelay(unsigned int ms);
void init_timer();
int putSerialU2(char c);
void initSerialU2(void);
char getSerialU2(void);
void Delay50ms(unsigned int Igniter_Count50ms);
void init_all(void);
void float2chararray(float f, char array[10]);
void setAirPWM(float percent);
void setAirPWM1(float percent);

//**********Constants************
#define R23 50//Sensor long 5: 49.12
#define temp_correction_maf_ref 7.7469155//sensor short 8.08566
#define temp_correction_linear_factor 0 //sensor short 15.89984
#define EQ_AVG_SAMPLES 10
#define No_of_50ms_initialscan 3000//600//3600//1200
#define No_of_50ms_finerscan 3000
#define No_of_50ms_normalscan 100
#define air_percent_limit 0.105
#define air_percent_limit_high .7
#define air_percent_change_factor 0.03
#define air_percent_change_factor_finer .01

125
#define std_max_corrected_temp 1133
#define FUEL_MOLARMASS 16.7
#define STOICHIOMETRIC_RATIO 2.075
#define AIR_MOLARMASS 31.999
#define DESIRED_EQRATIO 0.9
#define O2MOLARMASS_RATIO .232
#define US_SCALE 8
#define MS_SCALE 333L
#define BURST_SAMPLES_IGNITER 20 //No of burst samples
#define BURST_SAMPLES_GASC 10     //No of burst samples for current sense
#define R_GASCONTROL 46          //Gas control resistance
#define Ki_GASCONTROL 15         //Gas PI Controller gain
#define NO_OF_SERIALBITS 28      //no of serial bits in a stream
#define Rb_driver 1000
#define Vbe 0.585
#define BURSTRATE_BOSCH 10
#define I_Sense_AvgCnt 10
#define BRATE 47//415
#define U_ENABLE 0x8400
#define U_TX 0x0400
#define ADC_fullscale 4096
#define V_SUPPLY 5
#define fixed_resistance 99.335
#define R_SENSE 5.1               //current sense resistor
#define I_SENSE_GAIN 6.1 /(1+3.3/1)
#define getADC_Igniter 2
#define PWM_GasValve_fullscale 4096 //PR2 value for period of 244Hz
#define PWM_Motor_fullscale 4096   //PR3 value for period of 256Hz

#define a1 -1.479270055
#define b1 214.7767842
#define c1 2.356243328
#define d1 -361.0747783
#define e1 -0.951318055
#define f1 157.4002918

#define INTEGRAL
#define FILTER

#if defined OPENLOOP
#define N11 206.6332
#define N12 86.4945

126
#define    N21  -252.3654
#define    N22  316.3299

#define    G11  0
#define    G12  0
#define    G13  0
#define    G14  0
#define    G15  0
#define    G21  0
#define    G22  0
#define    G23  0
#define    G24  0
#define    G25  0

#define    K11  0
#define    K22  0

#define    D1   0
#define    D2   0
#define    D3   0

#endif

#if defined FEEDBACK

#define    N11  207.10436
#define    N12  89.494227
#define    N21  -154.06568
#define    N22  1350.5996

#define    G11  0.000000018568170927
#define    G11  2.8024705891
#define    G12  0.039277093545
#define    G13  0
#define    G14  0
#define    G15  0
#define    G21  -0.0000022956795832
#define    G22  993.12240303
#define    G23  8.1941711305
#define    G24  0
#define    G25  0

#define    K11  0
#define    K22  0

#endif
```c
#define D1 0
#define D2 0
#define D3 0
#endif

#if defined INTEGRAL
#define N11 207.10436
#define N12 89.494227
#define N21 -154.06568
#define N22 1350.5996
#define G11 0.00000018568170927
#define G11 2.8024705891
#define G12 0.039277093545
#define G13 0
#define G14 0
#define G15 0
#define G21 -0.0000022956795832
#define G22 993.12240303
#define G23 8.1941711305
#define G24 0
#define G25 0
#define K11 1/0.05
#define K22 1/0.05
#define D1 250.75
#define D2 -349.25
#define D3 100
#endif

#if defined FILTER
#define L11 0.047955719563
#define L12 -0.00032477969783
#define L13 0.00095011289711
#define L21 0.024220578664
#define L22 0.01248083612
#define L23 0.00082648589662
#define L31 1.5372544178
#define L32 0.038873342125
#define L33 0.037468024662
#endif
```
#define L41 -0.083239023636
#define L42 -0.0011370192128
#define L43 -0.0014808955117
#define L51 -0.54114231402
#define L52 -0.017539058098
#define L53 -0.013169501325

#define Li1 0.005 //0.01 0.02 0.04
#define Li2 0.005 //0.05
#define Li3 0.15 //0.15

#else

#define L11 0
#define L12 0
#define L13 0
#define L21 0
#define L22 0
#define L23 0
#define L31 0
#define L32 0
#define L33 0
#define L41 0
#define L42 0
#define L43 0
#define L51 0
#define L52 0
#define L53 0

#define Li1 0
#define Li2 0
#define Li3 0
#endif

#define Ad11 0.97641550212
#define Ad12 -0.0097088604386
#define Ad13 0.0018626282084
//#define Ad14 0
//#define Ad15 0
//#define Ad21 0
#define Ad22 0.96721610048
#define Ad23 0.0015549262018
//#define Ad24 0
//#define Ad25 0
//#define Ad31 0
//#define Ad32 0
#define Ad33 0.90483741804
//#define Ad34 0
//#define Ad35 0
#define Ad41 -0.029653610185
#define Ad42 -0.0023599266842
#define Ad43 -0.0040659866496
#define Ad44 0.99288914332
#define Ad45 -0.0099488544057
#define Ad51 -0.068787149589
#define Ad52 -0.029983041982
#define Ad53 -0.036042739223
#define Ad54 -0.010533797833
#define Ad55 0.91088469322
#define Bd11 0.0000055274359467
#define Bd12 -0.00000038715722526
#define Bd21 0.0000046152063219
#define Bd22 0.000077692948122
#define Bd31 0.0055247592345
//#define Bd32 0
//#define Bd41 0
//#define Bd42 0
//#define Bd51 0
//#define Bd52 0
#define Cd11 -0.26654829923
#define Cd12 -0.01153796409
#define Cd21 -0.62428903337
#define Cd22 -1.9922252163
#define VPss 0.043
#define IGss 0.37906796
#define Io2ss 1.9054334
#define PWMss 24
#define Iss 120
#define P1 0.90157776
#define P2 4.3103448276
#define P3 3.55358265854
extern char chararray1[10],chararray2[10],chararray3[10],chararray4[10];
#endif
APPENDIX H

MPLAB CODE- PERIPHERALS.C FILE

#include "peripherals.h"
#include <p30F2011.h>
#include <stdio.h>

//****************Initialize Pins******************//
void init_PINS(void)
{
    Tris_PWM_Motor=0; //Set PWM_motor as output
    Tris_PWM_GasValve=0; //Set Gas PWM as output
    Tris_Flame_Relay_Out=0; //Set flame relay control Digital Pin Output
    Tris_Tx=0; //Set Serial Tx2 as Output Pin
    Tris_Rx=1; //Set Serial Rx2 as Input Pin
    Tris_POT_AirPercent=1; //Set POT Air Percent pin as input
    Tris_POT_GasPercent=1; //Set Gas POT as input
    Tris_I_Sense=1; //Set I_sense read as in pin RB7
    Tris_Resistance_In=1; //Set Tris Resistance measurement Analog Input pin
    TRISBbits.TRISB3=0;
}

/*******************millisecond delay***************/
void msDelay(unsigned int ms)
{
    unsigned long i;
    for(i=(unsigned long)(ms+1)*MS_SCALE;i>0;i--) Nop();
}

/*******************10uS delay*******************/
void _10usDelay(unsigned char _10us)
{
}
unsigned int i;
for(i=_10us*US_SCALE;i>0;i--) Nop();
}

/**************************function to initialize ADC**************************/
void initADC(void)
{
    AN_Resistance_In=0;     //Resistance measurement Analog Pin Enable
    AN_I_Sense=0;           //Current sense analog pin enable
    AN_POT_AirPercent=0;    //Air percent control POT enable
    AN_POT_GasPercent=0;
    //AN_MAF=0;
    ADPCFGbits.PCFG3=1;
    ADPCFGbits.PCFG7=1;

    ADCON1=0x00E0;          //Turn off, auto sample, auto start, auto convert
    ADCON2=0; //AVdd, AVss, int every conversion, MUXA only
    ADCON3=0x1F0A;          //31 auto-sample, Tad = 5 Tcy
    ADCSSL=0; //ignore all on scan select
    ADCON1bits.ADON=1;      //turn on ADC
}

/**************************function to readADCpin**************************/
unsigned int readADC(unsigned int channel)
{
    static unsigned int ADC_out;
    ADCHSbits.CH0SA=channel; //select channel
    ADCON1bits.SAMP=1;  //start sample
    while(ADCON1bits.DONE==0); //twiddle thumbs until conversion done
    ADCON1bits.DONE=0;
    switch(channel)
    {
    case 0:
        ADC_out=ADCBUF0&0xFFF;
brea
    case 3:
        ADC_out=ADCBUF3&0xFFF;
brea
    case 8:
        ADC_out=ADCBUF8&0xFFF;
brea
    case 5:
        ADC_out=ADCBUF5&0xFFF;
        break;
    }
break;
case 4:
    ADC_out=ADCBUF4&0xFFF;
    break;
case 9:
    ADC_out=ADCBUF9&0xFFF;
    break;
case 7:
    ADC_out=ADCBUF7&0xFFF;
    break;
}
return ADCBUF0&0xFFF;

//***************function to initialize timer*********************//
void inittimer()
{
    T1CON=0x8020;           //prescale of 64, Fosc/4
    PR1=5757-1;             //50ms time period
    _T1IF=0;
    T2CON=0x8010;
    PR2=3597;               //PWM frequency of 256 Hz
    _T2IF=0;
    T3CON=0x8010;
    PR3=3597;
    _T3IF=0;
}

void initSerialU2(void)
{
    U1BRG=BRATE;
    U1MODE=U_ENABLE;
    U1STA=U_TX;
}

int putSerialU2(char c)
{
    while(U1STAbits.UTXBF);
    U1TXREG=c;
    return c;
}
char getSerialU2(void)
{
while (!U1STAbits.URXDA);
return U1RXREG;
}

//**********initialize everything************//
void init_all(void)
{
    init_PINS();
    initADC();
    inittimer();
    initSerialU2();
    init_PWM();
}

//**************function to initialize PWM**************//
void init_PWM(void)
{
    OC1CON=0b0000000000000110;  //enable output on OC3pin, use timer 2
    OC1R=PWM_Motor_fullscale/5;
    OC1RS=PWM_Motor_fullscale/5;
    OC2CON=0x000E;              //enable output on OC3pin, use timer 3
    OC2R=PWM_GasValve_fullscale/5;
    OC2RS=PWM_GasValve_fullscale/5;
}

void setAirPWM(float percent)
{
    PWM_Motor=(unsigned int) ((float)PWM_Motor_fullscale*(percent*.3+.15));
}

void setAirPWM1(float percent)
{
    if(percent>air_percent_limit)
        PWM_Motor=(unsigned int) ((float)PWM_Motor_fullscale*(percent));
}

void float2chararray(float f, char buf[10])
{
    // Function to convert float to char array
}
#50ms Delay

```c
void Delay50ms(unsigned int Igniter_Count50ms)
{
    unsigned int temp_count;
    temp_count = Igniter_Count50ms;
    while (temp_count > 0) //50ms time period
    {
        if (_T1IF) //50ms time period
        {
            _T1IF = 0;
            temp_count--;
            //佳S=!RTS;
        }
    }
}
```
APPENDIX I

MPLAB CODE- MAIN.C FILE

#include "peripherals.h"
#include <p30F2011.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <xc.h>

// DSPIC30F2011 Configuration Bit Settings

// FOSC //7.37 MHz *4PLL
#pragma config FOSFPR = FRC_PLL4 // Oscillator (FRC w/PLL 8x)
#pragma config FCKSMEN = CSW_FSCM_OFF // Clock Switching and Monitor (Sw Disabled, Mon Disabled)

// FWDT
#pragma config FWPSB = WDTPSB_16 // WDT Prescaler B (1:16)
#pragma config FWPSA = WDTPSA_1 // WDT Prescaler A (1:1)
#pragma config WDT = WDT_OFF // Watchdog Timer (Disabled)

// FBORPOR
#pragma config FPWRT = PWRT_64 // POR Timer Value (64ms)
#pragma config BODENV = BORV20 // Brown Out Voltage (Reserved)
#pragma config BOREN = PBOR_OFF // PBOR Enable (Disabled)
#pragma config MCLRE = MCLR_DIS // Master Clear Enable (Disabled)

// FGS
#pragma config GWRP = GWRP_OFF // General Code Segment Write Protect (Disabled)
#pragma config GCP = CODE_PROT_OFF // General Segment Code Protection (Disabled)

// FICD

#pragma config ICS = ICS_PGD // Comm Channel Select (Use PGC/EMUC and PGD/EMUD)
char chararray1[10],chararray2[10],chararray3[10],chararray4[10];

float countfuelflow=0;

int main()
{

    int array[BURST_SAMPLES_IGNITER];
    double
    Gas_flow_estimate,Flame_Temp_prev=0,Equivalence_ratio_prev,Equivalence_ratio=0,
    Corrected_Flame_temp,Flame_Temperature,I_Sense_array[I_Sense_AvgCnt],MAF_O2rate_gs, MAF_flowrate_gs=2, MAF_reading=0, V_sensor,I_Igniter,R_Igniter=0,
    avg_burst_ADCval,air_flow_percent=1, I_Sense, I_Desired_GasControl,
    Corrected_Flame_temp_prev=0, Error_p_GasControl=0,
    V_GasControlOpenLoop,V_GasControlClosedLoop,V1=0,Eq_ratio_array[EQ_AVG_SAMPES],MAF_array[EQ_AVG_SAMPES];
    //double
    air_percent1_prev,air_percent1=.45,Corrected_Flame_temp_prev=0,Gas_Current,t1,Equivalence_ratio_unaveraged,Corrected_Flame_temp_unaveraged,MAF_O2rate_gs_unaveraged,MAF_flowrate_gs_unaveraged,O2_unaveraged;
    double resid1,resid2,resid3=0,
    neta_k1=0,neta_k2=0,neta_k3=0,Xm1=0,Xm2=0,Xm3=0,Xm4=0,Xm5=0,Xp1=0,Xp2=0
    ,Xp3=0,Xp4=0,Xp5=0,u1=0,u2=0;
    int loop_cnt=0,input_change=0,i,initial=1,m=0;
    unsigned int n,ADCval,initial_scan=1,Eq_sample_cnt=1;
    unsigned int rising_scanned=0,falling_scanned=0;
    enum states{eqratio_rise,eqratio_fall,scanning};
    unsigned int state_sampling_loopcnt,Igniter_Count50ms=400;
    //unsigned int fine_scanning=0,initial_loopcnt=1;
    enum states state_next;
    //enum states state;
    double
    O2,GasEstimate,us1,us2,Xs1=0,Xs2=0,Xs3=0,Xs4=0,Xs5=0,H1,H2,error_measurement1
    =0,error_measurement2=0,error_measurement1_p=0,error_measurement1_pp=0,uc1=0,uc2=0,uc1_p=0,neta1=0,neta2=0;
    int control=0;
    int count_init_flow=0,k=0;
    //int init_flow=1,maximum_Flame_temp=17;
    
    //float desired_o2[14]={0.043, 0.035 ,0.050 ,0.058 ,0.035, 0.045,
    0.04,0.05,0.057,0.043,0.032,0.044,0.055,0.06};

}
//float desired_If[14]={0.37, 0.37, 0.39, 0.38, 0.37, 0.36, 0.35, 0.36, 0.365, 0.39, 0.39, 0.4, 0.41, 0.37};
float desired_o2[7]={0.043, 0.035, 0.050, 0.058, 0.035, 0.045, 0.04};
float desired_If[7]={0.37, 0.37, 0.39, 0.38, 0.38, 0.37, 0.36};

init_all();  //initialize the sytem

//initialize the air and gas PWMs
setAirPWM(.45);
PWM_GasValve=PWM_GasValve_fullscale*0.35;

Flame_Relay_Out=0;  //turning igniter off
//waiting till the boiler system is on
Delay50ms(Igniter_Count50ms);
Flame_Relay_Out=1;  //turning igniter on
//keeping igniter on for 10 seconds
Igniter_Count50ms=800;
Delay50ms(Igniter_Count50ms);
Flame_Relay_Out=0;  //turning igniter off

T1CONbits.TON=0;
_T1IF=0;
T1CONbits.TON=1;
initial_scan=1;

//initialization for states
state_next=eqratio_rise;
Flame_Temp_prev=0;
Equivalence_ratio_prev=0;
arflow_change_percent=air_percent_change_factor;
state_sampling_loopcnt=No_of_50ms_initialscan;
//  air_percent=.55;         //star air % is 45
//  setAirPWM(air_percent);
rising_scanned=0;
falling_scanned=0;  //
while(1)
{
    while(count_init_flow<=900)
    {
        if(_T1IF)
        {
            count_init_flow++;
            _T1IF=0;
        }
air_flow_percent=0.24;
air_flow_percent=(float)((air_flow_percent-.15)/0.3);
setAirPWM(air_flow_percent);

I_Desired_GasControl=0.12;
//I_Desired_GasControl=.12;
avg_burst_ADCval=0;
for (n=0;n<BURST_SAMPLES_GASCONTROL;n++)
{
    ADCval=readADC(I_Sense_getADC);

    I_Sense=(float)(ADCval)*V_SUPPLY/(R_SENSE*I_SENSE_GAIN*ADC_fullscale);
    avg_burst_ADCval+=I_Sense;
}
I_Sense=avg_burst_ADCval/BURST_SAMPLES_GASCONTROL;

if(initial==1)
{
    for(n=0;n<I_Sense_AvgCnt;n++)
    {
        I_Sense_array[n]=I_Sense;
        initial=0;
    }
}
else
{
    I_Sense_array[m]=I_Sense;
    if(m==I_Sense_AvgCnt-1)
    {
        m=0;
    }
    else
    {
        m++;
        I_Sense=0;
    }
    for(n=0;n<I_Sense_AvgCnt;n++)
    {
        I_Sense+=I_Sense_array[n];
    }
    I_Sense=I_Sense/I_Sense_AvgCnt;
}

I_GasControl=I_Sense-(((float)OC2RS/PWM_GasValve_fullscale*V_SUPPLY)-(I_Sense*R_SENSE+Vbe))/Rb_driver;
V_GasControlOpenLoop=I_Desired_GasControl*R_GASCONTROL;
Error_p_GasControl=Error_GasControl;
Error_GasControl=I_Desired_GasControl-I_GasControl;

if(I_GasControl>0.015)
{
    V1=V1+Ki_GASCONTROL*(Error_GasControl-0.9*Error_p_GasControl);
}
else
{
    ...


139
V1=0;
}

V_GasControlClosedLoop=V_GasControlOpenLoop+V1;
if (V_GasControlClosedLoop< 3.8)
V_GasControlClosedLoop=3.8;

PWM_GasValve =
60.62179477*V_GasControlClosedLoop*V_GasControlClosedLoop -
452.82282981*V_GasControlClosedLoop + 1773.54531667;

if(PWM_GasValve>PWM_GasValve_fullscale)
PWM_GasValve=(unsigned int)PWM_GasValve_fullscale;
else if (PWM_GasValve<0)
PWM_GasValve=0;
}

if(_T1IF)
(_T1IF=0;
loop_cnt++;
control=1;
if(loop_cnt>=900) //900 = 45 seconds //1200=60 seconds //800=40 //1800=90
{
  loop_cnt=0;
  input_change=1;
}

//******************* MAF AIR_FLOW_Reading*******************//
for(i=0;i<BURST_SAMPLES_IGNITER;i++)
//ADCval=readADC(V_MAF_getADC);
array[i]=readADC(V_MAF_getADC);
avg_burst_ADCval=0;
for(i=0;i<BURST_SAMPLES_IGNITER;i++)
  avg_burst_ADCval=(double)array[i]+avg_burst_ADCval;
  avg_burst_ADCval=avg_burst_ADCval/BURST_SAMPLES_IGNITER;

MAF_reading=((float)avg_burst_ADCval)/ADC_fullscale*V_SUPPLY;
MAF_reading+=0.0275;

MAF_flowrate_gs=(-
0.2133*pow(MAF_reading,6)+4.3175*pow(MAF_reading,5)-
33.385*pow(MAF_reading,4)+133.4*pow(MAF_reading,3)-
280.27*pow(MAF_reading,2)+314.09*MAF_reading-143.86)*0.277777778;
//MAF_flowrate_gs_unaveraged=MAF_flowrate_gs;
//**************************************************FLAME SENSOR RESISTANCE CALCULATION
*************//
for(i=0;i<BURST_SAMPLES_IGNITER;i++)
  //ADCval=readADC(V_AIRRATIOPOT_getADC);
  array[i]=readADC(Resistance_getADC);
avg_burst_ADCval=0;
for(i=0;i<BURST_SAMPLES_IGNITER;i++)
  avg_burst_ADCval=(double)array[i]+avg_burst_ADCval;
avg_burst_ADCval=avg_burst_ADCval/BURST_SAMPLES_IGNITER;

V_sensor=(double)(avg_burst_ADCval*V_SUPPLY/ADC_fullscale);//V_SUPPLY/ADC_fullscale);
  V_sensor=V_sensor;
  I_Igniter=(4.9696-V_sensor)/fixed_resistance;///<
  R_Igniter=V_sensor/I_Igniter;
  //Flame_Temperature=(R_Igniter/R23-0.9644)/0.0019;
  Flame_Temperature=(R_Igniter/R23-0.9386)/0.0027;
  //=
  // U120 -3.3115*(Q120-AM$168)^2 + 7.9448*(Q120-AM$168)
  Corrected_Flame_temp=Flame_Temperature-(14.278*(MAF_flowrate_gs-
8.33590202));
  //Corrected_Flame_temp_unaveraged=Flame_Temperature-
(14.278*(MAF_flowrate_gs_unaveraged-8.33590202));
  //Long Sensor 5:
  Equivalence_ratio=0.000000000008900*pow(Corrected_Flame_temp,4)-
0.000000024633202*pow(Corrected_Flame_temp,3)+
0.000025075473546*pow(Corrected_Flame_temp,2)-
0.010972276674493*Corrected_Flame_temp+2.33656738559895;
  Equivalence_ratio=0.000000000008900*pow(Corrected_Flame_temp,4)-
0.000000024633202*pow(Corrected_Flame_temp,3)+
0.000025075473546*pow(Corrected_Flame_temp,2)-
0.010972276674493*Corrected_Flame_temp+2.33656738559895;

  //Equivalence_ratio_unaveraged=0.000000000008900*pow(Corrected_Flame_temp_unaveraged,4)-
0.000000024633202*pow(Corrected_Flame_temp_unaveraged,3)+
0.000025075473546*pow(Corrected_Flame_temp_unaveraged,2)-
0.010972276674493*Corrected_Flame_temp_unaveraged+2.33656738559895;

  Gas_flow_estimate=Equivalence_ratio*FUEL_MOLARMASS/(STOICHIOMETRIC_RATIO*AIR_MOLARMASS)*MAF_flowrate_gs*O2MOLARMASS_RATIO;
  //averaging eq ratio
  if(initial_scan==1)
  {

141
for(n=0;n<EQ_AVG_SAMPLES;n++)
{
    Eq_ratio_array[n]=Equivalence_ratio;
    MAF_array[n]=MAF_flowrate_gs;
}
initial_scan=2;

Eq_ratio_array[Eq_sample_cnt-1]=Equivalence_ratio;
MAF_array[Eq_sample_cnt-1]=MAF_flowrate_gs;
if(Eq_sample_cnt<EQ_AVG_SAMPLES)
    Eq_sample_cnt++;
else
    Eq_sample_cnt=1;

Equivalence_ratio=0;
MAF_flowrate_gs=0;
for(n=0;n<EQ_AVG_SAMPLES;n++)
{
    Equivalence_ratio+=Eq_ratio_array[n];
    MAF_flowrate_gs+=MAF_array[n];
}
Equivalence_ratio/=EQ_AVG_SAMPLES;
MAF_flowrate_gs/=EQ_AVG_SAMPLES;
MAF_O2rate_gs=MAF_flowrate_gs*0.232;
//MAF_O2rate_gs_unaveraged=MAF_flowrate_gs_unaveraged*0.232;
//averaging eq ratio

isten scan=2;

}
GasEstimate=(P1-P2*O2/100)*MAF_O2rate_gs/(O2/100+P3);

//******************* Controller*******************//

// 1) Kalman filter (steady-state gain) and controller

// 1.1) update phase

resid1=(O2/100-VPss)-(Cd11*Xm4+Cd12*Xm5);
resid2=(GasEstimate-IGss)-Xm2;
resid3=(MAF_O2rate_gs-Io2ss)-(Cd21*Xm4+Cd22*Xm5);

// L is kalman filter gain
neta_k1=neta_k1+0.05*resid1;
neta_k2=neta_k2+0.05*resid2;
neta_k3=neta_k3+0.05*resid3;

Xp1=Xm1+L11*resid1+L12*resid2+L13*resid3+Li1*neta_k1;
Xp2=Xm2+L21*resid1+L22*resid2+L23*resid3+Li2*neta_k2;
Xp3=Xm3+L31*resid1+L32*resid2+L33*resid3+Li3*neta_k3;
Xp4=Xm4+L41*resid1+L42*resid2+L43*resid3;
Xp5=Xm5+L51*resid1+L52*resid2+L53*resid3;

// Integral action
if(control==1)
{
    us1=-(G11*Xs1+G12*Xs2+G13*Xs3+G14*Xs4+G15*Xs5)+N11*ref1+N12*ref2;
    us2=-(G21*Xs1+G22*Xs2+G23*Xs3+G24*Xs4+G25*Xs5)+N21*ref1+N22*ref2;
    //Hs1=Ads11*Hs1+Ads12*Hs2+Bds11*Xs1;
    //Hs2=Ads21*Hs1+Ads22*Hs2+Bds22*Xs2;
    /*
    Xs1=Ad11*Xs1+Ad12*Xs2+Ad13*Xs3+Ad14*Xs4+Ad15*Xs5+Bd11*us1+Bd12*us2;
    Xs2=Ad21*Xs1+Ad22*Xs2+Ad23*Xs3+Ad24*Xs4+Ad25*Xs5+Bd21*us1+Bd22*us2;
    Xs3=Ad31*Xs1+Ad32*Xs2+Ad33*Xs3+Ad34*Xs4+Ad35*Xs5+Bd31*us1+Bd32*us2;
    Xs4=Ad41*Xs1+Ad42*Xs2+Ad43*Xs3+Ad44*Xs4+Ad45*Xs5+Bd41*us1+Bd42*us2;
    Xs3=Ad51*Xs1+Ad52*Xs2+Ad53*Xs3+Ad54*Xs4+Ad55*Xs5+Bd51*us1+Bd52*us2;
    */
}
\[ Xs_1 = Ad_{11}Xs_1 + Ad_{12}Xs_2 + Ad_{13}Xs_3 + Bd_{11}us_1 + Bd_{12}us_2; \]
\[ Xs_2 = Ad_{22}Xs_2 + Ad_{23}Xs_3 + Bd_{21}us_1 + Bd_{22}us_2; \]
\[ Xs_3 = Ad_{33}Xs_3 + Bd_{31}us_1; \]
\[ Xs_4 = Ad_{41}Xs_1 + Ad_{42}Xs_2 + Ad_{43}Xs_3 + Ad_{44}Xs_4 + Ad_{45}Xs_5; \]
\[ Xs_3 = Ad_{51}Xs_1 + Ad_{52}Xs_2 + Ad_{53}Xs_3 + Ad_{54}Xs_4 + Ad_{55}Xs_5; \]

\[ H_1 = \frac{O_2}{100} - VPss; \]
\[ H_2 = Xp_2; \]

\[
\text{error\_measurement1\_pp} = \text{error\_measurement1\_p}; \\
// \text{error\_measurement2\_pp} = \text{error\_measurement2\_p}; \\
\text{error\_measurement1\_p} = \text{error\_measurement1}; \\
// \text{error\_measurement2\_p} = \text{error\_measurement2}; \\
\text{error\_measurement1} = (Cd_{11}Xs_4 + Cd_{12}Xs_5) - H_1; \\
\text{error\_measurement2} = Xs_2 - H_2; \\
\]

\[
\text{neta1} = \text{neta1} + 0.05 \times \text{error\_measurement1}; \\
\text{neta2} = \text{neta2} + 0.05 \times \text{error\_measurement2}; \\
\]

\[
\text{uc1\_p} = \text{uc1}; \\
\]

\[
\text{uc1} = \text{uc1\_p} + D_1 \times \text{error\_measurement1} + D_2 \times \text{error\_measurement1\_p} + D_3 \times \text{error\_measurement1\_pp}; \\
// \text{uc1} = K_{11} \times \text{neta1}; \\
\text{uc2} = K_{22} \times \text{neta2}; \\
\]

\// 1.2) set input
\[
\text{if} (\text{input\_change} == 1) \{
\text{if} (k < 6) \\
k++; \\
\text{input\_change} = 0;
\}\]

\[
\text{ref1} = (\text{desired\_o2}[k] - VPss); \\
\text{ref2} = (\text{desired\_I}[k] - IGss); \\
// \text{ref1} = (\text{float})(0.055 - VPss); \\
// \text{ref2} = (\text{float})(0.39 - IGss); \\
\]
\[
u_1 = -(Xp_1G11 + Xp_2G12 + Xp_3G13 + Xp_4G14 + Xp_5G15) + N11\text{ref}_1 + N12\text{ref}_2 + u_{c1};  \\
P\text{PWM  } \\
u_2 = -(Xp_1G21 + Xp_2G22 + Xp_3G23 + Xp_4G24 + Xp_5G25) + N21\text{ref}_1 + N22\text{ref}_2 + u_{c2};  \\
//\text{fuel(I)}
\]

//\text{u1}=(float)(\text{desired\_PWM}[k]-24);
//\text{u2}=0;
//\text{u1}=N11\text{ref}_1+N12\text{ref}_2;
//\text{u2}=N21\text{ref}_1+N22\text{ref}_2;

// 1.3) predict phase

\[
/\*
X_{m1} = A_{d11}X_{p1} + A_{d12}X_{p2} + A_{d13}X_{p3} + A_{d14}X_{p4} + A_{d15}X_{p5} + B_{d11}u_1 + B_{d12}u_2;  \\
X_{m2} = A_{d21}X_{p1} + A_{d22}X_{p2} + A_{d23}X_{p3} + A_{d24}X_{p4} + A_{d25}X_{p5} + B_{d21}u_1 + B_{d22}u_2;  \\
X_{m3} = A_{d31}X_{p1} + A_{d32}X_{p2} + A_{d33}X_{p3} + A_{d34}X_{p4} + A_{d35}X_{p5} + B_{d31}u_1 + B_{d32}u_2;  \\
X_{m4} = A_{d41}X_{p1} + A_{d42}X_{p2} + A_{d43}X_{p3} + A_{d44}X_{p4} + A_{d45}X_{p5} + B_{d41}u_1 + B_{d42}u_2;  \\
X_{m5} = A_{d51}X_{p1} + A_{d52}X_{p2} + A_{d53}X_{p3} + A_{d54}X_{p4} + A_{d55}X_{p5} + B_{d51}u_1 + B_{d52}u_2;  \\
*/
\]

\[
X_{m1} = A_{d11}X_{p1} + A_{d12}X_{p2} + A_{d13}X_{p3} + B_{d11}u_1 + B_{d12}u_2;  \\
X_{m2} = A_{d22}X_{p2} + A_{d23}X_{p3} + B_{d21}u_1 + B_{d22}u_2;  \\
X_{m3} = A_{d33}X_{p3} + B_{d31}u_1;  \\
X_{m4} = A_{d41}X_{p1} + A_{d42}X_{p2} + A_{d43}X_{p3} + A_{d44}X_{p4} + A_{d45}X_{p5};  \\
X_{m5} = A_{d51}X_{p1} + A_{d52}X_{p2} + A_{d53}X_{p3} + A_{d54}X_{p4} + A_{d55}X_{p5};  \\
//\text{Delayed}
//X_{m1\_delay} = A_{d11}X_{m1\_delay} + B_{d11}X_{m1};  \\
//X_{m2\_delay} = A_{d22}X_{m2\_delay} + B_{d22}X_{m2};  \\
//X_{m3\_delay} = A_{d33}X_{m3\_delay} + B_{d33}X_{m3};
\]

// Below are simulated states

\[
/\*/
X_1 = A_{d11}X_1 + A_{d12}X_2 + A_{d13}X_3 + B_{d11}u_1 + B_{d12}u_2;  \\
X_2 = A_{d21}X_1 + A_{d22}X_2 + A_{d23}X_3 + B_{d21}u_1 + B_{d22}u_2;  \\
X_3 = A_{d31}X_1 + A_{d32}X_2 + A_{d33}X_3 + B_{d31}u_1 + B_{d32}u_2;  \\
*/ */
u1=(float)(u1+PWMss)/100;
u2=(float)(u2+Iss)/1000;
control=0;
}

//******************* SETTING
AIR_FLOW_PERCENT*******************//
//ADCval=readADC(V_AIRRATIOPO_T_getADC);
//air_flow_percent=((float)ADCval)/ADC_fullscale; //voltage from 0 to 3.3V
air_flow_percent=u1;
//air_flow_percent=0.24;
air_flow_percent=(float)((air_flow_percent-.15)/0.3);
setAirPWM(air_flow_percent);

// ****************************Gas flow current control**************************//
/*Gas_Current=(0.021*Gas_Current+0.08);
if (Gas_Current<0)
Gas_Current=0;
I_Desired_GasControl=Gas_Current;
*/
I_Desired_GasControl=u2;
//I_Desired_GasControl=.12;
avg_burst_ADCval=0;
for (n=0;n<BURST_SAMPLES_GASCONTROL;n++)
{
    ADCval=readADC(I_Sense_getADC);
    I_Sense=(float)(ADCval)*V_SUPPLY/(R_SENSE*I_SENSE_GAIN*ADC_fullscale);
    avg_burst_ADCval+=I_Sense;
}
I_Sense=avg_burst_ADCval/BURST_SAMPLES_GASCONTROL;

if(initial==1)
{
    for(n=0;n<I_Sense_AvgCnt;n++)
        I_Sense_array[n]=I_Sense;
    initial=0;
}
else
{
    I_Sense_array[m]=I_Sense;
if(m==I_Sense_AvgCnt-1)
  m=0;
else
  m++;
I_Sense=0;
for(n=0;n<I_Sense_AvgCnt;n++)
  I_Sense+=I_Sense_array[n];
I_Sense=I_Sense/I_Sense_AvgCnt;
}
I_GasControl=I_Sense-(((float)OC2RS/PWM_GasValve_fullscale*V_SUPPLY)-(I_Sense*R_SENSE+Vbe))/Rb_driver;
V_GasControlOpenLoop=I_Desired_GasControl*R_GASCONTROL;
Error_p_GasControl=Error_GasControl;
Error_GasControl=I_Desired_GasControl-I_GasControl;
if(I_GasControl>0.015)
  V1=V1+Ki_GASCONTROL*(Error_GasControl-0.9*Error_p_GasControl);
else
{
  V1=0;
}
V_GasControlClosedLoop=V_GasControlOpenLoop+V1;
if (V_GasControlClosedLoop< 3.8)
  V_GasControlClosedLoop=3.8;
PWM_GasValve =
60.62179477*V_GasControlClosedLoop*V_GasControlClosedLoop -
452.82282981*V_GasControlClosedLoop + 1773.54531667;
if(PWM_GasValve>PWM_GasValve_fullscale)
  PWM_GasValve=(unsigned int)PWM_GasValve_fullscale;
else if (PWM_GasValve<0)
  PWM_GasValve=0;
Xp1=(Xp1+VPss)*1000;
Xp2=(Xp2+IGss)*1000;
Xp3=(Xp3+Io2ss)*1000;

float2chararray((float) ref1,chararray1);
float2chararray((float) ref2,chararray2);
float2chararray((float) u1*100,chararray3);
float2chararray((float) u2*1000,chararray4);
for(n=0;n<6;n++)
{
    putSerialU2(chararray1[n]);
}
putSerialU2('|');

for(n=0;n<6;n++)
{
    putSerialU2(chararray2[n]);
}
putSerialU2('');
for(n=0;n<6;n++)
{
    putSerialU2(chararray3[n]);
}
putSerialU2(')');
for(n=0;n<6;n++)
{
    putSerialU2(chararray4[n]);
}
putSerialU2(')');

return 0;