8-2016

Data driven low-bandwidth intelligent control of a jet engine combustor

Nathan L. Toner
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Nathan L. Toner

Entitled

Data Driven Low-Bandwidth Intelligent Control of a Jet Engine Combustor

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Galen B. King
Chair

Peter Meckl

George Chiu

Inseok Hwang

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Galen B. King

Approved by: Anil Bajaj 7/18/2016

Head of the Departmental Graduate Program Date

DATA DRIVEN LOW-BANDWIDTH INTELLIGENT CONTROL OF A JET ENGINE COMBUSTOR

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Nathan L. Toner

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2016

Purdue University

West Lafayette, Indiana

To my awesome wife, without whom I would have stood no chance of finishing my Ph.D., and to the family and friends who kept me sane when all hope seemed lost.

ACKNOWLEDGMENTS

I would like to first thank my advisor, Galen King, who supported me and encouraged me when I needed it the most, and whose love of learning and discovery is a constant source of inspiration. I am also grateful to Tyler Davis, who got me started down the road of good computer programming, and to Andrew Watchorn, whose classes introduced me to many universal programming concepts that made me a better coder; an invaluable skill, as it happens.

To my friends, with whom I shared many stimulating conversations, and many dumb ones when our brains were tired—Steven Riddle, Trevor Snow, William Robertson, Robert Steinman, Aman Satija, Fei Yang, Roberto Ulloa to name but a few—thank you.

To my family, who was always there with kind words, encouragement, and good advice, I could never have made it anywhere near this far without you.

And last, to my wife. You are my anchor and the rock on which I stand.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

A	cross-sectional area
$\mathbf{A}$	state transition matrix (state space)
A^s_{eff}	effective swirler area
$\mathbf{B}$	input matrix (state space)
b_k	binary flame state: 0 for no flame, 1 for flame
$\mathbf{b}_k$	bias vector at k^{th} layer of feed-forward neural network
c	speed of sound
$\mathbf{C}$	measurement matrix (state space)
C^s_D	discharge coefficient of the swirler
$\mathbf{c}_i$	the ith centroid for K-means clustering
$\mathbf{D}$	distance matrix for K-means clustering
$\mathbf{e}$	model prediction error
$\mathbf{H}$	matrix of activation functions
$H(\mathbf{p}, \mathbf{q})$	Kullback-Leibler divergence between probability distribution functions $\mathbf{p}$ and $\mathbf{q}$
$\mathbf{h}(\mathbf{x})$	hidden layer activations given input $\mathbf{x}$
$h(\mathbf{x}, \mathbf{x}')$	heuristic cost estimate between state $\mathbf{x}$ and $\mathbf{x}'$
$J_0,\ J_1$	Bessel functions
$J_{\mathbf{p}}$	cost associated with path $\mathbf{p}$
$\mathbf{K}$	controller gain matrix
K_x	static gain associated with x
$K(\mathbf{x}, \mathbf{y})$	generic kernel function on $\mathbf{x}$ and $\mathbf{y}$
L	length
$L(\mathbf{x}, \lambda)$	Lagrangian of $\mathbf{x}$ parameterized by λ
$\dot{\mathbf{m}}$	mass flow rate vector

$\dot{m}_x$	mass flow rate of x
N_v	number of vanes
$P(\mathbf{p}, \mathbf{q})$	perplexity between probability distribution functions $\mathbf{p}$ and $\mathbf{q}$
$\mathbf{p}(s)$	path through the operating space parameterized by s
$p'(t)$	measured combustor pressure signal
$p(x)$	marginal probability distribution function of x
$p(\mathbf{x}, t)$	pressure at location $\mathbf{x}$, time t
$p(x, y)$	joint probability distribution function of x and y
p_i	the ith prime number
$\hat{p}_i(t)$	i^{th} observed oscillatory mode of $p'(t)$
Pr	Prandtl number
$P_x(t)$	dynamic pressure readings at x, psi
Q	reflection coefficient for a semi-infinite line
$\mathbf{q}$	feature vector augmented with inputs $\mathbf{u}$
$\mathbf{Q}$	state error weighing matrix
$Q(\mathbf{x}, t)$	heat release at location $\mathbf{x}$, time t
R	radius
$\mathbf{R}$	input effort weighing matrix
R_i^s	inner radius of the swirler
R_o^s	outer radius of the swirler
$r(x)$	rectified linear unit activation function on x
s	path length along state trajectory
$\mathbf{S}$	a sequence of operating conditions
t	time, seconds
$\mathbf{t}$	vector of training labels
$\mathbf{T}$	matrix of training label data
t_d or δ	time delay, seconds
t_k	label associated with the kth state vector
T_v	vane thickness

$\mathbf{U}$	sequence of control efforts
$\mathbf{u}(t)$	control effort vector at time t
V	volume
$\bar{v}_i$	mean dynamic pressure power at the ith microphone
V_x	voltage associated with x
W	wave shear number
$\mathbf{w}_0$	optimal hyper-plane weights
$\mathbf{W}_k$	weight matrix at k^{th} layer of feed-forward neural network
$\mathbf{x}$	state vector, feature vector, or spacial coordinate
$\mathbf{X}$	set of experimental measurements
$\mathbb{X}_B$	set of points composing a decision boundary
$\mathbf{x}_d$	desired operating condition
y	propagation constant
$\mathbf{y}$	measured operating condition or classification vector
$\tilde{\mathbf{y}}$	classification output as a probability distribution function
$\hat{\mathbf{z}}$	estimated or predicted value of $\mathbf{z}$
$\mathbf{z}_i$	the ith support vector
α	gradient descent learning rate
$\boldsymbol{\beta}$	output weight matrix of ELM network
γ	ratio of specific heats
δ	threshold value
ϵ	inflation factor
ζ	damping ratio
θ_v	vane angle with respect to flow direction
$\mu_{A_{j,i}}(x_j)$	membership value of the jth input and i^{th} fuzzy rule
ν	kinematic viscosity
ρ	density
σ_i	standard deviation of dynamic pressure at the ith microphone
$\sigma(x)$	sigmoid activation function on x

τ	time constant, seconds
ϕ_i	fuel-air equivalence ratio of the ith fuel line
$\boldsymbol{\phi}(\mathbf{x})$	support vector machine nonlinear mapping function
Ω	volume of integration
ω	angular frequency, radians per second
ω_n	natural frequency, radians per second
$\aleph$	training set comprising measurements and labels

ABBREVIATIONS

ACC	active combustion control
ADA*	anytime dynamic A*
Adam	adaptive momentum estimate (gradient descent optimizer)
AG-ELM	adaptive growth extreme learning machine
ARA*	anytime repairing A*
CC	command and control
ELM	extreme learning machine
FELM	fuzzy extreme learning machine
FFT	fast Fourier transform
FIS	fuzzy inference system
FLM	fuzzy learning machine
FSVM	fuzzy support vector machine
HCCI	homogeneous charge compression ignition
I-ELM	incremental extreme learning machine
KL-divergence	Kullback-Leibler divergence
LDI	linear direct injection
LIF	laser-induced florescence
LPM	liters per minute
LQG	linear quadratic Gaussian (regulator)
LQR	linear quadratic regulator
LS-SVM	least-squares support vector machine
LSTM	long short-term memory network
MFC	mass flow controller
MPC	model predictive control
OPC	operating point control

OP-ELM	optimal-pruning extreme learning machine
OS-ELM	online-sequential extreme learning machine
PCA	principle component analysis
PI	proportional-integral (control)
PSVM	proximal support vector machine
relu	rectified linear unit
RNN	recurrent neural network
SLFN	single hidden layer feed-forward neural network
SSA	singular spectrum analysis
SVM	support vector machine
t-SNE	t-distributed stochastic neighbor embedding
UML	Universal Modeling Language
VI	virtual instrument (LabVIEW program)

ABSTRACT

Toner, Nathan L. Ph.D., Purdue University, August 2016. Data Driven Low-Bandwidth Intelligent Control of a Jet Engine Combustor. Major Professor: Galen B. King, School of Mechanical Engineering.

This thesis introduces a low-bandwidth control architecture for navigating the input space of an un-modeled combustor system between desired operating conditions while avoiding regions of instability and blow-out. An experimental procedure is discussed for identifying regions of instability and gathering sufficient data to build a data-driven model of the system's operating modes. Regions of instability and blow-out are identified experimentally and a data-driven operating point classifier is designed. This classifier acts as a map of the operating space of the combustor, indicating regions in which the flame is in a "good" or "bad" operating mode. A data-driven predictor is also designed that monitors the combustion process in real time and provides a prediction of what operating mode the flame will be in for the next measurement. A path planning algorithm is then discussed for planning an input trajectory from the current operating condition to the desired operating condition that avoids regions of instability or blow-out in the input space. An adaptive layer is incorporated into the path planning algorithm to ensure that the path planner can update its trajectory when new information about the operating space becomes available.

1. INTRODUCTION

A jet engine combustor adds energy to a system by mixing air and fuel together and converting this air-fuel mixture's chemical potential energy to heat through combustion. In a linear direct injection (LDI) combustor, fuel is injected directly into an air stream. The air is often passed through a swirler assembly to make it turbulent prior to injecting the fuel. The air and fuel mix together as they pass through a venturi, and the mixture is ignited at the exit of the venturi. See, for example, [1, 2] for typical LDI combustor designs. Combustion typically occurs in a cylindrical or annular cavity called the combustion chamber. The resulting flame either attaches to the venturi or remains detached in the combustion chamber. Air is often added to the combustion chamber downstream of the swirler assembly to complete the combustion process [3]. This configuration is referred to as swirl-stabilized combustion, and is found in many industrial combustor applications. Many modern combustors combine multiple small combustor assemblies into a single combustor cup in order to improve air-fuel mixing and reduce emissions [2, 4, 5].

Full-scale jet engine combustors typically have multiple inputs that can be adjusted, including the mass flow rate of one or more air inlets and the mass flow rate of one or more fuel lines. Typically air flow rate is specified in liters per minute (LPM), and fuel flow rate is determined by the fuel-air equivalence ratio, ϕ_i. Fuel-air equivalence ratio is the ratio between the fuel-to-oxidizer ratio of the mixture and the stoichiometric fuel-to-oxidizer ratio (1.1) [6]. A combustion controller may be used to adjust these inputs to regulate the operating condition of the combustor to some desired condition while avoiding or attenuating any unstable or otherwise undesirable dynamic responses of the combustion process.

$$\phi_i = \frac{m_{\text{fuel}}/m_{\text{ox}}}{\left(m_{\text{fuel}}/m_{\text{ox}}\right)_{\text{st}}} \tag{1.1}$$

The interaction of multiple turbulent mixing and combustion processes, along with the interactions between the heat release of the flame and the acoustic properties of the combustion chamber, result in very complex dynamics governing LDI combustion. This complexity leads to difficulty in developing accurate physics-based models of a combustion process that are also tractable for use for controller design [7, 8], and necessitates the use of finite element modeling methods [9–12] or laboratory-based flame imaging or optical point measurement techniques [13–16] to characterize the response of a given combustion system to various operating conditions. One phenomenon of particular interest in the design and control of combustion systems is the type of instability that occurs due to the interaction of the heat release of the flame with the acoustic properties of the combustion chamber. Instabilities of this nature are known as thermoacoustic instabilities and cause significant discrete-frequency oscillations in the combustor, shortening its lifespan and potentially causing catastrophic failure [14, 17, 18]. These instabilities are self-excited modes of oscillation within the combustor, wherein combustion excites acoustic modes of the system, which in turn feed back into the combustion process when at the appropriate phase [19]. The mechanisms that lead to thermoacoustic instabilities have been extensively studied to inform better combustor design [13, 20–22].

The Rayleigh criterion is often used to describe the conditions under which thermoacoustic instabilities develop in a combustion system [20]. The Rayleigh criterion states that when unsteady heat is added in phase with pressure fluctuations, a condition exists where the pressure is amplified [23]. The normalized Rayleigh index can be defined in terms of the spacial and time-dependent pressure $p(\mathbf{x}, t)$, heat release $Q(\mathbf{x}, t)$, and their means $\bar{p}(\mathbf{x})$ and $\bar{Q}(\mathbf{x})$ (1.2).

$$\mathcal{R} = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \left(\frac{\int_\Omega \left(p(\mathbf{x}, t) - \bar{p}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}}{\int_\Omega \bar{p}(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \right) \left(\frac{\int_\Omega \left(Q(\mathbf{x}, t) - \bar{Q}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \right)}{\int_\Omega \bar{Q}(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \right) \mathrm{d}t \tag{1.2}$$

In (1.2), $\mathbf{x}$ is the spacial coordinate within the combustor, t is time, and Ω is the volume of integration. A value of $\mathcal{R} > 0$ indicates a region where an amplifying relationship exists between thermal and acoustical energy, leading to instability. A

value of $\mathcal{R} < 0$ indicates a region where a damping relationship exists between thermal and acoustical energy. Most efforts in combustion control and combustor design focus on the attenuation of the power of thermoacoustic instabilities.

Attenuating thermoacoustic instabilities is accomplished by interrupting the coupling between the flame and the acoustic modes of the combustion chamber. Passive approaches to this problem include designing the fuel injector assembly or combustion chamber geometry to reduce the likelihood of instabilities occurring at desired operating conditions [24, 25], and removing power from the acoustic modes of the chamber by adding Helmholtz resonators, acoustic liners, and other damping technologies [25–27]. The problem with these passive approaches is that they typically only work for a small range of operating conditions [8]. Active attenuation of combustion instabilities is the natural extension from a design-based approach.

Many modern combustion control systems fall under the category of active combustion control (ACC). These systems utilize high-bandwidth controllers to actively attenuate thermoacoustic instabilities as they are encountered [8, 28, 29]. ACC techniques have been shown to dramatically reduce the acoustic power of resonance modes in both laboratory-scale and full-scale combustors. For laboratory-scale combustors, high-fidelity actuation is typically accomplished with speakers connected to the combustion chamber [28, 30–32]. These acoustic controllers reduce pressure fluctuation amplitude by detecting a resonance mode and actuating the speakers at a phase-shifted copy of this resonance, thus adjusting the pressure component of the Rayleigh criterion (1.2). The phase shift is chosen such that maximum attenuation occurs [19]. This technique has the drawback of potentially altering the acoustic characteristics of the combustion chamber, making control more difficult. Furthermore, these actuators lack sufficient robustness for use in industrial-scale combustion systems. Other methods of changing the system boundary conditions, for example by changing the inlet or exhaust boundary condition, are also employed, but suffer from power limitations at larger scales [8], and are thus not widely adopted in industrial combustors.

For full-scale combustors, actuation is typically accomplished with high-speed modulation of fuel input or with spill valves [33–35], thus adjusting the heat release component of the Rayleigh criterion (1.2). Early attempts at fuel modulation employed on-off automotive fuel injectors, which are nonlinear [36]. Solenoid valves were adopted next for their linear performance, but suffered from poor bandwidth and insufficient control authority [34]. Magneto-restrictive valves and other high-bandwidth actuators have been used more recently and show promise for full-scale applications [37, 38]. Many of these actuators have the drawback of lacking sufficient robustness or requiring frequent maintenance [8], and are thus not widely adopted in industrial combustors.

Most combustion controllers found in industrial applications can be classified as operating point controllers (OPC). OPC techniques regulate a combustor's operating condition to fixed paths through the combustor's operating space in order to get from one desired operating condition to another while avoiding (or powering through) combustion instabilities. Operating conditions and paths are chosen to maintain performance criteria including power output, pollutant emissions, and noise levels [39].

This thesis details the design and implementation of a low-bandwidth OPC strategy to circumvent regions of instability or blow-out in the operating space of a combustor rather than actively cancel them using high-bandwidth control [39]. This strategy avoids the necessity for high-fidelity actuators that suffer from the potential drawbacks discussed previously. The operating space of the combustor system is explored experimentally and an operating point classifier is developed to differentiate "good" and "bad" regions of the operating space. A path planning algorithm utilizes this map *a priori* to determine a trajectory $\mathbf{p}(s)$ from the current operating point $\mathbf{x}$ to a desired operating point $\mathbf{x}_d$ that avoids regions of instability or blow-out while seeking to minimize a total path cost $J_{\mathbf{p}}$. An operating point predictor is developed to monitor the combustion process *in situ* and predict upcoming combustion modes. This information may be used by the path planning algorithm to improve the operating point trajectory in real time. Basic linear quadratic regulator (LQR) control on

the mass flow controllers (MFC) providing air and fuel to the combustion chamber is used to regulate the MFCs to the desired operating point trajectory.

2. PROPOSED CONTROLLER ARCHITECTURE

The proposed controller architecture comprises an operating point classifier built from experimental data that defines a decision boundary between "good" and "bad" operating conditions within the combustor's operating space, a path planning algorithm that uses this classifier map *a priori* to determine a trajectory between a starting condition and desired condition that avoids bad operating conditions, a predictor that monitors the combustion process *in situ* and attempts to predict upcoming operating modes using more data than the *a priori* classifier, and a controller that regulates the system to the desired operating point trajectory. This process is illustrated in Figure 2.1, wherein circles represent discrete operating conditions at which the system has been tested, the gray oval represents a region of instability that has been detected, and the path from the current operating point to the desired point is given by arrows between operating points.

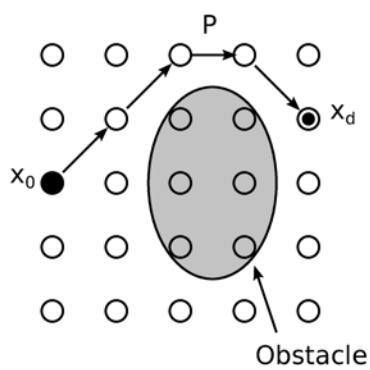


Figure 2.1. Illustration of the proposed controller moving the combustion system through its operating space while avoiding identified "obstacles".

The components of the proposed control architecture and their interactions are illustrated in a block diagram in Figure 2.2. Herein, the focus is on the path planner,

system, sensors, and predictor. A simple control loop following the desired trajectory through the operating space is assumed.

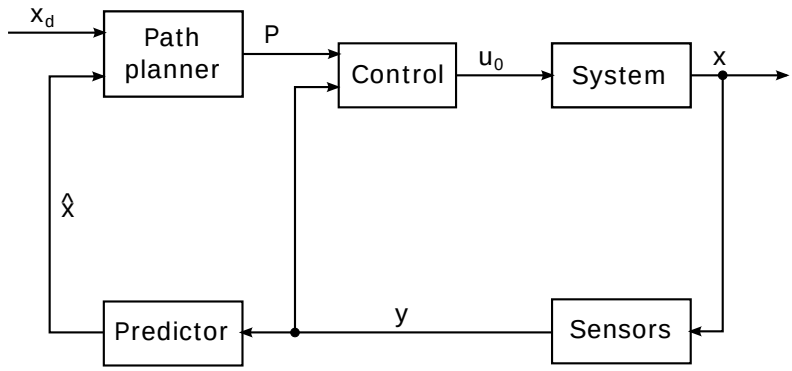


Figure 2.2. Proposed controller block diagram.

In Figure 2.2, $\mathbf{x}_d$ is the desired operating condition, $\mathbf{x}$ is the current operating condition of the system, $\mathbf{y}$ is the measured operating condition of the system, $\hat{\mathbf{x}}$ is the upcoming operating condition of the system predicted by the predictor, $\mathbf{P}$ is the operating point sequence representing the desired trajectory generated by the path planner, and $\mathbf{u}_0$ is the first element of this input sequence. The components of this block diagram will be discussed in the following. The components and interactions of the controller are illustrated in a universal modeling language (UML) [40] sequence diagram shown in Figure 2.3, wherein a model predictive control (MPC) structure is shown.

UML sequence diagrams like Figure 2.3 show the sequential interaction between different processes or entities. Each entity is identified with a label at the top of the diagram and its lifeline is represented by a dashed line extending downward from the label. The active time of an entity (with respect to the control architecture) is represented by a solid box on the lifeline. Calls from one entity to another are represented by solid horizontal arrows and return messages are represented by dashed arrows. A self-call is represented by a looped arrow. Sub-processes (usually stemming from self-calls) are shown as multiple boxes stacked on top of the entities lifeline.

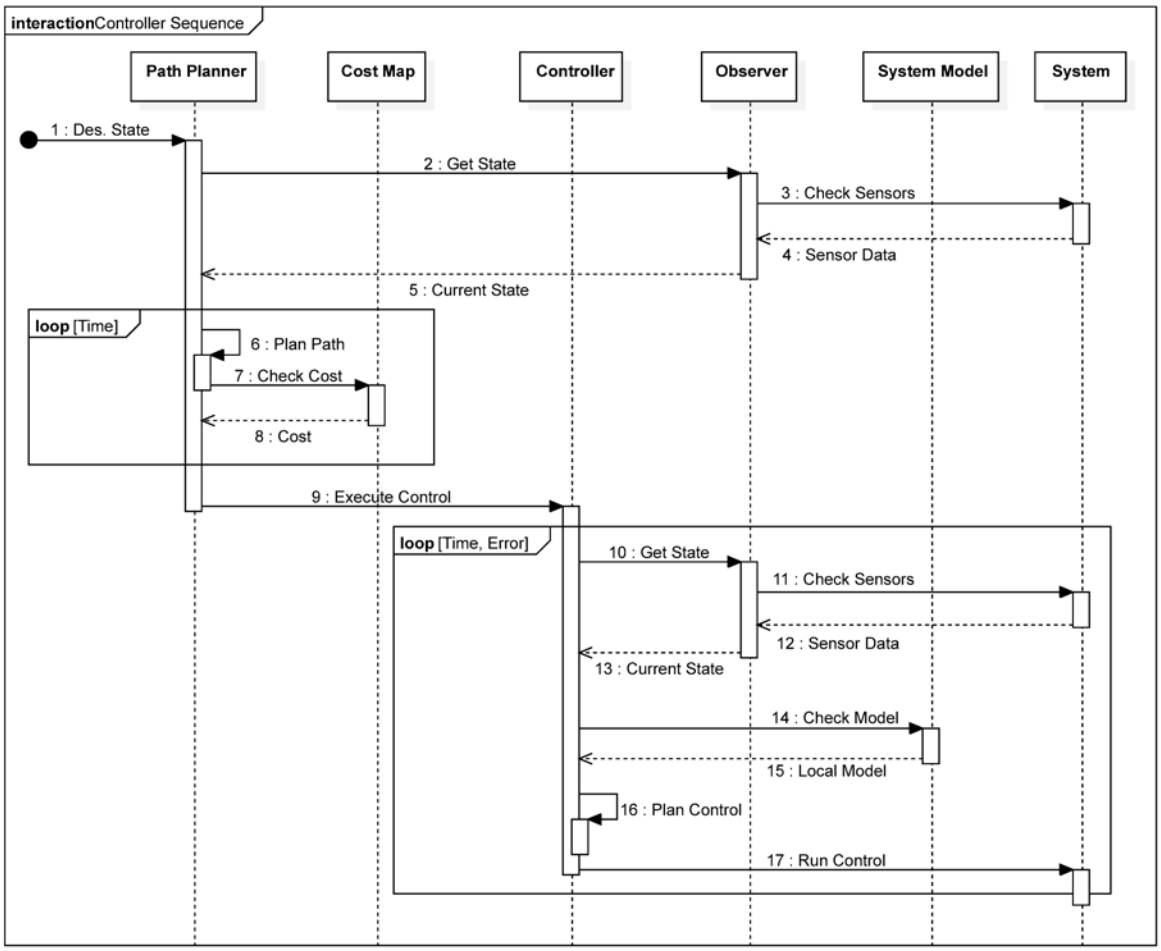


Figure 2.3. Proposed controller sequence diagram.

Loops within the sequence are identified by boxes around the looped processes and are labeled with the stop condition for the loop.

Figure 2.3 shows a single iteration of the process that the controller undergoes each time it receives a new desired operating point. The whole process will repeat as the system is moved along the operating point trajectory. If a more optimal trajectory is determined by the path planner or new information is incorporated into the operating space map, the controller will simply follow the new trajectory from its current operating point to the desired point.

3. NINE-ELEMENT RESEARCH COMBUSTOR

A multi-element combustor comprising nine separate LDI combustion elements—each element comprising an air swirler, fuel line, and venturi—was designed and built for developing the proposed intelligent control system. The combustor was designed to be fed by a single air line and three separately controllable fuel lines, and to meet dimensional constraints imposed by existing laboratory equipment. A drawing showing the overall dimensions of the combustor assembly is shown in Figure 3.1, where units are inches.

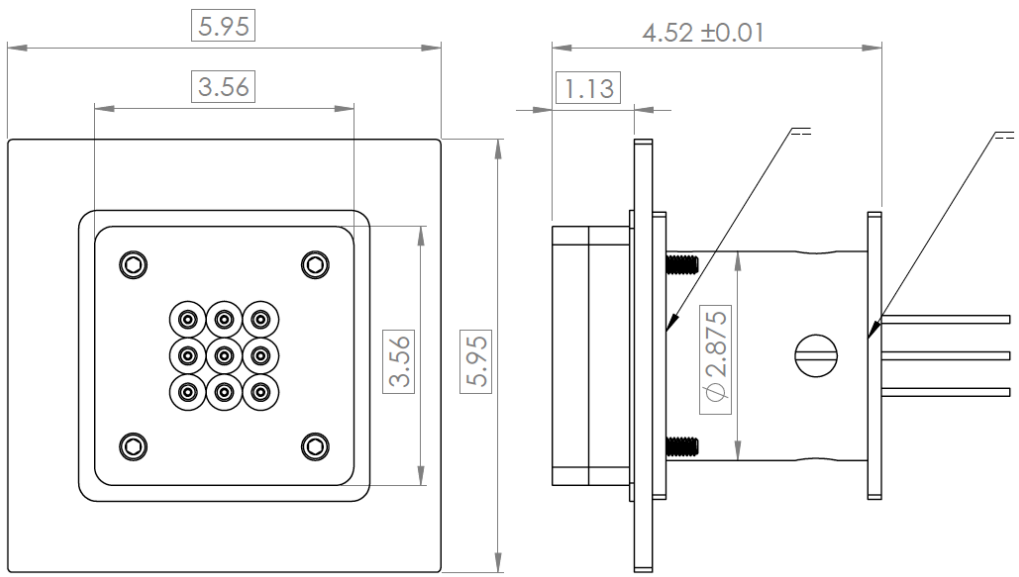


Figure 3.1. Nine-element combustor assembly dimensions in inches.

The multi-element combustor has the following characteristics that are desirable for the proposed controller:

- It is a complicated system that is very difficult to model.

- It has an operating space that includes regions of flame instability, lean blow-out, and rich blow-out.
- It must move through this operating space according to the dynamics of the mass-flow controllers and combustion chamber.
- As the system heats up, its dynamics and the nature of the unstable regions of its operating space will likely change, requiring a controller that can adapt to new information.

The complicated dynamics of the system lend themselves well to a controller that can learn a dynamic model of the system through experimental data. The regions of instability and blow-out in the operating space can be viewed as obstacles when planning a trajectory from the current operating condition to a desired operating condition.

3.1 Design

Fuel is fed from three separate mass flow controllers through nine fuel rods that extend from the base of the combustor to the exit of the venturi plate. Air is fed from a single mass flow controller through multiple air lines into a cylindrical pipe. The air then travels through a sintered plate and a honeycomb flow straightener to ensure uniform airflow. A pressure tap immediately following the sintered plate provides a measurement of the air pressure prior to air-fuel mixing and combustion. Air then passes through a nine-element swirler puck and exits into the venturi plate. The fuel rods pass through holes in the center of each swirler element before exiting upstream of the venturi plate where air and fuel are mixed. The enclosure around the swirler puck press fits inside of the combustion chamber, sealing that end of the chamber. A mounting plate was added to the design to enable easier mounting to experimental apparatus. The components of the system can be seen in a section view shown in Figure 3.2.

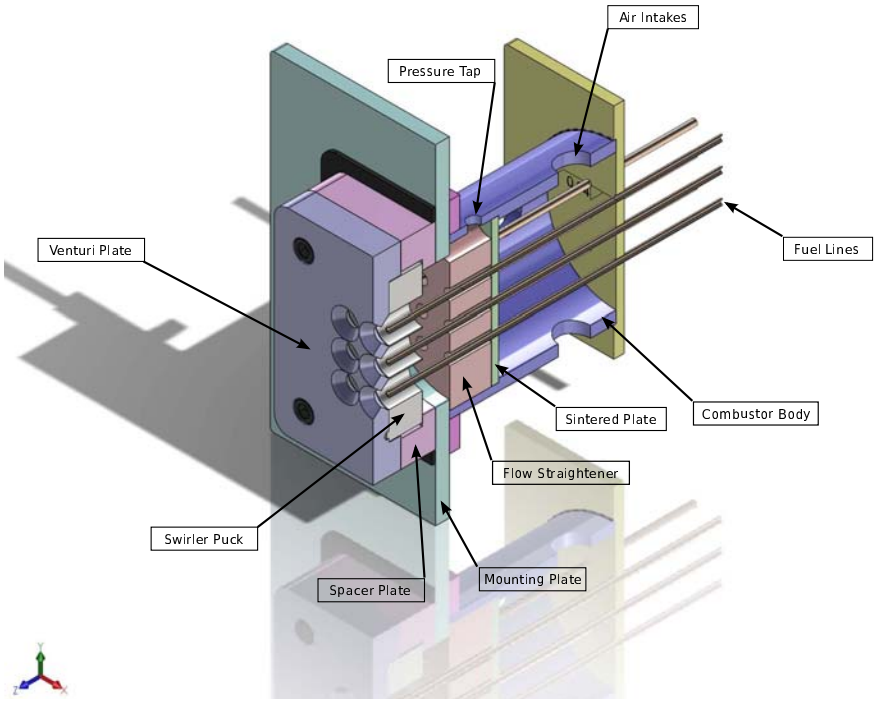


Figure 3.2. Section view of the nine-element combustor assembly.

The combustion chamber is a 3 foot long stainless steel square tube with a 4×4 inch outer cross section (3.56×3.56 inch inner cross section). Transparent viewing windows begin 1 inch from the bottom of the combustion chamber (where the combustor assembly is installed). The swirler puck in the combustor assembly is sandwiched between the venturi plate and a spacer plate. The spacer plate and venturi plate provide a 1 inch standoff that fits inside of the combustion chamber and raises the exit of the venturi plate to the level of the transparent viewing window as shown in Figure 3.3. Note that Figure 3.3 does not show the full length of the combustion chamber.

The nine elements of the combustor assembly were divided into three fuel stages as shown in Figure 3.4. The center element is an independent fuel stage referred to as the "pilot" stage. The remaining elements are divided into two fuel stages according

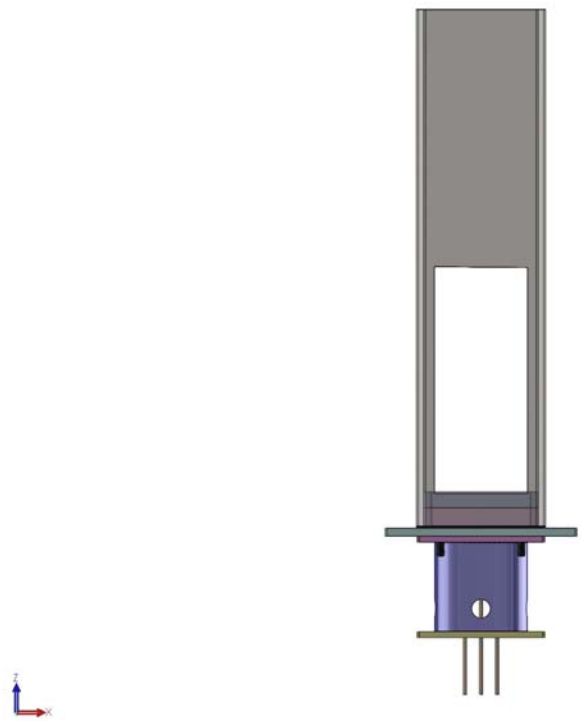


Figure 3.3. Nine-element combustor assembly with combustion chamber. The combustor assembly is designed to raise the base of the flame to the level of the bottom of the viewing window, as shown here.

to their distance from the center, and are referred to as the "middle" and "outer" stages. The equivalence ratio of each fuel stage, ϕ_i, can be set independently.

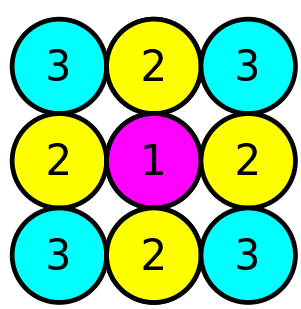
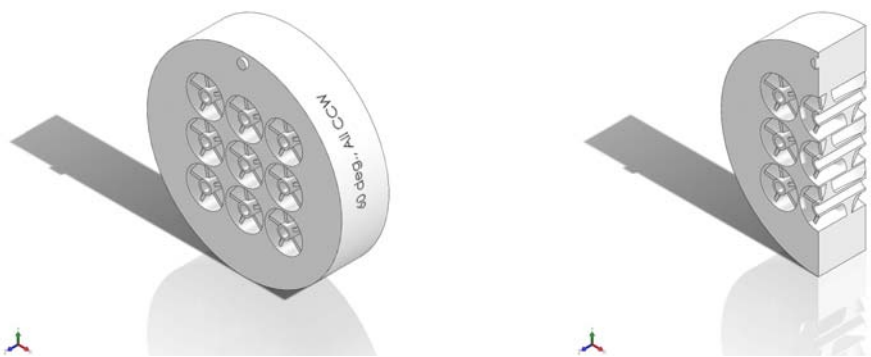


Figure 3.4. Illustration of the nine-element combustor fuel stage layout. The center "pilot" fuel line is independent, and then the middle ring and outer ring are each on separate fuel lines.

The swirler puck was designed with nine 60° vane angle swirler elements with five vanes per element. Each element has a channel through the center through which the fuel line passes. In the current swirler puck, all swirler elements have counter-clockwise rotation. The rotational direction, vane angle, and number of vanes of each swirler element are design parameters that can be adjusted to change the characteristics of the combustor [41]. The swirler puck is shown in Figure 3.5(a), and its cross section clearly showing the fuel channels through the center of each swirler element is shown in Figure 3.5(b).



(a) (b)

Figure 3.5. (a) Nine-element swirler puck with 60° vane angle and (b) its cross section.

The swirler puck has an effective area of $A^s_{\text{eff}} = 0.4387$ in^2. The effective area of the swirler is the total area of the swirler normal to air flow, minus any blockage due to swirler vanes and the fuel rod passage through the center, and multiplied by a discharge coefficient [42]. If the inner radius of the swirler is defined as the radius of the outer wall of the fuel passage R^s_i, the outer radius of the swirler is R^s_o, the

thickness of the swirler vanes is T_v, and the number of swirler vanes is N_v, then the effective area of the swirler A^s_{eff} can be calculated as shown in (3.1).

$$\begin{aligned} A_{\text{annulus}} &= \pi\Big(\left(R^s_o\right)^2 - \left(R^s_i\right)^2\Big) \\ A_{\text{blocked}} &= T_v N_v \left(R^s_o - R^s_i\right) \\ A_{\text{geometric}} &= \left(A_{\text{annulus}} - A_{\text{blocked}}\right)\cos\theta_v \\ A^s_{\text{eff}} &= A_{\text{geometric}} \times C^s_D \end{aligned} \tag{3.1}$$

In (3.1), θ_v is the vane angle with respect to the flow direction, and C^s_D is the discharge coefficient of the swirler, which was assumed to be $C^s_D = 0.6$ for this design. The effective area of a single swirler element viewed from the top is shaded blue in Figure 3.6.

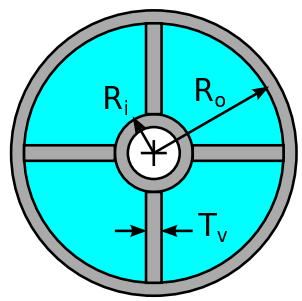


Figure 3.6. Illustration of swirler effective area. Effective area is shaded blue, and is the total swirler element area minus the fuel line area and the area taken up by the swirler vanes and walls.

3.2 Manufacturing and Assembly

The nine-element combustor was manufactured in the Mechanical Engineering student machine shop with the following exceptions:

- Welding of end plates to the cylindrical air chamber was done by Purdue University Research Machining Services.
- 3D printing of the swirler puck was done by the College of Engineering I2I Lab.

- The flow straightener was cut from aluminum honeycomb by Kent Machining, Inc. in Pendleton, IN using an electrostatic discharge machining process.

The majority of the combustor was manufactured from mild steel. The venturi plate was made from 304 stainless steel due to the high-temperature reacting environment to which it is exposed. Machining was done by hand on a three-axis manually-controlled milling machine, and on a manually-controlled lathe. Final reshaping of the combustor standoff for fitting into the combustion chamber was accomplished by hand using a grinding wheel and file. Manufacturing drawings for the various components of the nine-element combustor can be found in Appendix A. The completed nine-element combustor assembly is shown in Figure 3.7.

(a) Isometric View (b) Side View

Figure 3.7. Finished nine-element burner assembly.

Swagelok tube fittings were used to connect air and fuel lines, and to connect the static pressure gauge upstream of the swirler and venturi assembly. Swagelok tube fittings and union tee fittings were used to create fuel manifolds that split fuel coming from the air mass flow controller and from two of the fuel mass flow controllers into four lines each. These lines were then connected to the air intake fittings on the main cylinder of the combustor assembly, and to the inner and outer ring of fuel stages shown in Figure 3.4.

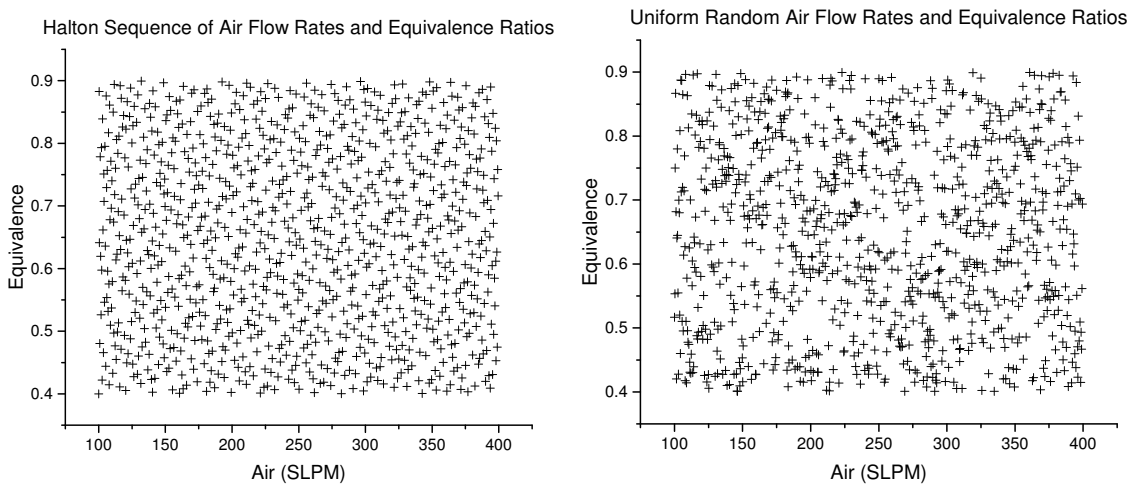
4. EXPERIMENTAL PROCEDURE

The nine-element combustor system has four inputs, $\mathbf{u} \in \mathbb{R}^4$, that can be controlled: the mass flow rate of air to the system and the mass flow rates of each of the three fuel stages. Fuel mass flow rates are determined from a desired overall equivalence ratio, ϕ_t, and percentages of total fuel flow for each fuel stage. The system was tested at various input conditions to build a map of the operating space that can then be used to determine regions of instability and blow-out, and to inform the design and implementation of the proposed controller. The tested operating conditions were chosen using an algorithm based on a low-discrepancy sequence (Section 4.1). A LabVIEW program—called a virtual instrument (VI)—was built to run the system through each of these operating conditions and record response data (Section 4.2). The LabVIEW program controlled the air and fuel mass flow rates using mass flow controllers connected to the system (Section 4.3), and recorded operating condition information read by static and dynamic pressure transducers, thermocouples, and a high-temperature probe (Section 4.4).

4.1 Choosing Operating Conditions

Low-discrepancy or quasi-random sequences like the Halton sequence and variations thereof [43] are deterministic sequences that possess random-like qualities. These sequences have been shown to improve the convergence of numeric integration and modeling techniques like support vector machines (SVM), extreme learning machines (ELM), and single-layer fuzzy neural networks (SLFN); they tend to have random-like properties while more uniformly covering a region of interest like a section of the operating space of a system [44, 45]. A comparison of a two-dimensional 1000-point Halton sequence and pseudo-random sequence for air mass flow rates and

fuel-air equivalence ratios is shown in Figure 4.1. The Halton sequence shown in Figure 4.1(a) covers the area of interest more uniformly than the random sequence shown in Figure 4.1(b), avoiding the regions of clumped or sparse sample points.



(a) Halton sequence

(b) Uniform pseudo-random sequence

Figure 4.1. Comparison of Halton sequence and uniform random sequence for generating 1000 air mass flow rate and overall equivalence ratio test points. Note that the Halton sequence (a) covers the region more uniformly with fewer gaps and clumps than the pseudo-random sequence (b).

An M-element Halton sequence [46] over a range of the input space $\mathbf{u} \in \mathbb{R}^m$ of a system is built by designating a prime number, p_i, associated with each dimension of the input space for $i = 1, 2, \ldots, m$. Any number N can be written in p-ary notation as (4.1).

$$N = e_M p^M + \cdots + e_1 p + e_0 \qquad 0 \leq e_j \leq p - 1 \tag{4.1}$$

Thus N can be represented by the base-p integer string $e_M \cdots e_1 e_0$. The Halton sequence then takes the radical inverse of this (4.2), which generates a very uniformly-distributed sequence on the interval $[0,\ 1]$ for each prime p_i.

$$R_p(N) = \frac{e_0}{p} + \frac{e_1}{p^2} + \cdots + \frac{e_M}{p^{M+1}} \tag{4.2}$$

The sequence $\mathbf{S}$ is then built for m dimension by generating a unique radical inverse sequence for each coordinate using the first m primes (4.3).

$$\mathbf{S}_N = \begin{bmatrix} R_2(N) & R_3(N) & R_5(N) & R_7(N) \end{bmatrix} \tag{4.3}$$

For the combustion system, the operating conditions of interest were determined using a Halton sequence to set the air mass flow rate $\dot{m}_A$ and total fuel-air equivalence ratio ϕ_t between certain desirable bounds by scaling the Halton sequence. The ith Halton sequence S_i can be scaled by the minimum and maximum values of interest for the ith input, $u_{i,\min}$ and $u_{i,\max}$ as in (4.4).

$$S_i^*(j) = u_{i,\min} + S_i(j) \times \left(u_{i,\max} - u_{i,\min}\right) \tag{4.4}$$

In (4.4), $S_i(j)$ and $S_i^*(j)$ are the jth element of the original and scaled sequence respectively, with $j \in [1, M]$ for the ith dimension. The limits of interest for the air mass flow rate and total fuel-air equivalence ratio are:

- $\dot{m}_A \in [1, 4]\%$ pressure drop across the combustor assembly.
- $\phi_t \in [0.4, 0.9]$

Note that "% pressure drop" is the percentage increase in pressure over atmospheric measured just up-stream of the combustor assembly, and is a common method of specifying the air flow rate through a combustor assembly. For our system, pressure drop was approximately equal to the voltage input to the air MFC. Three additional Halton sequence elements, $c_i \in [0, 1]$ for $i = 1, 2, 3$, were generated and then normalized (4.5) to represent the proportion of the total fuel flow rate provided by each MFC.

$$\tilde{c}_i = \frac{c_i}{\sum_{j=1}^{3} c_j} \tag{4.5}$$

The total fuel flow rate was determined from the air mass flow rate and desired total fuel-air equivalence ratio ϕ_t (4.6), and then split between each of the fuel MFCs using the normalized fuel proportions $\tilde{c}_i$ (4.7). If any individual MFC flow rate generated in this way was below $\approx 5\%$ of the MFCs maximum flow rate, that flow rate

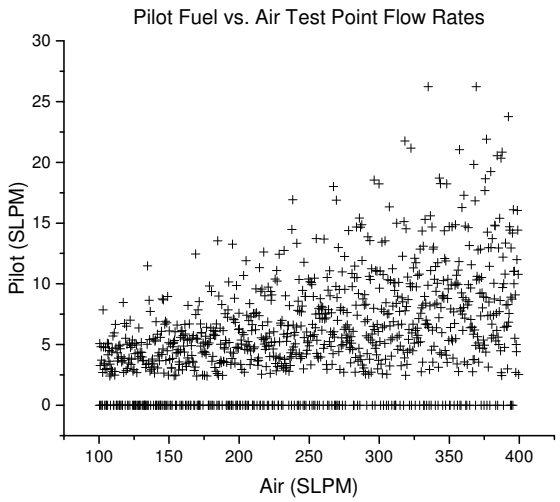
was set to zero, and another MFC's flow rate (typically the middle fuel line's MFC) was increased to maintain the desired overall flow rate. Checks were performed to ensure that the overall total fuel flow rate after these adjustments met the desired value.

$$\dot{m}_F = \frac{\phi_t \dot{m}_A}{\left(\dot{m}_{\text{fuel}}/\dot{m}_{\text{ox}}\right)_{\text{st}}} \tag{4.6}$$

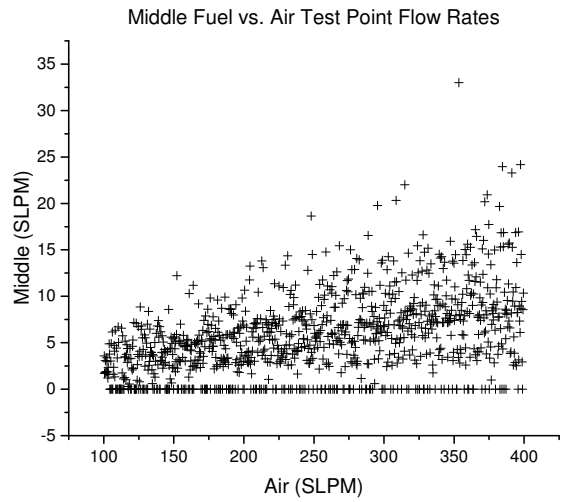
$$\dot{m}_i = \tilde{c}_i \dot{m}_F \tag{4.7}$$

The resulting four-dimensional operating conditions of interest were the scaled mass flow rates of air and the three fuel lines. This sequence can be denoted $\mathbf{S}^*$. A Python script was written to determine this input sequence, and the sequence of operating conditions was saved to a colon-delimited text file to be used by a LabVIEW VI (Section 4.2). The resulting fuel mass flow rates are plotted against the associated air mass flow rate for each of the three fuel lines in Figure 4.2. The middle fuel line shown in Figure 4.2(b) does not exhibit as clear a gap between low flow rates and zero because this fuel line was used to balance small flow rates as mentioned previously.

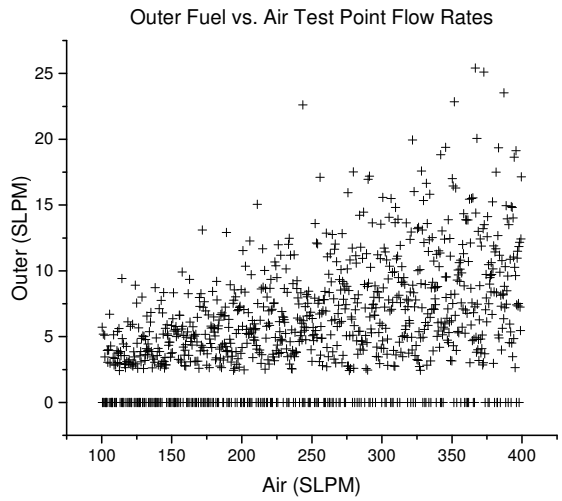
Note that the Halton sequence requires a large number of data points M if the prime numbers p_i are large. What constitutes "large" must be determined heuristically, but an insufficient number of points in the sequence for larger prime numbers will result in clear patterns and higher-discrepancy in the sequence. As a different prime number should be used for each dimension of a sequence, this issue can arise in higher-dimensional sequences. For this reason, modifications to the Halton sequence have been developed that further reduce discrepancy, including the scrambled Halton sequence [43], which randomly permutes the sequence for each prime $R_p(N)$ resulting in lower-discrepancy sequences in higher dimensions. For this experiment, the largest prime number used was $p_5 = 11$, and the 1000 data point sequence was still sufficiently random-like.



(a) Pilot



(b) Middle



(c) Outer

Figure 4.2. 1000 Halton-sequence-generated fuel and air flow rate test points for each of three fuel lines (see Figure 3.4). Note that fuel flow rates exhibit a ramp-like structure because the Halton sequence generates quasi-random fuel-air equivalence ratios, and the fuel-air equivalence ratio depends linearly on the air flow rate.

4.2 Running the Experiment

A LabVIEW command and control (CC) VI was created to automate experiments on the nine-element burner. The VI commands the system to move between the oper-

ating conditions of interest generated by the Halton sequence discussed in Section 4.1, and records the readings from various sensors monitoring the system as it transitions between and settles on operating conditions. If a blow-out is detected, the VI first goes to its current target operating condition to attempt to reignite at that condition, and then returns the system to a known stable operating condition if the target condition could not be lit. It waits for the flame to be re-ignited before moving on to the next operating condition of interest. Operating conditions are selected sequentially from the Halton sequence, resulting in a quasi-random walk through the operating space.

The VI was written as an object-oriented state machine, wherein different "states" are designed to perform specific tasks by calling class methods, and an overall state machine manages transitions between these states to ensure that the experiment is run properly. An object-oriented approach was chosen to simplify data handling using the encapsulation of data within a class. Two main classes were used: a Data Handler class that connects to the database and records experimental data in a queue, and an Experiment class that handles running the experiment and placing data on the queue. The Experiment object used in the VI comprises a Data Handler object. A UML class diagram for these two classes is shown in Figure 4.3.

A UML activity diagram detailing the operation of the LabVIEW CC VI is shown in Figure 4.4. This diagram shows the states of the VI in boxes and decision branches as diamonds. Arrows between elements on the diagram indicate the flow of the program between states and decision branches.

The *Initialize* state in Figure 4.4 waits for the user to specify an operating condition input file name, database connection information, and data field information. Once submitted, this state activates the sensors and mass flow controllers for the system, opens a connection to the input file, opens a connection to the database that is used to store experimental data, and spawns a process that will write data to this database as it is recorded. Figure 4.5 shows an activity diagram for this data writing process. The next state takes initial atmospheric pressure readings using a static

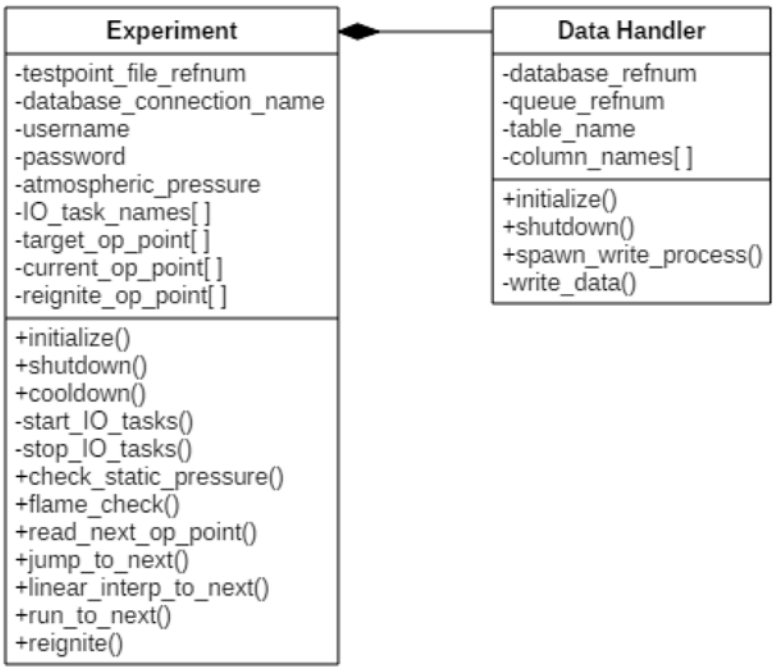


Figure 4.3. LabVIEW experiment automation VI class diagram.

pressure sensor and waits for the user to launch the experiment. The atmospheric pressure is stored for later use.

Once the experiment is launched, the VI moves the system to a known stable ignite condition. The system then waits until a flame is detected (not shown for this first step in the activity diagram) before moving on to load the input file containing the operating conditions of interest for the experiment. The VI reads the next input condition in this file (the target), and determines a step-wise linear path between the current operating condition and the target operating condition. It then commands the system to move step-wise along this path, only recording response data when the transients of the MFCs have settled, or when a blow-out occurs. This method was chosen to simplify data analysis as the input conditions should be stationary for any given reading. Once the end of the input file is detected, the VI calls a shutdown state which turns off the mass flow controllers (keeping air running for cool-down), deactivates the sensors, closes resources, and exits, checking for errors as it does so.

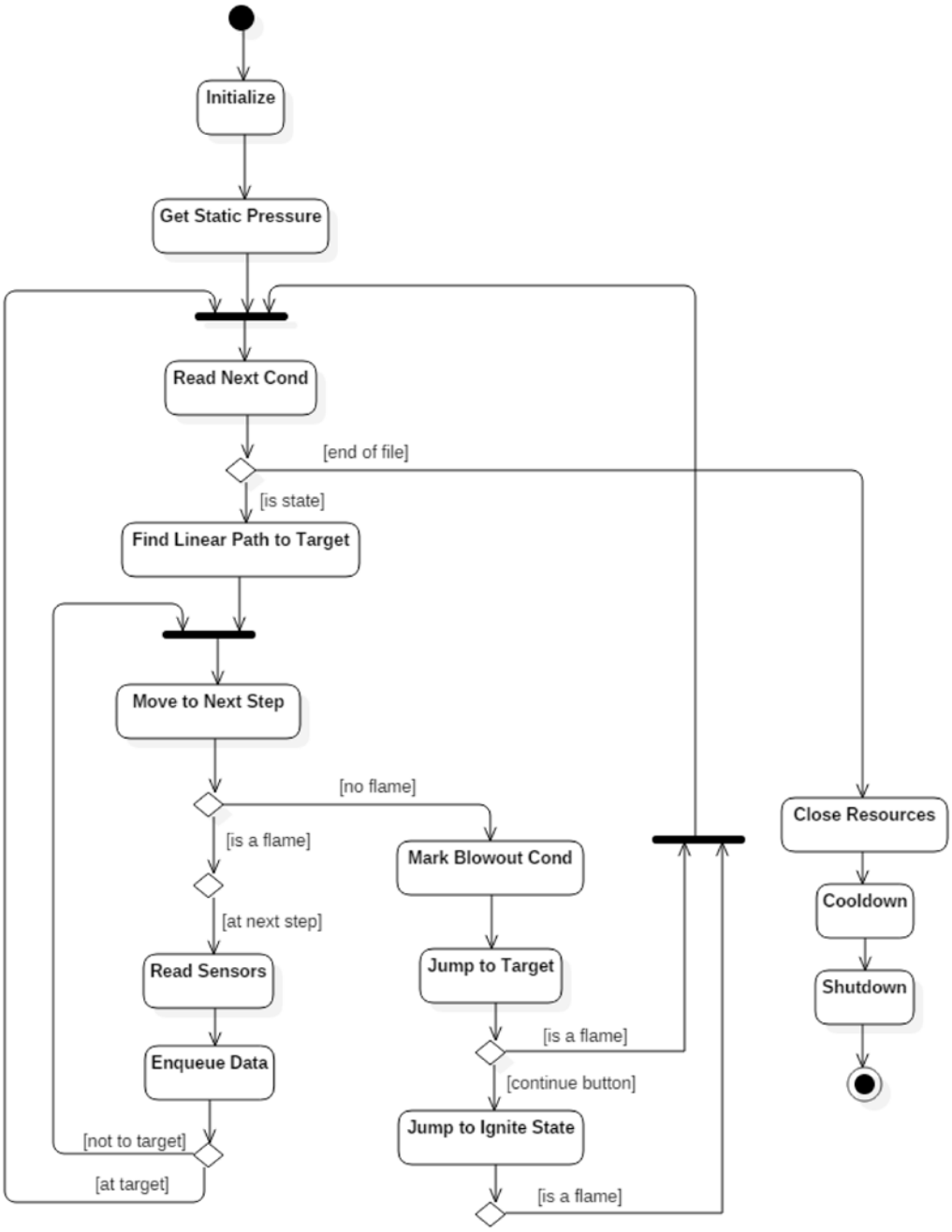


Figure 4.4. LabVIEW experiment automation VI activity diagram.

As long as a flame is detected, the VI generates a linear path between the current operating condition and the next and commands the mass flow controllers to follow this path, pausing at evenly-spaced steps and recording sensor readings as it goes.

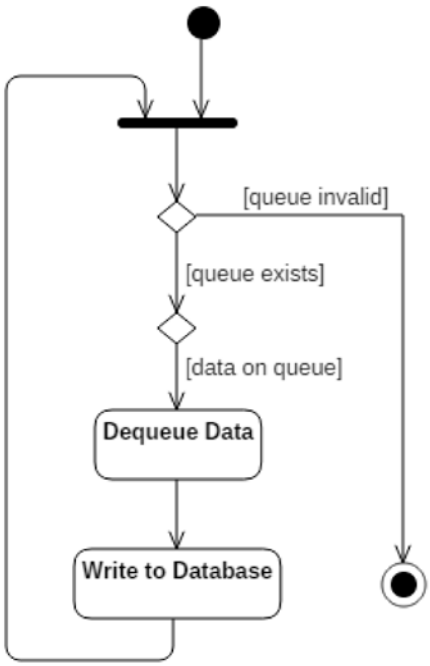


Figure 4.5. LabVIEW data writing VI activity diagram.

If at any time the flame blows out, the operating condition at blow-out is recorded along with the flame state, and the system is first transitioned to the target operating condition for re-ignition. The system is then transitioned to a known stable condition if it cannot be re-ignited at the target condition. Data is saved in an Oracle® MySQL database and processed with a Python script (Chapter 5).

4.3 Hardware Setup

The flame from the nine-element combustor is contained within a 36 inch long combustion chamber with a 3.56 inch square interior cross section. Fused quartz windows are installed along the sides of the combustion chamber, beginning 1 inch from the bottom. The nine-element combustor is designed to fit inside of this combustion chamber so that the exit of the venturi plate is brought to the level of these viewing windows. This is illustrated in Figure 3.3.

The nine-element combustor is mounted on a three-axis translation table illustrated in Figure 4.6. The combustor is mounted on the central red-colored platform in the figure. This table enables the combustion chamber to be moved relative to the centerline of a CCD camera or laser, enabling various axial and radial locations of the flame to be measured using techniques like chemiluminescence and laser-induced florescence (LIF). The position of the table is controlled manually by a separate LabVIEW VI, and is set to a constant position for these experiments.

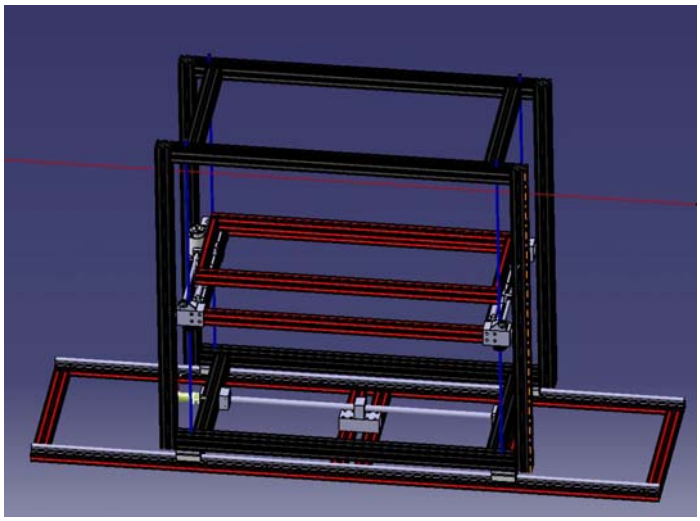


Figure 4.6. Three-axis translation table for experimental procedures.

Air and fuel are connected to the combustor via flexible hoses. Air and fuel mass flow rates are controlled using Porter mass flow controllers as given below. Each air and fuel line uses a single dedicated mass flow controller so that all lines can be independently adjusted. For the middle and outer fuel lines, the single fuel flow coming from the MFC is split into four lines using a manifold made from Swagelok® T-junctions.

1. Air: Porter Model 203A 500 SLPM flow rate calibrated for air.

2. Fuel line 1: Porter Model 251, 40 SLPM flow rate calibrated for methane.

3. Fuel line 2: Porter Model 251, 40 SLPM flow rate calibrated for methane.

4. Fuel line 3: Porter Model 251, 40 SLPM flow rate calibrated for methane.

The mass flow controllers are in turn connected to a Porter PCIM4 computer interface module. The interface module sets a flow-rate based on a 0–5 V input signal and uses an integrated PI controller to regulate the mass flow controller to this flow rate. Each mass flow controller returns a voltage $V_{\text{out}} \in [0,\ 5]$ VDC that is proportional to and approximately linear with respect to the flow rate through the MFC. The components and interactions of the lab hardware are shown schematically in Figure 4.7.

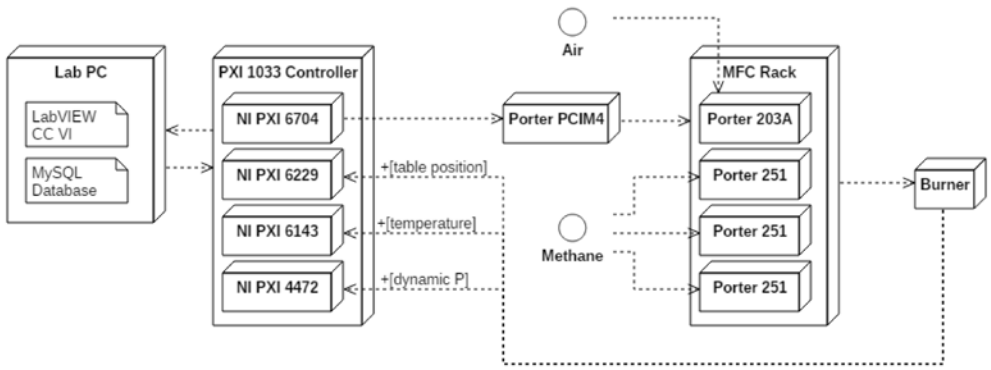


Figure 4.7. Schematic of lab hardware and interactions.

The MFCs were characterized by recording the step response of each MFC for various input voltages using the voltage feedback from the MFC to measure the response. The 2 V step response of the air MFC is shown in Figure 4.8, where the input is the voltage command sent to the MFC, and the response is the voltage signal read from the MFC, and is proportional to the flow rate through the MFC as discussed in the following. Additional step responses can be found in Appendix B.

The flow rate from the MFCs exhibit a time-delayed first- or second-order response with respect to input voltage, the general transfer functions for which are given by (4.8) and (4.9) respectively. In the following, t_d is the time delay in seconds, τ is the first-order time constant in seconds, ω_n is the natural frequency of a second-order system in radians per second, and ζ is the damping ratio of a second-order system.

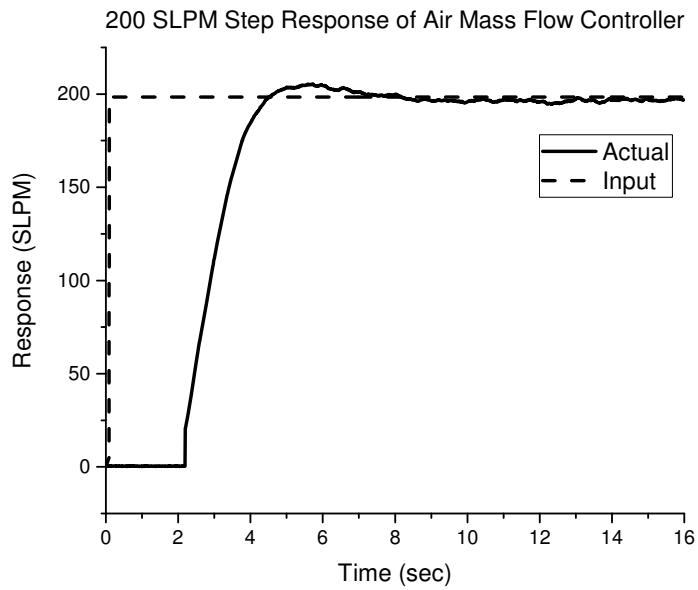


Figure 4.8. 2 V step responses of air mass flow controller.

The static gain K_{MFC} is given by (4.10) where $V_{\max} = 5$ VDC, $V_{\min} = 0$ VDC, $\dot{m}_{\min} = 0$ SLPM, and $\dot{m}_{\max}$ varies for the air and fuel MFCs and is given in the preceding. The model parameter values for each of the MFCs are summarized in Table 4.1.

$$G_{P1}(s) = \frac{Q(s)}{V(s)} = e^{-t_d s}\frac{K_{MFC}}{\tau s + 1} \tag{4.8}$$

$$G_{P2}(s) = \frac{Q(s)}{V(s)} = e^{-t_d s}\frac{K_{MFC}}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1} \tag{4.9}$$

$$K_{MFC} = \frac{\dot{m}_{\max} - \dot{m}_{\min}}{V_{\max} - V_{\min}} \tag{4.10}$$

We can develop a controller to better regulate the MFCs to the desired mass flow rate. To do so, we must first linearize the transfer function (4.8) and (4.9) by replacing the time delay term with a Padé approximation (4.11), where the numerator and denominator can be of arbitrary order m and n respectively. The numerator and denominator are respectively defined by (4.12) and (4.13) [47]. For the first-order system, a Padé approximation of order $m = n = 2$, and for the second-order systems,

Table 4.1. Mass flow controller model parameters.

MFC	K_{MFC}, SLPM/V	τ, sec	ζ	ω_n, rad/s	t_d, sec
Air	99.21	N/A	0.7054	1.35	1.8
Pilot	7.967	0.520	N/A	N/A	0.546
Middle	7.964	0.516	N/A	N/A	0.194
Outer	7.972	0.239	N/A	N/A	0.306

a Padé approximation with numerator order $m = 3$ and denominator order $n = 2$ fit the response data well. This substitution results in the linearized system dynamics given in state-space form (4.14) for the first-order system and (4.15) for the second-order system.

$$e^{-t_d s} \approx \frac{\sum_{i=1}^{m} p_i}{\sum_{i=1}^{n} q_i} = \frac{1 - \frac{3t_d}{5}s + \frac{3t_d^2}{20}s^2 - \frac{t_d^3}{60}s^3}{1 + \frac{2t_d}{5}s + \frac{t_d^2}{20}s^2} \tag{4.11}$$

$$p_i = (-1)^i \frac{(m+n-i)!m!}{(m+n)!i!(m-i)!} \tag{4.12}$$

$$q_i = \frac{(m+n-i)!n!}{(m+n)!i!(n-i)!} \tag{4.13}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{-12}{\tau t_d^2} & \frac{-12\tau - 6t_d}{\tau t_d^2} & \frac{-6\tau - t_d}{\tau t_d} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \frac{12}{\tau t_d^2} \end{bmatrix} V \tag{4.14}$$

$$Q = K_{MFC} \begin{bmatrix} 1 & \frac{-t_d}{2} & \frac{t_d^2}{12} \end{bmatrix} \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \ddot{x} \end{bmatrix}^T$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-20\omega_n^2}{t_d^2} & \frac{-8t_d\omega_n^2-40\zeta\omega_n}{t_d^2} & \frac{-t_d^2\omega_n^2-40t_d\zeta\omega_n-20}{t_d^2} & \frac{-2t_d\zeta\omega_n-8}{t_d} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{20\omega_n^2}{t_d^2} \end{bmatrix} V \qquad (4.15)$$

$$Q = K_{MFC} \begin{bmatrix} 1 & \frac{-3t_d}{5} & \frac{3t_d^2}{20} & \frac{-t_d^3}{60} \end{bmatrix} \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \ddot{x} & \dddot{x} \end{bmatrix}^T$$

A step response for the air mass flow controller is shown against the step response of the model for this mass flow controller using (4.15) in Figure 4.9.

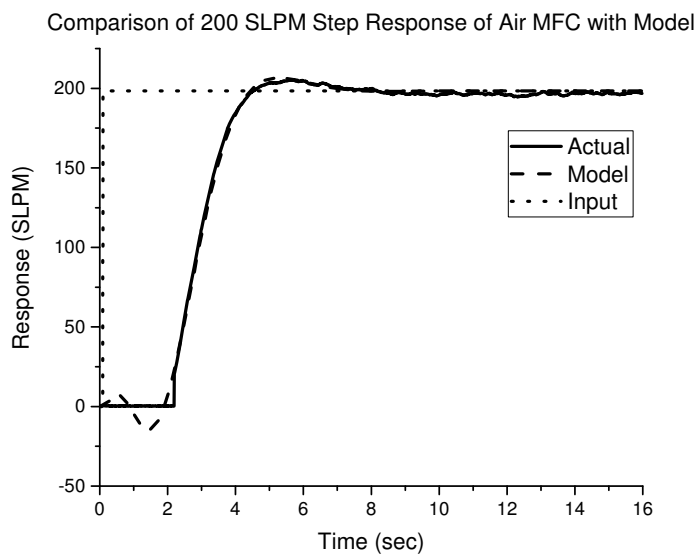


Figure 4.9. Comparison of actual and Padé approximation model step response of air mass flow controller.

A simple linear quadratic regulator (LQR) can then be designed using the state space equation (4.14) or (4.15). The control effort for the LQR is defined as $u = -\mathbf{Kx}$ and the control law $\mathbf{K}$ is determined by solving the algebraic Riccati equation (4.16) [48] assuming the state space equations (4.14) and (4.15) can be expressed as

$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$ and $Q = \mathbf{C}\mathbf{x}$. In (4.16), $\mathbf{Q}$ is a state error weighing matrix, and $\mathbf{R}$ is an input effort weighing matrix.

$$\begin{aligned} \mathbf{K} &= \mathbf{R^{-1}B^{T}P} \\ 0 &= \mathbf{A^{T}P} + \mathbf{PA} - \mathbf{PBR^{-1}B^{T}P} + \mathbf{Q} \end{aligned} \tag{4.16}$$

In the event that there is significant noise in the system—i.e. $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k + \mathbf{v}_k$ and $Q_k = \mathbf{C}\mathbf{x}_k + \mathbf{w}_k$ where $\mathbf{v}_k$ is Gaussian white process noise, $\mathbf{w}_k$ is Gaussian white measurement noise, and equations are shown in discrete time—a linear quadratic Gaussian regulator (LQG) can be used [49]. An LQG is an LQR where the state of the system in the presence of noise is estimated using a Kalman filter. The control effort the the LQG is defined as $u_k = -\mathbf{K}\hat{\mathbf{x}}_k$ where $\hat{\mathbf{x}}_k$ is the Kalman-estimated state of the system (4.17).

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}u_k + \mathbf{L}_k\Big(Q_{k+1} - \mathbf{C}\big(\mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}u_k\big)\Big) \tag{4.17}$$

The Kalman gain $\mathbf{L}_k$ is iteratively determined by solving the following discrete Riccati equation (4.18) [48].

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{A}\Big(\mathbf{P}_k - \mathbf{P}_k\mathbf{C}^T\big(\mathbf{C}\mathbf{P}_k\mathbf{C}^T + \mathbf{W}\big)^{-1}\mathbf{C}\mathbf{P}_k\Big)\mathbf{A}^T + \mathbf{V} \\ \mathbf{P}_0 &= E\Big[\big(\mathbf{x}_0 - \hat{\mathbf{x}}_0\big)\big(\mathbf{x}_0 - \hat{\mathbf{x}}_0\big)^T\Big] \end{aligned} \tag{4.18}$$

Where $\mathbf{V}$ and $\mathbf{W}$ are the estimated covariance matrices of the process noise and measurement noise respectively. The Kalman gain $\mathbf{L}_k$ at time k is then given by (4.19).

$$\mathbf{L}_k = \mathbf{A}\mathbf{P}_k\mathbf{C}^T\big(\mathbf{C}\mathbf{P}_k\mathbf{C}^T + \mathbf{W}\big)^{-1} \tag{4.19}$$

By observation, process and measurement noise were minimal for the MFCs and an LQR controller was sufficient to regulate the MFCs to desired set points.

4.4 Instrumentation

Local dynamic pressure measurements were taken at four points along the axis of the combustion chamber using GRAS 46BD 1/4 inch CCP microphone sets. Mi-

crophones were placed approximately logarithmically, with tighter spacing between microphones near the primary combustion zone, and wider spacing further away. A median-filtered mean measured power from the closest microphone to the combustor was used as a reliable indicator of whether a flame was present. An ambient dynamic pressure measurement was taken near the combustion chamber using a GRAS 46AQ 1/2 inch CCP random incidence microphone set. The dynamic pressure sensor locations are shown in Figure 4.10, where all units are inches unless otherwise specified.

The GRAS 46BD microphones have a safe operating temperature range up to 70 C, much lower than the temperature of the combustion chamber. For this reason, a method of making local dynamic pressure measurements without jeopardizing the sensors was required. The GRAS microphones were installed several inches away from the combustion chamber behind a heat shield using remoting tubes [50–52] to transfer dynamic pressure fluctuations within the combustion chamber to the microphones. Each remoting tube comprises a 15 cm standoff tube that transfers pressure fluctuations from the combustion chamber to the pressure transducer, and a 30 m open damping tube tube after the pressure transducer that provides a semi-infinite termination. The semi-infinite termination helps to reduce the effects of high-frequency wave reflections off the end of the remoting tube. The end of this coil was left open to atmosphere, as the combustion process takes place at atmospheric pressure and so bulk fluid flow through the remoting tube was not a concern. A Swagelok® T-junction is used to connect the pressure transducer to the standoff tube and damping tube, and a Swagelok® weld fitting is used to connect the remoting tube assembly to the combustion chamber. This setup is illustrated in Figure 4.11, where units are inches unless otherwise specified. For a full drawing of the combustion chamber assembly with remoting tubes, see Appendix A.

The dynamic pressure fluctuations measured by a GRAS microphone, $P_m(t)$, are the convolution of the pressure fluctuations within the combustion chamber, $P_p(t)$, with the dynamics of the remoting tube, $P_r(t)$ (4.20). Equivalently, the frequency

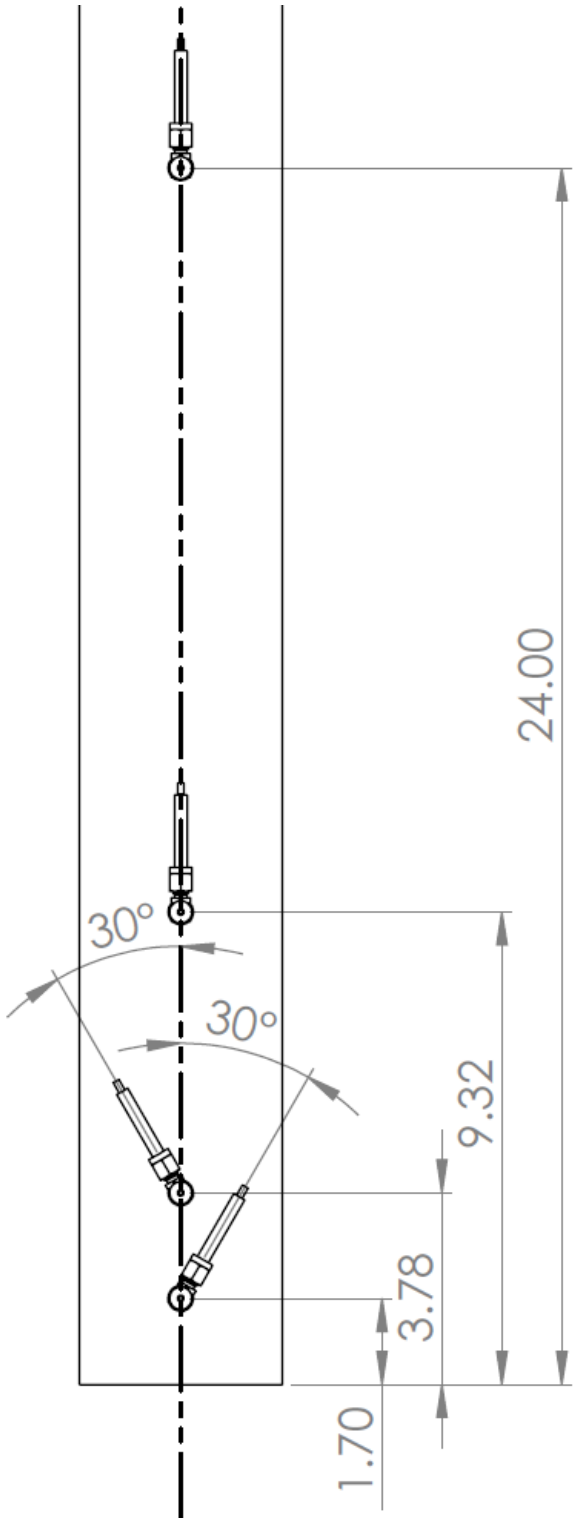


Figure 4.10. Illustration of dynamic pressure measurement locations, measurements are in inches and degrees.

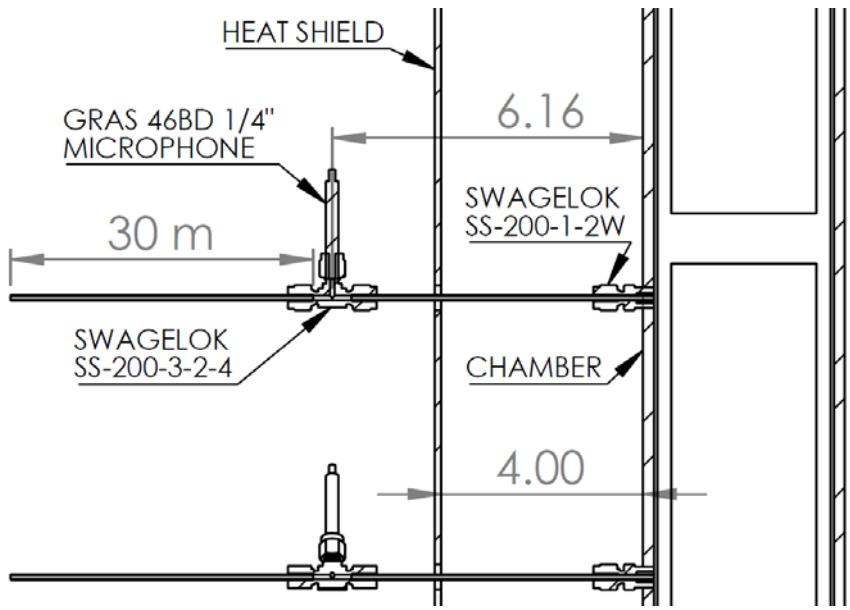


Figure 4.11. Illustration of dynamic pressure remoting assembly with heat shield.

response measured by a microphone, $G_m(j\omega)$, is the frequency response of the combustion process, $G_c(j\omega)$, in series with that of the remoting tube, $G_r(j\omega)$ (4.21). To recreate the approximate dynamic characteristics of the combustion process, the response measured by the microphones must be deconvolved with an approximate model of the dynamics of the remoting tube.

$$P_m(t) = P_c(t) * P_r(t) \tag{4.20}$$

$$G_m(j\omega) = G_c(j\omega) \cdot G_r(j\omega) \tag{4.21}$$

The remoting tube adds some volume to the system that is being measured, and this volume can be characterized by a frequency response function, $G_r(j\omega)$. Pressure waves traveling through a tube are attenuated primarily via boundary layer interactions [51], and visco-thermal effects [50]. For the combustor operating at atmospheric pressure, we can assume that no bulk fluid flows through the remoting tube, and so visco-thermal effects are the dominant attenuating factor [52,53]. Bergh and Tildeman [53] developed the standard model for pressure fluctuations within a series connection of N tubes with N volumes assuming no bulk fluid flow through

the tubes. Using the simplifying assumptions of Samuelson [50, 54]—primarily that the tube is homogeneous except at the transducer, which adds a small volume, V, to the system—the frequency response function of the standoff tube with semi-infinite termination can be expressed as (4.22).

$$G_r(j\omega) = \frac{P_1}{P_0}(j\omega) = \Big(\cosh(yL_{01}) + Q\sinh(yL_{01}) + \big(\sinh(yL_{01})\tanh(yL_{1e})\big)\Big)^{-1} \tag{4.22}$$

where L_{01} is the distance from the wave source to the pressure transducer (standoff length), L_{1e} is the distance from the pressure transducer to the semi-infinite tube termination, Q is a reflection coefficient (4.23), and y is the propagation constant (4.24).

$$Q = \frac{\omega V}{cA}\left[\left(1 + \frac{2(\gamma-1)J_1(E)}{EJ_0(E)}\right)\left(\frac{2J_1(W)}{WJ_0(W)} - 1\right)\right]^{-\frac{1}{2}} \tag{4.23}$$

$$W^2 = -R^2\frac{j\omega\rho}{\mu} = -R^2\frac{j\omega}{\nu}$$

$$E = \mathrm{Pr}W$$

$$y = \frac{\omega\left(1 + \frac{2(\gamma-1)J_1(E)}{EJ_0(E)}\right)^{\frac{1}{2}}}{c\left(\frac{2J_1(W)}{WJ_0(W)} - 1\right)^{\frac{1}{2}}} \tag{4.24}$$

In (4.23) and (4.24), the zero and first-order Bessel functions are denoted J_0 and J_1 respectively. Their arguments are the dimensionless shear wave number, W, and the product, E, of this shear number with the Prandtl number Pr. The transducer volume is denoted V, and represents the additional cavity volume introduced where the pressure transducer intersects the remoting tube assembly. The cross-sectional area and radius of the standoff tube are denoted A and R respectively, and the speed of sound is denoted c. Analysis of these models and empirical study of remoting tube configurations shows that shorter standoffs reduce attenuation of pressure fluctuations and are preferable over longer standoffs, larger radius tubes increase the bandwidth of the remoting tube assembly, and a very long damping tube must be used to avoid distortion at lower frequencies [51, 55, 56]. The necessary length of the damping tube increases as the radius of the remoting tube increases.

To determine minimum standoff tube lengths, 1 m long 1/8 inch stainless steel tubes with a 2.29 mm inner diameter were connected to the combustion chamber and the system was run to a steady external temperature. Thermocouples were then inserted into the tubes and slowly moved closer to the combustion chamber until the operating temperature limit of the microphones was reached. Several centimeters was then added to this length as a buffer. The standoff tubes were then cut, and the remoting tubes and heat shield assembled with thermocouples in place of the pressure transducers. The system was again run to a steady external temperature to ensure that the pressure transducer would not exceed its maximum operating temperature in this final configuration. Finally, the thermocouples were replaced with the pressure transducers so that dynamic pressure measurements could be made.

Temperatures at the outer walls of the combustion chamber were measured using four Omega 5TC-GG-K-24-36 24 AWG K-type thermocouples with braided glass insulation at locations corresponding to each of the dynamic pressure probes. The exit temperature of the flame was monitored with an Omega TJ36-CAXL-38U-18 Super-OMEGACLAD® XL heavy duty transition junction K-type thermocouple probe with a 3/8 inch diameter.

A GE PMP-4060 static pressure transducer was used to measure the absolute atmospheric pressure and the absolute pressure upstream of the swirler puck (see Figure 3.2). The difference between these two measurements corresponds to the pressure drop across the swirler and venturi assembly. This pressure drop was used to determine the overall mass flow rate through the combustor, and is fed back to the mass flow control loop to maintain a desired overall flow rate. The pressure transducers are configured to measure pressures in the range $P \in [11.5, 17.5]$ psi.

Thermocouples and pressure transducers are connected to a National Instruments PXI-1033 controller. The PXI also handles sending commands to the computer interface modules. The PXI interfaces with the desktop running the CC VI through an NI PXI-1033 Integrated MXI Express card, enabling 110 MB/s sustained throughput between the PXI and PC. The PXI has one high-speed analog input card for interfacing

with dynamic pressure transducers, one analog output card, and two general-purpose data acquisition cards installed:

- NI PXI-4472: 8-channel, 24-bit, 102.4 kS/s simultaneous analog input
- NI PXI-6704: analog output
- NI PXI-6229: 32-channel multifunctional DAQ
- NI PXI-6143: 8-channel, 16-bit, 250 $^{\mathrm{kS}}/_{\mathrm{s}}$ multifunctional DAQ

5. DATA PROCESSING

One significant hurdle to using conventional machine learning techniques with our data is that our data is not yet labeled. By understanding the structure of the data, we can employ unsupervised learning techniques like K-means clustering to assign categories to different operating modes of the combustion chamber. The analysis and categorization of the data set is discussed in the following.

Experimental data was stored in an Oracle® MySQL database and processed using custom Python scripts. The data that was recorded every 0.01 seconds included the desired and actual mass flow rates for each of the four mass flow controllers (air and three fuel lines) in LPM, the outer combustion chamber wall temperatures at four locations along the flame axis in degrees Centigrade, the static pressure drop across the venturi plate as a percent of atmospheric, and the raw dynamic pressure readings from the four microphones placed along the flame axis using remoting tubes as discussed in Section 4.4. Dynamic pressure readings were taken at a rate of 10 kHz, so 100 dynamic pressure readings from each microphone were recorded for each reading from the thermocouples, static pressure sensor, etc.

5.1 Power Spectra and Frequency Analysis

Data was segmented into 0.5 second windows, with each window overlapping by 50%—the first window contained data recorded between 0 and 0.5 seconds, the second window between 0.25 and 0.75 seconds, and so on. A fast Fourier transform (FFT) was performed on each window, resulting in the power spectra varying over the duration of the experiment. Each spectrum was scaled to decibels and plotted on a time vs. frequency "waterfall" as shown in Figures 5.1–5.5 where the color represents the decibel magnitude of a specific frequency at a specific time. Frequencies range between

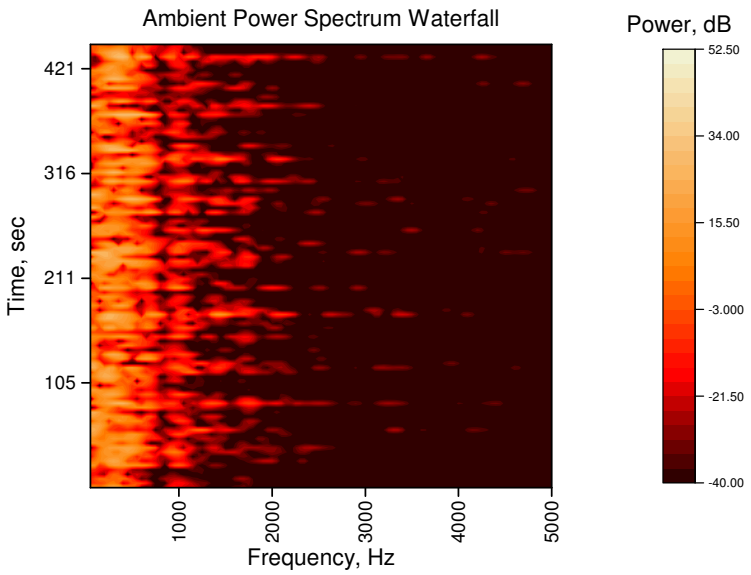


Figure 5.1. Power spectrum waterfall plot of ambient microphone signal. The ambient microphone sits outside of the combustion chamber, approximately 1 m away, and exhibits the richest dynamics in the lower frequency ranges.

0 on the left to 5 kHz on the right. Time begins at 0 at the bottom and increases going up.

The vertical bands most prominent in Figure 5.2 do not change with operating condition, and represent structure imposed by the frequency response dynamics of the physical combustion chamber: these bands exist because similar power was recorded at nearly the same frequency throughout most of the duration of the experiment. Evenly spaced bands most likely represent harmonics of the combustion chamber. By observing the response of the microphones measuring inside of the combustion chamber (Figure 5.2–5.5), we see that the primary resonance frequency of the combustion chamber is approximately 250 Hz, with measurable harmonics up to 1500 Hz (six times the primary) and higher at some operating conditions. The primary resonance mode occurs in a region of very rich frequency response, but after approximately 400 Hz, the harmonics of this mode dominate the response.

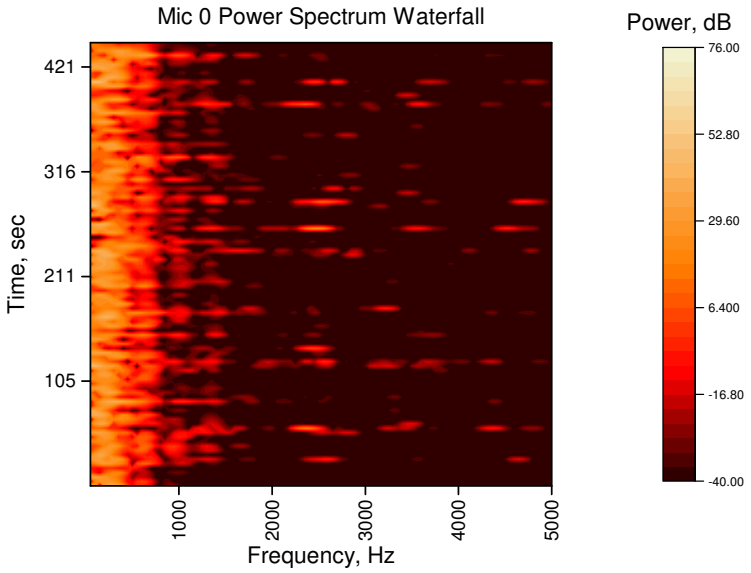


Figure 5.2. Power spectrum waterfall plot of the microphone 0 signal. Note the vertical banding in the signal, indicating structure in the frequency response of the chamber.

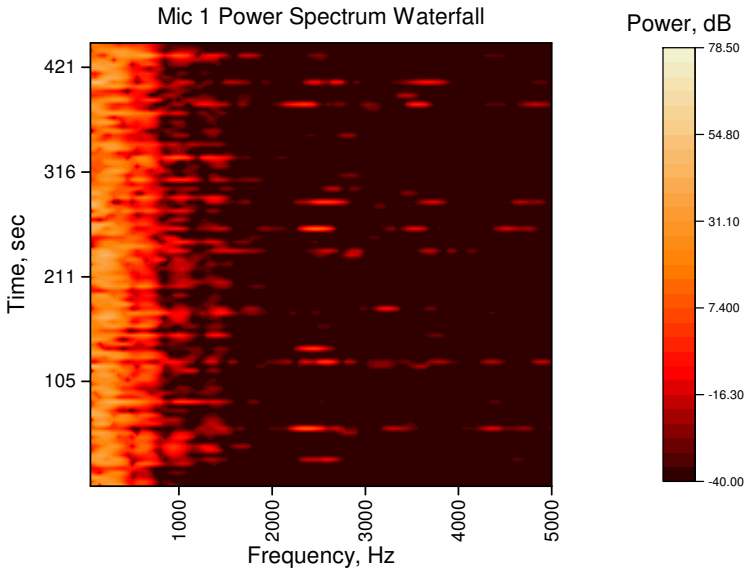


Figure 5.3. Power spectrum waterfall plot of the microphone 1 signal. Note that vertical banding is still present, but is slightly less distinct than the banding in Figure 5.2.

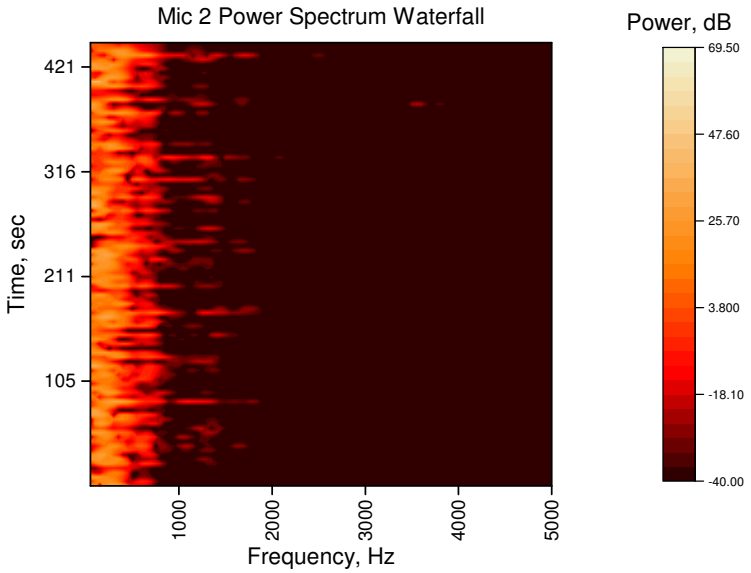


Figure 5.4. Power spectrum waterfall plot of the microphone 2 signal. Here the visible structure of the signal is significantly reduced as the microphone moves further from the combustor assembly.

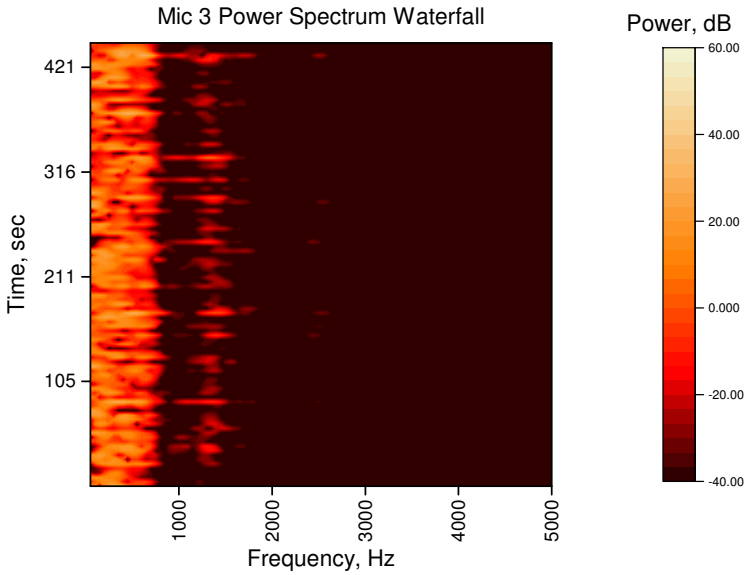


Figure 5.5. Power spectrum waterfall plot of the microphone 3 signal. Here all but the highest harmonics of the combustion chamber have blended together.

Horizontal bands represent changes in the response of the system as the input conditions changed over time. Note that dynamic pressure data was only recorded when the input conditions were not changing; thus each horizontal line in the waterfall plots of Figures 5.1–5.5 represents the power spectra for a specific, stationary input condition. By observation, the combustion chamber exhibited four distinct categories of combustion: attached flames, detached stable flames, detached flickering flames (local extinction), and complete blow out. Flames that are on the verge of attaching or detaching, and intermittently flicker between these two modes, could be considered a fifth category. Very little data was collected for blow-out conditions, and so these do not show up well in the waterfall plots. Lower power conditions typically correspond to detached stable flames, which are much quieter than attached or flickering flames. Higher power conditions without very high-frequency dynamics typically correspond to attached flames, as these are louder than detached flames. The conditions showing the highest-frequency dynamics typically correspond to flickering flames, which exhibit characteristics of a square wave and have high-frequency harmonics. These flickering states often precede a blow-out, and can be seen as long horizontal bands of high power primarily in Figures 5.1–5.3.

The structure observable in these figures and the previous discussion implies that this data can be used to cluster experimental data into flame mode categories. This process will be discussed shortly.

5.2 Mutual Information Analysis

Mutual information is a measurement of the similarity between the joint probability distribution function of two variables, x and y, and the naïve joint probability assuming the two distribution functions are independent. In other words, it is the number of bits of information saved if one understands the relationship between x and y rather than assuming that they are independent. More generally, it can be used as a distance measure between two processes [57, 58].

The mutual information between two discrete distributions X and Y is defined as (5.1), where $p(x)$ (or $p(y)$) is the marginal probability distribution function of x (or y), and $p(x, y)$ is the joint probability distribution function of x and y.

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{5.1}$$

Figure 5.6 shows an auto-mutual information waterfall plot where the y-axis shows the measurement time, and the x-axis shows a time delay, δ. The color of the plot indicates the degree of similarity between two samples, delayed by δ indicated by the x-axis.

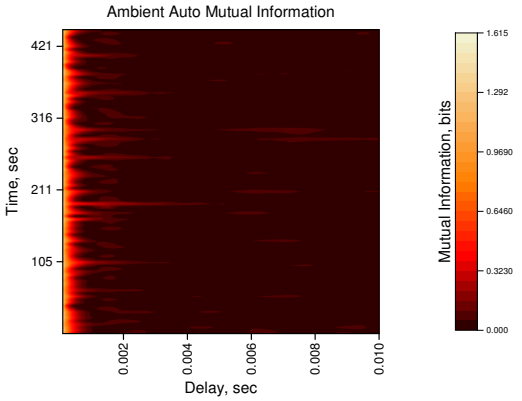
There is not much consistent structure in the auto-mutual information; however, the wider horizontal bands of high mutual information typically correspond to local extinction flame modes (flickering), and precede a blow out. There is a faint vertical band present in all microphones around $\delta = 0.002$ seconds. This perhaps indicates wave reflections occurring within the combustion chamber, or quasi-periodicity in the flame. Significant vertical banding in the auto-mutual information would imply some periodic structure of the flame. As is, very little periodic structure appears to exist. The flame is, perhaps, chaotic.

5.3 K-Means Clustering

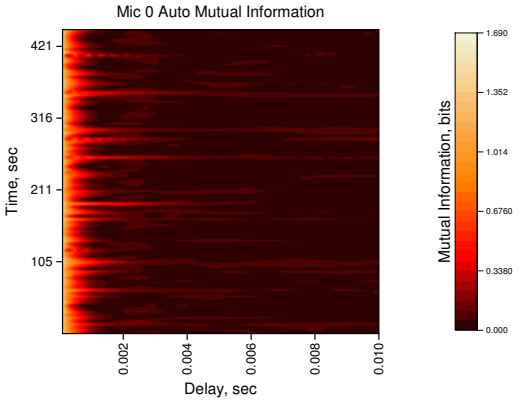
K-means clustering is an unsupervised classification technique whereby unlabeled data is divided into K classes such that the in-class sum-squared distance of points in that class to the class centroid is minimized [59]. If the data set is given as $\mathbf{X} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \dots & \mathbf{x}_n \end{bmatrix}$ where $\mathbf{x}_i$ is the ith sample and there are n samples, the K-means algorithm can be summarized by the following:

1. Initialize K centroids $\mathbf{c}_j$ with $j \in [1, K]$ at the locations of K randomly-selected samples from the data set.

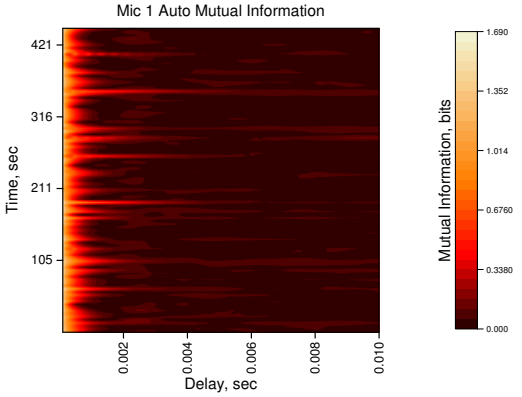
2. Calculate the distance matrix $\mathbf{D} = \begin{bmatrix} d_{i,j} \end{bmatrix}_{n \times K}$ from every point in the data set to each of the K clusters: $d_{i,j} = (\mathbf{x}_i - \mathbf{c}_j)^T (\mathbf{x}_i - \mathbf{c}_j)$.



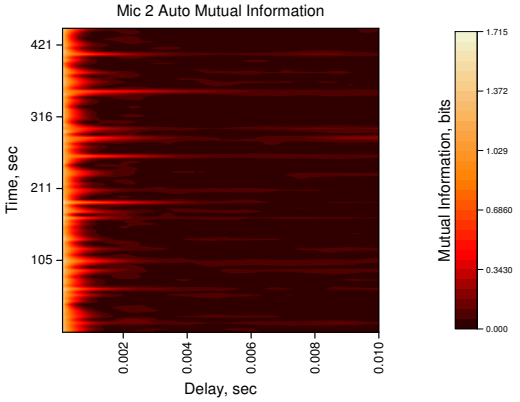
(a) Ambient



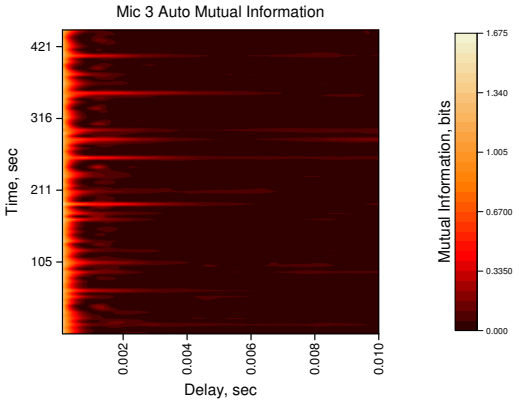
(b) Mic 0



(c) Mic 1



(d) Mic 2



(e) Mic 3

Figure 5.6. Waterfall plots of auto-mutual information for the ambient dynamic pressure sensor and each of the dynamic pressure sensors along the axis of the combustion chamber.

3. Assign a label y_i to each point in the data corresponding to the cluster to which it has the minimum distance: $y_i = \underset{j \in [1,K]}{\arg\min}\, d_{i,j}$

4. Update the cluster centroid locations to the mean location of all of the points within that cluster: $\mathbf{c}_j = \frac{\sum \mathbf{x}_i}{N_j} : \{\mathbf{x}_i | y_i = j\}$. Note that here N_j is the number of elements with label j.

5. If the maximum number of iterations has been reached, break.

6. If any cluster data point assignment has changed, return to step 2.

7. If no assignment has changed, break.

What data is considered in determining distance to a centroid affects how well the algorithm separates data into distinct categories [60, 61]. Using the data structure analysis discussed in the preceding, flame status $b_k \in \{0, 1\}$ static pressure $P_k \in \mathbb{R}$, mass flow rate of each MFC $\dot{\mathbf{m}}_k \in \mathbb{R}^4$, and mean power measured by each microphone $\bar{\mathbf{v}}_k \in \mathbb{R}^5$ were chosen as the features of interest for K-means clustering. Note that the flame status, b_k, was determined using a threshold on the median-filtered mean power reading from the closest microphone to the combustor in the combustion chamber. This proved to be a reliable measurement of the presence of a flame. Additional tests were conducted with more features, but did not converge to meaningful clusterings. A TensorFlow [62] script was written to perform K-means clustering with $K = 2, 3, \ldots, 7$ centroids to determine the appropriate number of categories for the data set. K-means clustering is sensitive to the choice of initial centroid locations [63], and so a Monte-Carlo method was employed to run the K-means algorithm many times for each value of K with randomly-selected starting centroids. The algorithm is shown in pseudocode below.

```
mean_distances = []
for K in range(2, 8):
  MC_history = []
  for _ in range(MC_repeats):
```

```
    centroids, assignments, distances = run_k_means(data, K)
    MC_history.append(mean(distances[:, assignments]))
    record centroid and assignment history too...
  mean_distances.append(min(MC_history))
  keep track of best centroids and assignments...
plot(range(2, 8), mean_distances)
```

The main K-means function, `run_k_means()`, runs a number of iterations of the K-means algorithm and quits if centroid locations cease to change. Its sub-function, `initialize_centroids()`, randomly chooses points from the input data set as initial centroids. This random selection of initial centroids is repeated for each iteration of the Monte-Carlo run for training K-means with a given value of K. These two functions are shown in pseudocode below.

```
def run_k_means(data, K):
  centroids = initialize_centroids(data, K)
  cluster_assignments = zeros(len(data))
  for i in range(max_iterations):
    # Find closest centroids
    for j in range(len(data)):
      for k in range(len(centroids)):
        distances[j, k] = (data[j] - centroids[k])**2
    best_centroids = argmin(distances, axis=1)  # find closest centroid
    if best_centroids == cluster_assignments:
      break  # no change in assignments, we're done
    cluster_assignments = best_centroids

    # Update centroid locations
    for i in range(len(centroids)):
      centroids[i] = mean(data[best_centroids == i])
  return (centroids, cluster_assignments, distances)
```

```
def initialize_centroids(data, K):
  centroids = []
  for i in range(K):
    centroids[i] = data[random_int(0, len(data))]
  return centroids
```

The mean distance measure to closest centroids for the best performing clustering for each value of K is shown in Figure 5.7.

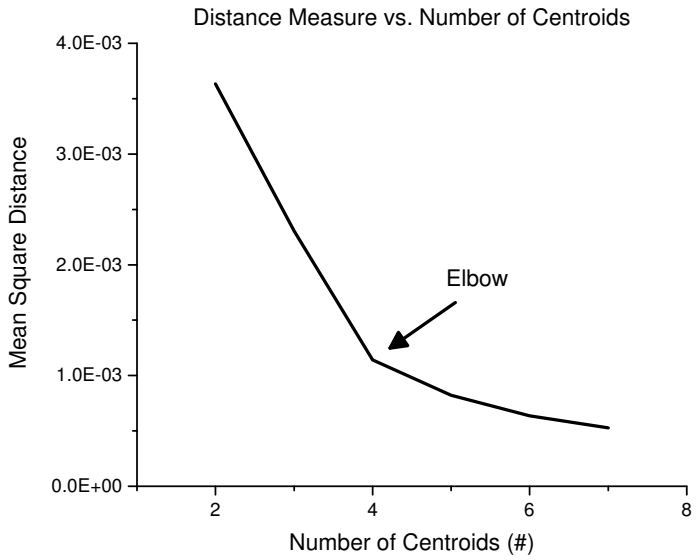


Figure 5.7. Mean distance measure of all data points to their nearest centroid vs. the number of centroids. The elbow indicates the optimum number of labels for clustering the data set.

The shape of Figure 5.7 is typical of K-means clustering applications, and the elbow indicated at $K = 4$ in the figure is the point at which we reach optimal clustering. Fit usually improves beyond this point, but this is typically due to over-fitting—a single distinct group may end up with multiple centroids, reducing the mean distance of data points within that group to their nearest centroid, but resulting in multiple identified classes for the same distinguishable class in the data set. With four clusters, it is easy to assign physical meaning to each cluster. The four clusters correspond to

operating conditions wherein the flame is attached, detached and stable, detached and flickering (local extinctions), and totally blown out. Note that thermoacoustic instabilities were either not observed, or were difficult to distinguish from local extinction conditions. This is likely due to insufficient power from the burner for exciting thermoacoustic instabilities in the chosen combustion chamber. The clustering of the data points is visualized in Figures 5.8 and 5.9, in which the higher-dimensional data has been projected onto a plane using the t-distributed stochastic neighbor embedding (t-SNE), which attempts to find the projection that minimizes the divergence between the probability distribution functions of the higher-order and lower-order data sets [64–66]. Both of these figures show projections of higher-dimensional data onto a two-dimensional manifold. For Figure 5.8, the kth operating condition is represented by a 12-feature vector comprising the label $t_k \in \{0, 1, 2, 3\}$, flame state $b_k \in \{0, 1\}$, static pressure $P_k \in \mathbb{R}$, current flow rates of the four mass flow controllers $\dot{\mathbf{m}}_k \in \mathbb{R}^4$, and mean power measured by each of the five microphones $\bar{\mathbf{v}} \in \mathbb{R}^5$. For Figure 5.9, the standard deviation of the power spectra measured by each of the five microphones $\boldsymbol{\sigma} \in \mathbb{R}^5$ is included as well, thus representing each operating condition as a 17-feature vector.

Figure 5.8 clearly shows the separation between operating conditions when represented as 12-feature vectors. Here, each operating mode is grouped into a distinct region of the graph, with the exception of blow-out states which are intermixed with attached and flickering modes. This matches the observation that blow-out typically occurred after the flame began to flicker and die, or when the flame was attached and air flow rates increased to the point that the flame was extinguished. Furthermore, Figure 5.8 shows flickering conditions situated between attached and detached modes; often the transition between these two modes involved intermittent reattachment of the flame which has similar characteristics to a detached flickering flame.

From Figure 5.9 we see that the t-SNE algorithm does not differentiate between stable 17-feature operating conditions, which makes some sense as attached and detached operating conditions both occur throughout the operating space of the com-

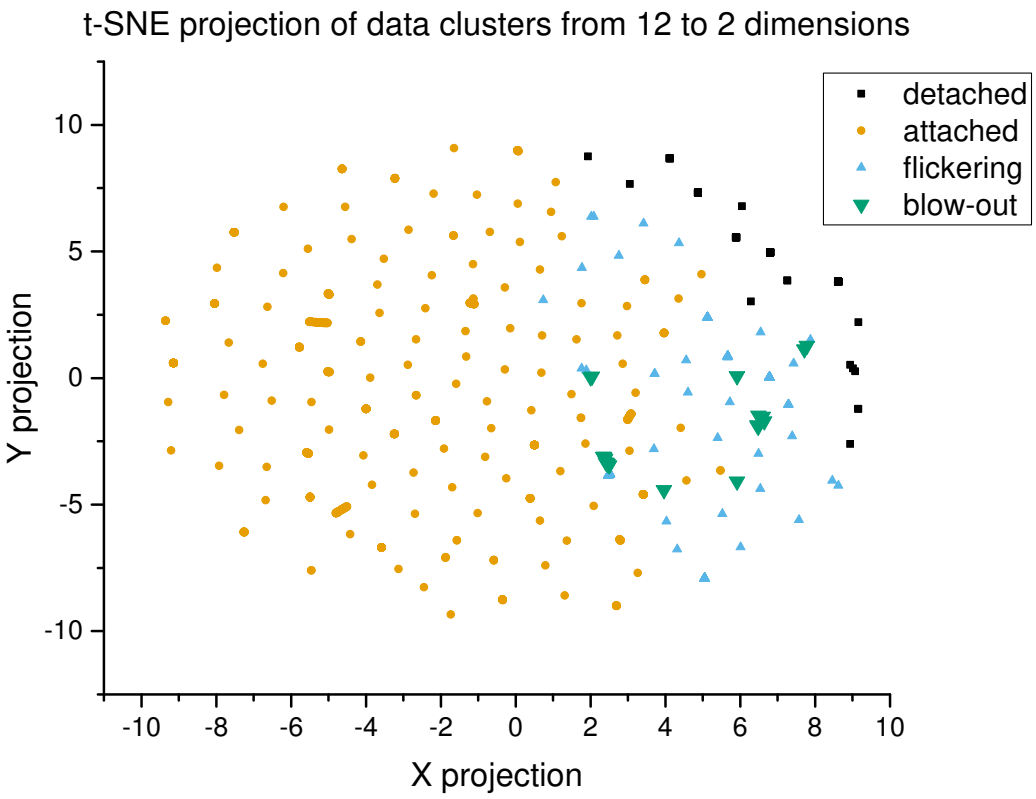


Figure 5.8. t-SNE projection of labeled 12-dimensional data points onto two dimensions. In this projection, operating modes are more clearly separated, with blow-outs intermixed with the flickering and attached conditions.

bustor and do not seem to be confined to specific regions. Blow-out conditions seem to form a distinct group on the right-hand side of Figure 5.9, and flickering states tend to aggregate towards this group. This matches the trend that the flame often begins to detach and flicker before blowing out. Flickering states are also seen intermixed with attached and detached flames, likely corresponding to those conditions where the flame is transitioning between attached and detached, and thus intermittently attaching.

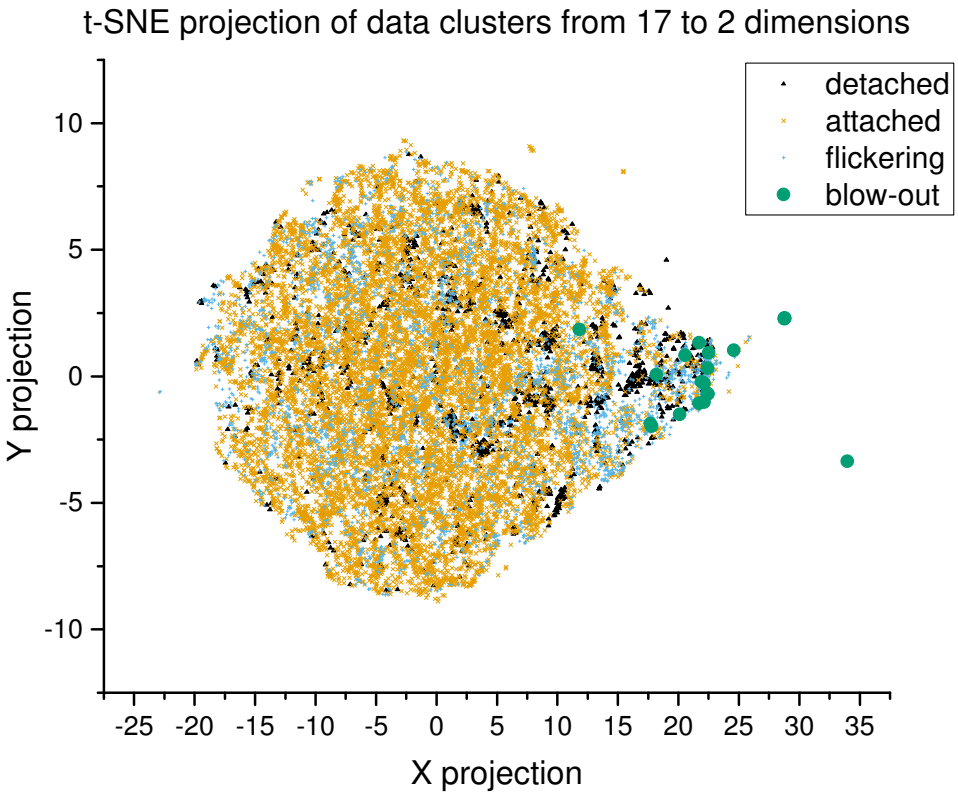


Figure 5.9. t-SNE projection of labeled 17-dimensional data points onto two dimensions. Note that operating modes under this projection are generally intermixed, but blow-out conditions tend toward one corner of the projection, with flickering conditions aggregating closer to blow-outs.

6. IDENTIFYING OBSTACLES IN THE OPERATING SPACE

In order to effectively plan trajectories between desired operating conditions while avoiding regions of instability or blow out, we must develop a map of the operating space using our experimental data. To differentiate regions of desirable performance in the operating space of the combustor from regions of instability or blow out, we first label our training data set as in Section 5.3. A data-driven classifier can then be trained using this labeled data set to predict the boundaries between different combustion modes in the operating space of the combustor.

6.1 Classification Algorithms

The problem of identifying desirable and undesirable regions within the operating space of the combustion process from experimental data can be reduced to a binary classification problem once the experimental data has been labeled. N labels can be grouped into two categories corresponding to desirable and undesirable conditions, thus binary classification. A decision boundary can then be modeled using any number of binary classification algorithms such as support vector machines (SVM), extreme learning machines (ELM), or fuzzy learning machines (FLM) [67]. This decision boundary can be used to identify regions that should be avoided by the controller (obstacles), and to predict the classification of new operating conditions online as the system runs.

6.1.1 Support Vector Machine

The support vector machine and its variants [68–70] have been used extensively for binary and multi-class classification problems. SVM has two major learning features:

1. The training data are first mapped onto a higher-order feature space through some nonlinear mapping function $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x}) \colon \mathbb{R}^n \to \mathbb{R}^\nu$ with $\nu > n$.

2. An optimization method is then employed to maximize the linear separation margin between two different categories within this feature space while minimizing misclassification error [68, 71].

SVMs have been applied to flame quality classification in industrial boilers [72] and a least-squares SVM was applied to modeling bio-diesel engine performance and emission characteristics [73].

If the hyper-plane separating two different data categories in ν-dimensional feature space is $\mathbf{w}_0 \cdot \mathbf{z} + b_0 = 0$, it has been shown that the weights $\mathbf{w}_0$ for the optimal hyper-plane in the feature space can be written as a sum of a limited number of the feature vectors, $\mathbf{z}_i$, called the support vectors (6.1).

$$\mathbf{w}_0 = \sum_{\text{support vectors}} t_i \alpha_i \mathbf{z}_i \tag{6.1}$$

The decision boundary of the binary SVM classifier can then be expressed as (6.2), wherein $\hat{t}$ is the predicted label of the transformed operating condition $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{z}_i \cdot \mathbf{z}$ is the dot product between support vector $\mathbf{z}_i$ and vector in feature space to be classified $\mathbf{z}$. More generally, the dot product in (6.2) can be replaced by other kernel functions, $K\big(\mathbf{z}, \mathbf{z}_i\big)$, like the radial basis function $K\big(\mathbf{z}, \mathbf{z}_i\big) = \exp\Big\{ -\frac{|\mathbf{z}-\mathbf{z_i}|^2}{\sigma^2} \Big\}$.

$$\hat{t} = f(\mathbf{z}) = \operatorname{sign}\left(\sum_{\text{support vectors}} t_i \alpha_i \mathbf{z}_i \cdot \mathbf{z} + b_0 \right) \tag{6.2}$$

Given the training set $\aleph = \big\{ (t_i,\ \mathbf{z}_i) \mid t_i \in \{-1, 1\},\ \mathbf{z}_i \in \mathbb{R}^\nu,\ i = 1, \ldots, l \big\}$, the hyper-plane that maximizes the separation between classes while minimizing misclassification error can be found by minimizing (6.3) subject to the constraints (6.4) [68].

$$\frac{\mathbf{w}^2}{2} + cF\left(\sum_{i=1}^{l} \xi_i \right) \tag{6.3}$$

$$\begin{aligned} t_i\big(\mathbf{w} \cdot \mathbf{z}_i + \mathbf{b}\big) &\geq 1 - \xi_i \qquad & i = 1, \ldots, l \\ \xi_i &\geq 0 \qquad & i = 1, \ldots, l \end{aligned} \tag{6.4}$$

In (6.3), c is a constant, $F(u)$ is a monotonically convex function, e.g. $F(u) = u^2$, and ξ_i are slack variables used to allow for some misclassification for systems that are not completely linearly separable in the feature space.

The computational complexity of SVM algorithms is usually at least quadratic with the number of training examples. The traditional SVM therefore does not scale well to complex problems with large training sets. The least-squares SVM (LS-SVM) [70] and proximal SVM (PSVM) [69] provide fast implementations of the traditional SVM by utilizing equality constraints to result in a least-squares training solution and avoid quadratic programming [71].

6.1.2 Extreme Learning Machine

Extreme learning machines (ELM) are a class of single-hidden-layer feed-forward neural networks (SLFN) wherein the input weights are randomly assigned and the output weights and hidden layer parameters are determined analytically using a least-squares approach [71, 74–76]. This training approach gives ELMs the advantage of being extremely fast to train compared with systems that require numeric optimization algorithms. SLFNs are capable of approximating any continuous function [77] and implementing any classification application [78]. Thus the ELM, a class of SLFN, can be used generally for classification and continuous function approximation applications [71, 74]. ELMs have been used to model the stable operating envelope of an unstable homogeneous charge compression ignition (HCCI) engine [67], for predicting combustion phasing in HCCI engines [79], for modeling bio-diesel engine performance and emission characteristics [73], and for many other applications.

The decision boundary of a binary ELM classifier takes the form (6.5), wherein $\hat{t}$ is the predicted label for input $\mathbf{x}$, $\mathbf{h}(\mathbf{x})$ is the hidden-layer output corresponding to the input to the ELM $\mathbf{x}$, and $\mathcal{B}$ is the output weight matrix between the hidden layer and the ELM output layer.

$$\hat{t} = \text{sign}\big(f(\mathbf{x})\big) = \text{sign}\big(\mathbf{h}(\mathbf{x})\mathcal{B}\big) \tag{6.5}$$

Given a training set $\aleph = \big\{(\mathbf{t}_i,\ \mathbf{x}_i) \mid \mathbf{t}_i \in \mathbb{R}^m,\ \mathbf{x}_i \in \mathbb{R}^n,\ i = 1, \ldots, N\big\}$ where $\mathbf{t}_i$ are the outputs (labels) associated with the training inputs, and with activation function $h(\mathbf{x})\colon \mathbb{R}^n \to \mathbb{R}$ and L hidden neurons, the output weight matrix $\mathcal{B}$ that minimizes training error and has minimum norm can be found using (6.6).

$$\hat{\mathcal{B}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{6.6}$$

In (6.6) the matrix $\mathbf{H}$ is an $N \times L$ matrix of the hidden layer activation functions (6.7) for N training samples and $L \leq N$ hidden neurons. The matrix $\mathbf{T}$ is an $N \times m$ matrix of the training output data (6.8) for an m-dimensional output vector. The output weight matrix $\hat{\mathcal{B}} \in \mathbb{R}^{L \times m}$. In the case of a binary classifier, $\mathbf{t}_i = t_i \in \{-1,\ 1\}$ is the label associated with each training input $\mathbf{x}_i$. The expression $\mathbf{H}^{\dagger} = \big(\mathbf{H}^H\mathbf{H}\big)^{-1}\mathbf{H}^H$ is the Moore-Penrose pseudo-inverse of $\mathbf{H}$ [80], and $\mathbf{H}^H$ is the conjugate transpose of $\mathbf{H}$.

$$\mathbf{H} = \begin{bmatrix} h\big(\mathbf{w}_1\mathbf{x}_1 + b_1, \zeta_1\big) & \cdots & h\big(\mathbf{w}_L\mathbf{x}_1 + b_L, \zeta_L\big) \\ \vdots & \cdots & \vdots \\ h\big(\mathbf{w}_1\mathbf{x}_N + b_1, \zeta_1\big) & \cdots & h\big(\mathbf{w}_L\mathbf{x}_N + b_L, \zeta_L\big) \end{bmatrix}_{N \times L} \tag{6.7}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} \tag{6.8}$$

The training algorithm for an ELM given the training set $\aleph$ described previously can be summarized as:

1. Randomly assign input weight vectors $\mathbf{w}_i$, input weight scalars b_i, and hidden layer parameters $\boldsymbol{\zeta}_i$ for $i = 1, \ldots, L$.

2. Calculate the hidden layer output matrix $\mathbf{H}$ (6.7).

3. Calculate the minimum-norm least squares output weight vector $\hat{\mathcal{B}}$ using (6.6).

6.1.3 Fuzzy Classification

For systems where uncertainty may exist as to which category a given data point falls into, fuzzy learning machines such as the fuzzy SVM (FSVM) [81] and fuzzy ELM (FELM) [82, 83] have been developed. The FSVM and FELM apply a fuzzy membership function to each input data point, allowing different inputs to make different contributions to the overall learning of the decision boundary. In this way the decision boundary can account for uncertainty in the membership of the training data in each category.

The FELM method applies the ELM approach to a fuzzy inference system (FIS). An SLFN is created with L fuzzy rules as its hidden nodes. The parameters of the hidden nodes are randomly initialized, and the output is the inner product of the firing strength of these rules with an output weight matrix $\mathbf{B}$ (for vector output $\mathbf{y}$), or output weight vector $\boldsymbol{\zeta}$ (for scalar output t). This network is illustrated in Figure 6.1.

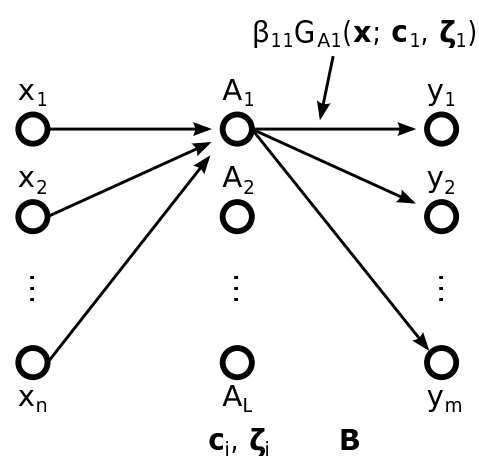


Figure 6.1. Fuzzy extreme learning machine network illustration.

Triangular membership functions like that shown in Figure 6.2 are commonly used in FIS [84]. The ith membership function is parameterized by its center for the jth input variable $c_{j,i}$, and by its left and right bounds $\boldsymbol{\zeta}_{j,i} = \begin{bmatrix} a_{j,i} & b_{j,i} \end{bmatrix}$.

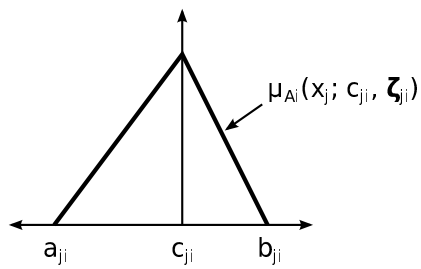


Figure 6.2. Triangular fuzzy membership function.

The membership value of $\mu_{A_{j,i}}(x_j)$ of the jth input value and the ith triangular fuzzy rule is given by (6.9):

$$\mu_{A_{j,i}}\big(x_j;\ c_{j,i},\ \boldsymbol{\zeta}_{j,i}\big) = \begin{cases} \frac{a_{j,i}-x_j}{a_{j,i}-c_{j,i}} & a_{j,i} \leq x_j \leq c_{j,i} \\ \frac{b_{j,i}-x_j}{b_{j,i}-c_{j,i}} & c_{j,i} < x_j \leq b_{j,i} \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

The firing strength of the ith rule to the input $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^n$ is then given by (6.10).

$$\begin{aligned} R_i\big(\mathbf{x};\ \mathbf{c}_i,\ \boldsymbol{Z}_i\big) = \mu_{A_{1,i}}\big(x_1;\ c_{1,i},\ \boldsymbol{\zeta}_{1,i}\big) \otimes \mu_{A_{2,i}}\big(x_2;\ c_{2,i},\ \boldsymbol{\zeta}_{2,i}\big) \otimes \cdots \\ \otimes\, \mu_{A_{n,i}}\big(x_n;\ c_{n,i},\ \boldsymbol{\zeta}_{n,i}\big) \end{aligned} \tag{6.10}$$

In (6.10), the $\otimes$ operator represents a fuzzy `AND` operation: $\mu_1 \otimes \mu_2 = \min(\mu_1, \mu_2)$ [84]. The matrix $\mathbf{Z}_i = \begin{bmatrix} \boldsymbol{\zeta}_{1,i} & \boldsymbol{\zeta}_{2,i} & \cdots & \boldsymbol{\zeta}_{n,i} \end{bmatrix}$ contains the parameters of the ith membership function for each of the n inputs. The firing strength of each of the L fuzzy rules can be normalized by (6.11).

$$G_i\big(\mathbf{x};\ \mathbf{c}_i,\ \mathbf{Z}_i\big) = \frac{R_i\big(\mathbf{x};\ \mathbf{c}_i,\ \mathbf{Z}_i\big)}{\sum_{i=1}^{L} R_i\big(\mathbf{x};\ \mathbf{c}_i,\ \mathbf{Z}_i\big)} \tag{6.11}$$

The label $\hat{t}$ for an input $\mathbf{x}$ predicted by the fuzzy model is then given by the sign of the inner product of the normalized firing strengths of each of the L fuzzy rules with an output weight vector $\boldsymbol{\beta} \in \mathbb{R}^L$ (6.12).

$$\begin{aligned} \hat{t} &= \mathbf{G}_A \boldsymbol{\beta} \\ \mathbf{G}_A &= \begin{bmatrix} G_1(\mathbf{x};\ \mathbf{c}_1,\ \mathbf{Z}_1) & \cdots & G_L(\mathbf{x};\ \mathbf{c}_L,\ \mathbf{Z}_L) \end{bmatrix} \in \mathbb{R}^L \end{aligned} \tag{6.12}$$

The decision boundary of the FELM classifier is then given by the set of points $\mathbb{X}_B$ where the decision function (6.12) equals zero (6.13).

$$\mathbb{X}_B = \left\{ \mathbf{x} \mid \mathbf{G}_A \boldsymbol{\beta} = 0 \right\} \tag{6.13}$$

The FELM training algorithm is a direct extension of the ELM algorithm to FIS:

1. Randomly assign fuzzy membership function parameters $c_{j,i}$ and $\boldsymbol{\zeta}_{j,i}$ for each of the n inputs and L fuzzy rules.

2. Calculate the normalized firing strength G_i of each of the fuzzy rules (6.11).

3. Calculate the minimum-norm least squares output weight vector $\hat{\boldsymbol{\beta}}$ using an approach similar to (6.6).

6.1.4 Multi-Layer Feed-Forward Neural Networks

With recent advancements in high-efficiency backwards propagation techniques, multi-layer feed-forward neural networks and convolutional neural networks—wherein all interconnecting weights and biases are trained—have become quite popular [62]. The extreme learning machine (Section 6.1.2) and fuzzy learning machine (Section 6.1.3) are examples of single-hidden-layer feed-forward networks in which the input weights and biases are untrained. A fully-trained multi-layer network generalizes this concept to include multiple hidden layers while training all weight matrices and bias vectors. Any nonlinear differentiable function can be used as the activation function for hidden neurons, but some notable functions include the rectified linear unit (relu), whose activation $r(x)$ is given by (6.14), the sigmoid function (6.15), and the hyperbolic tangent function, $\tanh(x)$.

$$r(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{6.14}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.15}$$

For a network with M hidden layers, the activation (output) of layer $k-1$ can be denoted $\mathbf{h}_{k-1} \in \mathbb{R}^{n_{k-1}}$. The activation of layer k can then be determined by (6.16), where $\mathbf{W}_k$ is an $n_k \times n_{k-1}$ matrix of trainable weights connecting layer $k-1$ to layer k, $\mathbf{b}_k \in \mathbb{R}^{n_k}$ is a vector of trainable biases, and $f_k(\cdot)$ is the nonlinear activation function for layer k. The first layer of a network is the input layer, thus $\mathbf{h}_0 = \mathbf{x}$. The final layer of the network is the output layer; thus with M hidden layers, the output $\hat{\mathbf{y}}$ is given by (6.17) where $\mathbf{W}_O \in \mathbb{R}^{m \times n_M}$ and $\mathbf{b}_O \in \mathbb{R}^m$ are the output weights and biases respectively, and $\mathbf{h}_M$ is the final hidden layer activation of the network. This network structure is illustrated in Figure 6.3.

$$\mathbf{h}_k = f_k\big(\mathbf{W}_k \mathbf{h}_{k-1} + \mathbf{b}_k\big) \tag{6.16}$$

$$\hat{\mathbf{y}} = \mathbf{W}_O \mathbf{h}_M + \mathbf{b}_O \tag{6.17}$$

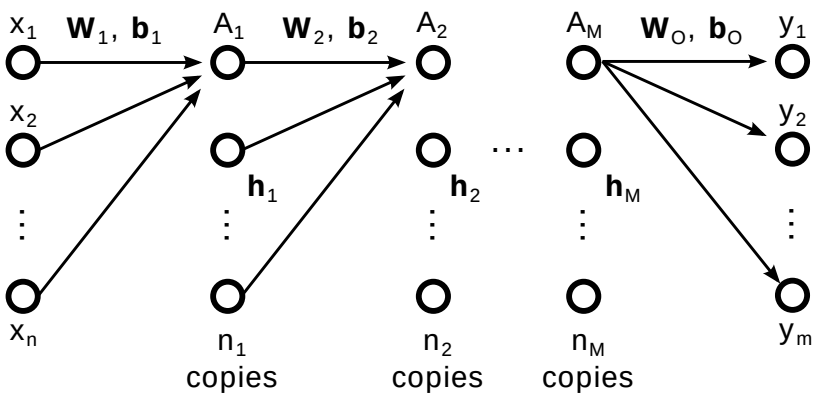


Figure 6.3. Multi-layer feed-forward neural network structure with M hidden layers where each hidden layer is defined by its activation function A_i, and is connected to the previous layer by weights $\mathbf{W}_i$ and biases $\mathbf{b}_i$.

For classification applications, we represent our data labels as one-hot encoded vectors, which for m labels have m elements that are all zero except for the element corresponding to the appropriate label, which is one (6.18). We then want to output a probability distribution representing our confidence in how well the input vector

matches each of our possible output categories. To convert the output vector $\hat{\mathbf{y}}$ to a probability distribution function $\tilde{\mathbf{y}}$, the softmax function (6.19) is used.

$$\mathbf{y}_k^i = \begin{cases} 1 & i = t_k \\ 0 & \text{otherwise} \end{cases} \tag{6.18}$$

$$\tilde{y}_i = \gamma(\hat{\mathbf{y}})_i = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{m} e^{\hat{y}_j}} \tag{6.19}$$

To train the network, we first define a performance metric J that measures how well our network is predicting our training data. A common choice of performance metric is the mean Kullback-Leibler divergence (KL-divergence) between the predictions and the one-hot training labels (6.20), where $\ln(\cdot)$ is typically either the base-2 logarithm or natural logarithm of the argument (natural logarithm for this discussion) [57,58]. The KL-divergence between two probability distribution functions is at a minimum $H(\mathbf{p}, \mathbf{q}) = 0$ when $\mathbf{p} = \mathbf{q}$.

$$H(\mathbf{y}_k, \tilde{\mathbf{y}}_k) = \sum_{i=1}^{m} y_k^i \ln \left(\frac{y}{\tilde{y}} \right)_k^i \tag{6.20}$$

Note that for our system, the length of the output m is equal to the number of flame modes identified with our clustering algorithm (Section 5.3), K; thus $K = m$. Any back propagation algorithm can then be used to perform a stochastic gradient descent training process to determine the network weights $\mathbf{W}_i$ and biases $\mathbf{b}_i$ with $i = 0, 1, \ldots, N$ that minimize the training loss (6.20). Stochastic gradient descent involves running many iterations of back propagation training on small training batches taken from the overall training set. This allows for much faster iterations of back propagation that, in the aggregate, converge to a minimum cost.

6.2 Application to the Combustor System

Using the operating point classification discussed in Section 5.3, training data point $\mathbf{x}_i$ can be associated with a label t_i depending on whether its existing label

from K-means classification associates it with a blow out or otherwise undesirable operating mode (6.21).

$$t_i = \begin{cases} 0 & \text{undesirable mode} \\ 1 & \text{desirable mode} \end{cases} \tag{6.21}$$

Data labels were combined such that detached, stable flames were considered "good" and flickering flames, blow-out, and attached flames were all considered "bad". This ensured that there was roughly the same number of good and bad data points, and allows us to attempt to train a classifier that can be used as a map to move through the combustor's operating space while maintaining a detached stable flame. This label was then converted to a two-element one-hot vector (6.18).

Several classifiers were trained using Tensorflow [62] with various structures. Given the fact that the classifier is to be used as a map for planning a trajectory through the operating space of the combustor, only information about the space that we can know *a priori* can be used in training this map. Thus, the inputs to the classifier $\mathbf{x}_k \in \mathbb{R}^4$ were solely the flow rates to each of the four MFCs, and the outputs of the classifier were the estimated probabilities that each input condition will be "good" or "bad". The model performance was sensitive to the initial conditions of the model parameters, and so a Monte-Carlo training approach was used to select the best model out of many training sessions.

The structure that performed best was a two-hidden-layer neural network with 256 hidden nodes in each layer, and all elements fully connected. The first hidden layer was activated with the relu function (6.14), and the second hidden layer was activated with the hyperbolic tangent function $\tanh(\mathbf{x})$. Weight matrices and biases connecting the layers were trained by gradient descent using the adaptive momentum estimate (Adam) optimizer [85]. The Adam gradient descent algorithm takes a variable-sized step in the direction of the gradient of the cost function with respect to the parameters for each training batch, with step size dependent on estimates of the first- and second-order moments of the gradient. If we define our parameters as a vector $\boldsymbol{\theta}$, and given a cost function $J(\boldsymbol{\theta})$, the estimated first- and second-order moments of the gradient

of the cost function at step k are (6.22) and (6.23) respectively. Here $\mathbf{g}_k = \nabla J(\boldsymbol{\theta})$ is the gradient of the cost function with respect to the model parameters, $\boldsymbol{\theta}$, and β_1 and β_2 are optimizer parameters that can be tuned, but are typically chosen to be close to 1.

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1)\mathbf{g}_k \tag{6.22}$$

$$\mathbf{v}_k = \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2)\mathbf{g}_k^2 \tag{6.23}$$

The authors note that the estimates of the first- and second-order moments of the gradients are biased toward zero when initialized as vectors of zeros. They counteract this bias by finding a bias-corrected estimate of the first- and second-order moments of the gradients: (6.24) and (6.25) respectively.

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{1 - \beta_1^k} \tag{6.24}$$

$$\hat{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1 - \beta_2^k} \tag{6.25}$$

The parameter vector is then updated according to the Adam update law (6.26), where α is the learning rate of the algorithm, and ϵ is another parameter that can be tuned. The variable-sized step through the parameter space is given by $\Delta\boldsymbol{\theta}_k = \frac{\alpha}{\sqrt{\hat{v}_k}+\epsilon}\hat{\mathbf{m}}_k$ in (6.26).

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\alpha}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon}\hat{\mathbf{m}}_k \tag{6.26}$$

The authors propose default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. A typical value for the learning rate is $\alpha = 0.001$. Model parameters ($\boldsymbol{\theta}$ in the preceding) were the weight matrices and biases connecting layers of the network and were defined as $\mathbf{W}_1 \in \mathbb{R}^{256\times 4}$, $\mathbf{b}_1 \in \mathbb{R}^{256}$, $\mathbf{W}_2 \in \mathbb{R}^{256\times 256}$, $\mathbf{b}_2 \in \mathbb{R}^{256}$, $\mathbf{W}_O \in \mathbb{R}^{2\times 256}$, and $\mathbf{b}_O \in \mathbb{R}^2$. Weight matrices were initialized with Gaussian random numbers with zero mean and standard deviation of $\sigma = \frac{1}{\sqrt{n}}$ for an $n \times m$ matrix. Bias vectors were initialized to zero. The output of the classifier was converted to a probability distribution function with the softmax function (6.19).

After training, the mean perplexity (6.27) of the model on the validation set was found to be $P(\mathbf{y}, \tilde{\mathbf{y}}) \approx 1.58$, corresponding to a validation classification accu-

racy (number of correct classifications divided by total number of data points in the validation set) of 91%. Perplexity is a measure of how well an approximation of a probability distribution predicts the actual distribution; the smaller the perplexity, the better the approximation predicts the actual probability distribution function. If the two distributions are equal, the KL-divergence (6.20) will be 0, and perplexity will be 1. For this data set, the perplexity indicates how well the model's classification match the actual data point's classification on average.

$$P(\mathbf{p}, \mathbf{q}) = \frac{1}{n} \sum_{k=1}^{n} e^{H(\mathbf{p}_k, \mathbf{q}_k)} \tag{6.27}$$

7. PREDICTING THE OPERATING MODE OF THE COMBUSTOR

Given the classifier map discussed in Chapter 6, we can attempt to plan a path through the operating space of the combustor while maintaining some operating mode and avoiding regions of instability and blow-out. This map, however, is imperfect, and only utilizes *a priori* information in its prediction, specifically the flow rates to each of the four MFCs. We would like to utilize *in situ* information about the system—including additional sensor readings and time-series history—as the combustor runs to improve our prediction of nearby performance and predict whether operating conditions that we are approaching will differ from the conditions predicted by the classifier. Furthermore, many path planning algorithms utilize "look-ahead" information about the space through which we are planning a path in order to improve the path as it is traversed and to avoid unforeseen obstacles [86–88]. Thus having a predictor may aid in planning trajectories through the operating space of the combustor in real time. Recurrent neural networks (RNN) are a good candidate for incorporating additional sensor information in real time while also maintaining some "memory" of what has happened in the system for informing predictions; see, for example, [89–92].

In general, an RNN can be viewed as a chain of identical network elements, each feeding its output into the next element in the chain. At a given time step, measurements from the system are fed into the RNN element, which outputs a prediction based on this input and the previous value of the RNN. This process is illustrated in Figure 7.1, where $\mathbf{x}_i$ is the ith input, $\mathbf{y}_i$ is the ith prediction, and A is the RNN element.

Sometimes our system only requires that we incorporate information from recent measurements in our prediction of future measurements. This is the case illustrated in Figure 7.1, where x_k only depends on the previous N results, and N is small. For some systems, much more context is needed to predict future labels. This requires

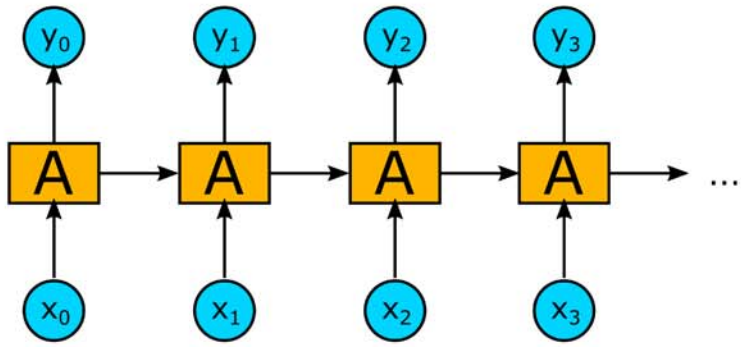


Figure 7.1. Recurrent neural network structure. Inputs come as a sequence of vectors $\mathbf{x}_i$, and the network outputs a result $\mathbf{y}_i$ for each input. The network then passes its output forward to the next iteration of the network, thus maintaining some memory. This can be viewed as a chain of identical network elements where each element passes its output to the next element in the chain.

that the network "remember" context for a longer time, and generally RNNs fail to learn these relationships [93]. This case is illustrated in Figure 7.2.

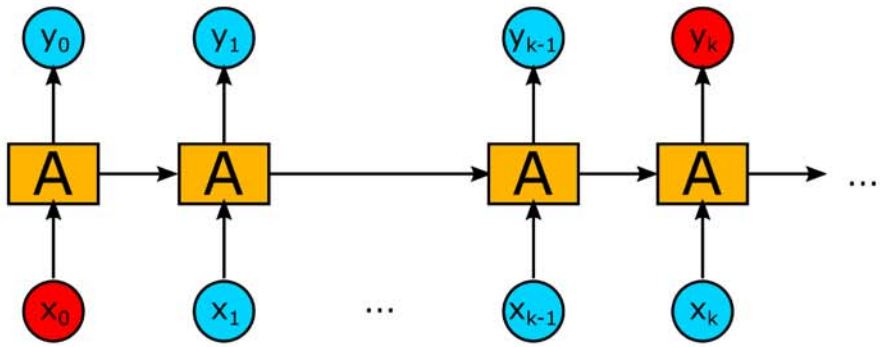


Figure 7.2. Recurrent neural network structure illustrating long-term dependency issue common to RNNs. A typical RNN structure would have difficulty in learning a long-term dependency between the red-shaded output $\mathbf{y}_k$ and a far-away previous input like the red-shaded $\mathbf{x}_0$.

The long short-term memory (LSTM) network is designed to circumvent the long-term dependency problem from which other RNNs suffer. LSTMs were first introduced in [94] and have since been expanded upon by many researchers [92, 95–98].

They are widely applicable to a large variety of problems that can be posed in terms of predicting elements in a sequence.

For typical RNNs, the network element is fairly simple: for example, a single tanh element. LSTMs possess the same chain structure as typical RNNs illustrated in Figure 7.1, but with a more complicated (and specifically designed) network cell. The LSTM maintains an internal state $\mathbf{h}_k \in \mathbb{R}^N$, and its output $\mathbf{g}_k \in \mathbb{R}^N$ and internal state are dependent on the current measurement $\mathbf{x}_k \in \mathbb{R}^n$, the previous output $\mathbf{g}_{k-1}$, and the previous state $\mathbf{h}_{k-1}$, all gated by sigmoid functions $\sigma(\alpha) = \frac{1}{1+e^{-\alpha}}$ as illustrated in Figure 7.3. Note that Figure 7.3 does not show all details of the LTSM node for the sake of illustration. The first step of the LSTM is to determine how much of its internal state to forget. The "forget gate" vector $\mathbf{f}_k$ at time step k is given by (7.1).

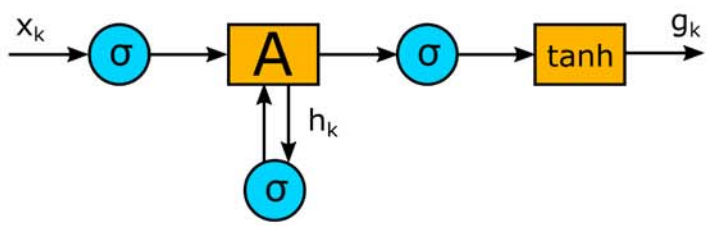


Figure 7.3. A single node of a long short-term memory network, illustrating the internal state and sigmoid gates on input, output, and state. Note that not all details of the LSTM node structure are shown here.

$$\mathbf{f}_k = \sigma\big(\mathbf{W}_f[\mathbf{g}_{k-1}, \mathbf{x}_k] + \mathbf{b}_f\big) \tag{7.1}$$

In (7.1) and subsequent equations, $[\mathbf{g}_{k-1}, \mathbf{x}_k] \in \mathbb{R}^{N+n}$ represents the concatenation of the previous network state vector with the current input vector. The next step is to determine how much new information to incorporate into the system state for the next iteration. The "input gate" vector $\mathbf{i}_k$ at time step k is given by (7.2), and an intermittent state vector $\tilde{\mathbf{h}}_k$ is given by (7.3).

$$\mathbf{i}_k = \sigma\big(\mathbf{W}_i[\mathbf{g}_{k-1}, \mathbf{x}_k] + \mathbf{b}_i\big) \tag{7.2}$$

$$\tilde{\mathbf{h}}_k = \tanh\big(\mathbf{W}_h[\mathbf{g}_{k-1}, \mathbf{x}_k] + \mathbf{b}_h\big) \tag{7.3}$$

The stored state is next updated as in (7.4), where the $*$ operator represents element-wise multiplication.

$$\mathbf{h}_k = \mathbf{f}_k * \mathbf{h}_{k-1} + \mathbf{i}_k * \tilde{\mathbf{h}}_k \tag{7.4}$$

The output of the LSTM element is then dependent on an "output gate" vector $\mathbf{o}_k$ (7.5) and the updated LSTM state $\mathbf{h}_k$ (7.6).

$$\mathbf{o}_k = \sigma\big(\mathbf{W}_o[\mathbf{g}_{k-1}, \mathbf{x}_k] + \mathbf{b}_o\big) \tag{7.5}$$

$$\mathbf{g}_k = \mathbf{o}_k * \tanh(\mathbf{h}_k) \tag{7.6}$$

The LSTM has a fairly broad parameter space comprising the weight matrices $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_h$, and $\mathbf{W}_o \in \mathbb{R}^{N\times(N+n)}$, as well as the bias vectors $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_h$, and $\mathbf{b}_o \in \mathbb{R}^N$. Finally, the prediction from the network $\hat{\mathbf{y}}_k \in \mathbb{R}^m$ is given by (7.7) where $\mathbf{W}_p \in \mathbb{R}^{m\times N}$ and $\mathbf{b}_p \in \mathbb{R}^m$.

$$\hat{\mathbf{y}}_k = \mathbf{W}_p\mathbf{g}_k + \mathbf{b}_p \tag{7.7}$$

A TensorFlow [62] script was written to train an LSTM network on experimental data using the Adam gradient descent optimization method [85]. The network state vector was given dimension $N = 64$, and the network was composed of ten layers. Input data used the same 11 features used to train K-means: flame state $b_k \in \{0, 1\}$, input conditions $\mathbf{x}_k \in \mathbb{R}^4$, static pressure $P_k \in \mathbb{R}$, and mean dynamic pressure power at each sensor $\bar{\mathbf{v}}_k \in \mathbb{R}^5$. The labeled experimental data set was divided into a training and validation set, with the last 30% of the data going into the validation set. Training labels were represented as one-hot vectors, i.e. vectors with as many elements as we have distinct training labels where all elements are zero except the ith element corresponding to label t_k (6.18). One-hot training labels can be viewed as a probability distribution function wherein only one bin has non-zero probability. The model performance was sensitive to the initial conditions of the model parameters, and so a Monte-Carlo training approach was used to select the best model out of many training sessions.

The prediction error of the network was determined by converting predicted labels $\hat{\mathbf{y}}_k$ into probabilities $\tilde{\mathbf{y}}_k$ with the softmax function (7.8) and then determining

the mean KL-divergence between the predictions and the one-hot training labels (6.20) [57, 58]. This is similar to the process used to train the classifier, discussed in Chapter 6.

$$\tilde{y}_k^i = \gamma(\hat{\mathbf{y}}_k)_i = \frac{e^{\hat{y}_k^i}}{\sum_{j=1}^m e^{\hat{y}_k^j}} \tag{7.8}$$

Note that for our system, the length of the output m is equal to the number of flame modes identified with our clustering algorithm (Section 5.3), K; thus $K = m = 4$. After training, the mean perplexity (6.27) of the model on the validation set was found to be $P(\mathbf{y}, \tilde{\mathbf{y}}) \approx 1.23$, corresponding to a validation prediction accuracy (number of correct predictions divided by the total number of data points in the validation set) of 92%. For this data set, the perplexity indicates how well the model's predicted labels match the actual data labels on average.

$$P(\mathbf{p}, \mathbf{q}) = \frac{1}{n} \sum_{k=1}^{n} e^{H(\mathbf{p}_k, \mathbf{q}_k)} \tag{7.9}$$

8. PATH PLANNING ALGORITHMS FOR COMBUSTOR CONTROL

We begin this section looking at path planning through a discrete operating space with costs associated with each operating point within the space. We assign high costs to unstable or blow-out conditions and low costs to stable conditions based on experimental data. This produces a discrete operating point map with costs associated with each measured operating condition. The problem of taking the system from an initial operating condition $\mathbf{x}_0 \in \mathbb{X} \subset \mathbb{R}^n$ to a desired operating condition $\mathbf{x}_d \in \mathbb{X}$ can be viewed as a path-planning problem in n-dimensional space. A path $\mathbf{x}_k = \mathbf{p}(k) \mid k = 0, \ldots, N$ may be designed to minimize the normalized discrete cost function $J_{\mathbf{p}}$ (8.1). By assigning high enough costs to conditions that have been identified as "bad", the minimum-cost path between two points will avoid these regions.

$$J_{\mathbf{p}} = \sum_{k=0}^{N} c\big(\mathbf{p}(k)\big) \tag{8.1}$$

In (8.1) $c(\cdot)$ represents a cost function $c\colon \mathbb{R}^n \to \mathbb{R}$ and N is the number of operating points in the path. For the operating point cost map method, $c(\mathbf{x})$ is simply the cost value associated with the condition $\mathbf{x}$. The Anytime Dynamic A* (ADA*) algorithm is an anytime replanning algorithm that can be used to find a path between $\mathbf{x}_k$ and $\mathbf{x}_d$ in real time, and to minimize the cost of this path as time allows. The ADA* algorithm comprises two main components:

1. Dynamic replanning in the presence of new information
2. Anytime path planning and optimization under time constraints

The first component, replanning, is the ability of the algorithm to incorporate new information into its optimal trajectory in an efficient manner. This is addressed by incorporating ideas from the D* [86] and D* lite [99] algorithms. These algorithms find an optimal path from a starting point $\mathbf{x}_0$ to a desired point $\mathbf{x}_d$ by maintaining

an estimate $J(\mathbf{x})$ of the cost from each operating condition to the goal condition. It also stores a one-step look-ahead cost $J_{\text{next}}(\mathbf{x})$ which satisfies (8.2).

$$J_{\text{next}}(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} = \mathbf{x}_d \\ \min_{\mathbf{x}_{\text{next}} \in \text{Succ}(\mathbf{x})} \Big(c(\mathbf{x}) + c(\mathbf{x}_{\text{next}}) + J(\mathbf{x}_{\text{next}}) \Big) & \mathbf{x} \neq \mathbf{x}_d \end{cases} \tag{8.2}$$

In (8.2), $\text{Succ}(\mathbf{x}) \subset \mathbb{X}$ are the operating points neighboring $\mathbf{x}$. A point is said to be consistent if $J(\mathbf{x}) = J_{\text{next}}(\mathbf{x})$, otherwise it is over-consistent if $J > J_{\text{next}}$, or under-consistent if $J < J_{\text{next}}$. The algorithm uses a heuristic and a priority queue "`OPEN`" to focus its search efficiently. The heuristic $h(\mathbf{x}, \mathbf{x}')$ estimates the cost of an optimal path between points $\mathbf{x}$ and $\mathbf{x}'$, and must be less than or equal to the actual least-cost path between points $\mathbf{x}$ and $\mathbf{x}'$. A good candidate for the heuristic is the Euclidean distance in n-dimensional space between $\mathbf{x}$ and $\mathbf{x}'$ (8.3). The priority queue `OPEN` always holds the inconsistent points, which need to be updated and made consistent.

$$h(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}')} \tag{8.3}$$

Operating points in the `OPEN` set are assigned a pair of key values, `key[0]` and `key[1]` based on their cost values J and J_{next}, and their heuristic $h(\mathbf{x}_0, \mathbf{x})$. Details about the key values are given in the following. If the cost of any point changes, the algorithm updates the values of J_{next} of any points affected by the cost change, and then places all inconsistent points on the priority queue. It then proceeds to update points on the queue in increasing priority until the queue is empty. Due to the design of the key value of the queue and the inclusion of the heuristic, the algorithm ensures that points along the path are processed efficiently, and that points that do not matter for the path are ignored. When operating point costs decrease, the key value `key[0]` ensures that only those points that are over-consistent and could potentially decrease the overall path cost will be processed. When operating point costs increase, the algorithm ensures that only those points that are under-consistent and affect the cost of the current path will be processed. Upon completion, the algorithm will return the optimal path between $\mathbf{x}_0$ and $\mathbf{x}_d$.

The second component of ADA^*, anytime planning, is the ability of the algorithm to provide a sub-optimal trajectory that successfully navigates between desired points (if feasible) at any time, using extra time to optimize this trajectory. This is addressed by incorporating ideas from the Anytime Repairing A* (ARA*) algorithm [100]. ARA* takes advantage of the idea of consistency discussed previously, but multiplies the heuristic $h(\mathbf{x}, \mathbf{x}')$ by an inflation factor $\epsilon \geq 1$. This means that the priority of any operating condition in the `OPEN` queue is dependent on the inflation factor ϵ. The algorithm begins by doing an A* search using a large initial inflation factor $\epsilon_0 >> 1$, but only visiting each point in the operating space once. If a point becomes inconsistent, rather than placing it back in the `OPEN` set, it is placed in a set of inconsistent points, "`INCONS`". On the next iteration of the algorithm, the points in the set `INCONS` are placed into `OPEN`, the inflation factor is reduced $\epsilon \leftarrow \epsilon - \delta$, and the process is repeated. By only visiting each point once in a search, the search is completed very quickly. Furthermore, by only reconsidering points from the previous search that became inconsistent, much of the previous search effort can be reused.

The ADA* algorithm combines these two into a single algorithm. Its main function is given in pseudo-code in the following. If no changes in operating point cost are detected, the main function plans an initial sub-optimal path, and then begins decreasing ϵ and updating the path until an optimal solution is found ($\epsilon = 1$). This is exactly the same as the ARA* algorithm. If point costs change significantly, however, then the current solution may no longer be good, and it may be costly to repair it completely. In these cases, the value of ϵ is increased so that a less optimal solution can be obtained quickly [101]:

```
def main():
  Initialize costs
  Insert desired point x_d into OPEN
  path = compute_or_improve_path()
  fork(controller())
  while x != x_d:
```

```
    if point cost has changed:
      for x in points_with_new_cost:
        Update cost(x)
        update_point(x)
      if change in cost is large:
        Increase eps or re-plan from scratch
    elif eps > 1:
      eps -= delta

    OPEN.update(INCONS)  # add INCONS points to OPEN set
    INCONS.clear()  # empty these sets
    CLOSED.clear()
    path = compute_or_improve_path()

    if eps == 1:
      wait for changes in point costs
    x = read_current_point()
```

The line `fork(controller())` calls some control function that seeks to maintain the system on the `path` generated by the path planning algorithm. The `while` loop inside of the `main()` function continually improves the optimality of the path while incorporating new information until the current operating condition matches the desired condition. The functions `compute_or_improve_path()` and `update_point(x)` are given in pseudo-code below:

```
def compute_or_improve_path():
  while min(key(s in OPEN)) < key(x_0) or J_next(x_0) != J(x_0):
    x = OPEN.remove_min()  # get point with minimum key from OPEN
    if J(x) > J_next(x):
      J(x) = J_next(x)
      CLOSED.add(x)
```

```
      update_point(x_prev) for x_prev in Predecessors(x)
    else:
      J(x) = Inf
      update_point(x)
      update_point(x_prev) for x_prev in Predecessors(x)

def update_point(point x):
  if x was not visited before:
    J(x) = Inf
  if x != x_d:
    # Find closest neighbor.
    J_next(x) = min(cost(x) + cost(x_next) + J(x_next))
  if x in OPEN:
    OPEN.remove(x)
  if J(x) != J_next(x):
    if x not in CLOSED:
      OPEN.add(x)
    else:
      INCONS.add(x)
```

Because changes in operating point costs may cause some points to become under-consistent, points need to be inserted into the `OPEN` set with a key value reflecting the minimum of their old cost and their new cost. Furthermore, under-consistent points need to use costs with un-inflated heuristics. The `key(x)` function is given in pseudo-code below. Note that a list of two values is returned by `key(x)`. Operating point $\mathbf{x}$ is said to have a smaller key than point $\mathbf{x}'$ if `key(x)[0] < key(x')[0]` or if `key(x)[0] == key(x')[0]` *and* `key(x)[1] < key(x')[1]`.

```
def key(point x):
  if J(x) > J_next(x):
    return [J_next(x) + eps*h(x_0, x); J_next(x)]
```

```
else:
  return [J(x) + h(x_0, x); J(x)]
```

A UML activity diagram illustrating the ADA* path planning algorithm [101] is shown in Figure 8.1. Boxes in this diagram represent actions taken by the algorithm with arrows indicating the flow of activity through the diagram. Diamonds represent conditional branches with each arrow coming out of the diamond labeled with that arrow's condition.

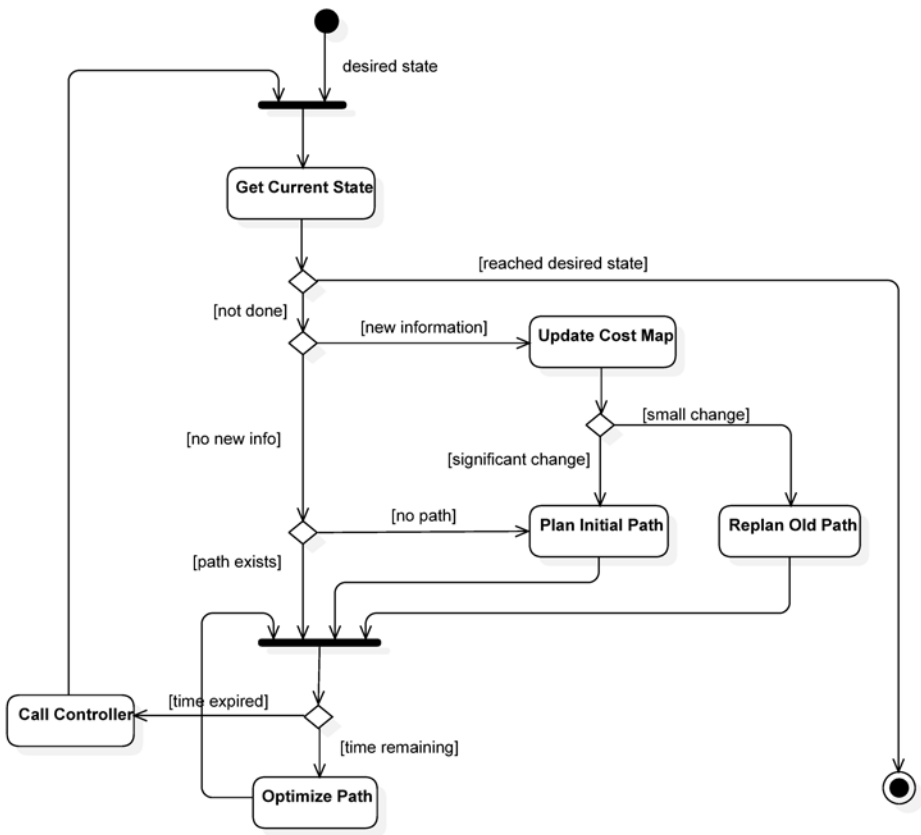


Figure 8.1. Path planner activity diagram.

Starting from the top of Figure 8.1, the ADA* algorithm determines the current operating condition and compares it with the desired condition. If the system has reached the desired condition, the algorithm stops planning; otherwise the algorithm checks whether new information has been incorporated into the operating point cost

model. If no new information is present, it either plans an initial sub-optimal path to the final condition, or spends a cycle optimizing the current path. If there is still time, the optimization process repeats, otherwise the algorithm calls the controller to follow the path.

If new operating point cost information is present (e.g., the system detects an unexpected instability and needs to update its operating point cost map accordingly, or the system predicts unexpected behavior in the future), the algorithm updates the operating point cost map. A heuristic factor is then used to determine whether the updated operating point cost map has a "significant" enough effect on the path cost. If the path cost changes significantly, the algorithm plans a new sub-optimal path from scratch and begins optimizing again; otherwise the algorithm re-plans the existing path, utilizing as much of the existing optimized path as possible.

Existing implementations of the ADA* and similar algorithms focus on path planning for autonomous ground and aerial vehicles, where on-board sensors can be used to look ahead and update information about the space around the vehicle [99, 102–106]. For the combustor control application, the sensors on the system can only measure the current operating condition, and so the algorithm is unable to take advantage of look-ahead information. For this reason, the predictor discussed in Chapter 7 is incorporated into the path planning algorithm to provide look-ahead predictions of the stability (cost) of future operating conditions along the path.

Additional considerations must be made for path planning with obstacles defined by a classifier as discussed in Chapter 6. In this case, the operating space is no longer a directed graph with costs associated with each move through the space, but a continuous space with mathematically-defined boundaries, $\mathbf{x}_B$, between conditions identified as "good" and "bad" given by (8.4) for the ELM and (8.5) for the FELM. Note that we will treat the MLFN separately in the following.

$$\mathbf{x}_B \in \mathbb{X}_B = \{\mathbf{x} \mid \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = 0\} \tag{8.4}$$

$$\mathbf{x}_B \in \mathbb{X}_B = \{\mathbf{x} \mid \mathbf{G}(\mathbf{x}; \mathbf{c}, \mathbf{a})\boldsymbol{\beta} = 0\} \tag{8.5}$$

The ADA* algorithm builds a path sequentially in steps. This can be applied to a continuous space with obstacles defined by an ELM classifier by minimizing some cost function taking steps $\Delta\mathbf{x}$ of fixed length $\|\Delta\mathbf{x}\|$ in a local region around the current operating point. One approach to avoid crossing region boundaries is to determine the direction $\mathbf{r}(\mathbf{x}_2) = \frac{\mathbf{x}_2-\mathbf{x}_1}{\|\mathbf{x}_2-\mathbf{x}_1\|}$ that minimizes deviation from the vector between the starting point and the desired point $\mathbf{e}_d = \frac{\mathbf{x}_d-\mathbf{x}_1}{\|\mathbf{x}_d-\mathbf{x}_1\|}$ (8.6)

$$\mathbf{r} = \min_{\mathbf{x}_2} \frac{\mathbf{r}(\mathbf{x}_2)\cdot\mathbf{e}_d}{\big\|\mathbf{r}(\mathbf{x}_2)\big\|\big\|\mathbf{e}_d\big\|} \tag{8.6}$$

while meeting a condition that the minimum distance to the boundary point $d^2 = \big(\mathbf{x}_B - \mathbf{x}_2\big)^T\big(\mathbf{x}_B - \mathbf{x}_2\big) \geq \delta$ for positive distances, with distance sign defined by (8.7).

$$\operatorname{sign}\big(d(\mathbf{x})\big) = \begin{cases} +1 & \forall\mathbf{x} \mid \mathbf{h}(\mathbf{x})\boldsymbol{\beta} > 0 \\ -1 & \forall\mathbf{x} \mid \mathbf{h}(\mathbf{x})\boldsymbol{\beta} < 0 \end{cases} \tag{8.7}$$

The minimum distance to the ELM decision boundary defined by (8.4) can be found using Lagrange multipliers [107] by defining the Lagrangian $L(\mathbf{x},\lambda) = \big(\mathbf{x} - \mathbf{x}_2\big)^T\big(\mathbf{x} - \mathbf{x}_2\big) + \lambda\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ and setting its partial derivatives to zero (8.8).

$$\begin{aligned} \nabla L(\mathbf{x},\lambda) &= \mathbf{0} \\ \frac{\partial}{\partial\lambda}L(\mathbf{x},\lambda) &= \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = 0 \end{aligned} \tag{8.8}$$

The step size, $\|\Delta\mathbf{x}\|$, must then be tuned to ensure that the steps are small enough to avoid crossing over region boundaries. This process is illustrated in Figure 8.2.

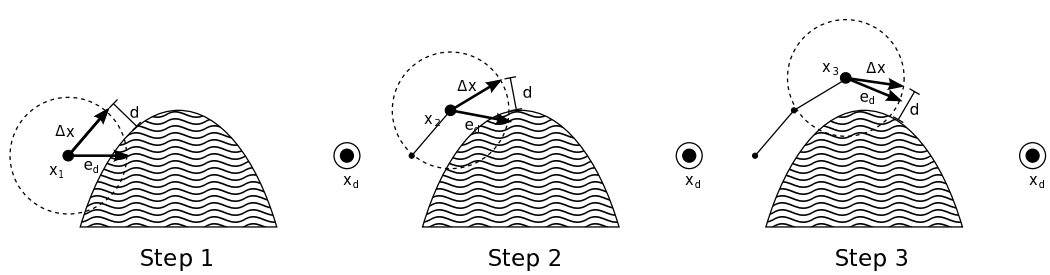


Figure 8.2. Path planning step in continuous domain.

This process could run into problems if non-convex obstacles are encountered. Improvements to the path planning algorithm applied to a space defined by a classifier like the ELM or FELM is an ongoing area of interest for this project. This process can be applied to the FELM by placing its decision boundary function (8.5) in the Lagrangian instead of the ELM decision boundary function.

One candidate for the overall cost function for the path is the total Euclidean path length plus some penalty based on the closeness of the path to obstacles. If straight-line steps of length $\|\Delta\mathbf{x}\|$ are taken, then the path length can be simplified to the total number of steps, N, times this length (8.9)

$$J_{\mathbf{p}} = N\|\Delta\mathbf{x}\| + \alpha \sum_{k=1}^{N} c\big(d(\mathbf{x_k})\big) \tag{8.9}$$

where $c(\cdot)\colon \mathbb{R} \to \mathbb{R}$ is some cost function that penalizes the path for being too close to obstacles, and α is a weighing factor used to adjust the relative weight of path length and the obstacles proximity penalty.

Using the MLFN classifier, it is more difficult to evaluate the decision boundary defined by (8.10), where $\mathbf{y}_B$ is an m-element uniform PDF indicating the boundary between classifications.

$$\begin{aligned} &\mathbf{x}_B \in \mathbb{X}_B \\ &\quad = \Big\{\mathbf{x} | f_M\big(\mathbf{W}_M f_{M-1}\big(\mathbf{W}_{M-1} f_{M-2}\big(\dots f_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)\dots\big) + \mathbf{b}_{M-1}\big) + \mathbf{b}_M\big) = \mathbf{y}_B\Big\} \end{aligned} \tag{8.10}$$

In this case, the *a priori* map of the operating space of the combustor can be discretized by evaluating it over a grid of possible input conditions, and a directed graph of the operating space's classification can be built. The ADA* algorithm can then be run without modification on the discretized map of the operating space, and nearest discrete points in the space can be updated with the predictor when predicted operating conditions do not match classification.

9. CONCLUSIONS

A nine-element research combustor was designed and built for the purpose of developing a data-driven classifier of the operating space of the combustor that could be used as an *a priori* map of the operating mode of the flame at a given operating condition, and for the purpose of developing a data-driven predictor that could use *in situ* information to predict upcoming flame modes. An experimental procedure was designed to characterize the operating space of this system. A LabVIEW command and control VI was developed to interface with existing lab hardware and automate this experimental procedure and data collection process.

Analysis of the structure of the experimental data was presented. K-means clustering was used to label experimental data points as "attached", "detached", "flickering", and "blow-out". A multi-layer feed-forward neural network operating point classifier was built using the labeled experimental data to act as a map of "good" (detached) and "bad" (attached, flickering, or blow-out) operating conditions throughout the space. The classifier exhibited 91% accuracy in labeling the operating class of the flame ("good" or "bad") when tested on a validation data set that was excluded from the training set. This indicates that *a priori* mapping of the operating modes of the combustor system is feasible, but not perfect. This operating point classifier could act as a map for planned *a priori* trajectories through the operating space of the system that avoid undesirable operating modes, but given the classifier's accuracy, real time updates to these trajectories would likely be required as unexpected behaviors become apparent.

To this end, a long short-term memory recurrent neural network predictor was built using the labeled experimental data set. This predictor monitors the operation of the combustor *in situ*, giving it access to significantly more information than the classifier—including operation history—that it can use to predict what flame mode

will occur next. The predictor is 92% accurate in predicting the operating mode of the next reading from the combustor on a validation data set, and predicts which specific mode identified by the K-means classifier will be seen; not just "good" or "bad" classifications. If only "good" and "bad" classifications are required, the predictor's accuracy improves. The predictor can be used as a substitute for "look-ahead" information about the operating space of the combustor, and can be used to inform a real-time path planning algorithm about predicted behaviors that do not match the *a priori* operating space map. While the predictor does provide information about upcoming operating conditions, two drawbacks exist:

1. The predictor only provides a probabilistic estimate of the upcoming operating condition.

2. As presented the predictor only predicts the operating mode of the next measurement of the combustion chamber.

The first drawback is intrinsic to the system and the fact that our path is temporal, not spacial, in nature—we cannot measure future operating modes directly. The second drawback can be addressed by training the predictor to output a series of upcoming predicted operating modes instead of just one. The training process can be modified to provide each training batch with a label batch that is itself a series of upcoming operating modes instead of only the next operating mode. This could be implemented with the current network structure and would require only minor changes to the Tensorflow model. This method would, however, require modifications to the way that data is collected. Currently, operating modes are characterized based on static measurements: the input condition is not changing while measurements are taking place. In order to predict a sequence of operating modes, training data would need to include measurements as the operating condition is changing along a prescribed path. These paths, then, would need to adequately blanket the region of interest in the operating space in much the same way that the Halton-sequence-generated operating conditions of interest currently do.

A controller architecture was introduced that utilizes the operating space classifier and operating mode predictor to plan a trajectory between desired operating conditions in the combustor's operating space, and to update this trajectory as predicted behavior deviates from the behavior of the *a priori* map. A path planning algorithm was introduced that determines a feasible and optimal path in real time through the operating space of the system that moves the system from its current operating condition to a desired condition while avoiding regions of instability and blow-out.

The accuracy of both the MLFN classifier and the LSTM predictor on the validation data set are promising considering the apparent complexity of the system. A path through the input space of the combustor planned using the classifier as an *a priori* map can then take advantage of the additional information available *in situ* by using the predictor to monitor the current operating conditions of the combustor as well as its history and determine what operating mode is most likely to occur next. When the predicted operating mode does not match the expected mode, the map can be updated with this new information and the anytime repairing path planning algorithm can provide a corrected path right away.

Both the *a priori* map and the *in situ* classifier provide an estimate of the current and future operating conditions of the combustor. Operating mode classifications are based on features present in the data that is collected, and, other than blow-out conditions, it is difficult to verify that an identified operating mode matches the actual operating mode for a given input. Furthermore, some gradient may exist between operating modes, such as the intermittent attachment and detachment of flames when transitioning between attached stable flames and detached stable flames. This makes it difficult to quantify the performance of the path planning algorithm using the classifier map and predictor.

The approach presented herein may be viable as a method for expediting the development of a new combustor system. The current development method for new combustor systems involves extensive testing to map out operating point transitions to achieve desired emissions and power output performance metrics while maintaining

safe operation. See, for example, [108] for some discussion of this testing procedure. It is hoped that a method similar to that presented herein could be implemented, incorporating emissions and power output measurements into the training data set, to automatically determine operating point transition paths that achieve performance and stability metrics on a full-scale combustor system. By automating this portion of the combustor design process, it is hoped that new designs may be completed more quickly and cheaply, thus enabling faster development of next generation combustor technologies to meet increasingly strict emissions regulations.

10. RECOMMENDATIONS

Recommended next steps are divided into three categories: those pertaining to the hardware, those pertaining to the experimental procedure, and those pertaining to the controller. First, it is recommended that a combustion chamber with smaller cross-sectional area be used for future analysis. This is a fundamental change to the combustion system, which will require all new data gathering, classifier design, and predictor design. The existing CC VI and Tensorflow scripts can be used with little or no modification to run these new experiments and training procedures. It is expected that this change will enable the combustor to more dramatically excite thermoacoustic instabilities and resonance modes in the chamber while also reducing the thermal capacity of the chamber; making temperature measurements more viable for use in analyzing the system's performance. This change would require a new combustion chamber, as well as some re-design or re-work of the nine-element combustor assembly so that it can be mated with the new chamber.

Experiments could be repeated with additional combustors with some modification of the CC VI and little to no modification of the Tensorflow scripts. In this way, one could study the generalizability of the current approach to other combustor systems. The CC VI and Python script used to generate test points would need to be modified to account for different input variables that may be adjusted for the different combustor systems, for different ranges of inputs that provide desired flow rates and fuel-air equivalence ratios, and for different stable re-ignite points.

A method for labeling data points as they are collected would improve confidence in the validity of the labels applied to data points, and may improve the performance of the classifier and predictor trained using these labels in correctly mapping out the operating space and predicting future operating modes. Currently data points are labeled with a "flame" or "no flame" status based on the median-filtered mean power

recorded at the dynamic pressure sensor closest to the combustor. If additional sensors or processing techniques could be applied to robustly identify operating conditions as "attached", "detached", or "flickering" *in situ*, then unsupervised classification of operating conditions would no longer be needed and any classifier or predictor trained from the data would have greater confidence in the validity of its outputs. Furthermore, the ability to identify operating conditions *in situ* would provide feedback as to whether the operating space map and predictor are correctly identifying conditions, and may provide an avenue by which these two components could be adapted online.

Attempts could be made to modify the experimental procedure to better canvas the operating space of the combustor. Currently, if the combustor blows out early in its move towards the next desired operating point, any ignitable operating conditions between the blow-out point and the desired operating point will be skipped, and the combustor will jump straight to the desired operating point to check for re-ignition. This could lead to missed stable operating conditions that may be reachable on other trajectories. Furthermore, the reliance on a single known re-ignite point may result in identified stable operating modes being biased towards this re-ignite point. One possible way to improve canvassing of the operating space would be to begin by randomly sampling operating conditions to find several stable re-ignition points and then choosing the re-ignition point from which we restart after each blow out and failed re-ignition at a target condition randomly from several options. This would effectively change the starting condition for the next test path through the operating space and would offer different starting conditions for most re-ignites.

An extension to improving canvassing of the operating space would be to canvas the operating space with a series of trajectories rather than stationary operating modes, and attempting to train a predictor based on these trajectories that can predict a series of upcoming operating modes. This could enable better look-ahead for the path planning algorithm. Furthermore, developing the classifier and predictor using non-stationary data might enable these components to better handle transient conditions in the combustor system.

The natural progression of this research would be to extend these methods to developing a data-driven *dynamic* model of the system that can be used to predict the system's response to a given input sequence. The goal of this model would be to capture the dynamic response characteristics of the chamber, not only the operating mode characteristics as the current classifier and predictor do. This presents several challenges, including determining how to characterize the dynamic response of the system from a given input. Techniques like principle component analysis (PCA) and singular spectrum analysis (SSA) may be viable tools for this characterization. Whatever technique proves viable, a data-driven dynamic model can be build from inputs $\mathbf{u}_k \in \mathbb{R}^m$ and response feature vectors $\mathbf{x}_k \in \mathbb{R}^n$ following a similar method to the classifier and predictor discussed herein.

An adaptive model predictive controller (MPC) [109,110] could next be developed to regulate the combustor operating point to the desired path. An MPC utilizes an internal model of the system and the current operating point to predict the evolution of the system over a finite prediction horizon. The control effort is calculated by optimizing the predicted performance of the system over a finite horizon to determine an optimal control sequence $\mathbf{U} = \begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{N-1}\end{bmatrix}$, and the first step of the control sequence $\mathbf{u}_0$ is applied to the plant. This process repeats for every sampling cycle. Typically in MPC, the predicted system response is described by a difference equation (10.1).

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \mathbf{f}\big(\hat{\mathbf{x}}_k, \mathbf{u}_k\big)\\ \mathbf{y}_k &= \mathbf{g}\big(\hat{\mathbf{x}}_k\big)\end{aligned} \tag{10.1}$$

Here we will refer to $\hat{\mathbf{x}}_k$ as the predicted state of the system at time k. The control and state sequences must satisfy $\mathbf{u}_k \in \mathbb{U} \subset \mathbb{R}^m$ and $\hat{\mathbf{x}}_k \in \mathbb{X} \subset \mathbb{R}^n$ respectively for all k. The state sequence resulting from the control sequence $\mathbf{U}$ is denoted $\widehat{\mathbf{X}} = \begin{bmatrix}\hat{\mathbf{x}}_0 & \hat{\mathbf{x}}_1 & \cdots & \hat{\mathbf{x}}_{N-1}\end{bmatrix}$. The control objective is to steer the system state sequence $\widehat{\mathbf{X}}$ such that the states follow a desired trajectory sequence $\mathbf{P} \in \mathbb{X}^N \subset \mathbb{R}^{n\times N}$. For

initial state $\mathbf{x}$, trajectory length N, input trajectory $\mathbf{U}$, and resulting predicted state trajectory $\widehat{\mathbf{X}}$, the cost can be defined as $J_{\mathrm{MPC}}(\mathbf{x}, \mathbf{U})$ (10.2)

$$J_{\mathrm{MPC}}(\mathbf{x}, \mathbf{U}) = \sum_{i=0}^{N-1} \left[\left(\mathbf{p}_i - \hat{\mathbf{x}}_i\right)^T \mathbf{Q} \left(\mathbf{p}_i - \hat{\mathbf{x}}_i\right) + \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i \right] \tag{10.2}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a matrix that can be used to adjust the weights of the error vector $(\mathbf{p}_i - \hat{\mathbf{x}}_i)$—for instance by weighing error in certain state variables more heavily than others—and $\mathbf{R} \in \mathbb{R}^{m \times m}$ is a matrix that can be used to adjust the weights of each element of the control effort $\mathbf{u}_i$. It is possible to utilize different values for these matrices at each step in the sequence, i.e. $\mathbf{Q}_i$ and $\mathbf{R}_i$, thus allowing us to weigh errors and control efforts differently throughout time. As a first attempt, it would be simpler to choose constant controller cost function weight matrices. The control problem then requires determining the control sequence $\mathbf{U}^0$ that minimizes a cost function J_{MPC} (10.3).

$$J^0_{\mathrm{MPC}} = \min_{\mathbf{U}} \left\{ J_{\mathrm{MPC}}(\mathbf{x}, \mathbf{U}) \mid \mathbf{u}_k \in \mathbb{U},\ k = 0, \ldots, N-1 \right\} \tag{10.3}$$

The control law then applies the first element of the optimal control sequence $\mathbf{u}^0_0$ to the system. On the next sampling cycle, the current state of the system $\mathbf{x}$ is updated and the process is repeated. The overall control system including the path planner but without the predictor discussed previously is illustrated in Figure 10.1.

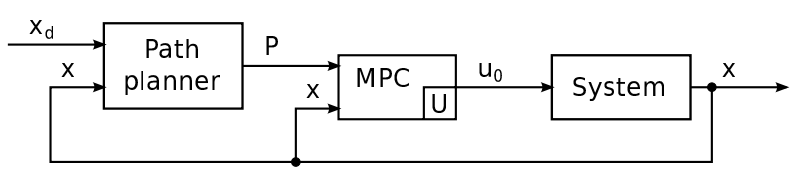


Figure 10.1. Path-following controller block diagram.

Obviously an important part of the model predictive control scheme is the model chosen to represent the system. For the combustion system in question, a good physics-based model does not exist or is computationally inefficient for control purposes. For this reason, it is proposed that a data-based model be developed using the feature vectors obtained from techniques like PCA and SSA on the same experiment

performed to train the state classifier, discussed in Chapter 4. A candidate model follows the same design approach as the ELM classifier, but incorporates input data to predict the state of the system at the next time step (10.4)

$$\hat{\mathbf{x}}_{k+1} = \mathbf{h}^T\big(\mathbf{q}_k, \mathbf{W}\big)\mathcal{B} \tag{10.4}$$

where $\mathbf{q}_k = \begin{bmatrix}\mathbf{x}_k^T & \mathbf{u}_k^T\end{bmatrix}^T \in \mathbb{R}^{n+m}$ is the feature vector augmented by adding the inputs to the end, $\mathbf{W} = \begin{bmatrix}\mathbf{w}_1 & \cdots & \mathbf{w}_M\end{bmatrix} \in \mathbb{R}^{(n+m)\times M}$ is a matrix of the input weights for the M hidden nodes, $\mathbf{h}^T = \begin{bmatrix}h_1\big(\mathbf{q}, \mathbf{w}_1\big), & h_2\big(\mathbf{q}, \mathbf{w}_2\big) & \cdots & h_M\big(\mathbf{q}, \mathbf{w}_M\big)\end{bmatrix} \in \mathbb{R}^M$ is a vector of the M hidden nodes, and $\mathcal{B} = \begin{bmatrix}\boldsymbol{\beta}_1 & \cdots & \boldsymbol{\beta}_n\end{bmatrix} \in \mathbb{R}^{M\times n}$ is the output weight matrix from the M hidden nodes to the n features of $\mathbf{x}_{k+1} \in \mathbb{R}^n$. The radial basis function (10.5) is a candidate for the hidden node activation function.

$$h_i\big(\mathbf{q}, \mathbf{w}_i\big) = \exp\left\{\frac{-\big(\mathbf{q}^T\mathbf{w}_i - c_i\big)^2}{\sigma_i^2}\right\} \tag{10.5}$$

In (10.5), the parameter c_i represents the center of the ith hidden node, and parameter σ_i represents the width of the ith hidden node. For the ELM algorithm, these parameters and the input weights $\mathbf{w}_i$ are randomly initialized and held constant; only the output weights $\mathcal{B}$ are tuned.

Note that for the ELM, a more complicated hidden node is chosen such that a single-layer feed-forward network may be used with randomly initialized input weights and hidden node parameters while maintaining enough complexity to capture the dynamics we wish to model. MLFN structures may be viable as well. When all but the output weights of the network are initialized randomly, it is possible that the MLFN structure and ELM structure discussed here are equivalent for this problem. The proper structure for modeling the dynamics of the system is an open question.

Given L training sets $\aleph^j = \big\{(\mathbf{x}_i, \mathbf{u}_i)^j \mid \mathbf{x}_i^j \in \mathbb{X} \subset \mathbb{R}^n,\ \mathbf{u}_i^j \in \mathbb{U} \subset \mathbb{R}^m,\ i = 1, \ldots, N,\ j = 1, \ldots, L\big\}$, the ELM can be trained by first constructing the next-step feature matrix $\mathbf{X}' \in \mathbb{R}^{n\times L(N-1)}$ (10.6), the augmented feature matrix $\mathbf{Q} \in \mathbb{R}^{(n+m)\times L(N-1)}$ (10.7), and the matrices $\mathbf{W}$ and $\mathcal{B}$ discussed previously. Note that

the transposes of $\mathbf{X}'$ and $\mathbf{Q}$ are shown in (10.6) and (10.7) respectively in order to fit the expression on a single line.

$$\left(\mathbf{X}'\right)^T = \begin{bmatrix} \left(\mathbf{x}_2^1\right)^T \\ \left(\mathbf{x}_3^1\right)^T \\ \vdots \\ \left(\mathbf{x}_N^1\right)^T \\ \left(\mathbf{x}_2^2\right)^T \\ \vdots \\ \left(\mathbf{x}_N^2\right)^T \\ \vdots \\ \left(\mathbf{x}_2^L\right)^T \\ \vdots \\ \left(\mathbf{x}_N^L\right)^T \end{bmatrix} \tag{10.6}$$

$$\mathbf{Q}^T = \begin{bmatrix} \left(\mathbf{q}_1^1\right)^T \\ \left(\mathbf{q}_2^1\right)^T \\ \cdots \\ \left(\mathbf{q}_{N-1}^1\right)^T \\ \left(\mathbf{q}_1^2\right)^T \\ \cdots \\ \left(\mathbf{q}_{N-1}^2\right)^T \\ \cdots \\ \left(\mathbf{q}_1^L\right)^T \\ \cdots \\ \left(\mathbf{q}_{N-1}^L\right)^T \end{bmatrix} \tag{10.7}$$

The hidden node output matrix $\mathbf{H}(\mathbf{Q}, \mathbf{W}) \in \mathbb{R}^{L(N-1)\times M}$ (10.8) is next calculated using the M hidden nodes with randomly assigned centers c_i and widths σ_i for $i \in \{1, \ldots, M\}$.

$$\mathbf{H}(\mathbf{Q}, \mathbf{W}) = \begin{bmatrix} \mathbf{h}^T(\mathbf{q}_1^1, \mathbf{W}) \\ \vdots \\ \mathbf{h}^T(\mathbf{q}_{N-1}^1, \mathbf{W}) \\ \mathbf{h}^T(\mathbf{q}_1^2, \mathbf{W}) \\ \vdots \\ \mathbf{h}^T(\mathbf{q}_{N-1}^2, \mathbf{W}) \\ \vdots \\ \mathbf{h}^T(\mathbf{q}_1^L, \mathbf{W}) \\ \vdots \\ \mathbf{h}^T(\mathbf{q}_{N-1}^L, \mathbf{W}) \end{bmatrix} \tag{10.8}$$

Based on these definitions, the following theorem can be stated [76]:

Theorem 10.0.1 *Given some small positive value $\delta > 0$ and activation function $h\colon \mathbb{R} \to \mathbb{R}$ which is infinitely differentiable, there exists $M \leq N$ such that for N arbitrary distinct samples $\left(\mathbf{x}_i, \mathbf{t}_i\right) \mid \ \mathbf{x}_i \in \mathbb{R}^p,\ \mathbf{t}_i \in \mathbb{R}^n$, for any input weight vectors $\mathbf{w}_i \in \mathbb{R}^n$ and any hidden layer parameters $b_i \in \mathbb{R}$ chosen according to any continuous probability distribution function, then with probability one, $\|\mathbf{H}_{N\times M}\mathcal{B}_{M\times n} - \mathbf{T}_{N\times n}\| < \delta$.*

The proof for Theorem 10.0.1 is given in [76].

Finally, it has been proven that the minimum-norm least squares solution $\hat{\mathcal{B}}$ to the system $\mathbf{H}\mathcal{B} = \mathbf{X}'$ is given by (10.9), where $\mathbf{H}^\dagger = \left(\mathbf{H}^H\mathbf{H}\right)^{-1}\mathbf{H}^H$ is the Moore-Penrose pseudo-inverse, and $\mathbf{H}^H$ is the conjugate transpose of $\mathbf{H}$ [111, 112]. Note that the Moore-Penrose pseudo-inverse results in a matrix with transposed dimensions.

$$\hat{\mathcal{B}} = \left[\mathbf{H}(\mathbf{Q}, \mathbf{W})\right]^\dagger \left(\mathbf{X}'\right)^T \tag{10.9}$$

The solution (10.9) has the following properties:

1. It is the least-squares solution to $\mathbf{H}\mathcal{B} = \mathbf{X}'$:

$$\left\|\mathbf{H}\hat{\mathcal{B}} - \mathbf{X}'\right\| = \left\|\mathbf{H}\mathbf{H}^{\dagger}\mathbf{X}' - \mathbf{X}'\right\| = \min_{\mathcal{B}} \left\|\mathbf{H}\mathcal{B} - \mathbf{X}'\right\|$$

2. It has the smallest norm among all of the least squares solutions of $\mathbf{H}\mathcal{B} = \mathbf{X}'$:

$$\begin{aligned} &\left\|\hat{\mathcal{B}}\right\| = \left\|\mathbf{H}^{\dagger}\mathbf{X}'\right\| \leq \left\|\mathcal{B}\right\| \\ &\forall \mathcal{B} \in \left\{\mathcal{B} \mid \ \|\mathbf{H}\mathcal{B} - \mathbf{X}'\| \leq \|\mathbf{H}\mathbf{Z} - \mathbf{X}'\|, \ \forall \mathbf{Z} \in \mathbb{R}^{M\times n}\right\} \end{aligned}$$

3. The minimum-norm least squares solution is unique.

When used to predict the performance of the system, the predicted state update equation as a function of the augmented predicted state vector $\hat{\mathbf{q}}_k$ is given by (10.10).

$$\hat{\mathbf{x}}_{k+1} = \mathbf{h}\big(\hat{\mathbf{q}}_k, \mathbf{W}\big)\hat{\mathcal{B}} \tag{10.10}$$

The prediction error is defined as $\hat{\mathbf{e}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$, where $\mathbf{x}_k$ is the actual system feature vector at the kth time step. With this ELM model, the MPC discussed above could be a viable candidate for a model-based controller using a data-drive model for regulating the operating condition of the research combustor system.

The next step would be incorporating a method for updating the data-driven model of the system in the presence of unexpected system behavior. With the ELM model discussed previously, this simply requires that one be able to adjust the network weight matrix $\mathcal{B}$ with new operating data. One candidate method for updating the network weight matrix as new information becomes available is the online sequential ELM (OS-ELM) algorithm [82,83,113,114]. This algorithm trains the initial network as discussed previously, defining $\mathbf{P}_0 = \left(\mathbf{H}^H\mathbf{H}\right)^{-1}$, so that $\mathbf{H}^{\dagger} = \mathbf{P}_0\mathbf{H}^H$. The algorithm then updates the network weight matrix sequentially with each new data point. Given the new augmented feature vector $\mathbf{q}_{k+1}$ and system response feature vector $\mathbf{x}'_{k+1}$, the algorithm updates the output weight matrix sequentially according to (10.11).

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k\mathbf{h}\big(\mathbf{q}_{k+1}\big)\mathbf{h}^T\big(\mathbf{q}_{k+1}\big)\mathbf{P}_k}{1 + \mathbf{h}^T\big(\mathbf{q}_{k+1}\big)\mathbf{P}_k\mathbf{h}\big(\mathbf{q}_{k+1}\big)} \\ \hat{\mathcal{B}}_{k+1} &= \hat{\mathcal{B}}_k + \mathbf{P}_{k+1}\mathbf{h}\big(\mathbf{q}_{k+1}\big)\Big(\big(\mathbf{x}'_{k+1}\big)^T - \mathbf{h}^T\big(\mathbf{q}_{k+1}\big)\hat{\mathcal{B}}_k\Big) \end{aligned} \tag{10.11}$$

Rather than applying this algorithm with each new data point, a threshold δ could be introduced. The algorithm could then retrain the output weight matrix when the norm of the error vector $\|\hat{\mathbf{e}}_{k+1}\| > \delta$. The OS-ELM algorithm can be summarized as:

1. Randomly assign input weight vectors $\mathbf{w}_i$, optional biases $\mathbf{b}_i$, and hidden layer parameters $\boldsymbol{\zeta}_i$ for $i = 1, \ldots, L$ hidden nodes.

2. Calculate the hidden layer output matrix $\mathbf{H}$ (6.7).

3. Calculate the initial minimum-norm least squares output weight vector $\hat{\mathcal{B}}$ using (6.6).

4. Present the next data point $\aleph_{k+1} = \left(\mathbf{q}_{k+1}, \mathbf{x}'_{k+1}\right)$.

5. Calculate the new hidden layer activation vector $\mathbf{h}(\mathbf{q}_{k+1})$, predicted state $\hat{\mathbf{x}}_{k+1}$, and prediction error $\hat{\mathbf{e}}_{k+1}$.

6. If $\|\hat{\mathbf{e}}_{k+1}\| > \delta$, where δ is a heuristic prediction error threshold, update the output weight matrix $\hat{\mathcal{B}}_{k+1}$ using (10.11).

7. Set $k = k + 1$ and return to step 4.

The OS-ELM algorithm maintains a static neural network and only updates the output weight matrix when new information is available, or when prediction error is too high. Another possibility for adapting the system model online is to use an algorithm that modifies the underlying hidden layer of the neural network when prediction error is too high. Algorithms exist that seek to grow a network, choosing optimal hidden nodes at each iteration, or prune a very large network, leaving only the most important nodes for predicting system performance. Network growing algorithms including the incremental ELM (I-ELM) [75, 113] and adaptive growth ELM (AG-ELM) [115]. The I-ELM algorithm begins with a small network trained in exactly the same way as the traditional ELM, and then adds a single node at a time until an error requirement or the maximum network size is met. An enhanced I-ELM (EI-ELM) chooses new nodes from a set of candidate nodes based on which node has

the largest effect on the overall error. The AG-ELM algorithm adaptively builds a network from scratch one node at a time until some prediction error threshold or a maximum number of hidden nodes is reached.

The optimal pruning ELM (OP-ELM) [113, 116] algorithm seeks to prune a very large ELM so that only the most important nodes remain. The OP-ELM algorithm begins with a very large ELM trained in the same way as described previously but possibly incorporating various hidden layer activation functions, e.g. linear, sigmoid, Gaussian. OP-ELM then ranks the hidden layer neurons based on their contribution to reducing the prediction error of the ELM by applying the multi-response spare regression algorithm [117]. The best hidden nodes are then selected through leave-one-out validation.

While each of these algorithms seeks to choose the best hidden node at each step, they still build a static hidden node structure that stops adapting once the maximum network size or desired prediction error has been reached. Model structure and adaptive training algorithms are an open area of research, and what model and adaptation algorithm is best for this problem is an open question that may merit further study. This discussion of MPC and adaptive ELM is given for the sake of establishing some ground work toward the goal of developing a data-driven model-based controller for the combustor system. Many questions remain, and this is an area where future research may be merited.

LIST OF REFERENCES

LIST OF REFERENCES

[1] G. Bulat, K. Liu, G. Brickwood, V. Sanderson, and B. Igoe, "Intelligent operation of siemens (SGT-300) DLE gas turbine combustion system over an extended fuel range with low emissions," in *Proceedings of the ASME Turbo Expo 2011*, (Vancouver, British Columbia, Canada), pp. 917–925, ASME, Jan. 2011.

[2] H. C. Mongia, "N+3 and N+4 generation aeropropulsion engine combustors: part 6: operating conditions, target goals and lifted jets," in *Proceedings of the 49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, (San Jose, CA), AIAA, July 2013.

[3] R. D. Flack, *Fundamentals of jet propulsion with applications*, vol. 17. Cambridge University Press New York, 2005.

[4] K. M. Tacina, "Swirl-venturi lean direct injection combustion technology," in *Prceedings of the Spring Technical Meeting of the Central States Section of the Combustion Institute*, pp. 1–9, Apr. 2012.

[5] C.-M. Lee, "NASA project develops next generation low-emissions combustor technologies," in *Prceedings of the 51st AIAA Aerospace Sciences Meeting*, (Grapevine, TX), AIAA, Jan. 2013.

[6] K. K. Kuo, *Principles of combustion*. Hoboken, NJ: John Wiley, 2nd ed., 2005.

[7] R. Meyer, R. A. DeCarlo, S. Pekarek, and C. Doktorcik, "Gas turbine engine behavioral modeling," *Purdue Electrical and Computer Engineering Technical Reports*, vol. 14, Jan. 2014.

[8] A. P. Dowling and A. S. Morgans, "Feedback control of combustion oscillations," *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 151–182, 2005.

[9] P. Schmitt, T. Poinsot, B. Schuermans, and K. P. Geigle, "Large-Eddy Simulation and experimental study of heat transfer, nitric oxide emissions and combustion instability in a swirled turbulent high-pressure burner," *Journal of Fluid Mechanics*, vol. 570, pp. 17–46, Jan. 2007.

[10] H. Pitsch, "Large-Eddy Simulation of turbulent combustion," *Annual Review of Fluid Mechanics*, vol. 38, no. 1, pp. 453–482, 2006.

[11] G. Staffelbach, L. Y. M. Gicquel, G. Boudier, and T. Poinsot, "Large Eddy Simulation of self excited azimuthal modes in annular combustors," *Proceedings of the Combustion Institute*, vol. 32, no. 2, pp. 2909–2916, 2009.

[12] R. Garby, L. Selle, and T. Poinsot, "Large-Eddy Simulation of combustion instabilities in a variable-length combustor," *Comptes Rendus Mecanique*, vol. 341, pp. 220–229, Jan. 2013.

[13] K. K. Venkataraman, L. H. Preston, D. W. Simons, B. J. Lee, J. G. Lee, and D. A. Santavicca, "Mechanism of combustion instability in a lean premixed dump combustor," *Journal of Propulsion and Power*, vol. 15, pp. 909–918, Nov. 1999.

[14] H. J. Lee, J. G. Lee, B. Quay, and D. Santavicca, "Mechanism of combustion instability due to flame-vortex interactions in a lean premixed gas turbine combustor," in *Proceedings of the 49th AIAA/ASME/SAE/ASEE Joint Propulsion Conferences*, AIAA, July 2013.

[15] G. B. King, N. M. Laurendeau, and M. W. Renfro, "Two-Point Scalar Time-Series Measurements in Turbulent Partially Premixed Flames," Final Report 20090429217, AFOSR/NA, Arlington, VA, Feb. 2009.

[16] K. M. Kopp-Vaughan, S. G. Tuttle, M. W. Renfro, and G. B. King, "Heat release and flame structure measurements of self-excited acoustically-driven premixed methane flames," *Combustion and Flame*, vol. 156, pp. 1971–1982, Oct. 2009.

[17] J. J. Keller, "Thermoacoustic oscillations in combustion chambers of gas turbines," *AIAA Journal*, vol. 33, no. 12, pp. 2280–2287, 1995.

[18] S. Ducruix, T. Schuller, D. Durox, and S. Candel, "Combustion dynamics and instabilities: Elementary coupling and driving mechanisms," *Journal of Propulsion and Power*, vol. 19, no. 5, pp. 722–734, 2003.

[19] J. C. DeLaat, G. Kopasakis, J. R. Saus, C. T. Chang, and C. Wey, "Active combustion control for a low-emissions aircraft engine combustor prototype: Experimental results," *Journal of Propulsion and Power*, vol. 29, no. 4, pp. 991–1000, 2013.

[20] M. Harvazinski, W. Anderson, and C. Merkle, "Combustion instability diagnostics using the rayleigh index," in *Proceedings of the 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, (San Diego, CA), AIAA, Aug. 2011.

[21] P. Weigand, W. Meier, X. R. Duan, R. Giezendanner-Thoben, and U. Meier, "Laser diagnostic study of the mechanism of a periodic combustion instability in a gas turbine model combustor," *Flow, Turbulence and Combustion*, vol. 75, pp. 275–292, Dec. 2005.

[22] J. G. Lee and D. A. Santavicca, "Experimental diagnostics for the study of combustion instabilities in lean premixed combustors," *Journal of Propulsion and Power*, vol. 19, pp. 735–750, Sept. 2003.

[23] J. W. S. Rayleigh, "The explanation of certain acoustical phenomena," *Nature*, vol. 18, no. 455, pp. 319–321, 1878.

[24] G. A. Richards, D. L. Straub, and E. H. Robey, "Passive control of combustion dynamics in stationary gas turbines," *Journal of Propulsion and Power*, vol. 19, no. 5, pp. 795–810, 2003.

[25] Y. Huang and V. Yang, "Dynamics and stability of lean-premixed swirl-stabilized combustion," *Progress in Energy and Combustion Science*, vol. 35, pp. 293–364, Aug. 2009.

[26] V. Bellucci, C. O. Paschereit, P. Flohr, and F. Magni, "On the use of Helmholtz resonators for damping acoustic pulsations in industrial gas turbines," in *Proceedings of the ASME Turbo Expo 2011*, (New Orleans, LA), pp. 2001–GT–0039, ASME, June 2001.

[27] N. Tran, S. Ducruix, and T. Schuller, "Damping combustion instabilities with perforates at the premixer inlet of a swirled burner," *Proceedings of the Combustion Institute*, vol. 32, no. 2, pp. 2917–2924, 2009.

[28] A. M. Annaswamy, M. Fleifil, J. W. Rumsey, R. Prasanth, J.-P. Hathout, and A. F. Ghoniem, "Thermoacoustic instability: Model-based optimal control designs and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 8, pp. 905–918, Nov. 2000.

[29] S. Candel, "Combustion dynamics and control: Progress and challenges," *Proceedings of the Combustion Institute*, vol. 29, no. 1, pp. 1–28, 2002.

[30] C. O. Paschereit, E. Gutmark, and W. Weisenstein, "Structure and control of thermoacoustic instabilities in a gas-turbine combustor," *Combustion Science and Technology*, vol. 138, pp. 213–232, Sept. 1998.

[31] C. O. Paschereit, E. Gutmark, and W. Weisenstein, "Control of thermoacoustic instabilities and emissions in an industrial-type gas-turbine combustor," in *Proceedings of the 27th Symposium (International) on Combustion*, pp. 1817–1824, CI, 1998.

[32] A. S. Morgans and A. P. Dowling, "Model-based control of combustion instabilities," *Journal of Sound and Vibration*, vol. 299, pp. 261–282, Jan. 2007.

[33] J. M. Cohen and A. Banaszuk, "Factors affecting the control of unstable combustors," *Journal of Propulsion and Power*, vol. 19, no. 5, pp. 811–821, 2003.

[34] J. R. Seume, N. Vortmeyer, W. Krause, J. Hermann, C.-C. Hantschk, P. Zangl, S. Gleis, D. Vortmeyer, and A. Orthmann, "Application of active combustion instability control to a heavy duty gas turbine," in *Proceedings of the ASME ASIA '97 Conference & Exposition*, (Singapore), pp. 97–AA–119, ASME, Oct. 1997.

[35] C. E. Johnson, Y. Neumeier, M. Neumaier, B. T. Zinn, D. D. Darling, and S. S. Sattinger, "Demonstration of active control of combustion instabilities on a full-scale gas turbine combustor," in *Proceedings of ASME Turbo Expo 2001*, (New Orleans, LA), pp. 2001–GT–0519, ASME, June 2001.

[36] P. J. Langhorne, A. P. Dowling, and N. Hooper, "Practical active control system for combustion oscillations," *Journal of Propulsion and Power*, vol. 6, no. 3, pp. 324–333, 1990.

[37] Y. Neumeier and B. T. Zinn, "Experimental demonstration of active control of combustion instabilities using real-time modes observation and secondary fuel injection," in *Proceedings of the 26th Symposium (International) on Combustion*, pp. 2811–2818, CI, 1996.

[38] P. Barooah, T. J. Anderson, and J. M. Cohen, "Active combustion instability control with spinning valve actuator," in *Proceedings of ASME Turbo Expo 2002*, (Amsterdam, The Netherlands), pp. GT–2002–30042, ASME, June 2002.

[39] N. Docquier and S. Candel, "Combustion control and sensors: A review," *Progress in Energy and Combustion Science*, vol. 28, no. 2, pp. 107–150, 2002.

[40] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[41] K.-H. Yoo, J.-C. Kim, H.-G. Sung, L. Zhang, and V. Yang, "Flow dynamics in combustors with multi-element swirl injectors," in *Proceedings of the 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, (Orlando, FL), pp. AIAA–2011–786, AIAA, Jan. 2011.

[42] A. H. Lefebvre, *Gas turbine combustion*. CRC Press, 2nd ed., Sept. 1998.

[43] E. Braaten and G. Weller, "An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration," *Journal of Computational Physics*, vol. 33, pp. 249–258, Nov. 1979.

[44] N. Petrov, I. Jordanov, and J. Roe, "Identification of radar signals using neural network classifier with low-discrepancy optimisation," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC)*, (Cancun, Mexico), pp. 2658–2664, IEEE, June 2013.

[45] W. J. Morokoff and R. E. Caflisch, "Quasi-Monte Carlo integration," *Journal of Computational Physics*, vol. 122, pp. 218–230, Dec. 1995.

[46] J. H. Halton and G. B. Smith, "Algorithm 247: Radical-inverse quasi-random point sequence," *Communications of the ACM*, vol. 7, pp. 701–702, Dec. 1964.

[47] V. Hanta and A. Prochzka, "Rational approximation of time delay," *Institute of Chemical Technology in Prague. Department of computing and control engineering. Technicka*, vol. 5, no. 166, p. 28, 2009.

[48] P. Lancaster and L. Rodman, *Algebraic riccati equations*. Clarendon Press, 1995.

[49] M. Athans, "The role and use of the stochastic linear-quadratic-Gaussian problem in control system design," *IEEE Transactions on Automatic Control*, vol. 16, pp. 529–552, Dec. 1971.

[50] D. Straub, D. Ferguson, R. Rohrssen, and E. Perez, "Design considerations for remote high-speed pressure measurements of dynamic combustion phenomena," in *Proceedings of the 45th AIAA Aerospace Sciences Meeting and Exhibit*, (Reno, NV), pp. AIAA 2007–561, AIAA, Jan. 2007.

[51] G. Ferrara, L. Ferrari, and G. Sonni, "Experimental characterization of a remoting system for dynamic pressure sensors," in *Proceedings of the ASME Turbo Expo 2005*, (Reno-Tahoe, NV), pp. GT2005–68733, ASME, Jan. 2005.

[52] D. R. Englund and W. B. Richards, "The infinite line pressure probe," Technical Memorandum NASA TM-83582, National Aeronautics and Space Administration, Cleveland, OH, May 1984.

[53] H. Bergh and H. Tijdeman, "Theoretical and experimental results for the dynamic response of pressure measuring systems," Technical Report NLR-TR F.238, Nationaal Lucht-En Ruimtevaartlaboratorium, Jan. 1965.

[54] R. D. Samuelson, "Pneumatic instrumentation lines and their use in measuring rocket nozzle pressure," Technical Report RN-DR-0124, Aerojet-General Corporation, Sacramento, CA, July 1967.

[55] M. A. White, M. Dhingra, and J. V. R. Prasad, "Experimental analysis of a waveguide pressure measuring system," *Journal of Engineering for Gas Turbines and Power*, vol. 132, pp. 041603–041603, Jan. 2010.

[56] H. Zinn and M. Habermann, "Developments and experiences with pulsation measurements for heavy-duty gas turbines," in *Proceedings of GT2007*, (Montreal, Canada), pp. GT2007–27475, ASME, May 2007.

[57] T. M. Cover and J. A. Thomas, *Elements of information theory*. Hoboken, NJ: Wiley, 2006.

[58] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 5, no. 1, pp. 3–55, 1948.

[59] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[60] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the 21st International Conference on Machine Learning*, ICML, (Banff, Canada), ACM, 2004.

[61] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, June 2010.

[62] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*. 2015. Software available from tensorflow.org.

[63] P. S. Bradley and U. M. Fayyad, "Refining initial points for K-means clustering," in *Proceedings of the 15th International Conference on Machine Learning*, (San Francisco, CA), pp. 91–99, ICML, 1998.

[64] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[65] L. Van der Maaten, "Learning a parametric embedding by preserving local structure," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. 5 of *JMLR: W&CP*, (Clearwater Beach, FL), pp. 384–391, 2009.

[66] L. Van der Maaten and G. Hinton, "Visualizing non-metric similarities in multiple maps," *Machine learning*, vol. 87, no. 1, pp. 33–55, 2012.

[67] V. M. Janakiraman, X. Nguyen, J. Sterniak, and D. Assanis, "Modeling the stable operating envelope for partially stable combustion engines using class imbalance learning," *IEEE Transactions, in Review*, June 2013.

[68] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sept. 1995.

[69] G. M. Fung and O. L. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, pp. 77–97, May 2005.

[70] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293–300, June 1999.

[71] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, pp. 513–529, Apr. 2012.

[72] C. I. Torres, F. Hernandez, A. Trejo, and G. Ronquillo, "Support vector machines applied to a combustion process," in *Proceedings of the 9th Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, pp. 176–181, Nov. 2012.

[73] K. I. Wong, P. K. Wong, C. S. Cheung, and C. M. Vong, "Modeling and optimization of biodiesel engine performance using advanced machine learning methods," *Energy*, vol. 55, pp. 519–528, June 2013.

[74] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks, 2004*, vol. 2, pp. 985–990, IEEE, July 2004.

[75] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, pp. 879–892, July 2006.

[76] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.

[77] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[78] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, pp. 799–801, May 2000.

[79] A. Vaughan and S. V. Bohac, "An extreme learning machine approach to predicting near chaotic HCCI combustion phasing in real-time," *arXiv preprint arXiv:1310.3567*, Oct. 2013.

[80] G. Strang, *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 4th ed., 2006.

[81] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 464–471, Mar. 2002.

[82] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," in *Proceedings of the IASTED International Conference on Computational Intelligence*, (Calgary, Canada), IASTED, July 2005.

[83] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, pp. 1067–1072, Aug. 2009.

[84] Y. C. Shin, *Intelligent systems: modeling, optimization, and control*. No. 30 in Automation and control engineering, Boca Raton: CRC Press, 2009.

[85] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations 2015*, ICLR, 2015.

[86] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Aug. 1995.

[87] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310–3317, IEEE, May 1994.

[88] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 2472–2477, IEEE, May 2011.

[89] N. Subrahmanya and Y. C. Shin, "Constructive training of recurrent neural networks using hybrid optimization," *Neurocomputing*, vol. 73, pp. 2624–2631, Aug. 2010.

[90] Y. Xia and G. Feng, "A new neural network for solving nonlinear projection equations," *Neural Networks*, vol. 20, pp. 577–589, July 2007.

[91] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, pp. 78–80, Apr. 2004.

[92] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, IEEE, May 2013.

[93] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, Mar. 1994.

[94] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[95] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *Manuscript*, 2014.

[96] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, pp. 602–610, July 2005.

[97] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling.," in *Proceedings of Interspeech 2014*, (Singapore), pp. 338–342, ISCA, Sept. 2014.

[98] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber, "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks," in *Proceedings of the 9th International Conference on Document Analysis and Recognition*, vol. 1, pp. 367–371, 2007.

[99] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, vol. 1 of *ICRA*, pp. 968–975, IEEE, 2002.

[100] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems*, 2003.

[101] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proceedings of the 2005 International Conference on Automated Planning and Scheduling*, AAAI, 2005.

[102] M. Pivtoraiko and A. Kelly, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS, pp. 2611–2616, IEEE, Sept. 2008.

[103] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *Proceedings of the 2001 American Control Conference*, vol. 1, (Arlington, VA), pp. 43–49, AACC, June 2001.

[104] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, pp. 141–166, Feb. 2007.

[105] C. Sprunk, B. Lau, P. Pfaffz, and W. Burgard, "Online generation of kinodynamic trajectories for non-circular omnidirectional robots," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 72–77, IEEE, May 2011.

[106] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. . Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, pp. 425–466, Aug. 2008.

[107] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic Press, 2014.

[108] H. Mongia, "TAPS: A fourth generation propulsion combustor technology for low emissions," in *Proceedings of the AIAA International Air and Space Symposium and Exposition: The Next 100 Years*, (Dayton, OH), AIAA, July 2003.

[109] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, June 2000.

[110] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, pp. 2967–2986, Dec. 2014.

[111] C. R. Rao and S. K. Mitra, *Generalized inverse of matrices and its applications.* Probability and Mathematical Statistics Series, Wiley, 1971.

[112] D. Serre, *Matrices: Theory and applications.* No. 216 in Graduate Texts in Mathematics, New York: Springer, 2nd ed., 2010.

[113] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, pp. 107–122, May 2011.

[114] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. 1411–1423, Nov. 2006.

[115] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 365–371, Feb. 2012.

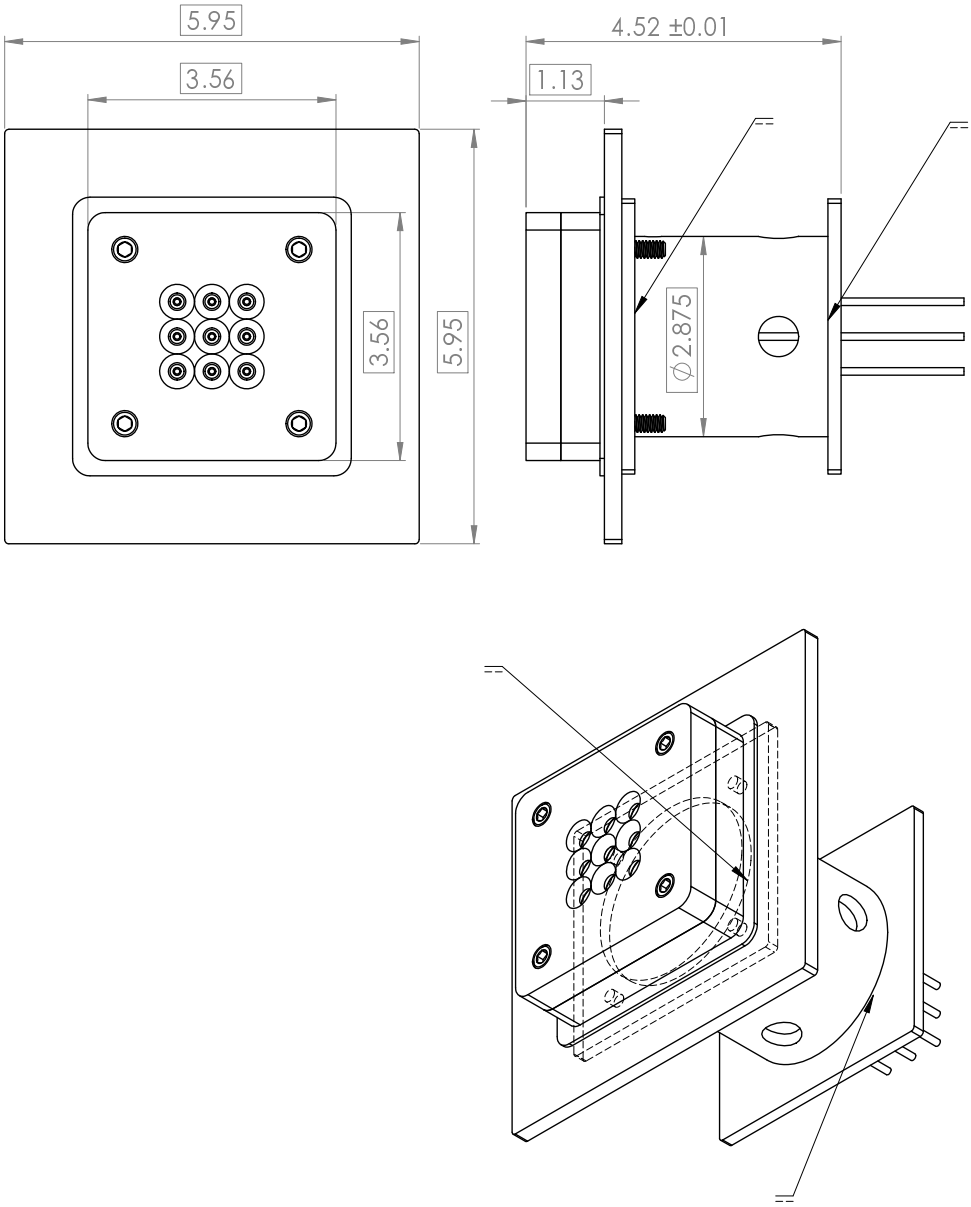
[116] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, pp. 158–162, Jan. 2010.

[117] T. Simila and J. Tikka, "Multiresponse sparse regression with application to multidimensional scaling," in *Artificial neural networks: Formal models and their applications ICANN 2005* (W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, eds.), no. 3697 in Lecture Notes in Computer Science, pp. 97–102, Berlin, Heidelberg: Springer, 2005.

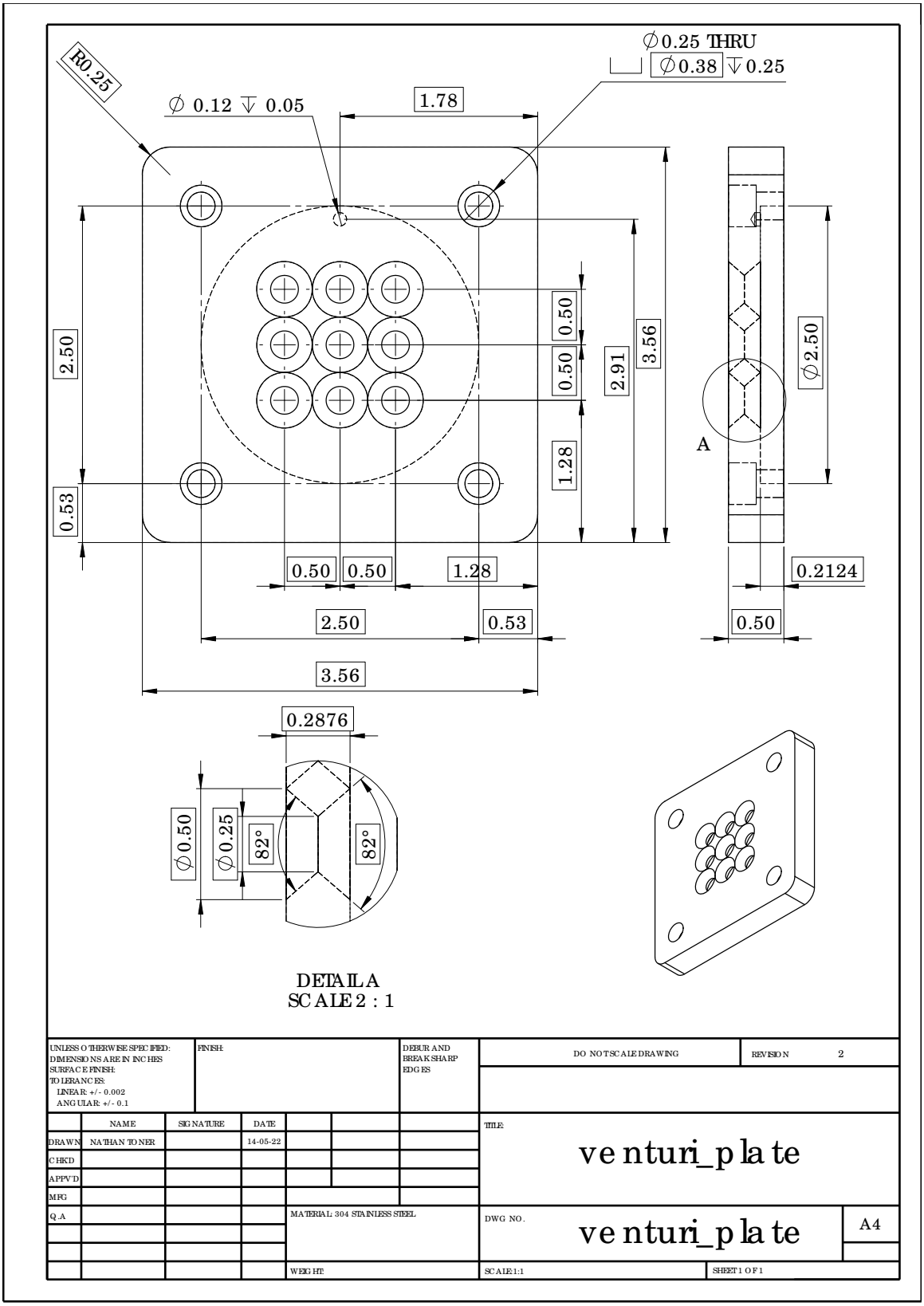
APPENDICES

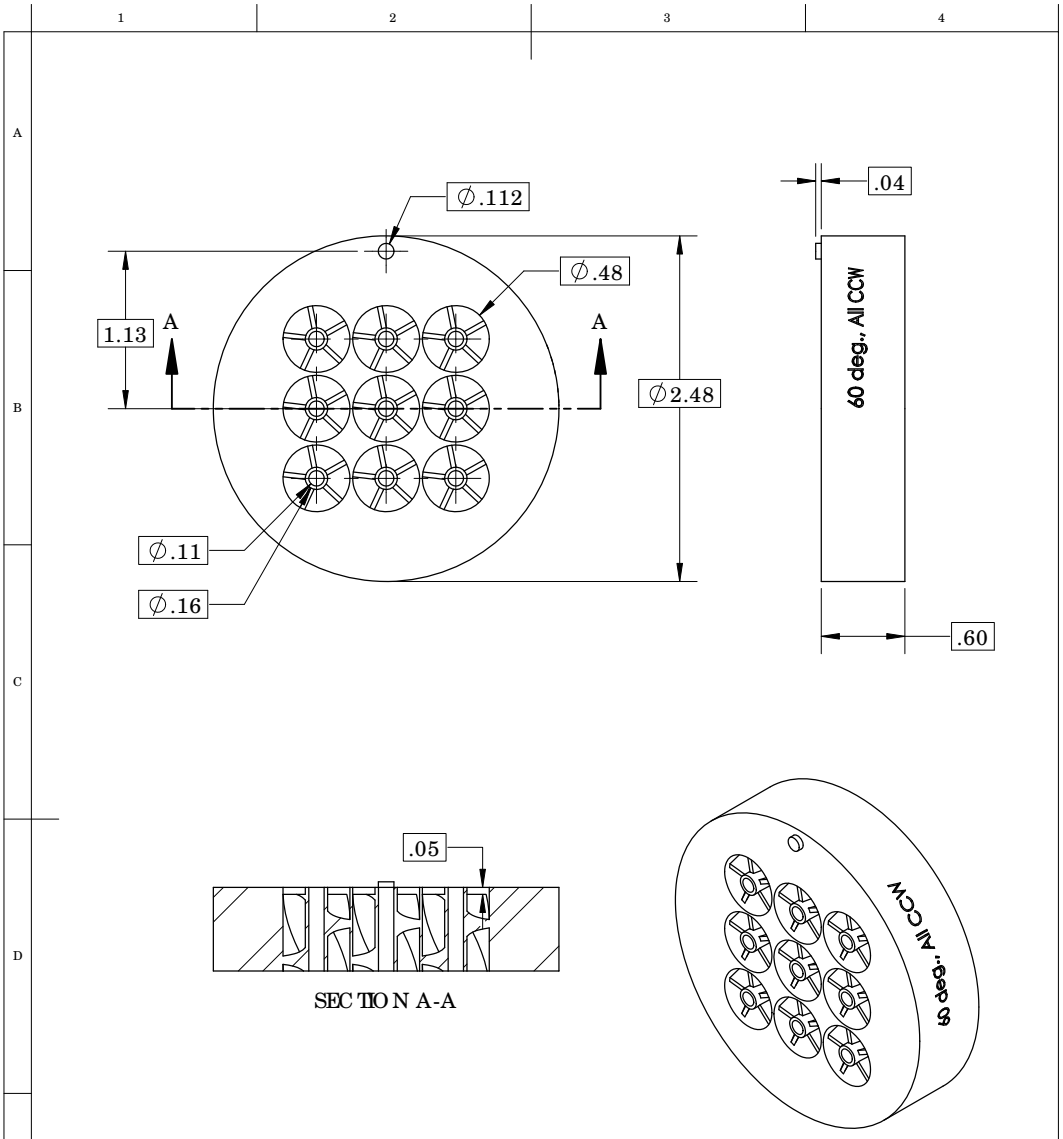
A. 9-ELEMENT BURNER DRAWINGS

Mechanical drawings of the components of the nine-element burner are included in the following. These drawings were used to manufacture the burner primarily in the mechanical engineering department's student machine shop, with exceptions noted in Chapter 3.

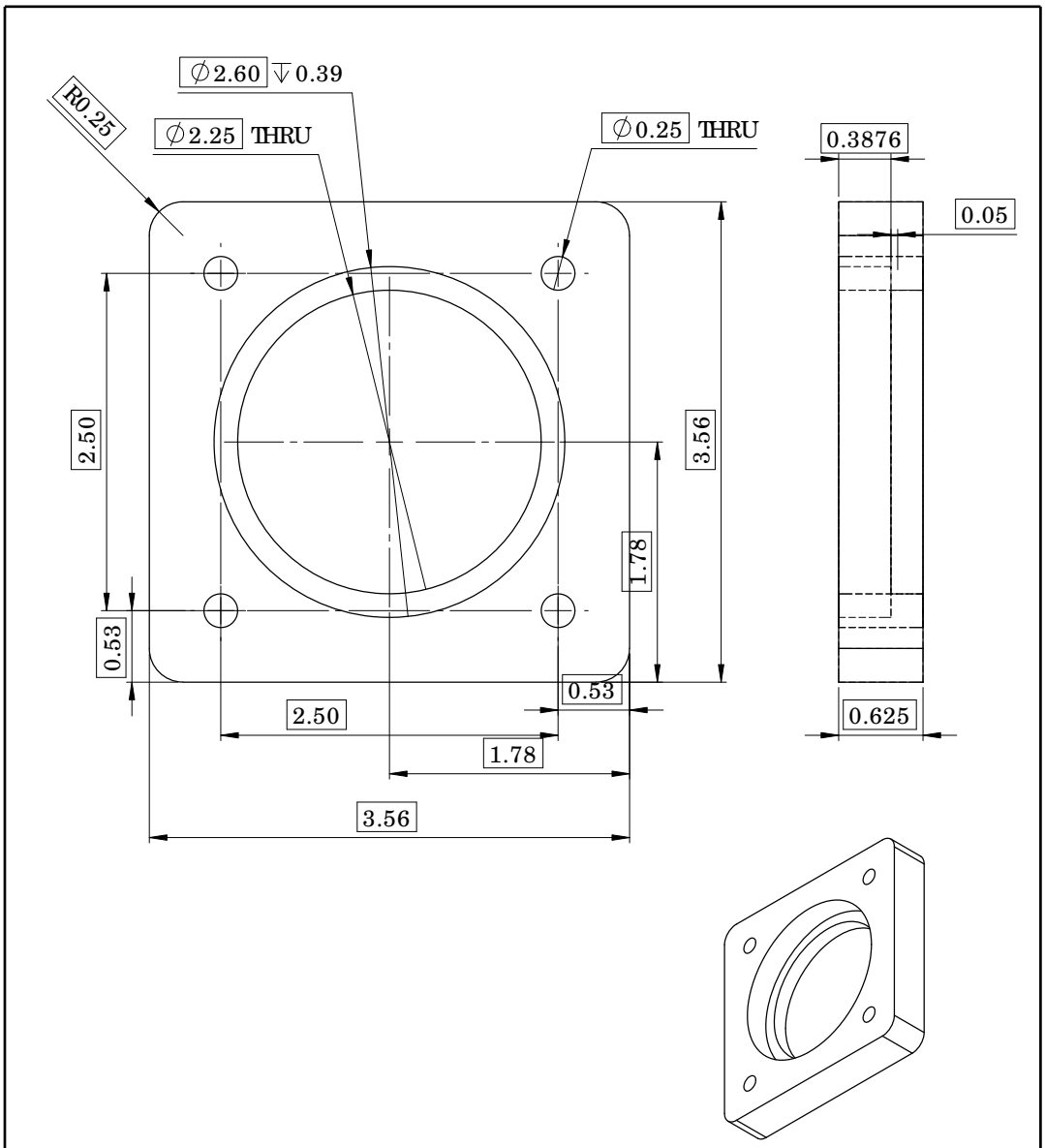


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +/-0.005 ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION 1
	NAME	SIGNATURE	DATE		TITLE:	9burner_square_assy
DRAWN	NATHAN TONER		14-02-14			
CHK'D						
APPV'D						
MFG						
Q.A				MATERIAL:	DWG NO.	9burner_square_assy A4
				WEIGHT:	SCALE:1:5	SHEET 1 OF 1

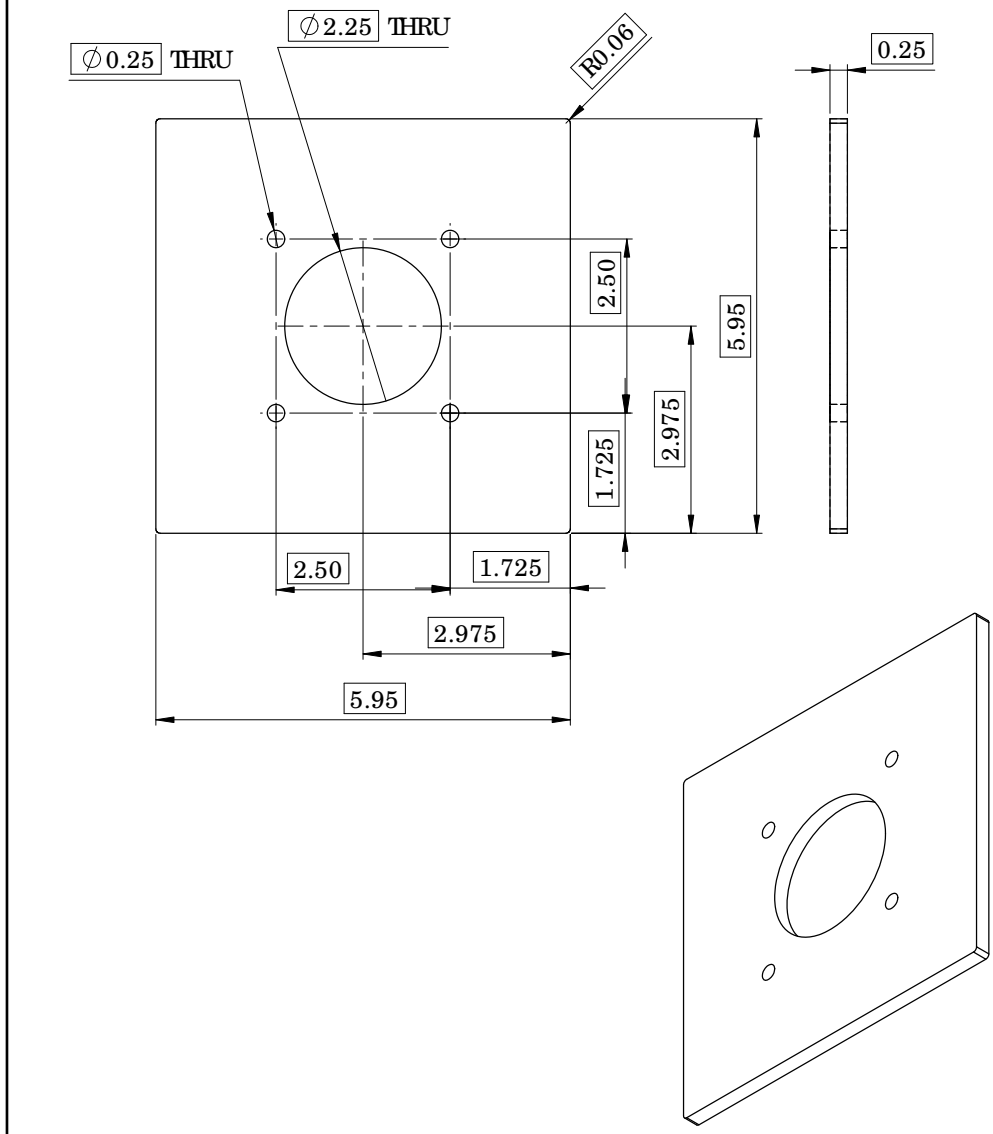

R0.25
∅0.25 THRU
∅0.38 ↧0.25
∅ 0.12 ↧ 0.05
1.78
2.50
0.53
0.50
0.50
2.91
3.56
1.28
A
∅2.50
0.50
0.50
1.28
2.50
0.53
3.56
0.2124
0.50
0.2876
∅0.50
∅0.25
82°
82°
DETAIL A
SCALE 2 : 1
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH:
TOLERANCES:
LINEAR: +/- 0.002
ANGULAR: +/- 0.1
FINISH:
DEBUR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION 2
NAME
SIGNATURE
DATE
DRAWN
NATHAN TONER
14-05-22
CHKD
APPV'D
MFG
Q.A
TITLE:
venturi_plate
MATERIAL: 304 STAINLESS STEEL
DWG NO.
venturi_plate
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1



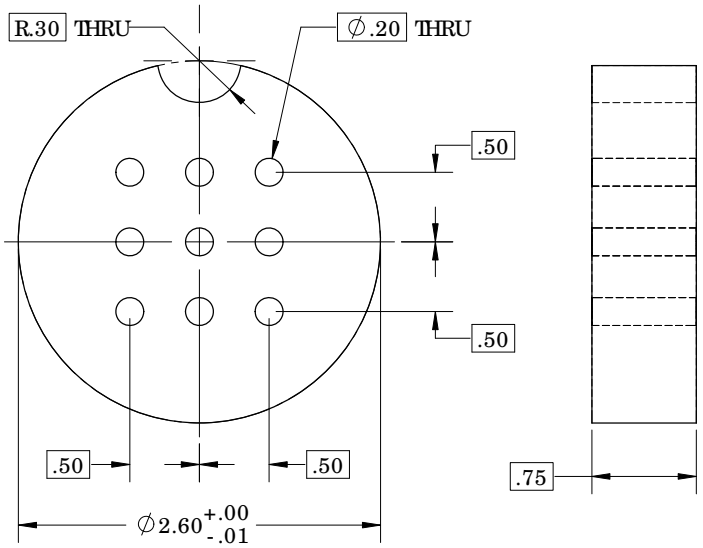
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION 0
	NAME	SIGNATURE	DATE		TITLE: swirler_60deg	
DRAWN	NATHAN L TONER		2015-03-30			
CHKD						
APPV'D						
MFG						
Q.A				MATERIAL: 3D-PRINTED RESIN	DWG NO. swirler_60deg	A4
				WEIGHT:	SCALE:1:1	SHEET 1 OF 1

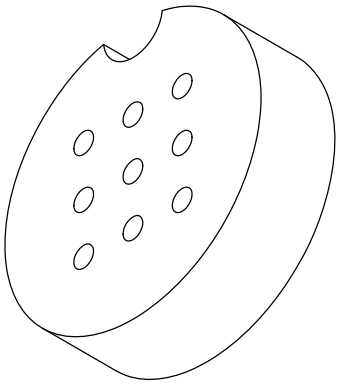


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +/- 0.002 ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION	1
	NAME	SIGNATURE	DATE		TITLE: spacer_plate		
DRAWN	NATHAN TONER		14-05-22				
CHKD							
APPV'D							
MFG							
Q.A				MATERIAL: LOW-CARBON STEEL	DWG NO. spacer_plate		A4
				WEIGHT:	SCALE:1:1	SHEET 1 OF 1	

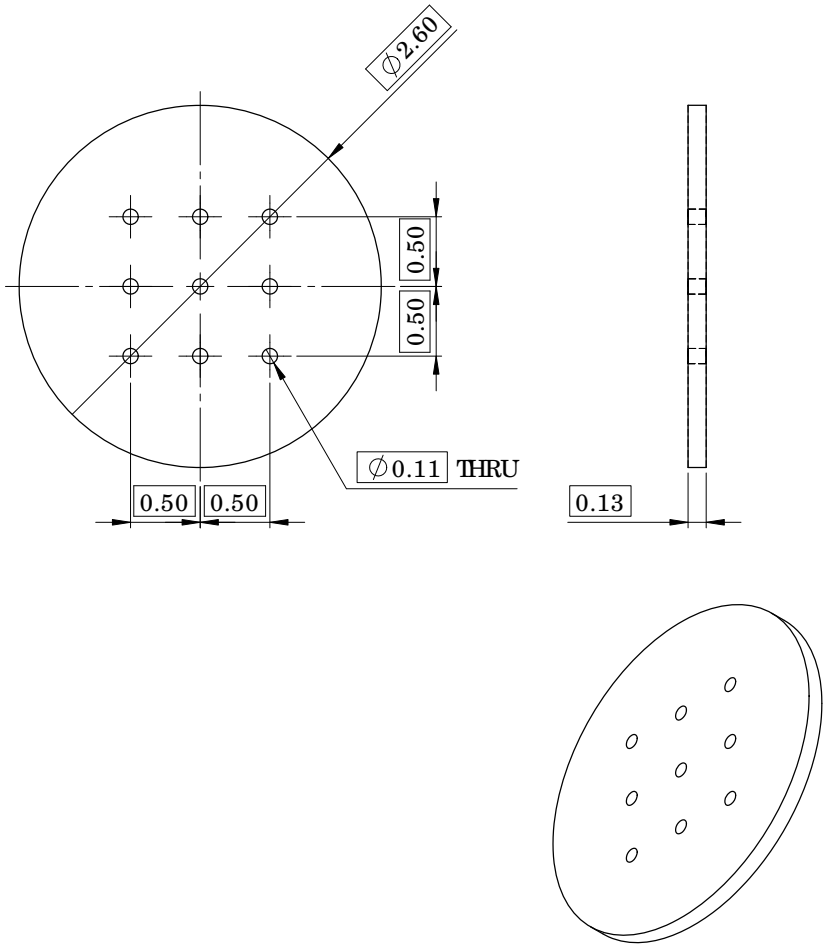


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +/- 0.005 ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION	0
	NAME	SIGNATURE	DATE		TITLE: mounting_plate		
DRAWN	NATHAN TONER		14-02-14				
CHKD							
APPV'D							
MFG							
Q.A				MATERIAL: LOW-CARBON STEEL	DWG NO. mounting_plate		A4
				WEIGHT:	SCALE:1:2	SHEET 1 OF 1	

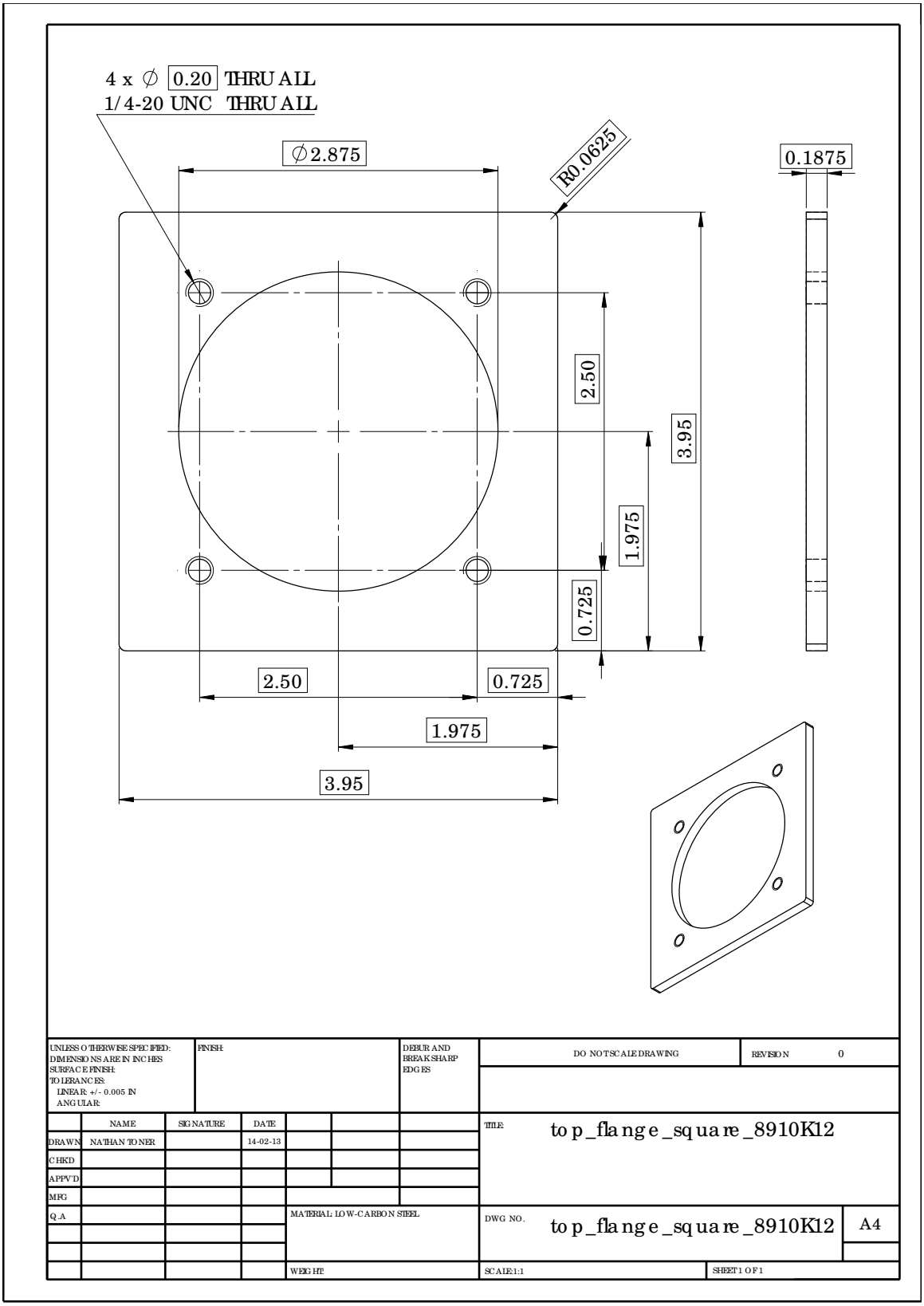


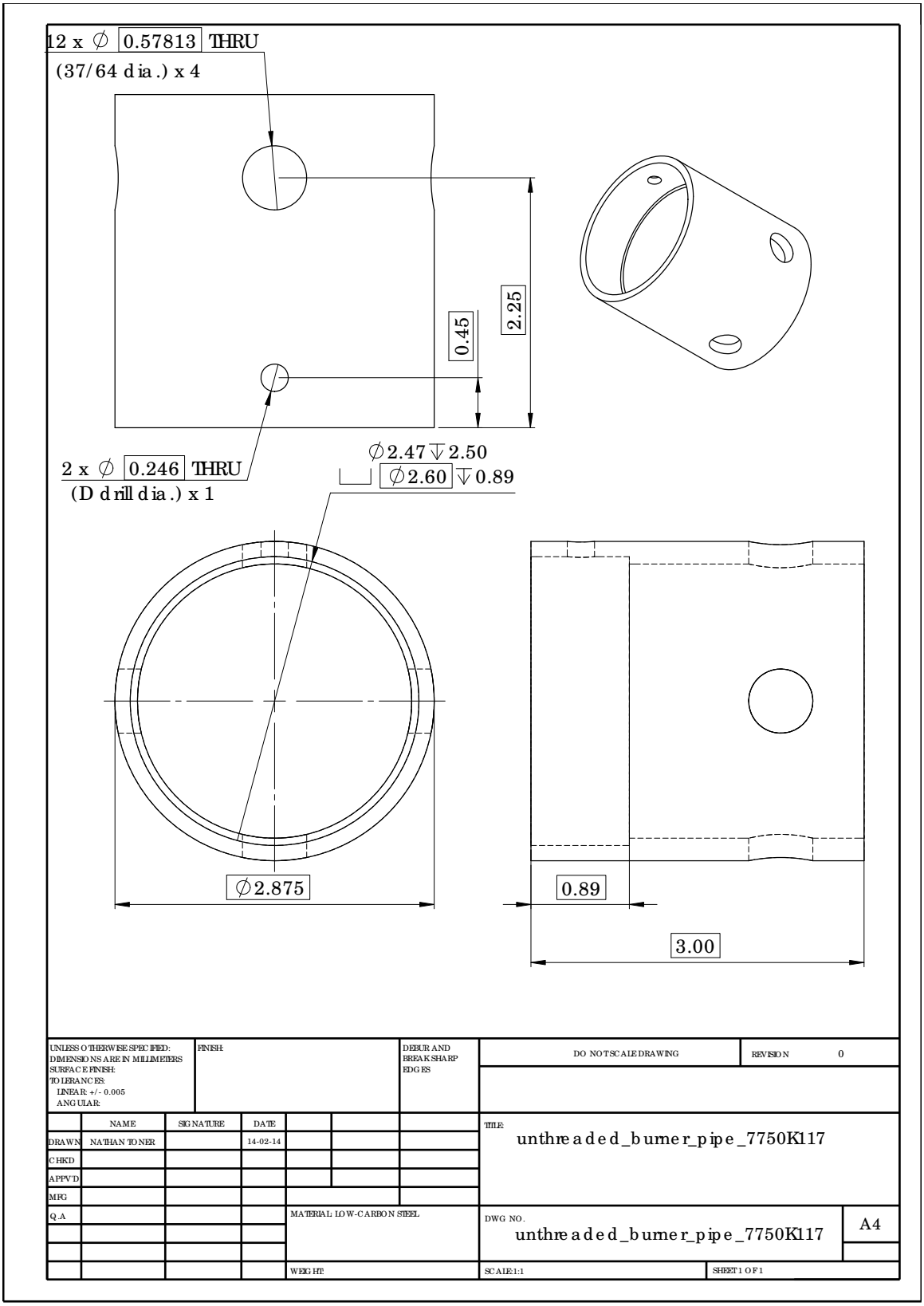


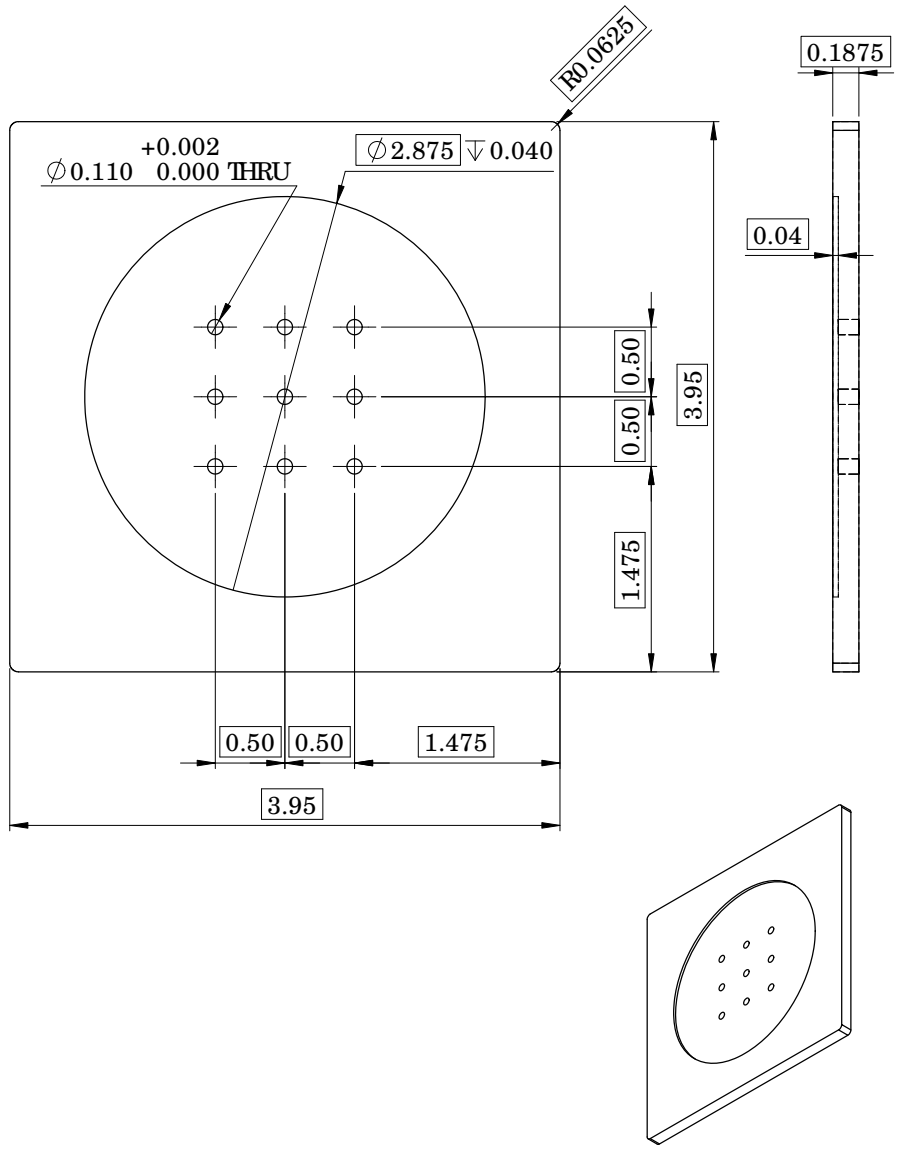
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +- 0.005 ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION 4
	NAME	SIGNATURE	DATE		TITLE:	
DRAWN	Nathan L Toner		2014-06-13		flow_straightener	
CHKD						
APPV'D						
MFG						
Q.A				MATERIAL: Al MESH, 1/8" x 0.002"	DWG NO. flow_straightener	A4
				WEIGHT:	SCALE:1:1	SHEET 1 OF 1



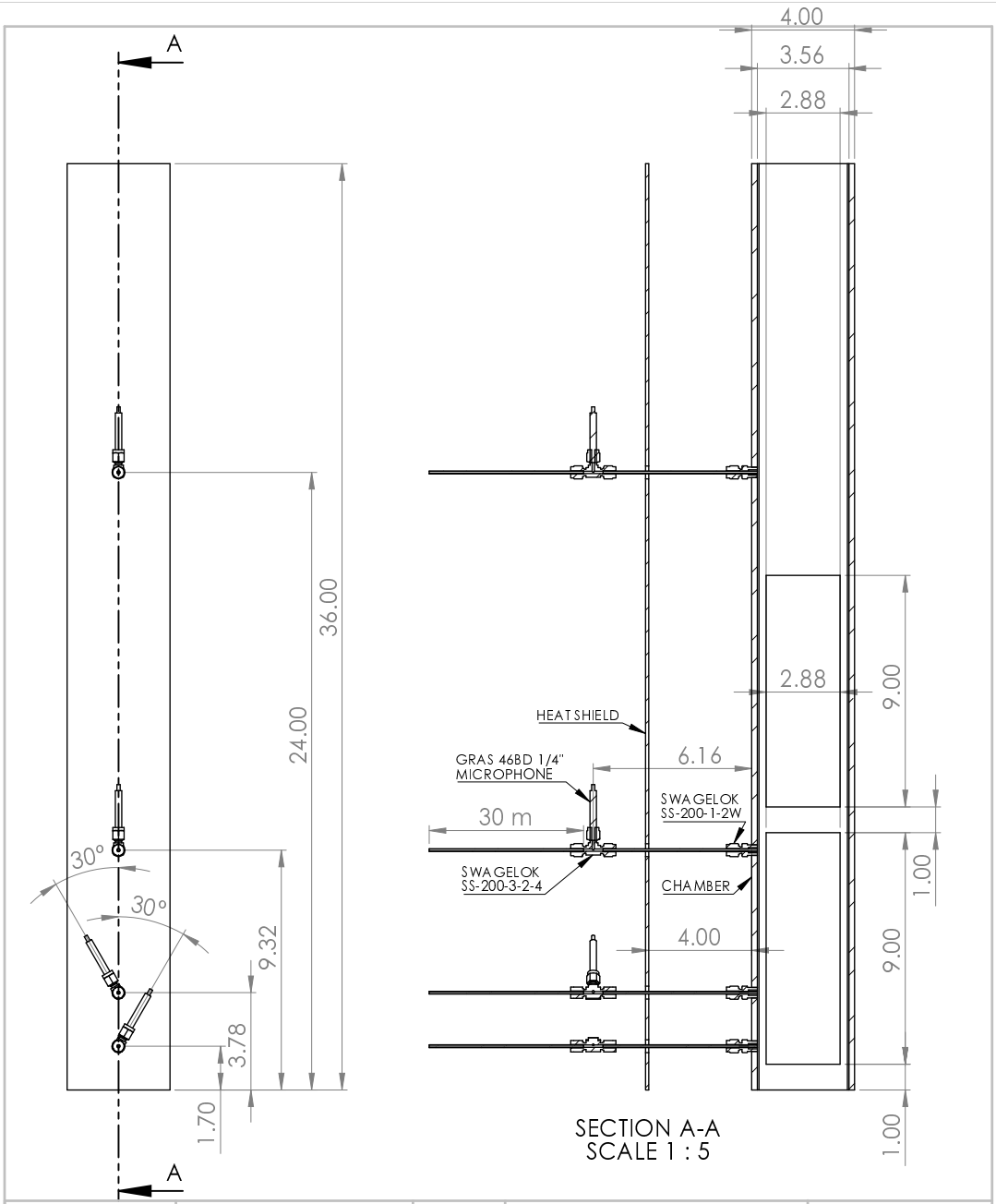
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +- 0.005 ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION 0
	NAME	SIGNATURE	DATE				TITLE: sintered_plate
DRAWN	Nathan L Toner		2014-06-13				
CHKD							
APPV'D							
MFG							
Q.A				MATERIAL: SINTERED 304 STAINLESS STEEL		DWG NO. sintered_plate	A4
				WEIGHT:		SCALE:1:1	SHEET 1 OF 1


4 x ∅ 0.20 THRU ALL
1/4-20 UNC THRU ALL
∅2.875
R0.0625
0.1875
2.50
3.95
1.975
0.725
2.50
0.725
1.975
3.95
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH:
TOLERANCES:
LINEAR: +/- 0.005 IN
ANGULAR:
FINISH:
DEBUR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION 0
NAME
SIGNATURE
DATE
DRAWN NATHAN TONER 14-02-13
CHKD
APPV'D
MFG
Q.A
TITLE top_flange_square_8910K12
MATERIAL: LOW-CARBON STEEL
DWG NO. top_flange_square_8910K12
A4
WEIGHT
SCALE:1:1
SHEET 1 OF 1


12 x ∅ 0.57813 THRU
(37/64 dia.) x 4
0.45
2.25
2 x ∅ 0.246 THRU
(D drill dia.) x 1
∅2.47 ↧2.50
∅2.60 ↧0.89
∅2.875
0.89
3.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR: +/- 0.005
ANGULAR:
FINISH:
DEBUR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION 0
NAME
SIGNATURE
DATE
DRAWN NATHAN TONER 14-02-14
CHKD
APPV'D
MFG
Q.A
TITLE:
unthreaded_burner_pipe_7750K117
MATERIAL: LOW-CARBON STEEL
DWG NO.
unthreaded_burner_pipe_7750K117
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: +/- 0.005 IN ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES			DO NOT SCALE DRAWING	REVISION 0
	NAME	SIGNATURE	DATE				TITLE:	
DRAWN	NATHAN TONER		14-02-13					
CHKD								
APPV'D								
MFG								
Q.A				MATERIAL: LOW-CARBON STEEL			DWG NO. bottom_endcap_square_8910K12	A4
				WEIGHT:			SCALE:1:1	SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING		REVISION: 0
	NAME	SIGNATURE	DATE		TITLE: chamber_assy		
DRAWN	NATHAN L TONER		2016-05-05				
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL: STAINLESS STEEL	DWG NO.		A4
				WEIGHT:	SCALE:1:20	SHEET 1 OF 1	

B. MASS FLOW CONTROLLER STEP RESPONSES

Step responses of the 40 SLPM fuel mass flow controllers are shown in Figures B.1–B.3. Note that the step responses for the pilot and middle fuel MFCs were taken by changing the input to the controller from 0 V to 2 V and recording their responses. These two MFCs exhibit some time delay when turning on from zero input, but exhibit close to first-order dynamics after that. The outer fuel line MFC, in contrast, exhibited excessive nonlinearity when turning on from zero input, and so in experiments a small (< 0.5 V) minimum input was maintained for this MFC.

When changing flow rates from non-zero operating conditions, all mass flow controllers exhibit approximately first-order dynamics with very little time delay, similar to that shown in Figure B.3.

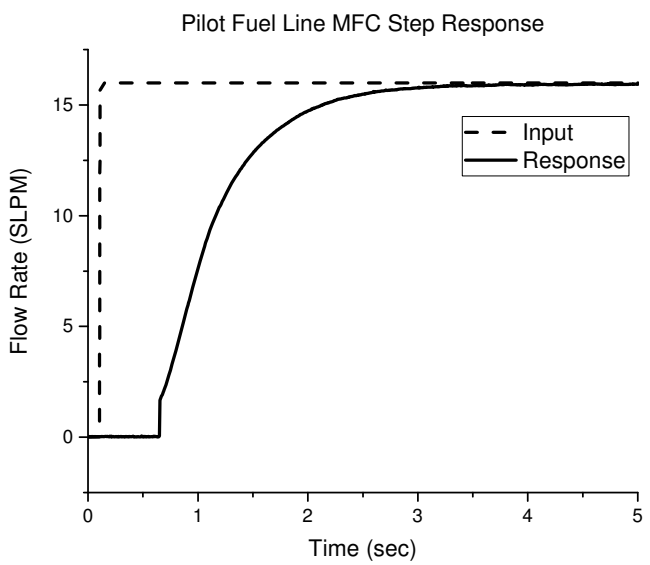


Figure B.1. 2 V step response of the pilot mass flow controller.

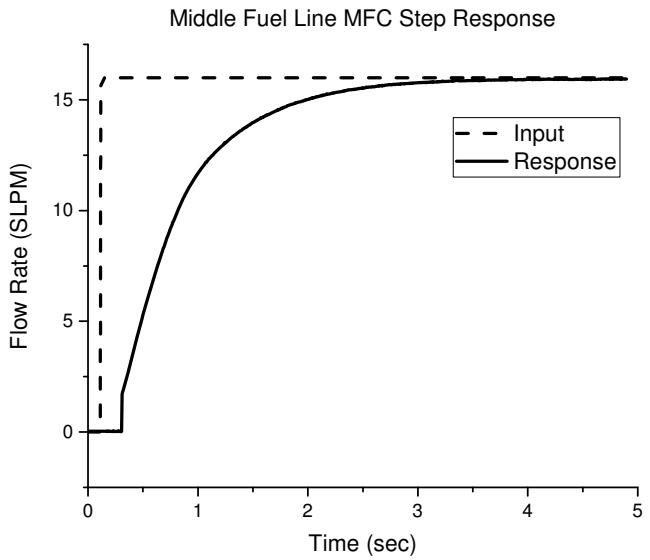


Figure B.2. 2 V step response of the middle mass flow controller.

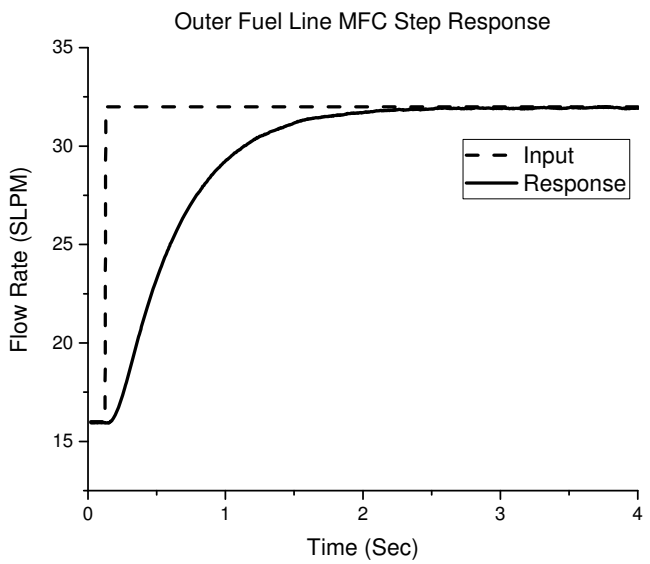


Figure B.3. 2 V step response of the outer mass flow controller. Note that this step response was taken from a non-zero initial steady-state condition by changing the input from 2 V to 4 V. This was done to avoid excessive nonlinearity in the outer fuel line mass flow controller observed when turning on.

VITA

VITA

Nathan Toner was born in Rochester, NY on January 28, 1985. After a disappointingly uneventful early childhood, he was pressed into service at the age of five as a gold minder in northern Ontario, where through cunning and business acumen uncharacteristic of one so young, he quickly rose to the top of the mining company that had enslaved him. Using his now vast fortune, he relocated an indigenous population of woolly mammoth to the hidden tropical crater located at the north pole; saving them from the looming threat of extinction and earning the title "Nate the Great". Having thus bankrupt himself, and reaching an age appropriate for college, he attended Purdue University in West Lafayette, IN, where he received a bachelors of science in mechanical engineering in 2008. He then worked for Cooper Power Systems in Milwaukee, WI and Greenwood, SC for two years until returning to Purdue in 2010 to pursue a doctorate in intelligent controls. In graduate school, Nate the Great fell into obscurity and has not made any additional significant contribution to global welfare to the present day. Some say he is only biding his time...