DEVELOPMENT OF A HIGH-TEMPERATURE, HI GH-PRESSURE, OPTICALLY ACCESSIBLE FLOW VESSEL AND SUBSEQUENT STUDY OF *n*-HEPTANE USING HIGH-SPEED VISUALIZATION TECHNIQUES

by

KEMAR CLIFTON JAMES

BRIAN FISHER, COMMITTEE CHAIR
AJAY K. AGRAWAL
MARCUS D. ASHFORD
SEMIH M. OLCMEN

A THESIS

Submitted in Partial fulfillments of the requirements
for the degree of Master of Science
in the Department of Mechanical Engineering
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2013

ABSTRACT

The focus of this work was to develop a continuous-flow vessel with extensive optical access for characterization of engine-relevant fuel-injection and spray processes. The spray chamber was designed for non-reacting experiments at pressures up to 1380 kPa and temperatures up to 200°C, which are characteristic of early direct-injection low-temperature combustion in diesel engines. Continuous flow of inert "sweep gas" enables acquisition of large statistical data samples and thus potentially enables characterization of stochastic spray processes. A custom flange was designed to hold a common-rail diesel injector, with significant flexibility to accommodate other injectors and injector types. This flexibility, combined with the continuous flow through the chamber, may enable studies of gas-turbine direct-injection spray processes in the future. Overall, the user can control and vary: injection duration, injection pressure, sweep-gas temperature, sweep-gas pressure, and sweep-gas flow rate. The user also can control frequency of replicate injections.

There are four flat windows installed orthogonally on the vessel for optical access. Optical data, at present, include global spray properties such as liquid-phase fuel penetration and cone angle. These measurements are made using a high-speed spray visualization system consisting of a fast-pulsed LED (light emitting diode) source and a high-speed camera. Experimental control and data acquisition have been set up and synchronized using custom LabVIEW programs. The culmination of this development effort was an initial demonstration experiment to capture high-speed spray visualization movies of n-heptane injections to determine liquid-phase fuel penetration length and spray cone angle. In this initial experiment, fuel-

injection pressure was ~120 MPa and the injection command pulse duration was 800 µs. At room conditions, liquid length and nominal spray cone angle were ~170 mm and ~14.5°, respectively. In contrast, with air flow in the chamber at 100 psi and 100°C, liquid length was considerably shorter at ~92 mm and spray cone angle was wider at ~16.5°. Future experiments will include the continuation of these measurements for a wider range of conditions and fuels, extension of high-speed imaging to vapor-phase fuel penetration using schlieren imaging techniques, and detailed characterization of spray properties near the injector nozzle and near the liquid length.

LIST OF ABBREVIATIONS AND SYMBOLS

BTU	British Thermal Units
GDI	Gasoline Direct Injection
TDC	Top Dead Center
LTC	Low-Temperature Combustion
CI	Compression-Ignition
HCCI	Homogenous Charge Compression-Ignition
SCCI	Stratified Charge Compression-Ignition
RCCI	Reactivity Controlled Compression-Ignition
IAGUI	Image Analysis Graphical User Interface
VI	Virtual Instrument
Sub-VI	Lower Level Virtual Instrument
DI	Direct Injection
DOI	Duration of Injection
GUI	Graphical User Interface
LED	Light Emitting Diode
IPT	Image Processing Toolbox
SADI	Stand-Alone Direct-Injection (driver)
SCFM	Standard Cubic Feet Per Minute

ACKNOWLEDGEMENTS

This research has been an amazing chapter of my life that would not have been possible without the support of key individuals who believed in me.

For their effort in making my dream one step closer I would like to give thanks to my friends and family for being my rock and motivation. I would also like to thank faculty adviser Dr. Fisher for taking me under his wing and putting up with my incessant questions, which were sometimes irrelevant but regardless of how trivial they were, he placed his best foot forward.

I would also like to express my gratitude to my committee members Dr. Ashford, Dr. Agrawal and Dr. Olcmen. Further gratitude is extended to the Department of Mechanical Engineering Graduate School for funding this training opportunity.

Finally this work and chapter of my life would not have been possible without continual support and encouragement from my extended family, fellow graduate students, friends and lab mates. You guys have definitely made the journey worthwhile and for all that you have done I ebulliently give thanks.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 MOTIVATION

Energy consumption is one of the primary areas that have significantly altered the way of life and the environment. The burning of fossil fuels to produce energy has had adverse effects on the environment due to the emission of toxic gases such as NOx , SOx and CO, in addition to unburned hydrocarbons and particulate matter [1-4]. Some of these gases not only contribute to global warming but also have been unequivocally linked to multiple health conditions including respiratory infections, heart disease and lung cancer [5-8] as seen in Figure 1.1. Mitigating these effects means not just meeting the energy demand but doing so more efficiently and cleanly, thus minimizing or eliminating these emissions for the same quantity of fuel burned.

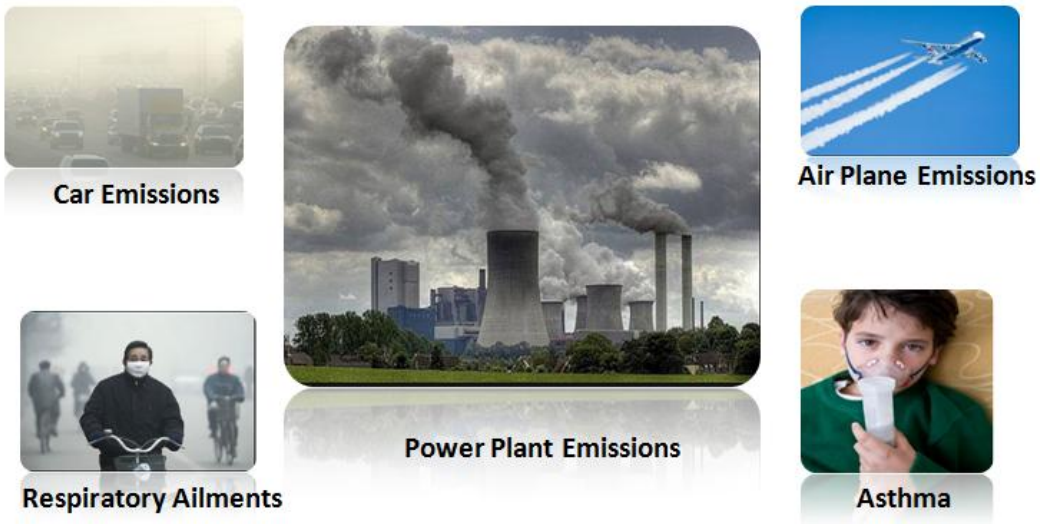

Figure 1.1: Various sources of emissions along with two health issues associated with air pollution.

Several means are currently in progress to combat this issue such as the use of renewable, nuclear, and other alternative resources. Unfortunately these alternative energy sources are unable to meet the U.S. or the world energy demand. In 2008 the world energy consumption was approximately 524 quadrillion BTUs, and it's expected to expand 56% between 2010 and 2040 [9], with fossil fuel projected to provide 80% of the energy for consumption. Therefore, the optimal and efficient use of fuels is currently an exciting area of research. One area in particular is the application of direct fuel injection across several sectors such as transportation and power generation, since these sectors combined account for over 68% of the U.S. energy consumption [10].

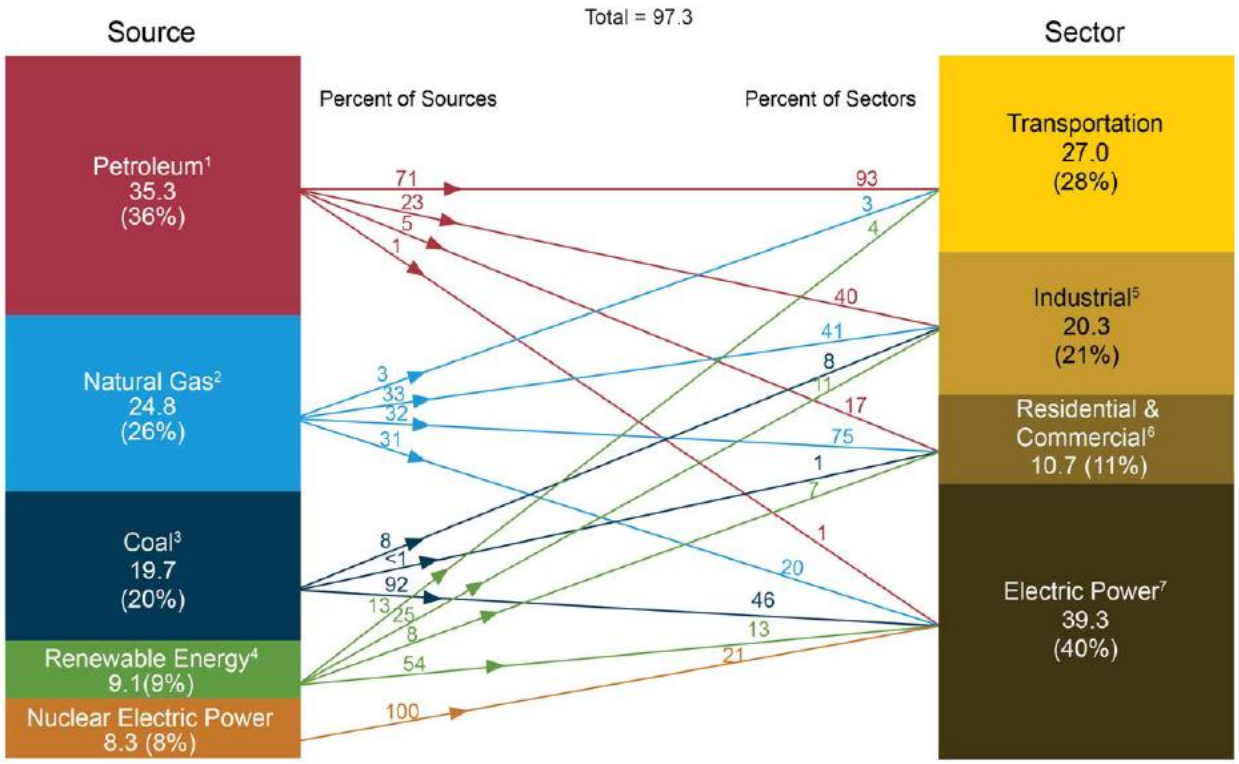

Figure 1.2: Primary energy consumption by source and sector, 2011 (Quadrillion Btu) [10]

Fuel-injection sprays are prevalent in practical combustion devices, including diesel and gas-turbine engines. In addition, spark-ignition engines are increasingly employing GDI (gasoline direct-injection) fuel sprays. It is therefore necessary to develop a greater understanding of fuel sprays, including break-up of the liquid fuel into droplets, entrainment of surrounding gas, droplet vaporization, penetration of liquid- and vapor-phase fuel, and fuel-air mixing. These processes set the initial conditions for combustion, and thus significantly affect emissions, fuel-flexibility, and efficiency of practical combustion systems.

Figure 1.3: Various applications of fuel injection, ranging from diesel and gasoline engine to gas turbine.

1.2 PLATFORMS USED TO STUDY FUEL SPRAYS

To probe the details of engine-relevant spray processes, it is necessary to have an experimental apparatus that replicates or at least approaches engine conditions. Furthermore, it is necessary to have optical access to the fuel spray. There are many types of systems that meet

these requirements. A useful review and discussion of engine-relevant experimental systems, with comparisons of their relative advantages and disadvantages, can be found in the literature [11]. Among the most useful and relevant systems for studies of fuel sprays are optical engines (*e.g.*, [12-15]), constant-volume chambers (*e.g.*, [16-19]), and constant-pressure flow vessels (*e.g.*, [20,21]).

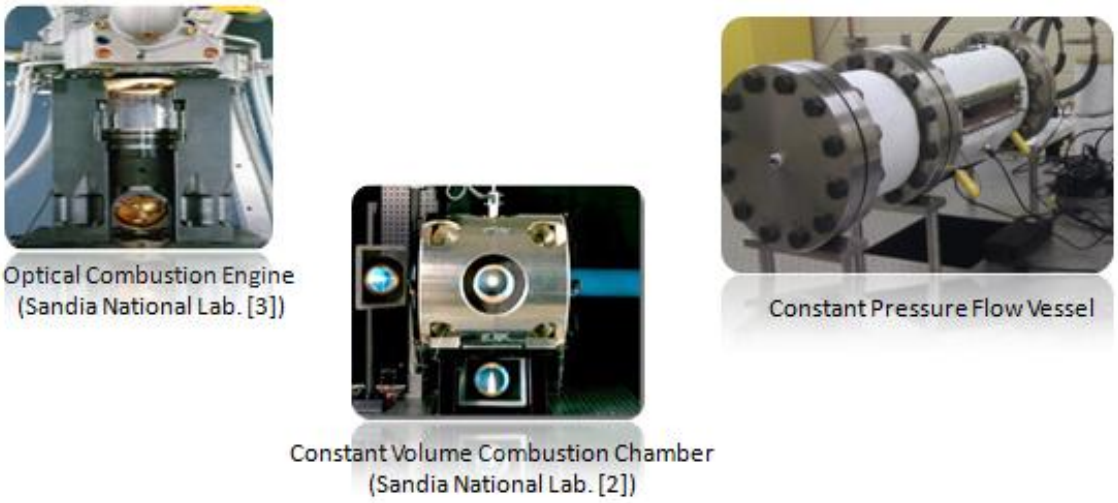

Figure 1.4: Common systems used to study the fuel spray and combustion characteristics

Optical engines most closely replicate engine conditions, but typically have limited optical access, are difficult to operate, and are difficult to model. Constant-volume chambers and constant-pressure flow vessels are more fundamental than engines, enable careful control of thermodynamic conditions, and allow extensive optical access. They remove transient conditions of a normal operating engine, allowing researchers to better isolate and observe the chemical and the fluid-mechanic processes. Further simplifying matters by introducing the spray under non-reacting conditions enables observation of fundamental physical properties of the spray using high-speed imaging and optical techniques. It is difficult to build up large statistical data sets in a *constant-volume chamber*, however, because the chamber must be exhausted and reset

before each injection event. On the other hand, it is typically more difficult to replicate engine-relevant thermodynamic conditions in a *constant-pressure flow vessel*, because it requires the flow gas to be pre-compressed and pre-heated upstream of the optical test section.

1.3 PREVIOUS WORK ON FUEL SPRAYS

Regardless of the type of experimental system, optical access enables detailed studies of sprays. Fuel sprays involve complex physical processes, which can be characterized in terms of local or global parameters. Local parameters include primarily droplet characteristics, such as size, velocity, and concentration. Droplet characteristics and behaviors have mostly been ignored, due to the prevalent belief that liquid-fuel vaporization is driven by entrainment of surrounding gas into the fuel jet and mixing of the gas with the fuel [22]. Under the so-called "mixing-limited vaporization" assumption, global spray properties such as cone angle (spreading angle) and liquid and vapor-phase penetration can be related to properties of the fuel and the ambient gas [15,22-28].

Naber and Seibers [22], in a constant-volume vessel, used schlieren imaging to observe and measure the spray penetration and spreading angle. The results evidently showed that there was strong correlation between ambient-gas density and the spray behavior, which was corroborated by prior research articles [24-28]. The articles also indicated a dependence of spray penetration on the ambient-gas density but exact nature of the relationship varied. In successive work, Siebers, used a combination of Mie scattering and schlieren imaging to extrapolate information on the liquid and vapor-phase fuel penetration [18]. Siebers demonstrated the

dependence (or independence) of liquid penetration length on parameters such as injection pressure, orifice diameter, fuel volatility, and ambient-gas temperature and density.

The liquid-phase penetration for fuel sprays under traditional diesel-engine conditions have also been extensively characterized for injection occurring near top dead center (TDC) [29-37]. Injection event occurring near TDC is associated with high and relatively steady ambient gas temperatures and densities. These conditions are similar to those studied in a constant-volume vessel, constant-volume chamber and even the optical accessible engine. Regardless of three common type of experimental system used to observe the global spray properties near TDC injection, there were several common spray trends observed throughout these works:

I. Liquid penetration length is independent of fuel-injection pressure.
II. Liquid penetration length increases linearly with orifice diameter.
III. Liquid penetration lengths are shorter for higher ambient gas densities and temperatures.
IV. Liquid penetration length increases as fuel volatility decreases
V. Spray penetration rate is slower when the ambient-gas density is higher
VI. The vapor-phase length extends beyond the maximum penetration length of the liquid-phase which oscillates about a mean value until the end of injection.

From the breadth of work presented, the maximum extent of liquid-phase fuel penetration or "liquid length," is arguably the most important global spray property. For a given engine and injector geometry, liquid length determines if liquid fuel impinges on the cylinder wall. Wall wetting is undesirable, as it contributes to lubrication dilution, reduced combustion efficiency,

and increased emissions [38-45]. Directly related to fuel penetration is the cone angle of the spray. A more-dispersed fuel spray, which has a wider cone angle, involves more entrainment of ambient gas and thus shortens the liquid length [22]. Vapor-phase fuel penetration is also important, because it affects overall fuel-air mixing in the combustion chamber and thus affects combustion efficiency and emissions (*e.g.*, [46]).

The conventional assumption about the nature of diesel sprays, namely the mixing-limited-vaporization assumption, has been applied to standard diesel-combustion conditions for quite some time. Recent developments, however, including advanced engine strategies (*e.g.*, low-temperature combustion, or LTC, by the means of early direct injection) and the advent of alternative fuels, have made additional research necessary.

1.4 LOW-TEMPERATURE COMBUSTION (LTC)

Low-temperature combustion refers to a class of engine-combustion strategies targeting cleaner and more efficient operation. The concept is illustrated in Figure 1.5. Conventional spark-ignition (SI) engines typically operate at near-stoichiometric conditions, where high temperatures lead to high NOx production. Conventional compression-ignition (CI) engines (*i.e.*, diesel engines), operate at a range of conditions simultaneously that tend to lead to high production of both soot and NOx. In addition, if temperatures become too low, conversion of CO to CO_2 suffers. Thus, the concept of LTC is to operate the engine in a region that is sufficiently lean to lower soot production, sufficiently low-temperature to reduce NOx production, and yet sufficiently high-temperature to maintain CO-to-CO_2 conversion [47]. There are several LTC

strategies that have been or continue to be topics of extensive research, including HCCI (homogenous charge compression-ignition), SCCI (stratified charge compression-ignition), and RCCI (reactivity controlled compression-ignition). One strategy for LTC in diesel engines is known as early direct-injection.

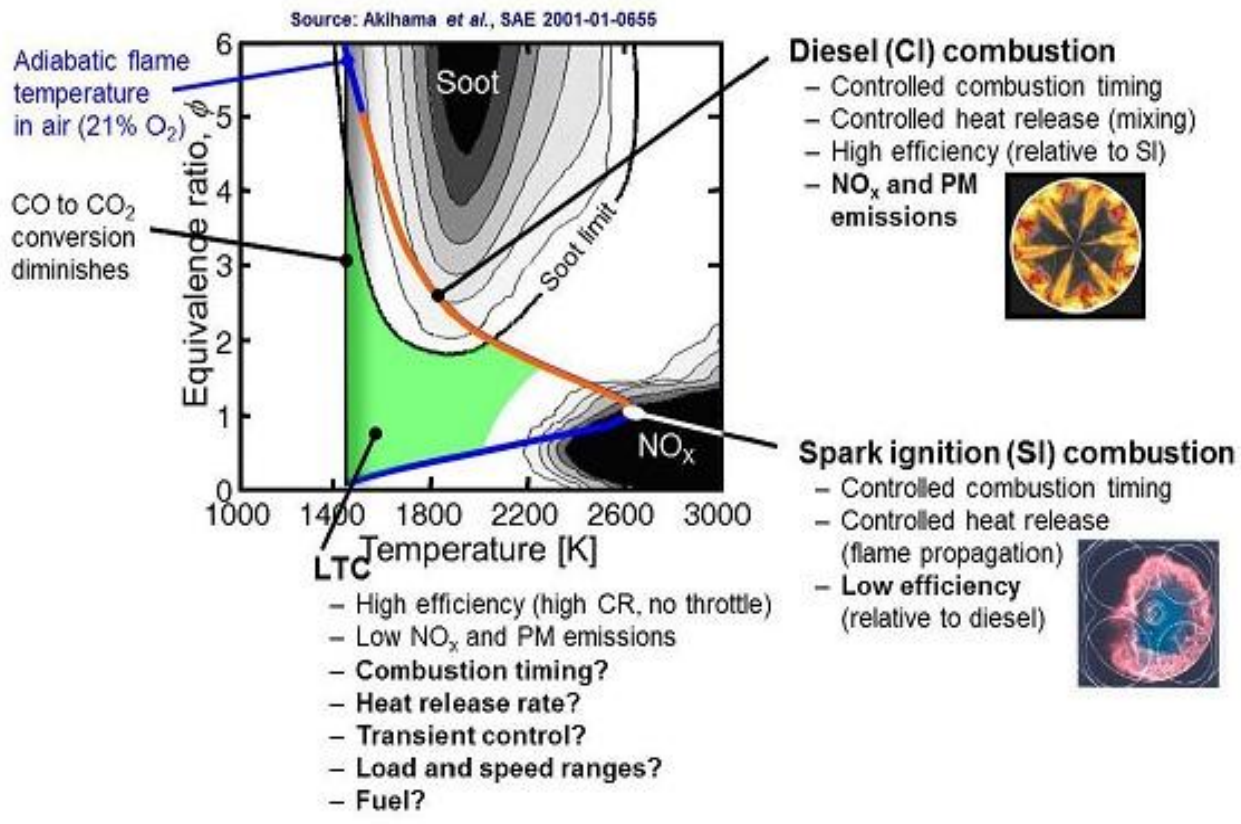

Figure 1.5: Illustration of low-temperature combustion (LTC) compared to conventional engine combustion (adapted from Akihama et al. [47]).

Early direct-injection LTC differs from conventional diesel combustion in several ways. First, fuel is directly injected earlier in the compression stroke compared to conventional practices, where the fuel is injected near TDC at density and temperature ranging from 20-30 kg/m^3 and (~900-1000 K) respectively. In the advanced LTC strategy, the injection event occurs at 30° to 70° before TDC at lower density and temperature ranging from ~5-10 kg/m^3 and 400-

500 K respectively. This increases pre-combustion mixing and simultaneously reduces particulate matter emissions and peak temperatures during combustion, thus minimizing nitrogen oxides formation [48-57]. Despite the benefits of this method compared to the conventional method, fuel is injected at lower in-cylinder ambient conditions where vaporization is poor and liquid impingement upon the cylinder liner and piston bowl are more likely to occur [58,59].

To attain suitable LTC performance wall impingement must be overcome, while achieving the low emission and efficiency gains of combustion [59]. Due to the absence of quantitative data in this area of early injection, however, successful application of this strategy is not yet fully achieved. In contrast, the conventional combustion method has extensive data that fully characterize diesel sprays and several blends. This archive of data even led to the development of a scaling law by Siebers [22], which predicts steady liquid penetration lengths quite accurately for a variety of fuels under a wide range of injection and ambient conditions relevant to conventional diesel engine operation. These data for the fuel spray characteristics have also contributed to the development of high-fidelity and time-resolved multidimensional models that have been extensively validated and used to predict compression-ignition and spark-ignition engine performance and emissions [60-70]. The goal of dedicated researchers heavily invested in LTC is to acquire this breadth of data and develop such models that can bridge the gap of knowledge and provide pertinent insight to allow effective implementation of the early direct-injection LTC strategy.

1.5 SUMMATION

One method of addressing the need for quantitative data relevant to early direct-injection LTC is to develop a continuous flow vessel that has extensive optical access to study liquid- and vapor-phase penetration, evaporation rate, and dispersion (*e.g.*, spreading or cone angle) of fuel sprays. The test rig described in this thesis falls in the category of *constant-pressure flow vessels*. This design enables frequent injections, which will allow for rapid buildup of large data sets and observation of stochastic behavior of sprays. This document will report on the conceptualization, design, assembly, and experimentation with a continuous-flow vessel. With this new experimental setup completed, it will be possible to study sprays for a wide variety of fuels and conditions and thus generate data to provide the missing gap of knowledge and to develop new models.

2. DESIGN CONCEPTS FOR FLOW VESSEL

This chapter details preliminary concepts for the design of the continuous-flow vessel. The first step was to define the requirements and parameters of the apparatus, as shown below.

Design Requirements:

1) Must be able to withstand pressure and temperature of 200 psi and 200ºC, respectively
2) Capable to mount without interfering with the optical accessibility
3) The design must be very economical, thus off-the-shelf parts are strongly encouraged
4) Optical accessibility that enables study of fuel sprays that are on the central axis of the chamber and also maximizes viewable length of spray
5) Must include method for spreading out and streamlining sweep-gas flow to minimize turbulence and large-scale flow structures

Functional Requirements:

1) Capability to vary pressure, temperature, and flow rate inside the vessel
2) Optical accessibility for high-speed spray visualization
3) Detachable windows for cleaning and replacement as necessary
4) Must be able to pass sweep gas through the vessel, thus allowing to system to be purged.
5) Must have ability to change injectors and/or injector types
6) Have ports for temperature and pressure sensors

2.1 INITIAL DESIGN CONCEPT

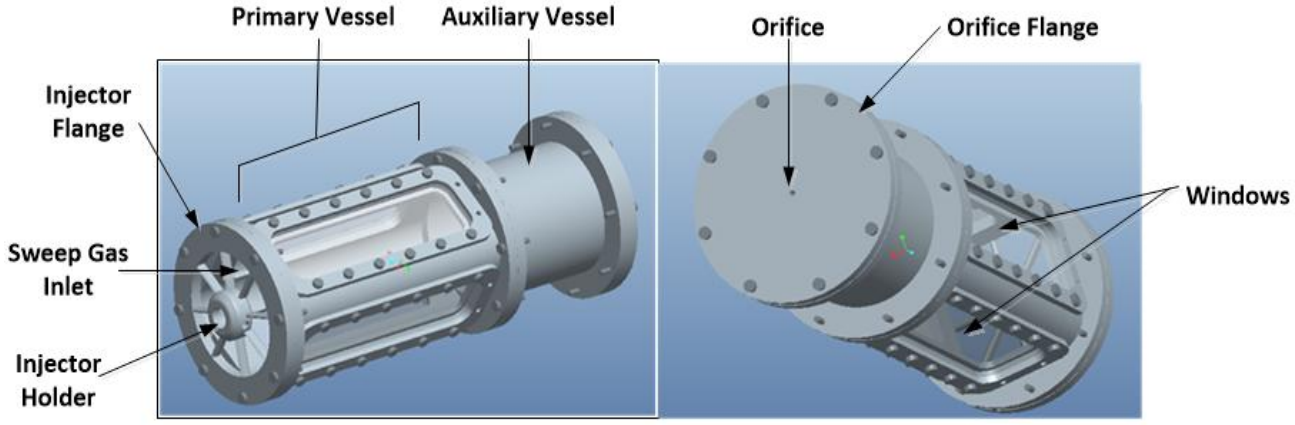

Figure 2.1: Isometric drawing for the first proposed vessel design.

Figure 2.1 illustrates the initial proposed design of the flow vessel. In this concept, the primary vessel would be made from a tube that would be flattened to accommodate four flat windows bolted to it. The injector flange would have an interchangeable injector holder; however, the entire front of the flange would have to be covered by a flexible hose or channel for the sweep-gas inlet. Though there were a few issues with this design, such as locating or making a flexible hose with sufficient size and ability to withstand the maximum pressure and temperature, some of the concepts were retained.

2.2 IMPROVED DESIGN CONCEPT

Building on the initial concept, the decision was made to enclose flow-conditioning hardware inside the main vessel section and have a bolt-on flange that would hold the injector

and have multiple ports to feed sweep gas into the vessel. This within itself had several design complexities that had to be addressed. The most important issue was that the upstream flow from the multiple inlet ports had to seamlessly combine in a uniform stream and velocity distribution before reaching the fuel-injection area. After addressing plausible flaws, an improved design concept was developed, as shown in Figures 2.2 and 2. 3.

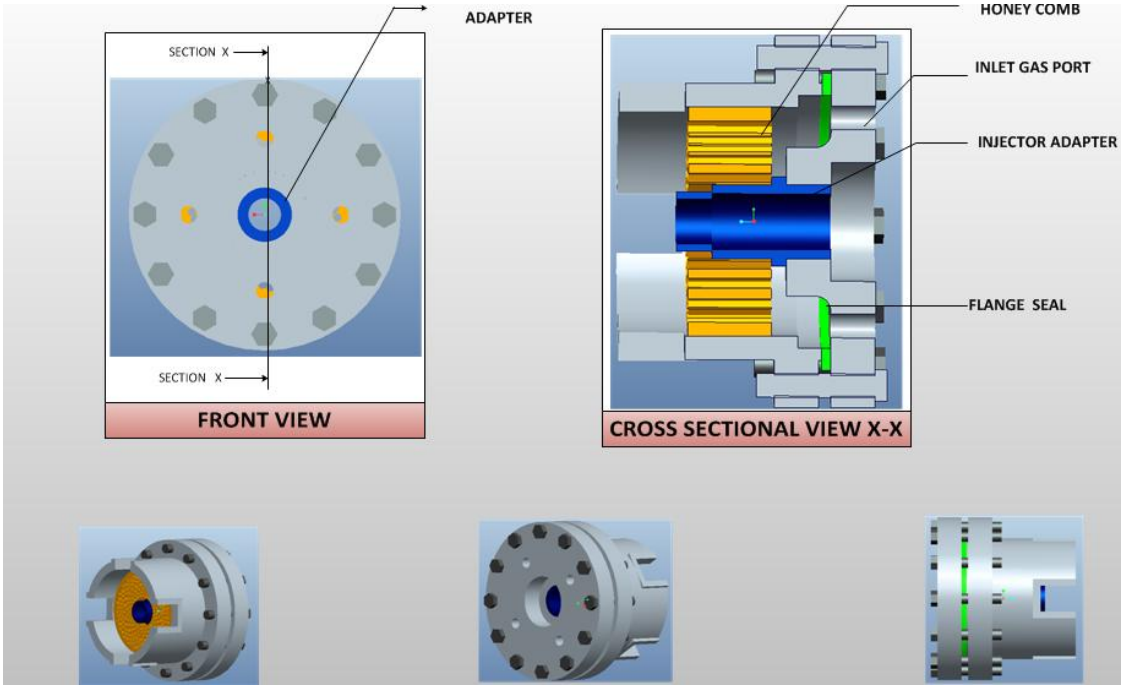

Figure 2.2: Injector flange in the improved design concept.

Figure 2.2 shows the small portion of the primary vessel with the newly designed injector flange and components. The four-inlet gas port would provide an easy means of introducing the sweep gas into the vessel; however, this would require several high-pressure flexible tubes and multiple fittings. The improved design showed promise, but there were of course a few concerns, including how to build the modified injector flange, integration of the injector adapter, obtaining flow-conditioning plates, and securing the flow-conditioning plates.

For the primary vessel, the new conceptual design kept it as simple as possible by using a cylindrical pipe with four rectangular flats for window mating, as illustrated in Figure 2.3. This eliminated the need to deform the tube, thus maintaining better structural integrity.

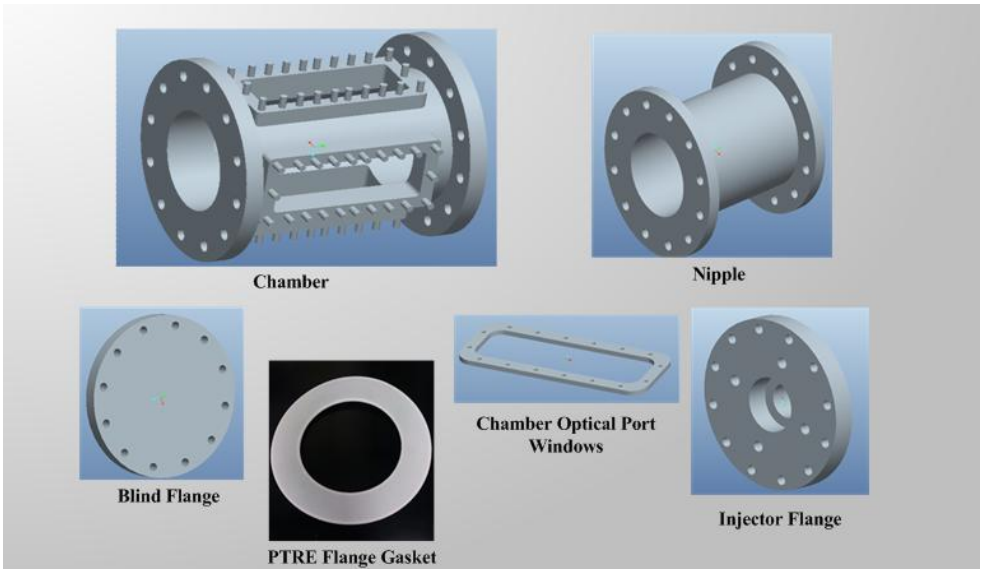

Figure 2.3: Major components for the improved vessel design concept.

2.3 FINAL DESIGN

In the next design stage, which ultimately led to the final design, the auxiliary vessel, orifice flange, optical port frame, and minor fittings remained the same as the previous design. Before the vessel was ready for fabrication, however, design modifications were made to the optical port connection, flow conditioning plates, and the injector holder/adapter. Once the final design concept was complete, the detailed design and fabrication were outsourced. The optical windows were fabricated by Rayotek Scientific, Inc. (San Diego, CA) and the vessel hardware

was fabricated by Fitz-Thors Engineering, Inc. (Bessemer, AL). The final design can be seen in Figures 2.4, 2.5, and 2.6.

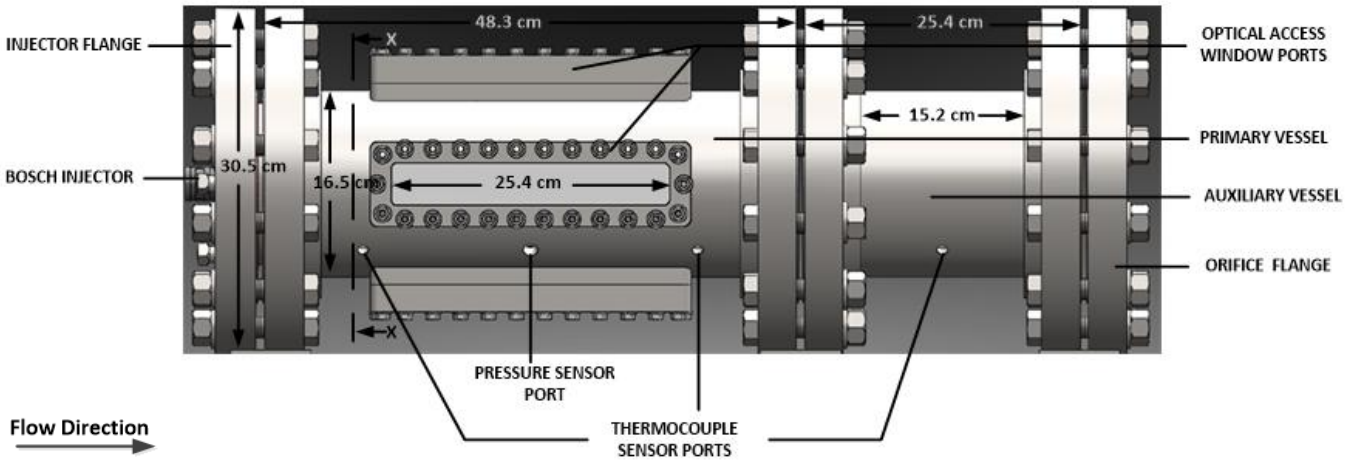

Figure 2.4. Side view of overall vessel assembly

Figure 2.4 shows a side view of the entire hardware assembly, which consists of two stainless steel pipe sections connected by stainless steel flanges. One pipe section constitutes the main, optically accessible chamber (labeled “primary vessel” in Figures 2.4 and 2.5) and the other pipe section is an auxiliary vessel ,implemented to mitigate turbulent effect on each injection . Including the flanges, the overall lengths of the primary and auxiliary vessels are ~48.3 cm (19 inches) and ~25.4 cm (10 inches), respectively. There are four small access ports along the length of the entire assembly. Three of the ports are designed for thermocouples to measure temperature. One port, located at the midpoint of the primary vessel, is designed for a pressure sensor.

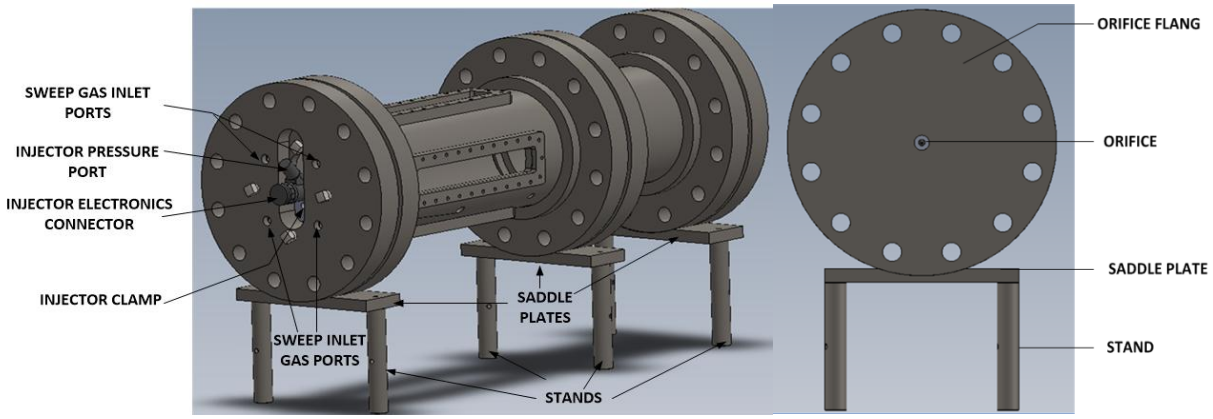

Figure 2.5 Isometric view and the end view of flow vessel components

The primary vessel has four orthogonal ports for optical access, as shown in Figures 2.4 and 2.5. Each port provides an open aperture that is nominally rectangular, with 1.3-cm (0.5-inch) width and 25.4-cm (10-inch) length. This geometry was designed primarily for experiments in which the fuel spray consists of a single jet oriented along the central axis of the cylindrical vessel. Four-way orthogonal access enables multiple simultaneous experiments, such as combined liquid-phase and vapor-phase penetration using elastic light scattering and schlieren imaging, respectively. Window assemblies, each consisting of a rectangular quartz window pre-installed on a stainless-steel housing, are bolted to the vessel to seal the system while providing optical access.

The sweep gas flows through the inside of the primary and auxiliary vessels, which have an inside diameter of 14.6 cm (5.76 inches), to continuously rinse the chamber and remove unreacted fuel vapor. The mass flow rate of sweep gas is controlled using a choked-flow orifice on the orifice flange. As long as the orifice is small enough to choke the flow, the mass flow rate can be calculated from choked-flow equations with knowledge of the pressure and temperature upstream of the orifice (*i.e.*, inside the vessel). By combining ideal-gas equations of state with

choked-flow relationships, assuming orifice size remains constant, it can be shown that volumetric flow rate (and therefore linear gas-flow velocity through the vessel) remains constant as pressure changes seen in Equation 3.

$$\frac{dm}{dt} = AP\sqrt{\frac{k}{RT}}\left(\frac{k+1}{2}\right)^{-\frac{k+1}{2k-2}} \quad (1)$$

$$\frac{dm}{dt} = \rho\frac{dv}{dt} \quad (2)$$

Recall $k = C_p/C_v$

$\rho = P/RT$

$\frac{dv}{dt} = AV$

Substituting equation (2) into (1) yields the following

$$\frac{dv}{dt} = ART\sqrt{\frac{k}{RT}}\left(\frac{k+1}{2}\right)^{-\frac{k+1}{2k-2}} \quad (3)$$

At the end of the entire assembly there is a flange (labeled “orifice flange” in Figure 2.4) with a central threaded hole. The hole enables installation of a pipe fitting with a small orifice drilled through the center. This design creates the flexibility to change orifice diameter by simply changing the installed fitting.

Streamlined flow inside the vessel is preferred, to avoid the unsteady effects of turbulence when studying fundamentals of fuel sprays. Use of a small exit orifice keeps flow rates, and thus flow velocities, low. A system was designed to condition the flow at the entry to the vessel assembly to further ensure streamlined conditions.

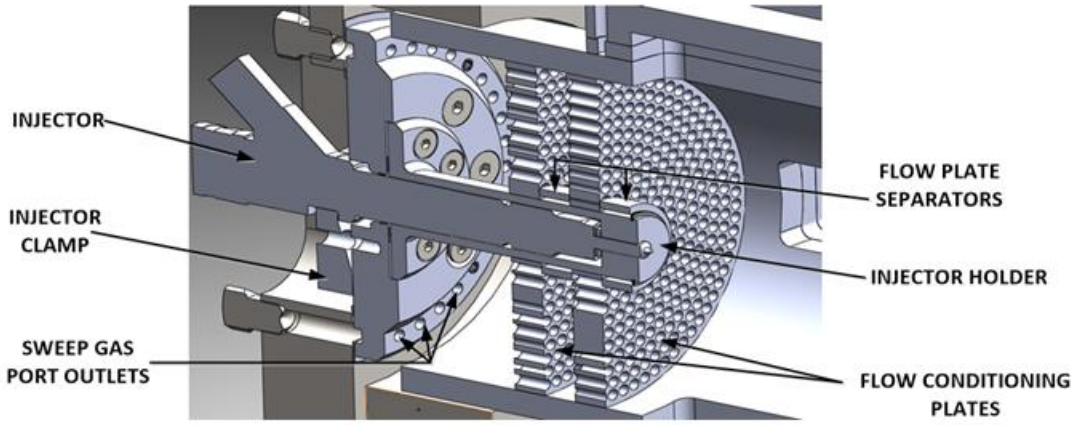

Figure 2.6. Section view of primary vessel entry area.

Figure 2.6, which shows a section view of the entry area, illustrates the flow-conditioning system. Gas flow can enter the primary vessel through as many as eight ports in the injector flange. Immediately upon entry, the flow encounters a plate that has a series of holes around its circumference (labeled as “sweep gas port outlets” in Figure 2.6), except in the areas where the flow enters. Thus, the flow encounters a stagnation plane and must spread out to pass through the holes. Once past this initial stagnation plate, the flow encounters a series of two ~2.5-cm-thick plates (labeled “flow-condition plates” in Figure 2.6), each with a radial pattern of 36 holes. These plates are designed to eliminate any large-scale disturbances and help streamline the flow as it enters the fuel-spray area.

3. COMPUTER AIDED DESIGN AND MODELING

Since the flow vessel is a custom-made platform that has not been tested previously, it was necessary to verify some of the design aspects. Most importantly, the strength of the vessel and the overall assembly is critical. In addition, the thermal and flow conditions are important for the intended experiments. At this stage, verification has been done using computational models. Experimental data, particularly for the conditions inside the vessel, can be gathered in the future to compare with the models.

3.1 FINITE ELEMENT ANALYSIS (FEA) FOR THE FLOW VESSEL

The flow vessel has to be strong enough to withstand a pressure and temperature of 200 psi and 200ºC, respectively. While the individual components were rated for these conditions, modifications and assembly considerations could alter the structural integrity of the vessel. Hence multiple finite element analyses were conducted using SolidWorks. Applying the prescribed conditions on the individual parts and the assembly provided very insightful information that also aided in minimization of thermal-expansion stresses in the final setup.

For the worst-case scenario, each component was assumed to approach the maximum global temperature of 200°C. With air at 200°C flowing through the vessel, the inner walls of the

vessel would no doubt be cooler. Nevertheless, this global temperature was coupled with 200 psi pressure on the inner walls to yield the FEA results seen in Figures 3.1 through 3.4.

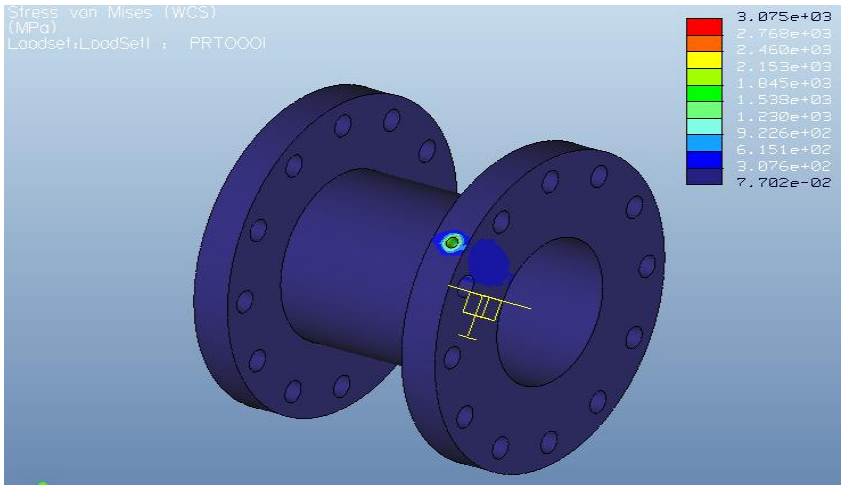

Figure 3.1: Stress distribution throughout the auxiliary vessel.

The contour distribution plot seen in Figure 3.1 shows that the only point that challenges the structural integrity of the auxiliary flange is the saddle plate screw hole. This is where the vessel attaches to mounting plates (saddles) shown in Figure 2.5, which in turn connect to an optical table. In reality the hole would not be restricted from displacement; however, to perform the analysis this was required by the SolidWorks. The auxiliary vessel shows no signs of failure under the prescribed conditions, since the mean von Mises stress is ~77 kPa and the yield strength of 304 stainless steel at 200 psi and 200°C is 159 MPa. To further verify the validity of the solution the FEA was run several times, increasing the number of elements with each run to ensure that the solution was grid-independent. In the final run, the solution was identical to the previous, the auxiliary vessel had a factor of safety exceeding 10 with over 100,000 elements used to perform the analysis.

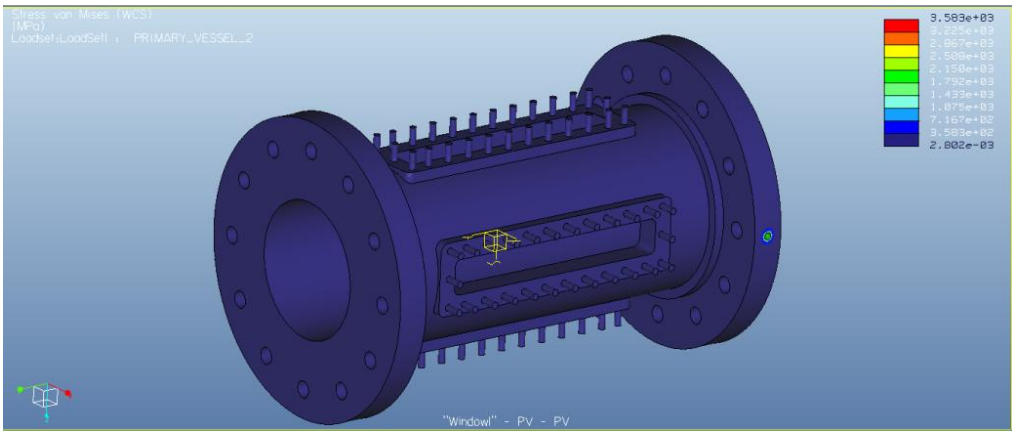

Figure 3.2: Stress distribution throughout the primary vessel.

The maximum stress occurs where the vessel saddle screw is located as seen in Figure 3.2, which corresponds to the same point of maximum stress in Figure 3.1. Since the saddle screw hole was fixed, it was not allowed to freely expand and displace compared to the other areas of the vessel and similar to the auxiliary vessel, because of the fixed constraint additional stresses that were being developed in the model. The primary vessel auto grid had over 150,000 elements assigned to the geometry to perform the analysis yielding the result in Figure 3.2. The result suggests that despite the four optical ports and sensor openings the structural integrity is still adequate to withstand the tremendous pressure and temperature that will be applied given that the mean von Mises stress is ~2.88 kPa.

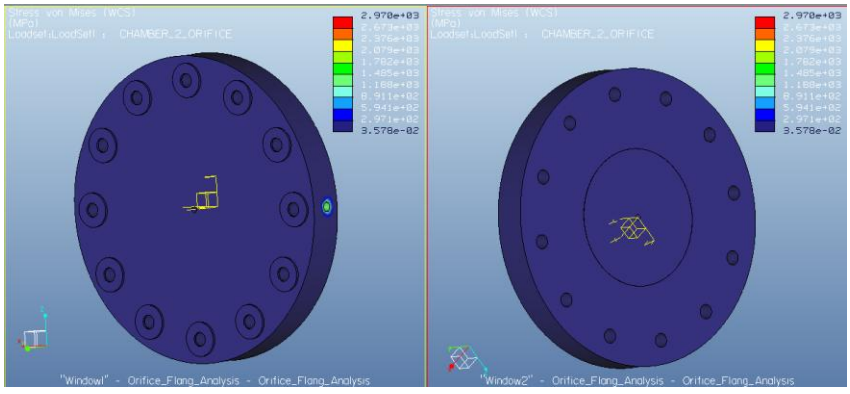

Figure 3.3: Stress distribution throughout the orifice flange.

Similar to the previous parts, the orifice flange seen in Figure 3.3 shows no signs of failure. This is not surprising, since the only modification made to this component was the addition of a tapped hole in the center of the flange to allow the interchangeability of various sized orifices. The final component, the injector flange, was meshed with 37,103 elements. As shown in Figure 3.4, FEA on the injector flange also produced very favorable results by predicting that it would also withstand the extreme conditions.

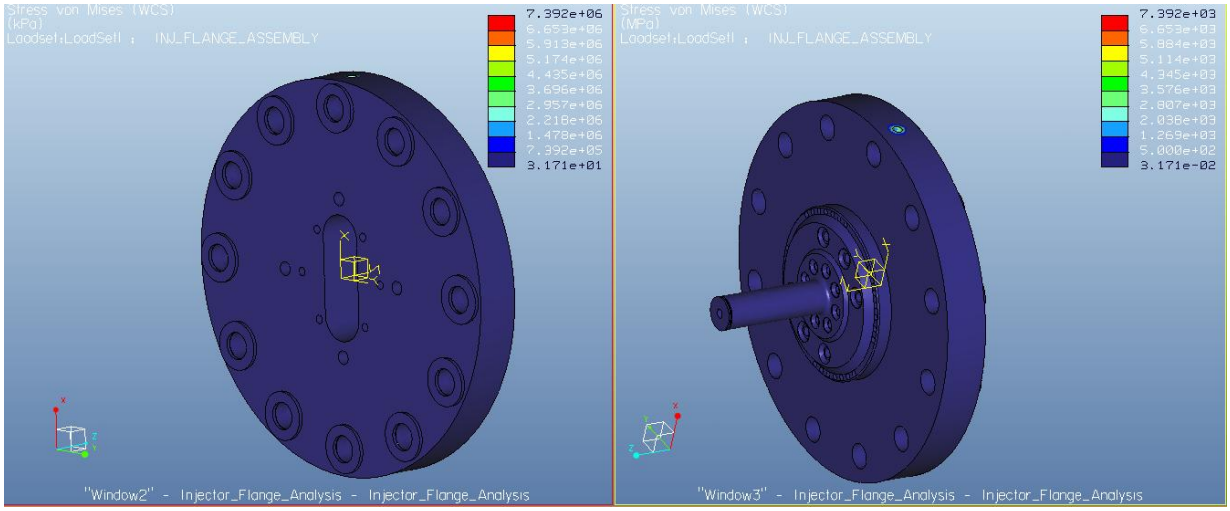

Figure 3.4: Stress distribution throughout the injector flange.

The other components not tested were bought to withstand the extreme operating conditions of the vessel and the four optical port windows were outsourced for fabrication to meet the desired specifications. In conclusion, the results obtained from the finite element analysis presented in this chapter suggests that the high-pressure, high-temperature flow vessel is fully capable to operate at the extreme temperature and pressure conditions required.

3.2 COMPUTIONAL FLUID DYNAMICS (CFD) ANALYSIS FOR THE SWEEP-GAS FLOW THROUGH THE VESSEL

The flow vessel's conceptual design and eventually the CAD model had several components implemented in the design phase that were assumed to aid in developing a uniform sweep-gas flow distribution. These components varied from the flow distributor on the injector flange, which allocates the inlet gas supply into 36 distributed outlet jets directed into the primary vessel to flow-conditioning plates that further distribute the flow and create a uniform cross-sectional velocity distribution. The final verification was to examine if the orifice location was appropriate and the swirl and tumbling events were far enough downstream as to not influence the fuel-spray characteristics.

Importing the CAD model of the vessel assembly into various CFD software such as Ansys Fluent and FloXpress within SolidWorks presented several issues. The biggest issue was that the full vessel assembly has over 300 hundred components, which made CFD modeling an impossible task given the computing power available. For simplicity, a similar internal flow path vessel was modeled, eliminating the unnecessary components that are irrelevant for the sweep-gas flow analysis. This simplified model was used to analyze how the vessel and sub-component geometry influence the flow. For the flow analysis, the inlet was specified as a pressure inlet with flow conditions of 400 K at 200 psi while the exit orifice was specified as an atmospheric pressure outlet and finally the wall temperature for the entire vessel was specified at 400k. A total of 462168 elements was used to model the entire fluid domain that converged after 619 iterations, the result obtained is illustrated in the following figures.

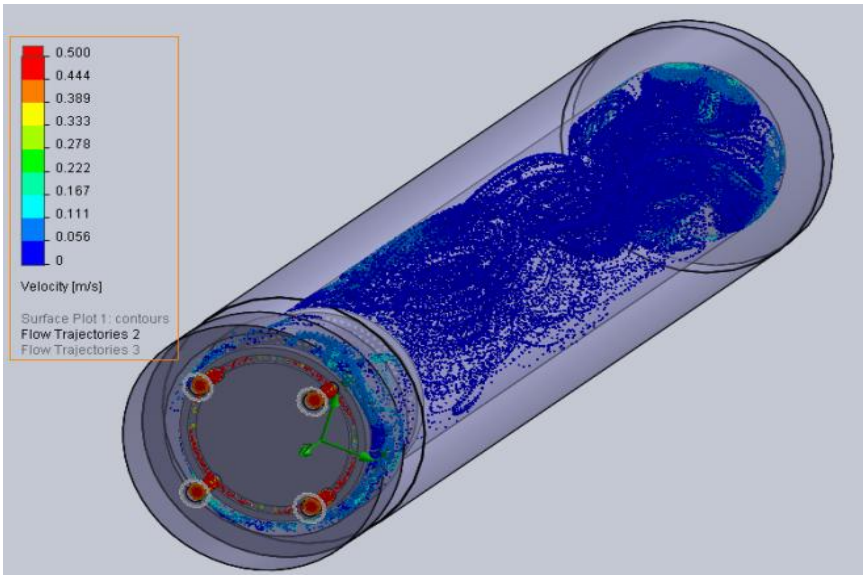

Figure 3.5: Isometric view of the sweep-gas flow through the vessel.

Computed results are shown in Figure 3.5. Results appear to show that the flow-conditioning plates create low velocity wave-like streamlines. The calculated velocity for the sweep gas throughout the primary vessel was approximately 2.7 cm/s which is remarkable close to the computed value from the simulation of 2.6 cm/s on average, thus validating that the result of the model is reasonable. Near the orifice flange at the end of the vessel, as shown in Figures 3.5 and 3.6, the flow becomes chaotic, forming large swirls and tumbles, before exiting through the orifice.

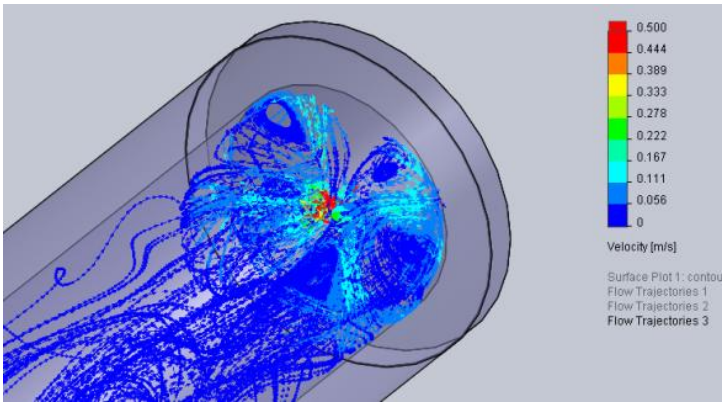

Figure 3.6: Sweep-gas flow simulation results near the orifice at the vessel exit.

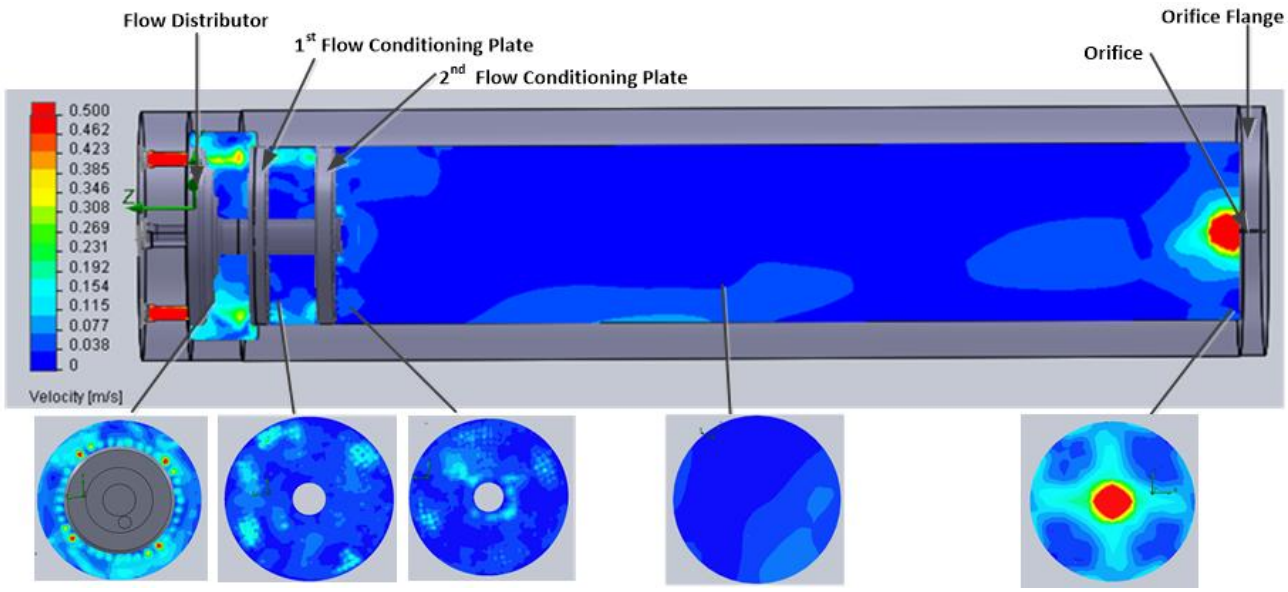

Figure 3.7: Cross-sectional views of the sweep-gas velocity profile at various points throughout the flow vessel at 400 K and 200 psi

Figure 3.7 without a doubt demonstrates that the injector flange flow distributor serves it purpose. It redistributes the flow, minimizes the sweep-gas inlet velocity, and produces small jets of air with wider spread angles. The first flow-conditioning plate further distributes the flow as expected. After the second flow-conditioning plate, the flow more streamlined, with very slight variation in the velocity profile. Changing the size of the exit orifice changed the characteristics of the tumbling and swirling regions near the exit, but these events were far enough downstream as to have significant influence on the fuel spray. Zooming in on the orifice flange in Figure 3.8 and viewing the velocity cross-sectional contours clearly illustrates areas of non-uniform velocity distribution near the exit. These areas in red and light blue are perhaps dominated by swirls and tumbles that are more prevalent inside the vessel near the orifice flange which can also be seen in Figure 3.6 as the sweep gas exit the vessel.

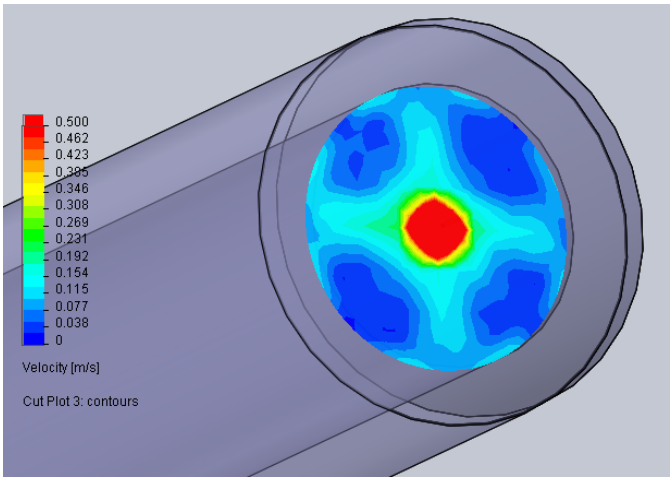

Figure 3.8: Velocity profile after near the exit of the orifice flange

In summary, results of the CFD model suggest that flow-modification components, including the initial entry-flow distributor and the two flow-conditioning plates, are effective in producing a flow in the main part of the vessel that is streamline and largely free of turbulence. In addition, the auxiliary vessel is effective in displacing flow disturbances further downstream such that the turbulence associated with flow exiting the system through a small orifice does not influence the spray in the optically accessible region.

4. SOFTWARE DEVELOPMENT IN LABVIEW AND MATLAB

4.1. DEVELOPMENT OF LABVIEW SOFTWARE FOR EXPERIMENTAL CONTROL AND DATA ACQUISITION

This chapter details the Labiew Virtual Instrument (VI), which was developed to acquire data and produce several synchronized trigger pulses. The data acquired by the VI include pressure and temperature in the vessel, as well as details about the experiment parameters (*i.e.*, metadata). Of the four synchronized trigger pulses produced, two control the camera gates (pre-trigger and trigger), the third triggers the injector, and the fourth triggers the pulse generator. After the pulse generator is triggered, it produces two synchronized bursts of TTL pulses, one for the camera (which indicates how many images to take) and the other for the LED, which toggles the on and off state of the light source.

FRONT PANEL AND CONTROLS

The front panel is created from a series of subVI's shown on the pallete menu. After careful consideration of the user interface and functional requirements, the subVI's were dragged and placed on an empty slate front panel which eventually evolved into what is seen in Figure 4.1.

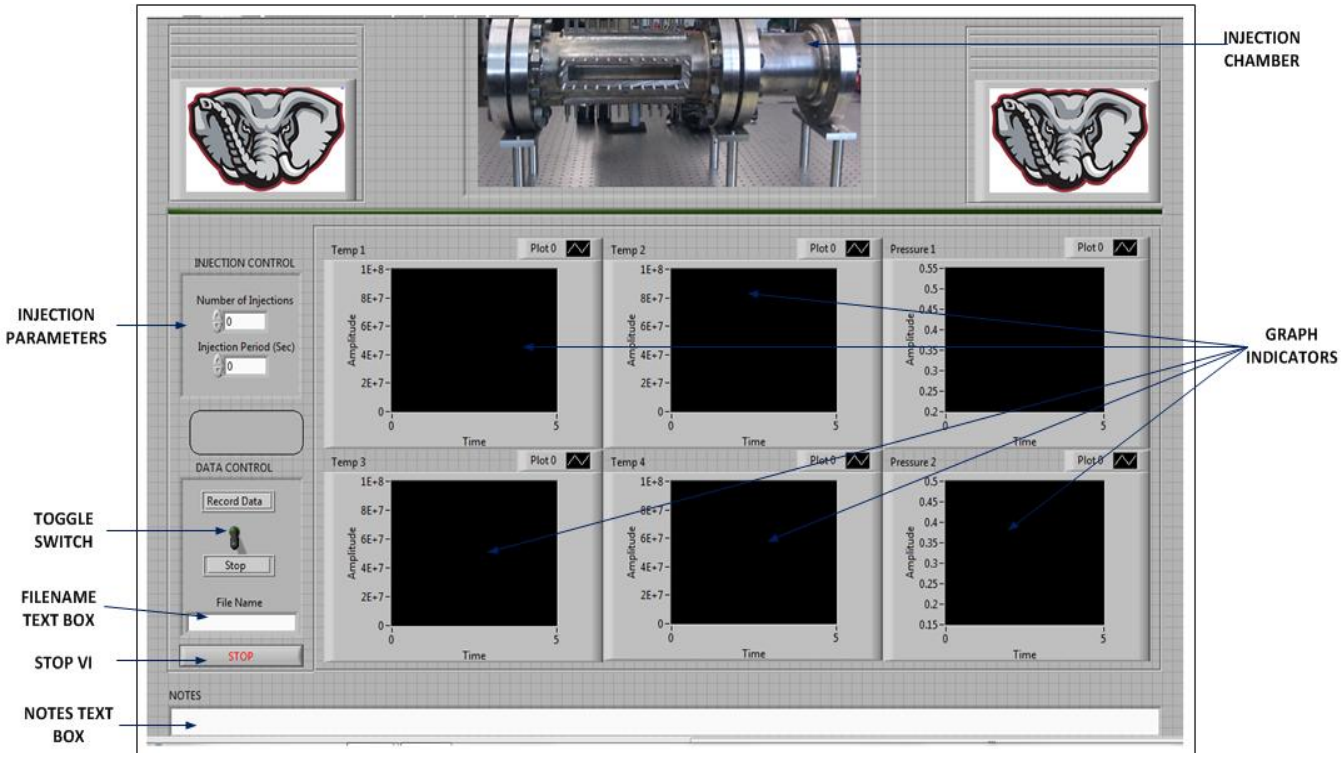

Figure 4.1: Front Panel of the spray chamber VI control.

When the VI is running the graph indicators will display the temperature and pressure readings. The top three graph indicators are currently the only ones that display the true data while the bottom three in the front panel are designed for future expansion. In the "injection parameter" region the user can define the number of injections intended for a particular experiment in addition to the period between each injection cycle. If both the number of injections and period between injection string input are left empty or zero, the toggle switch will not record the notes during the experiment since no injection occurred. However, if the injection parameters are specified, when the toggle switch is in the "on" state (up) there are three things that occur:

1) The temperature and pressure measurements are stored to a spreadsheet, where the filename is defined by the user in the filename text box.

2) The switch also serves as a trigger for the camera and the pulse generator, hence it is essential that the I/O ports are selected. If these ports are not selected an error message will pop up.

3) Finally, the switch also allows a subVI to record the injection parameters and any notes the user might have entered in the file note textbox before the experiment. The notes recorded are stored in a word document with the 'filename' specified by the user and '_notes' extension to it (see Figure 4.2).

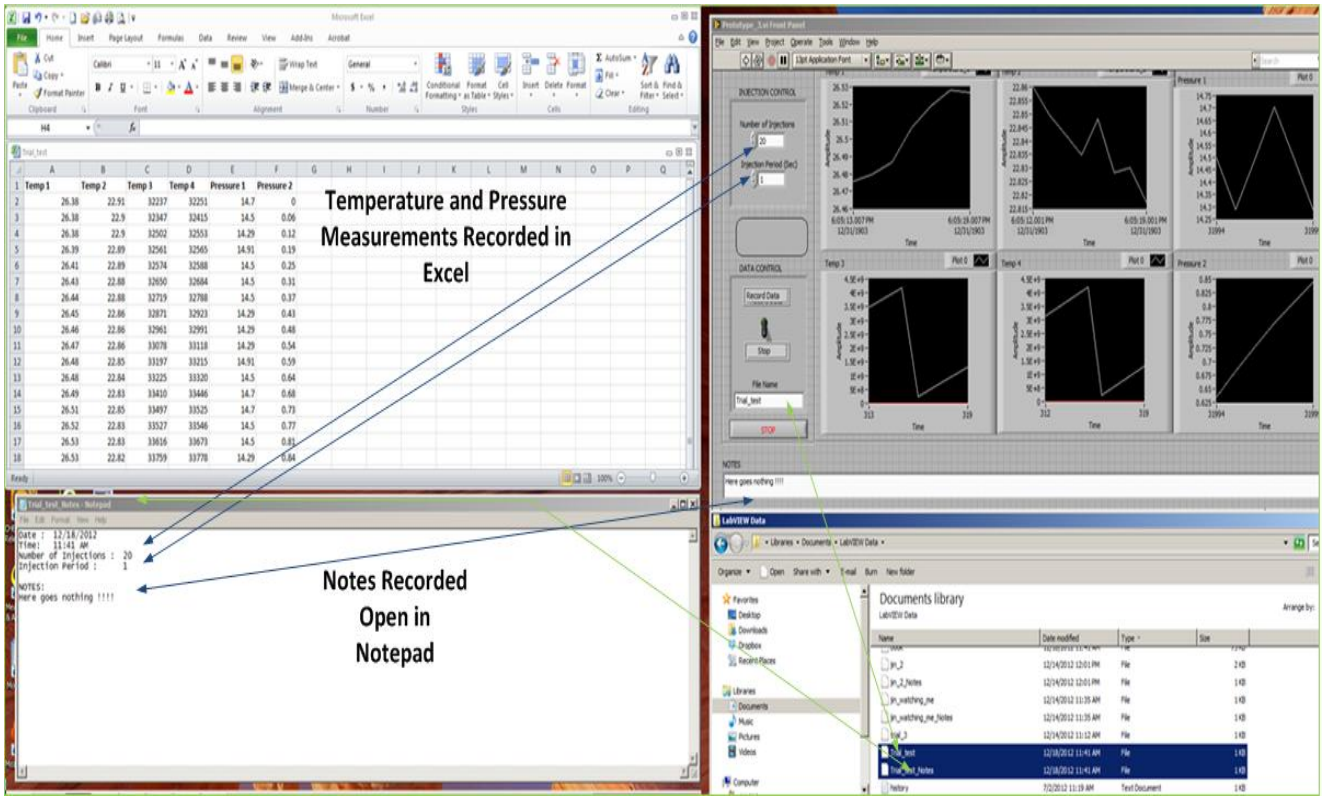

Figure 4.2: The VI Front Panel, spreadsheet, notepad, and the documents library.

BLOCK PANEL DEVELOPMENT

The block panel is the heart for the entire VI developed for data acquistion and control throughout the experiment, as illustrated in Figure 4.3.

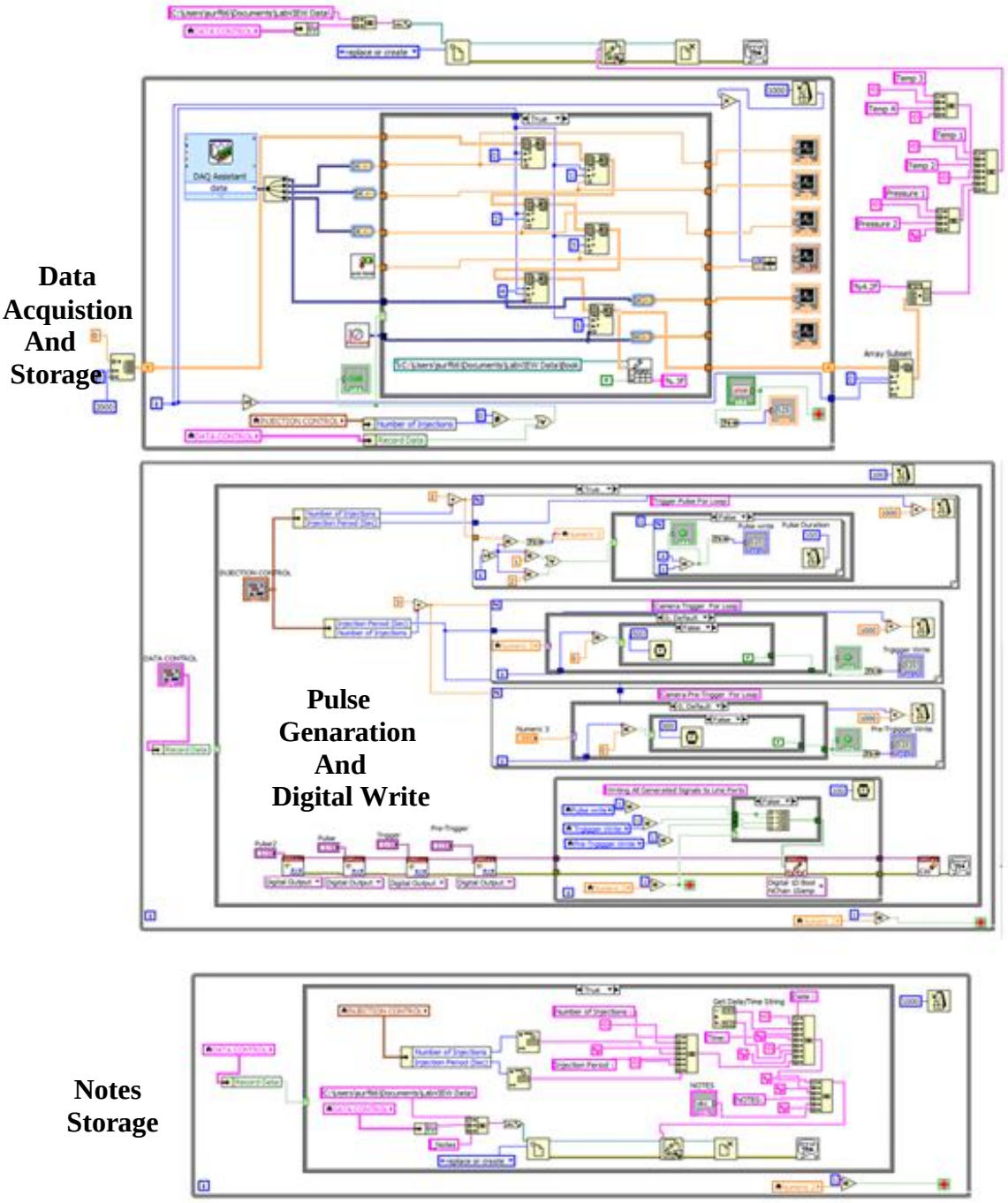

Figure 4.3: Block diagram for VI.

Figure 4.3 for explanation purposes can be divided into three main parts:

1) Data acquisition and storage
2) Pulse generation and digital write
3) Notes storage

A synopsis for each of these three main parts is given below in the following subsections.

(1) DATA ACQUISITION AND STORAGE:

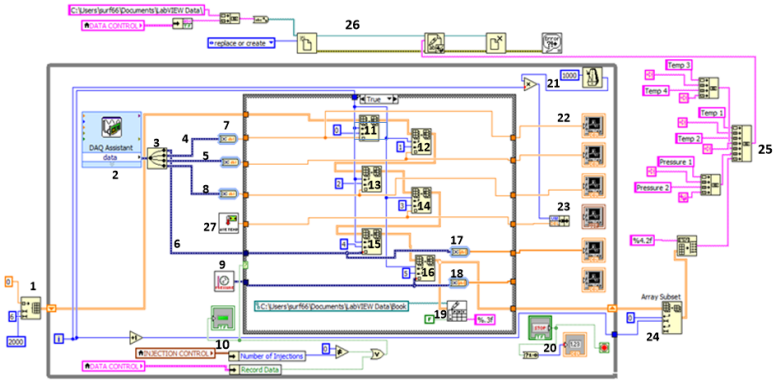

Figure 4.4: Data acquisition and storage section of the block panel view of the VI.

As the title suggests this section of the block panel VI is primarily responsible for acquiring the data from the DAQ Assistant. The temperature and pressure collected using the DAQ Assistant are obtained from the three K-type thermocouples and the pressure sensor. The data collected are split into four discrete channels and are used to build sub-arrays for each, where each the sub-array is appended every second. The instantaneous data are immediately displayed in the four graphic indicators while collected temperature and pressure data at the end of the experiment are stored to a user-defined Excel file and a backup Excel file called *Book* in the LabView Data folder.

(2) PULSE GENERATION AND DIGITAL WRITE:

The primary purpose of this section of the program is to trigger the pulse generator and to open the gate for the camera trigger and pre-trigger, thus allowing it to capture images when the TTL pulse train is sent to it in the *true state* of the case structure. The TTL pulse generated from the trigger pulse also serves to synchronize the LED light pulse and the injection with the camera. For simplicity, this section is further divided into four subsections inside the while loop for the *true state* of the case structure seen below in Figure 4.5.

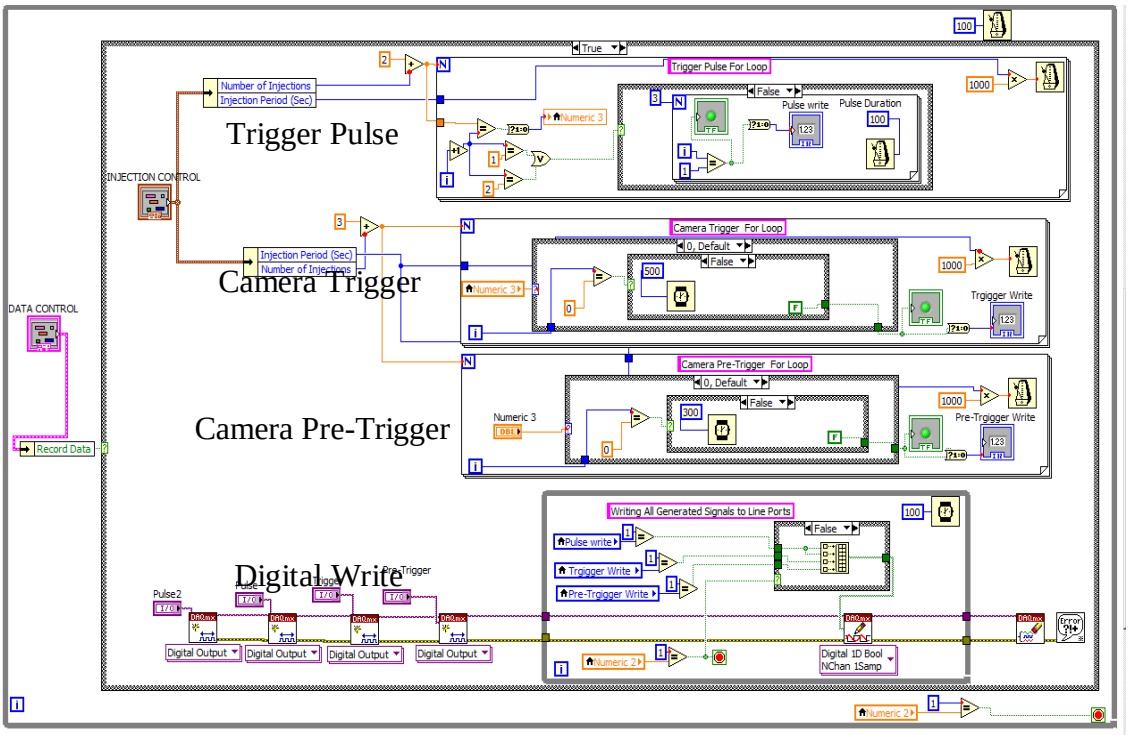

Figure 4.5: Pulse generation and digital write section of the VI's block panel.

The *Digital Write* is the final subsection inside the true case structure that allows all of the Boolean trigger signals generated within the virtual space to be transmitted outside the program to control the camera, the injector, and the LED. Hence, this section is a crucial component of the VI.

(3) NOTES STORAGE:

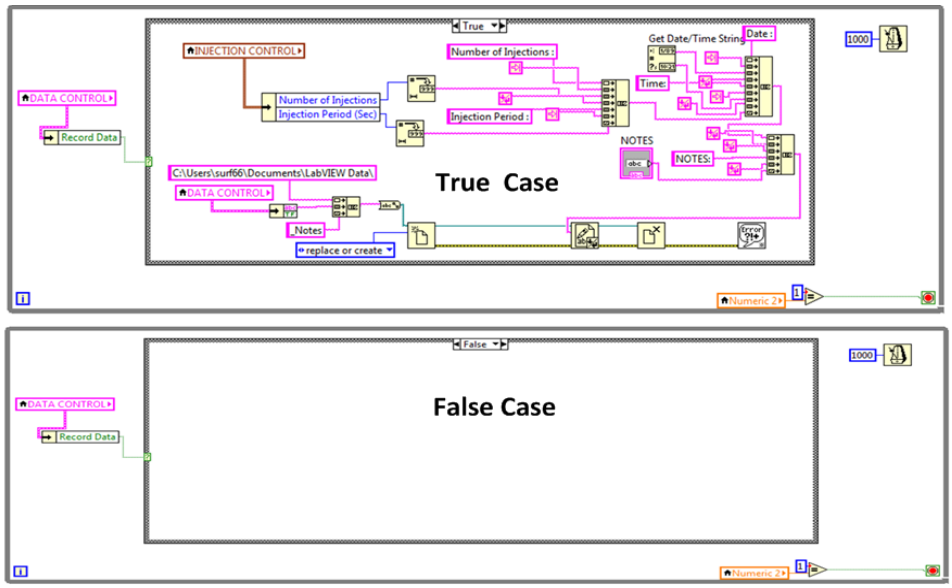

Figure 4.6: True and false case for the note storage VI.

Figure 4.6 shows the sub-component *Notes Storage* in both its true and false states where the states are control by the toggle switch on the front panel. In its true state of the case function it simply collects data from the front panel such as the injection parameters, filename, and notes entered by the user, converts everything to a string, and writes it to a file. For the false state of the case structure, however, nothing is done. The data recorded in this section also allow the user to keep meticulous details for each experiment ranging from observations written in the notes section, the time stamp that inherently occurs for each run, compressor run time, and other details that allow to trouble shoot probable cause for variation in the data. The file that is automatically created can be appended at any time, hence after the experiment any details that were left out can be conveniently entered in the document.

4.2 IMAGE ANALYSIS GRAPHICAL USER INTERFACE

The Image Analysis Graphical User Interface (IAGUI) is an essential tool that allows the user to extract information from the camera high-speed movie (stored as a CINE file). This information varies from simple boundary isolation and extraction to acquiring numerical information such as spray penetration length, cone angle, propagation rate, and origin. The IAGUI also allows the processing of JPEG images and has several built-in tools to optimize batch data processing and extraction.

The IAGUI was designed to be intuitive and user-friendly. So in order to open a desired file, simply clicking the *Open File* button will automatically open the computer file directory allowing the appropriate file to be selected. The selected file will then be immediately displayed; however, if the image is in the format of CIN or CINE the IAGUI will proceed through a guided process before displaying the desired image.

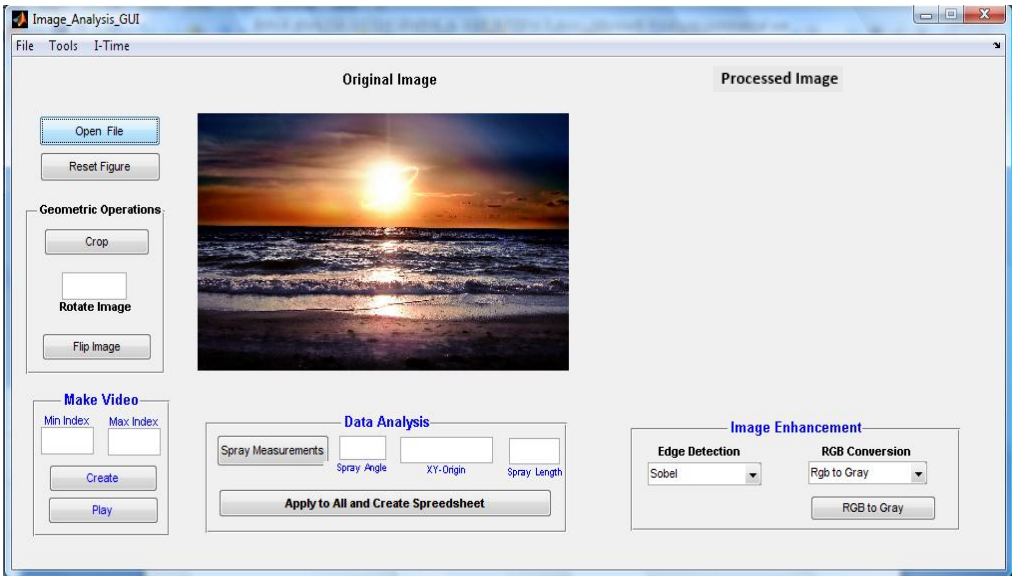

Figure 4.7: The IAGUI displaying a JPEG image.

Before any of the image loaded can be process they were first converted to gray scale, after conversion was completed the image could then be processed. Following this the IAGUI would display the processed image seen in Figure 4.8.

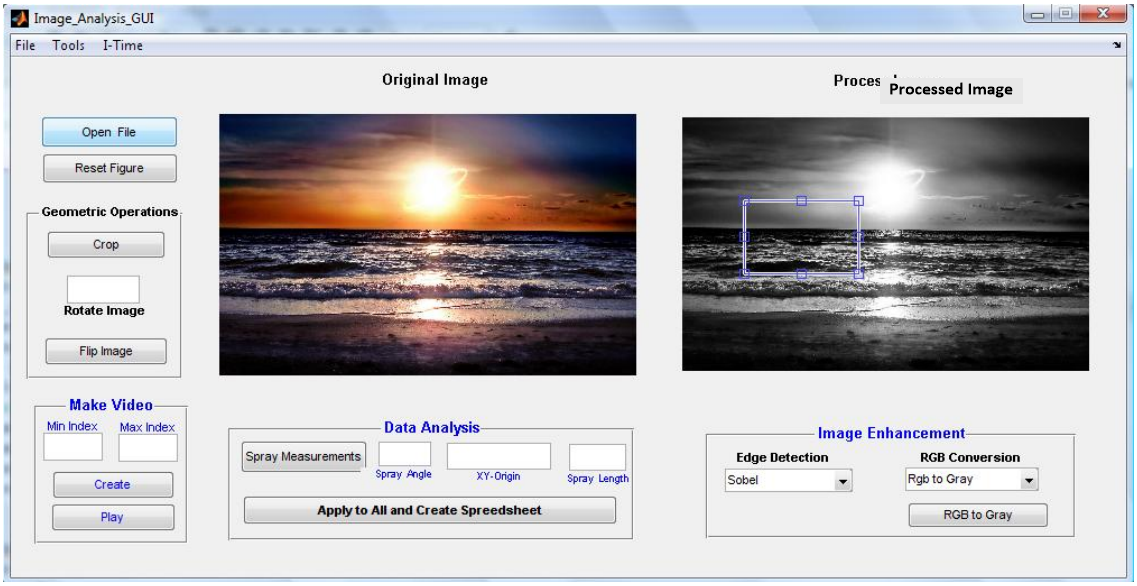

Figure 4.8: The IAGUI displaying the original and processed image with the interactive crop-rectangle.

In some cases, an image might have unnecessary areas that are not of interest or not quite in the appropriate orientation. In such cases, the *Geometric Operation* container is useful. The *Crop* push button, when selected, enables interactive choice of the crop rectangle with the six resizable handles. Once the desired region of interest is chosen, the image can be further rotated or flipped. Flipping the image is necessary in some cases, such as when determining the spray cone angle (its origin must always be on the left side; see Figure 4. 9). If the appropriate orientation is not taken into consideration before selecting the *Spray Measurement* button, an error will occur.

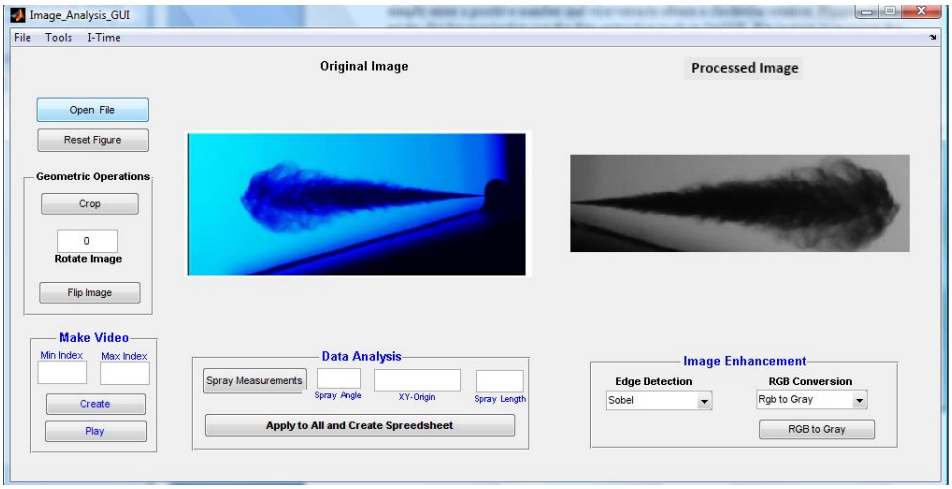

Figure 4. 9: The IAGUI displaying the original spray image changed to gray scale, flipped, and cropped in the *Processed Image* axes.

When the processed image is in the form illustrated in Figure 4.9, in order to extract numerical information from the spray, the *Image Enhancement* container and *Tools* on the tool bar are essential. For a quick approximation, the *Tools* menu has several options ranging from measurement of cone angle to finding the intensity profile across a few pixels in the processed image. The spray penetration length can be measured by selecting the indicated button. When the button is selected, a popup window appears which allows the user to select two extreme points of interest and then use the ruler on the tool bar to measure distance, as shown in Figure 4.10.

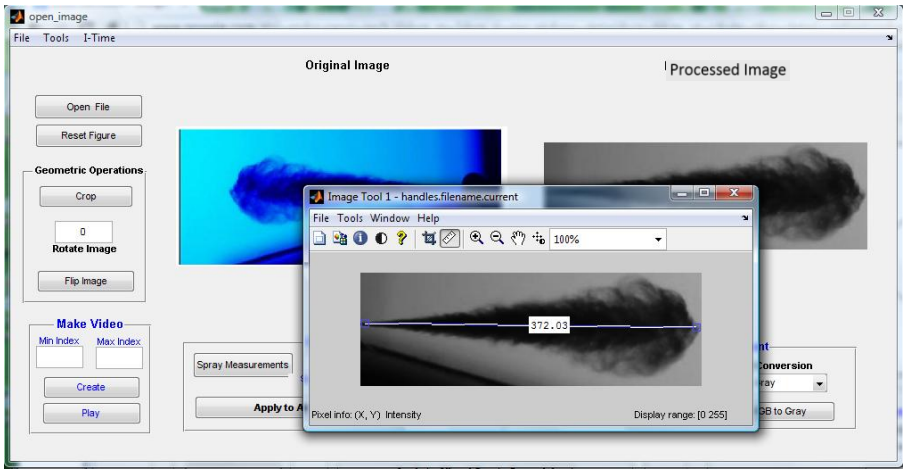

Figure 4.10: The IAGUI and the popup *Image Tool* GUI used to measure the spray length.

The popup *Image Tool* GUI is a built-in Matlab function that has several other image manipulation functions; however, it is primarily used in the IAGUI to measure the spray length. Similar to selecting the spray length from the main GUI and obtaining a rough estimate of the spray length, the cone angle (*i.e.*, spreading angle) also can be obtained. By selecting the *Spray Angle Measurement* option in *Tools* of the IAGUI a popup window will appear. This allows the user to place two handles (movable points) using the mouse at the upper and lower boundary of the spray to obtain a reasonable estimate of the spray cone angle. The final two functions in the IAGUI tools options are the *Intensity Line Plot* and the *FFT* (Fast Fourier Transform). The *Line Intensity Plot* function allows the user to trace a path along the processed image, after which an intensity 2D or 3D plot would be displayed. The *FFT* option, on the other hand, allows the ability to transform the processed image into the frequency domain to observe resonant or otherwise important frequencies, or as a tool to further mitigate image noise.

The primary purpose of these processing tools is to investigate individual images and "spot check" for specific observations. Use of these tools is impractical for batch processing of large numbers of images. Thus, the *Image Enhancement and Data Analysis* container was developed for batch processing and data extraction from the images. The *Image Enhancement* container has a drop down box for *Edge Detection* and *RGB* Conversion. At present, all fuel-spray images are captured in grayscale, and thus RGB options have not been used. In the future, however, for measurements such as those using rainbow schlieren imaging, this will be a useful option.

The *Edge Detection* dropdown box options are all useful in finding the boundary or edge of the spray, however the Canny method is the most popular, powerful and effective edge detection method, hence it tends to be the most reliable to accurately determine the spray

boundary[71]. This method is ideal because it addresses each individual pixel. In this method, an image is first smoothed using a Gaussian low-pass filter. Then a local gradient is computed for each point in the smoothed image to find the boundary. A threshold is then applied to isolate the boundary pixels and edge-linking algorithms are applied. Other methods tend to inflate or deflate the boundary before tracing, sometimes by several pixels, while the Canny method does little or no alteration to the image boundary before tracing it[71]. An example of the spray traced by the Canny edge-detection method is shown in Figure 4.11.

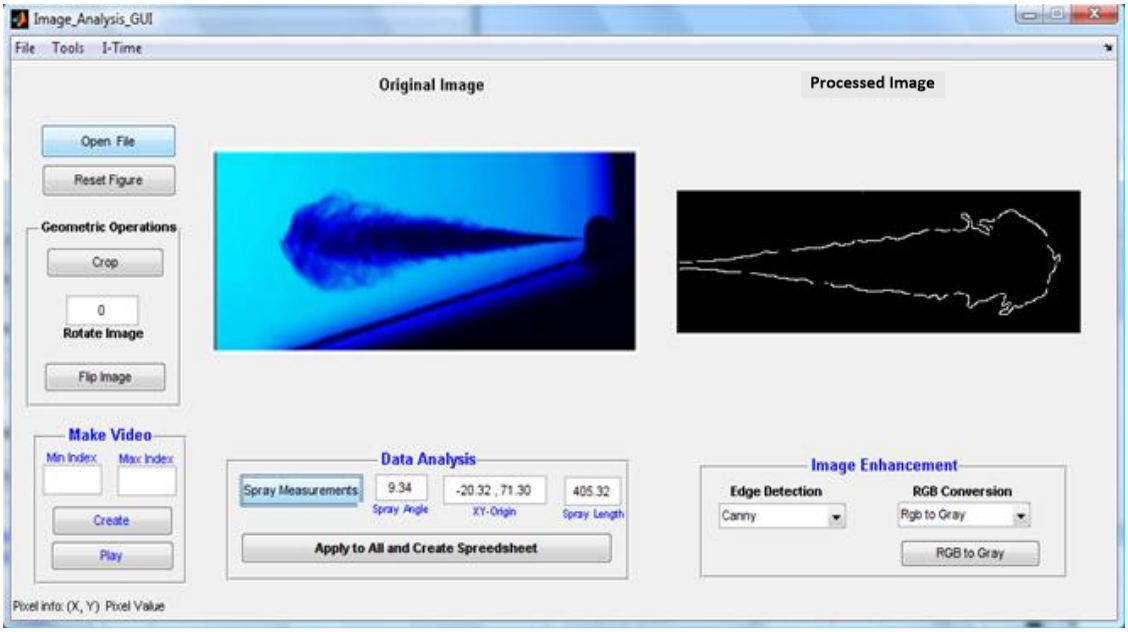

Figure 4.11: Original image and boundary outline of the processed image.

Once a well-defined boundary is obtained, the *Spray Measurement* button, when clicked, fits linear regression lines on both sides of the spray for 40% of its length and also finds the spray penetration length. The linear regression fit allows for a statically derived cone angle and spray origin, provides good mean to obtain this essential data. The distance from the computed spray origin to the most extreme point of the spray (*i.e.*, farthest to the right) is defined as the spray penetration length, which is essential to track the spray progression over time. Finally,

when all of the derived data are obtained they are displayed in the *Data Analysis* container, similar to Figure 4.11.

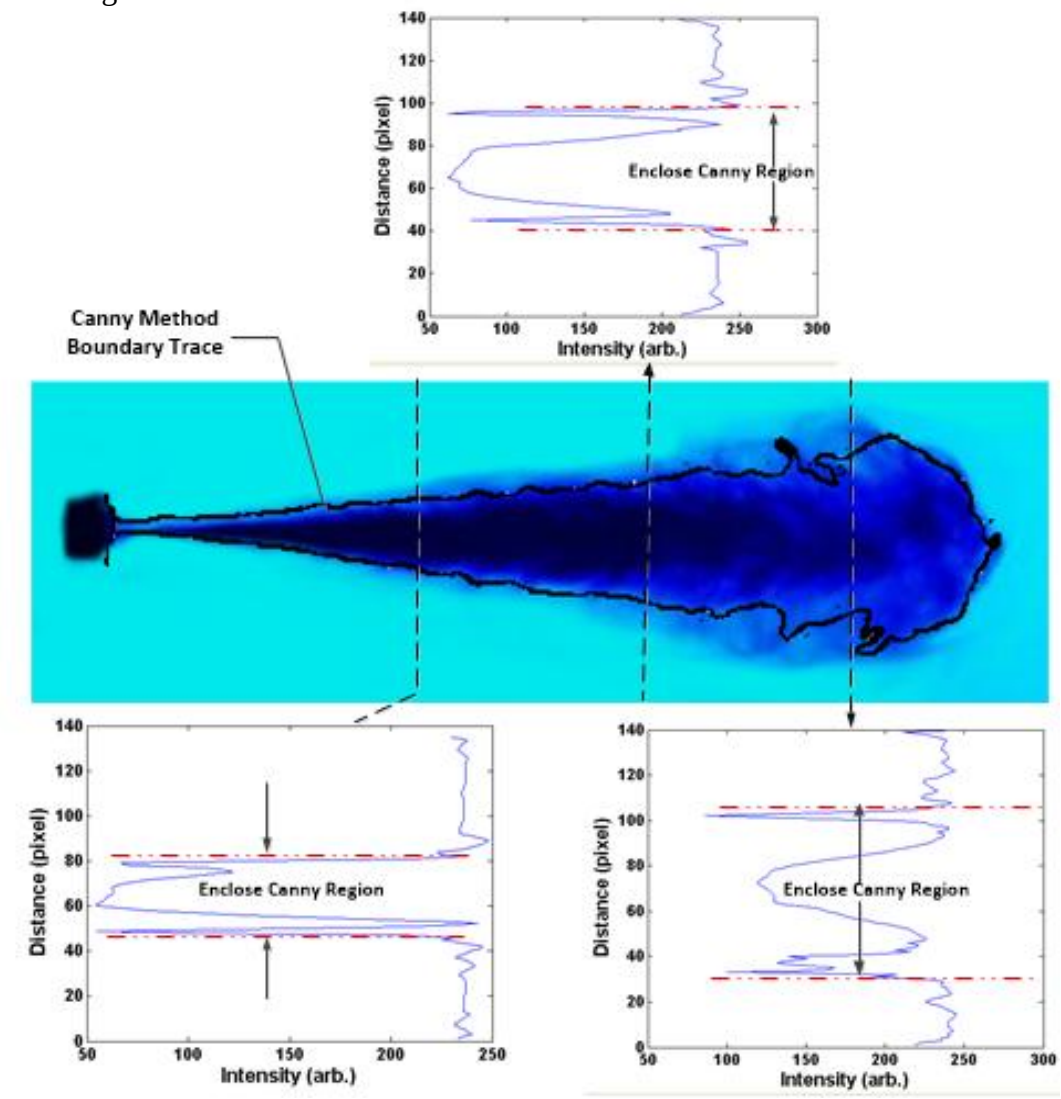

Figure 4.12: Extracted spray image superimposed with the Canny trace and subsequent intensity plots at various points along the axial spray length.

Figure 4.12 shows that the superimpose image of the Canny edge trace and the spray. Taking various cross sectional intensity line plot, intensity values exceeding 238, 233 and 232 where outside the boundary which accounts for an average 7% of the intensity range along each line plot. The reverse is true for gray scale spray image (spray is white), which would suggest

that 7% of the maximum intensity would be considered background and would be outside the Canny boundary trace.

The final feature in the *Data Analysis* container is the *Apply All and Create Spreadsheet* push button. This feature processes all of the desired images and extracts spray data such as the spray angle, origin, and length from each. These data extracted are then tabulated by the software and saved in a spreadsheet, which allows for further post-processing and generation of plots as desired.

Finally, the *Make Video* container seen in Figure 4.11 is used to convert a group of processed images into a AVI video file. The video file obtained from the processed images can be used for dynamic visualization of the spray as it propagates in time for further analysis and presentation purposes.

5. FINAL VESSEL ASSEMBLY AND EXPERIMENTAL SETUP

The final vessel assembly is shown in Figure 5.1. Design of the vessel was described in Chapter 2, and information on how it is assembled can be found in the Appendix. The purpose of this chapter is to describe the full experimental setup, including the peripheral systems that enable the desired experiments. These peripheral systems include those for the sweep gas, fuel injection, spray visualization, and control and data acquisition. Each peripheral system plays an imperative role to meet the functional requirements of the vessel and to accurately execute and record each experiment conducted. This chapter gives a synopsis of all of these subsystems and how they are effectively integrated with the final vessel assembly.

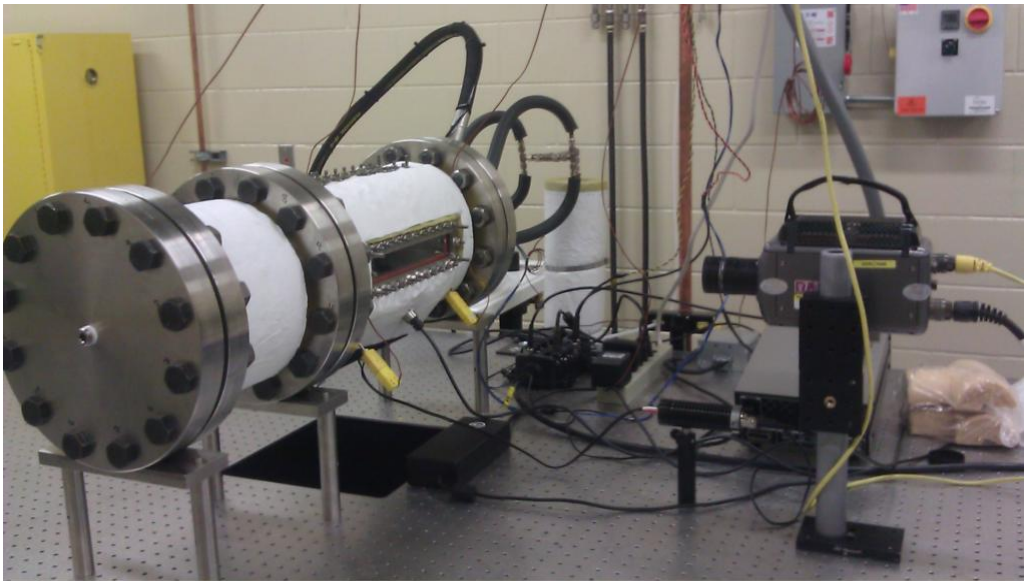

Figure 5.1: The final vessel setup.

5.1 SWEEP-GAS SYSTEM

The primary purposes of the gas-flow system are to continuously rinse the vessel with sweep gas, remove unreacted fuel vapor, and to set the thermodynamic conditions in the vessel. The gas-flow system and its components are illustrated in Figure 5.2.

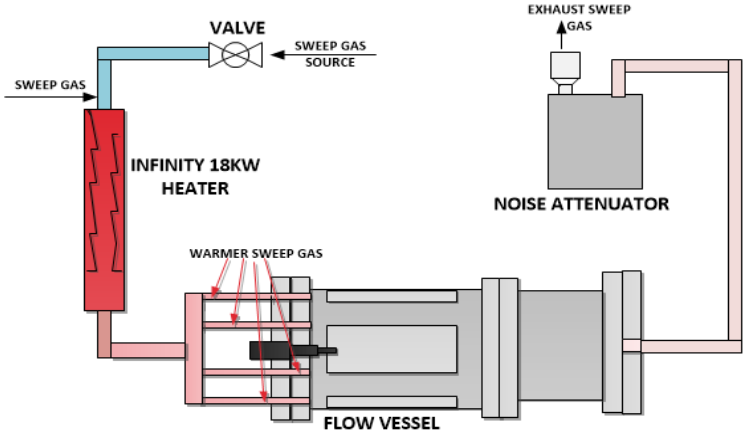

Figure 5.2. Schematic of gas-flow system.

Before entering the vessel assembly, the sweep gas passes through an electric heater to rise to a chosen temperature. The heater (model # CRES-MF-48-0180-K-3P-PTC; Infinity Fluids Corp.; Worcester, MA) has maximum power of 18 kW, which is capable of raising the temperature of ~56,630 standard cm^3/s (120 scfm) of air flow from room temperature to 250°C. Flow issuing from the auxiliary vessel through the choked-flow orifice creates a high-pitch whistle that easily exceeds 100 dB without sound mitigation. A system was therefore implemented to attenuate the sound, as well as to capture the "exhaust" from the vessel, as shown in Figure 5.2. A flexible high-temperature hose was attached to the exit orifice and directed into the bottom of a large water barrel located in a walk-in fume hood. The water serves as an effective sound attenuator, reducing the noise level to below 85 dB.

5.2 FUEL SYSTEM

The primary motivation for the design of this vessel and associated equipment is to study diesel-type fuel sprays. Therefore, the initial design includes a diesel fuel system. Main components of this system include a diesel injector, a high-pressure fuel cart, and an electronic injector driver unit. The injector is a Bosch CRIN3 solenoid-driven common-rail injector. It is the same injector that is used in the GM 6.6-liter LMM model engine, and it can operate at fuel pressures up to 180 MPa (~26,000 psig). For fundamental spray experiments, the injector is equipped with a nozzle that has a single orifice with 100-μm diameter located directly at the tip. This produces a single fuel jet oriented along the injector cylindrical axis of symmetry. The injector is mounted in the center of the injector flange such that the single fuel jet issuing from the injector tip is oriented along the central axis of the cylindrical vessel.

A fuel cart, made by Exergy Engineering LLC (Grand Rapids, MI), supplies high-pressure fuel to the injector. An air-driven pressure multiplier converts low-pressure fuel into high-pressure fuel and pumps it through a flexible high-pressure hose to the injector supply line. This system is capable of generating 200 MPa (~29,000 psig) using a normal shop-air supply of ~690 kPa (100 psig), and is designed for infrequent injections (< 1 Hz) since it supplies fuel to a spray vessel rather than a constantly-running engine.

An electronic SADI (stand-alone direct-injection) driver (Drivven, Inc.; San Antonio, TX) is used to drive and control the injector. This system provides the high-current electrical signal required to drive the solenoid and actuate the injector. A LabVIEW-style interface allows the user to specify the temporal profile of the electric current, with a great deal of flexibility to drive a wide variety of injectors. For the Bosch CRIN3 injector currently installed in the vessel, the electric-current profile is the typical peak-and-hold, though the hold portion is divided into two stages as shown in Figure 5.3.

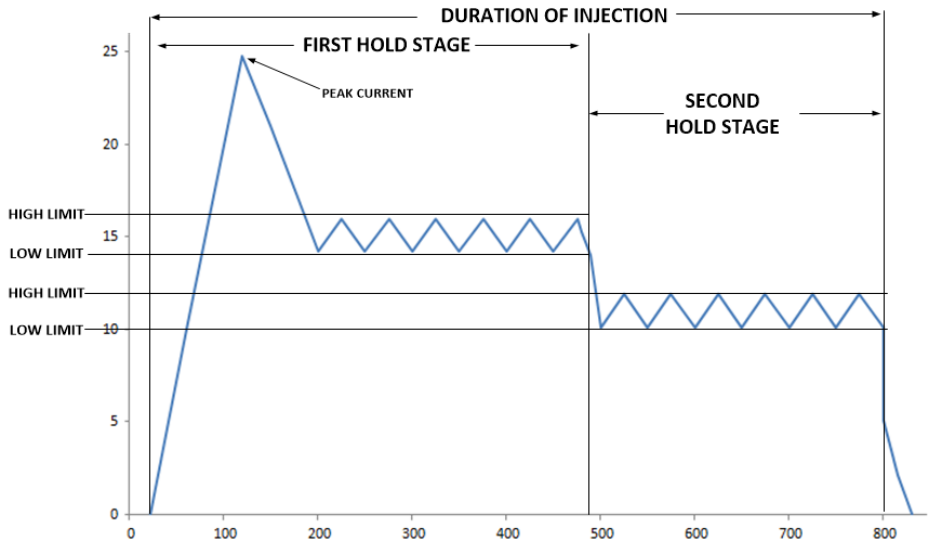

Figure 5.3: Schematic of electric-current profile to drive Bosch CRIN3 diesel injector.

The first hold stage has fixed time duration of 450 µs, which limits the shortest possible injection duration to ~500 µs. The second hold stage has variable time duration, which allows the user to define the time and thus define the overall injection duration (though it is shown as a fixed value in Figure 5.3). To protect the injector, driver software limits the command signal (*i.e.*, the electric current) to a duration of 5000 µs. Electric current is lower in the second-stage hold to minimize heat generation. Parameters of the electric-current profile are given in Table 5.1.

Table 5. 1: Shows the Current Profile Values for the Injector

Description	**Requirement**
Initial Peak	24.8 A
First hold stage high limit	16.0 A
First hold stage low limit	14.2 A
Second hold stage high limit	10.1
Second hold stage low limit	11.9 A
First hold stage duration	450 µs
62 max , 38 min	48V

5.3 SUMMARY OF EXPERIMENTAL CAPABILITIES

The continuous-flow spray-vessel assembly has been designed to allow variation of thermodynamic conditions within the vessel, namely sweep-gas pressure and temperature. Fundamentally, pressure is limited only by available pressure at the source, though the practical limit is the design pressure of 1380 kPa (200 psi). Temperature is technically limited by the 18-kW heating power and the chosen flow rate. The practical limit is the design temperature of 200°C, which is readily achievable for flow rates relevant to the intended experiments. For the current diesel-injection system, maximum fuel-injection pressure is 180 MPa. Duration of the injector-driver command signal (*i.e.*, *indicated* duration of injection) can be varied from ~500 μs to 5000 μs.

5.4 SPRAY-VISUALIZATION SETUP

The spray-visualization system consists of a high-speed camera and a light source. The camera is a Phantom v7.3 (Vision Research, Inc.; Wayne, NJ), with full resolution of 800 pixels × 600 pixels, maximum framing rate of 6,688 Hz at full resolution, maximum framing rate of 222,222 Hz at reduced resolution, and 22-μm pixel size. The light source is a custom-made high-repetition-rate pulsed LED emitter array (Lightspeed Technologies Inc.; Campbell, CA. Operating an LED in pulsed mode, with short durations of light output interspersed with longer durations of no light output, enables bursts of high-repetition-rate, high-intensity light. Output from the LED array is broadband white light with color temperature of 4600 K.

The LED-array light sheet enters the primary vessel through the bottom window and illuminates the fuel spray. The high-speed camera is positioned to image the spray through a side window. In this orthogonal configuration, the detected signal consists of light scattered elastically by liquid-fuel droplets. Thus, this experiment is set up to study liquid-phase fuel penetration, which is a global property of the liquid-phase portion of the fuel spray. This diagnostic does not provide information about individual fuel droplets. In the future, spray-visualization capabilities will be extended to imaging of vapor-phase penetration using schlieren imaging, as well as detailed studies of droplet and near-nozzle spray behavior.

5.5 CONTROL AND DATA ACQUISITION SYSTEM

To perform fuel-spray imaging experiments, it is necessary to simultaneously control conditions in the test section, control the fuel system and the injector, and control the spray-visualization system to synchronize imaging with fuel injection. It also is necessary to acquire relevant data, including pressure and temperature in the test section and fuel-spray images. A LabVIEW-based experimental-control and data-acquisition system was developed to accomplish these tasks. The architecture of the system is illustrated in Figure 5.4.

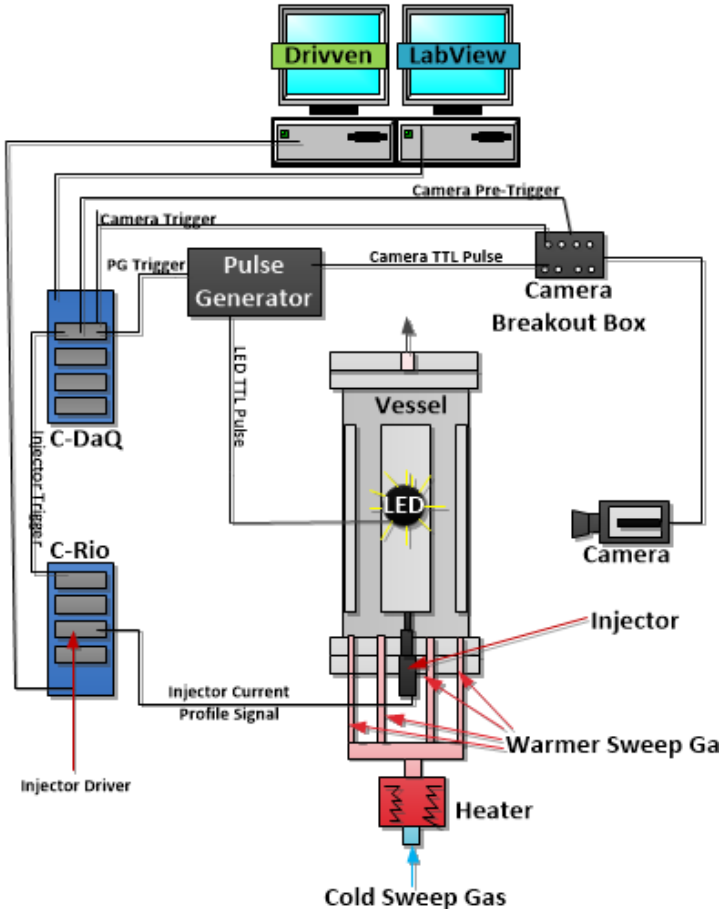

Figure 5.4: Schematic of entire experimental setup, including vessel assembly, spray-visualization system, and control and data-acquisition systems.

In the main LabVIEW program, the user specifies the number of injections and the time period between injections for an experiment. It is presumed that the user has warmed up the system by flowing sweep gas through the vessel assembly at the desired pressure and temperature. Once the experiment is initiated, identical digital triggers are sent to the injector driver and a digital pulse generator (DG645; Stanford Research Systems; Sunnyvale, CA). When triggered, the pulse generator sends out two trigger bursts, one to the LED system and one to the high-speed camera. The frequency and total number of pulses in these two trigger bursts determine the imaging frequency and the number of images acquired during the fuel-injection

event. Image files (movies) are recorded through Phantom software provided by Vision Research. The entire process is repeated for the number of injections specified. Pressure and temperature in the test section are recorded at 1 Hz throughout the entire experiment including all injections.

Injector-driver software (provided by Drivven) operates in parallel with the main LabVIEW program. In the driver software, the user can specify the electric-current profile (described above) and the duration of injection (DOI), and these parameters are communicated to the injector-driver hardware to ensure that the injector operates as desired each time it is triggered.

6. EXPERIMENTAL VALIDATION: HIGH-SPEED SPRAY VISUALIZATION OF *n*-HEPTANE FUEL SPRAYS

Throughout the process of design and development of the continuous-flow spray vessel, various components and subsystems were tested and verified. For example, the PID control system of the sweep-gas heater was tuned and a simple experiment was run to verify proper operation of the heater. The culmination of this entire development effort was the completion of baseline spray-visualization experiments with a single-component hydrocarbon fuel, *n*-heptane.

6.1 EXPERIMENTAL CONDITIONS

This research project culminated with high-speed spray-visualization experiments to determine bulk liquid-phase properties of *n*-heptane fuel sprays. Table 6.1 summarizes the conditions and relevant parameters for these experiments. Fuel parameters, including the indicated duration of injection and the rail pressure, were held fixed for all tests. To demonstrate the capabilities of the vessel, experiments were conducted for two different conditions as shown in Table 6.1. Test #1 was performed with quiescent air (*i.e.*, no flow) in the vessel at room conditions (*i.e.*, a "cold spray"). Test #2 was conducted with air flowing through the vessel at 690 kPa and 100°C.

Table 6.1: Conditions for spray-visualization experiments.

Condition/Parameter	**Value**
Fuel	*n*-heptane
Indicated DOI	800 μs
Flow rate	49.86 cm^3
Fuel-rail pressure	~12 MPa (17,500 psig)
Air pressure	**(1)** 100 kpa (14.7 psi)
	(2) 690 kpa (100 psi)
Air temperature	**(1)** 25°C
	(2) 100°C
Camera Frame Rate	14286 Hz
Camera Resolution	512 × 128 pixels
LED Pulse Width	2 μs

* Indicated DOI is duration of injector-driver command pulse.

Initial experiments in this vessel are focused on *n*-heptane as the fuel. This fuel is often used as a single-component chemical surrogate for diesel fuel, though it has much higher volatility and therefore vaporizes much faster. In addition, an *n*-heptane spray for a specific set of conditions is considered the baseline configuration for detailed spray experiments being conducted by a global research consortium known as the Engine Combustion Network (ECN). Led by Sandia National Laboratories, the ECN is a collaborative effort to provide an experimental database of diesel-relevant spray measurements for evaluation of experimental methods and for development and validation of computational models [29]. In short, *n*-heptane is relevant to what others are studying and is therefore a useful fuel for initial baseline spray experiments.

For these experiments, LED light was output with pulse width of 2 μs (for each pulse) and 70-μs period between each pulse. With this pulse period, sampling frequency was approximately 14,286 Hz. Using these settings it was possible to output the light in 35-pulse bursts, which was sufficient to image the full injection duration with one burst. The high-speed

camera acquired images synchronously with the LED light pulses, using exposure time of 6 μs to ensure capture of each 2-μs light pulse while minimizing integrated noise. Pixel resolution was set to 512 × 128, which was chosen to image the full width of the 25.4-cm window opening of the vessel when using a zoom lens set to focal length of 12.5 mm and an aperture of f/1.8. The conversion from pixel to true length was determined using a ~15.2-cm (6-inch) element placed in the exact plane of the spray. It was observed that the length of the calibration element was 345 pixels, yielding a conversion factor of approximately 0.44 mm/pixel.

6.2 RESULTS

Images were acquired for a total of five fuel-injection events for each condition. A sample *n*-heptane spray image is shown in Figure 6.1, with length scales superimposed for reference. Signal intensity (*i.e.*, brightness) indicates relative amount of liquid-phase fuel. As expected, there appears to be a dense central core of liquid fuel, surrounded by apparently decreasing concentration of liquid fuel in the radial and axial spray periphery. In this image, which was acquired near the end of injection, maximum liquid-phase penetration distance, or "liquid length," is approximately 13 cm and maximum width is nearly 3 cm.

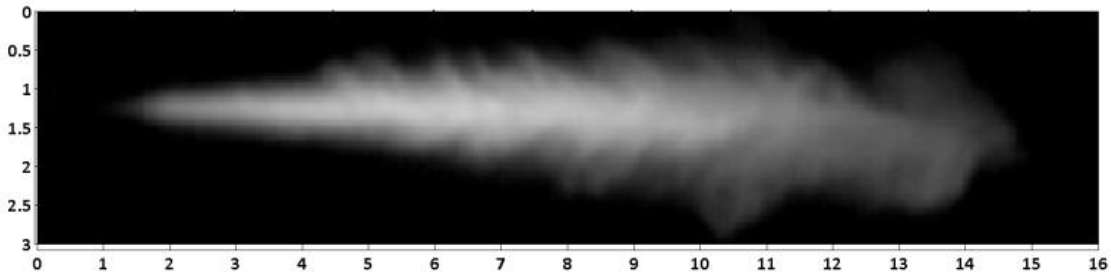

Figure 6.1. Sample *n*-heptane spray image captured by high-speed spray-visualization system. Scales shown have units of cm.

Based on built-in functions in the Matlab Image-Processing Toolbox (IPT), a custom GUI (graphical user interface) was developed in Matlab to post-process spray images as described in Chapter 4. Currently, the GUI is designed to find the boundary of the liquid-phase fuel using the Canny edge-detection method. The GUI then extracts liquid length and cone angle from the processed spray images. These important parameters are illustrated in Figure 6.2, based on the spray image shown in Figure 6.1. Cone angle describes fuel spreading and dispersion, which is dependent on air entrainment into the two-phase fuel jet. It is difficult to define a single value of cone angle to characterize the entire spray since it changes over the spray length. This will be a topic of future research with this experimental setup. For this study, cone angle was determined by fitting lines to the upper and lower boundaries for the half of the spray closest to the injector.

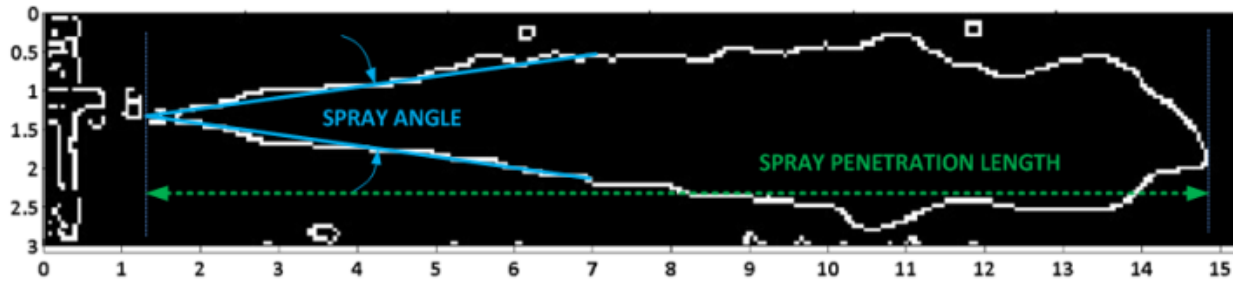

Figure 6.2: Illustration of liquid length and cone angle. Scales shown have units of cm.

Some clutter is visible at the origin of the spray in Figure 6.2, presumably due to the injector and other vessel surfaces. There are also a few small self-contained areas of signal that are either liquid or have been falsely identified as liquid. Regardless, liquid length is determined by the distance between the maximum liquid spray displacement and the origin. Liquid length and cone angle are plotted as functions of time in Figures 6.3 and 6.4, respectively. For each condition, liquid length (Figure 6.3) appears to have increased in a nearly linear fashion for the first ~0.2 ms then leveled off before reaching a maximum at end of injection. Actual duration of

injection as determined by these results was approximately 1.6 ms to 1.8 ms, where as indicated duration was only 0.8 ms. This behavior is typical and expected, however, for a common-rail diesel injector.

For quiescent room conditions, maximum liquid-phase penetration (*i.e.*, liquid length) was 169.6 mm on average over the five injections. Liquid length was extremely repeatable, with a standard deviation of 2.1 mm, which is equivalent to 1.2% relative standard deviation. For air-flow conditions of 690 kPa and 100°C, liquid length reached 92.2 mm on average with a standard deviation of 1.2 mm (1.3% relative). Repeatability is further highlighted by the dashed lines closely surrounding each liquid-length trace in Figure 9, which indicate the total "envelope" of the measurements, *i.e.*, the highest and lowest measured values at each point.

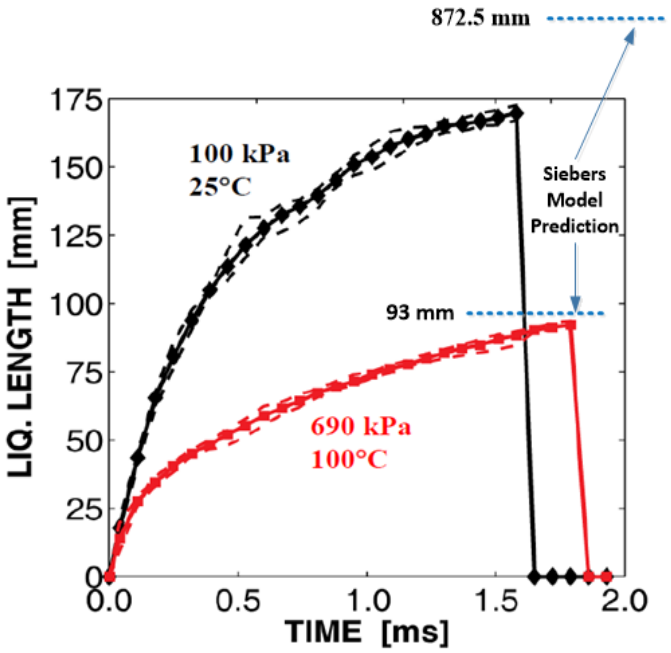

Figure 6.3: Liquid length *vs.* time for *n*-heptane sprays, averaged over five injections. Dashed lines indicate the full spread of acquired data

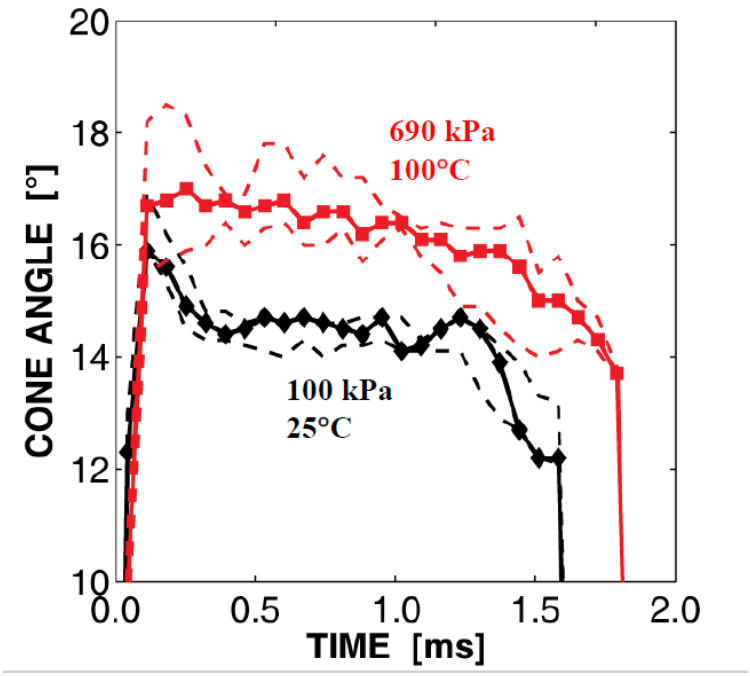

Figure 6.4: Cone angle *vs.* time for *n*-heptane sprays, averaged over five injections. Dashed lines indicate the full spread of acquired data.

Cone angle (Figure 6.4) displays interesting behavior. For room conditions, the first non-zero value after start of injection was approximately 12.9°. The cone angle then reached a transient peak value of 16.3° then decreased and established a somewhat steady value of 14.3°. This quasi-steady cone angle appears to have been established at approximately the same time at which the linear increase in liquid length ended. Cone angle then decreased at end of injection. For the elevated pressure/temperature conditions, behavior was similar with two exceptions. There was no initial transient peak and the quasi-steady value was somewhat higher at 16.5°.

The trends observed in these experiments were as expected. As pressure and temperature in the vessel increased, liquid length decreased and cone angle increased. These trends are consistent with the Siebers model, which is widely used to predict liquid lengths with generally good results [22]. The actual values of liquid length and cone angle, however, do not completely

agree between experiment and model. Using known or estimated injector parameters, fuel properties, and air-flow conditions, the Siebers model predicts a liquid length of 872.5 mm and a cone angle of 2.9° for room conditions. Thus the model predicts a much longer and narrower spray than was observed. For the elevated pressure/temperature conditions, the model predicts a liquid length of 93.0 mm and a cone angle of 6.5°. For these conditions, liquid lengths are actually in quite good agreement, though the predicted cone angle is much narrower than was observed.

A key aspect of the Siebers model, however, is the assumption of mixing-limited vaporization, in which evaporation of individual droplets is effectively ignored and liquid length is considered to be the downstream point at which sufficient air has been entrained into the fuel jet to completely vaporize the liquid fuel [22]. The model was developed for conventional diesel-engine conditions, which involve injection into very high pressure and temperatures characteristic of maximum-compression (*i.e.*, top-dead-center conditions). Siebers compared model results to available experimental data and noted that the model tended to predict longer liquid lengths than were seen experimentally at lower pressures and temperatures, though "lower" in this context is still considerably higher than in the current study [22]. Siebers also noted that the disparity between model and experiment increased as pressure and temperature decreased [22]. Thus, it is perhaps not surprising that the model predicts a much longer liquid length for room conditions than was observed experimentally. What remains unknown at this time is *why* this disparity exists between the Siebers model and experimental data. This is exactly why spray research in the vessel detailed in this thesis could prove to be extremely valuable.

7. CONCLUSIONS AND FUTURE WORK

7.1 SUMMARY AND CONCLUSIONS

A continuous-flow vessel with extensive optical access has been developed for characterization of engine-relevant fuel-injection and spray processes. Following were the major results of this work:

1. An assembly consisting of a primary vessel with optical access for spray visualization and an auxiliary vessel for exit-flow conditioning was designed and fabricated. The system has been designed for maximum pressure and temperature of 1380 kPa (200 psi) and 200°C, respectively.
2. A LabVIEW-based system has been developed to control the equipment and synchronize a high-speed imaging system with fuel injection. It also serves as a data-acquisition system for vessel pressure and temperature.
3. A spray-visualization system, consisting of a high-repetition-rate pulsed LED light source and a high-speed camera, has been set up to record time-resolved movies of fuel-injection events.
4. Initial spray-visualization experiments were performed to determine liquid lengths and spray cone angles for *n*-heptane at two different conditions. For room conditions (100 kPa, 25°C), liquid length and cone angle were on average 169.6 mm and 14.3°,

respectively. For 690 kPa and 100°C, liquid length and cone angle were on average 92.2 mm and 16.5°, respectively.

5. Experimental trends in liquid length and cone angle, with respect to thermodynamic conditions in the vessel, were in agreement with the well-established Siebers model. For the elevated pressure/temperature conditions, there was close agreement between measured and computed liquid lengths. For room conditions, the model predicted a much longer liquid length than was observed in the experiment.

Based on the vessel-development effort and initial baseline experiments, the following conclusions have been reached:

1. A detailed study of *n*-heptane spray properties is needed to establish a baseline data set for this vessel, which can be compared to work completed by others.
2. Results obtained in this vessel should prove particularly useful in advancing beyond the current understanding of fuel sprays for early-injection conditions.

7.2 FUTURE WORK

Future work will include use of the experimental setup to conduct high-speed imaging of both liquid- and vapor-phase fuel penetration over the range of achievable conditions for a variety of fuels. These detailed studies will begin with basic fuels, including primarily single-component hydrocarbons (*e.g.*, *n*-heptane, isooctane, *n*-decane, etc.). Studies will then proceed on multi-component blends of hydrocarbons, such as gasoline, diesel, and surrogate fuels.

Finally, studies will proceed to emerging alternative fuels such as biodiesel and its components. Once those fundamental experiments are complete for single injection events, further experiments will be conducted to observe multiple injections. In these studies, effects of varying injection duration and period between multiple injections will be studied. Beyond that, in the longer term, additional laser-based and optical-diagnostic techniques will be used to probe local spray properties such as fuel-droplet sizes and velocities.

Further aspects such as the evaporation rate of liquid fuel after end of injection and variability between spray events (*i.e.*, stochastic behavior) will be meticulously investigated using high-speed imaging. After gathering an extensive library of data for various fuels and experimental parameters for early-injection conditions, data obtained will be used to develop physics-based models that can predict the spray characteristics with high fidelity. Such models could be combined with the Siebers model, which is extremely useful for conventional engine conditions, and thus allowing researchers and designers to predict spray characteristics for a wide range of temperature and pressure conditions within the engine. With the ability to predict spray behavior for a wider range of conditions, it will be possible to develop engines and advanced engine-combustion strategies that result in cleaner and more-efficient operation.

9. REFERENCES

[1] Cengel, Y., & Boles, M. (2005)." Thermodynamics An Engineering Approach". (5th ed., pp. 113- 129). McGraw-Hill Highrr Education.

[2] N. Brehm, T. S. (1998). Nox Reduction in a Fuel Staged Combustor By Optimisation of the Mixing Process and Residence Time. *RTO MP-14* .

[3] Walsh, Philip P., "Gas turbine engineering handbook", 2nd edition, Fairfield NJ, 2004

[4] R. C. Steele, A. C. Jarrett, P. C. Malte, J. H. TOnouchi, and D. G. Nicol, "Variable affecting NOx formation in lean premixed combustion", Transactions of ASME, Vol. 119, pp. 102-107, January 1997

[5] Peden, D.B.,2001," Air pollution in asthma: effect of pollutants on airway inflammation "Ann Allergy Asthma Immunol 87(2001), pp. 12–17

[6] Sunyer, J,. 2001," Urban air pollution and chronic obstructive pulmonary disease: a review " Eur Respir J, Vol. 17, pp. 1024–1033.

[7] A Sydbom, A Blomberg, S Parnia *et al* .Health effects of diesel exhaust emissions Eur Respir J, Vol. 17, pp. 733–746.

[8] Nicolai,T.,1999," Air pollution and respiratory disease in children: what is the clinically relevant impact? ," Pediatr Pulmonol Suppl, Vol. 18 , pp. 9–13.

[9] http://www.eia.gov/todayinenergy/detail.cfm?id=12251#

[10] http://www.eia.gov/totalenergy/data/annual/pdf/sec2_3.pdf

[11] Baert, R.S.G., Frijters, P.J.M., Somers, B., Luijten, C.C.M., and de Boer, W., 2009, "Design and Operation of a High Pressure, High Temperature Cell for HD Diesel Spray Diagnostics: Guidelines and Results," SAE Paper 2009-01-0649.

[12] Espey, C. and Dec, J.E., 1993, "Diesel Engine Combustion Studies in a Newly Designed Optical-Access Engine Using High-Speed Visualization and 2-D Laser Imaging," SAE Paper 930971, SAE Trans. 102(4), pp. 703-723.

[13] Musculus, M.P.B., 2005, "Measurements of the Influence of Soot Radiation on in-Cylinder Temperatures and Exhaust Nox in a Heavy-Duty DI Diesel Engine," SAE Paper 2005-01-0925, SAE Trans. 114(3), pp. 845-866.

[14] Upatnieks, A., Mueller, C.J., and Martin, G.C., 2005, "The Influence of Charge-Gas Dilution and Temperature on DI Diesel Combustion Processes Using a Short-Ignition-Delay, Oxygenated Fuel," SAE Paper 2005-01-2088, SAE Trans. 114(4), pp. 773-785.

[15] Fisher, B.T. and Mueller, C.J., 2010, "Liquid Penetration Length of Heptamethylnonane and Trimethylpentane under Unsteady In-Cylinder Conditions," Fuel, 89(10), pp. 2673-2696.

[16] Oren, D.C., Wahiduzzaman, S., and Ferguson, C.W., 1984, "A Diesel Combustion Bomb: Proof of Concept," SAE Paper 841358.

[17] Naber, J.D. and Siebers, D.L., 1996, "Effects of Gas Density and Vaporization on Penetration and Dispersion of Diesel Sprays," SAE Paper 960034, SAE Trans. 105(3), pp. 82-111.

[18] Siebers, D.L., 1998, "Liquid-Phase Fuel Penetration in Diesel Sprays," SAE Paper 980809, SAE Trans. 107(3), pp. 1205-1227.

[19] Idicheria, C.A. and Pickett, L.M., 2007, "Quantitative Mixing Measurements in a Vaporizing Diesel Spray by Rayleigh Imaging," SAE Paper 2007-01-0647, SAE Trans. 116(3), pp. 490-504.

[20] Weber, J., Spiekermann, P., and Peters, N., 2005, "Model Calibration for Spray Penetration and Mixture Formation in a High Pressure Fuel Spray Using a Micro-Genetic Algorithm and Optical Data," SAE Paper 2005-01-2099.

[21] Ochoterena, R., Larsson, M., Andersson, S., and Denbratt, I., 2008, "Optical Studies of Spray Development and Combustion Characterization of Oxygenated and Fischer-Tropsch Fuels," SAE Paper 2008-01-1393.

[22] Siebers, D.L., 1999, "Scaling Liquid-Phase Fuel Penetration in Diesel Sprays Based on Mixing-Limited Vaporization," SAE Paper 1999-01-0528, SAE Trans. 108(3), pp. 703-728.

[23] Fisher, B.T., Knothe, G., and Mueller, C.J., 2010, "Liquid-Phase Penetration under Unsteady In-Cylinder Conditions: Soy- and Cuphea-Derived Biodiesel Fuels Versus Conventional Diesel," Energ. Fuels, 24(9), pp. 5163-5180.

[24] Dent JC. Basis for the comparison of various experimental methods for studying spray penetration (SAE Paper 710571). SAE Trans 1971;80:1881–4.

[25] Hay N, Jones JL. Comparison of the various correlations for spray penetration (SAE Paper 720776). 1972.

[26] Hiroyasu H, Arai M. Structure of fuel sprays in diesel engines (SAE Paper 900475). SAE Trans 1990;99:1051–61.

[27] Varde KS, Popa DM. Diesel fuel spray penetration at high injection pressures (SAE Paper 830448). SAE Trans 1983;92:369–408.

[28] Chiu WS, Shahed SM, Lyn WT. A transient spray mixing model for diesel combustion (SAE Paper 760128). SAE Trans 1976;85:502–12.

[29] Espey C, Dec JE. The effect of TDC temperature and density on the liquid-phase fuel penetration in a D.I. diesel engine (SAE Paper 952456). SAE Trans 1995;104:1400–14.

[30] Ricart LM, Xin J, Bower GR, Reitz RD. In-cylinder measurement and modeling of liquid fuel spray penetration in a heavy-duty diesel engine (SAE Paper 971591). SAE Trans 1997;106:1622–40.

[31] Zhang L, Tsurushima T, Ueda T, Ishii Y, Itou T, Minami T, et al. Measurement of liquid phase penetration of evaporating spray in a DI diesel engine (SAE Paper971645). 1997.

[32] Canaan RE, Dec JE, Green RM, Daly DT. The influence of fuel volatility on the liquid-phase fuel penetration in a heavy-duty D.I. diesel engine (SAE Paper 980510). SAE Trans 1998;107:583–602.

[33] Wang T-C, Han J-S, Xie X, Lai M-C, Henein NA, Bryzik W. Direct visualization of high pressure diesel spray and engine combustion (SAE Paper 1999-01-3496).1999.

[34] Desantes JM, Pastor JV, Payri R, Pastor JM. Experimental characterization of internal nozzle flow and diesel spray behavior. Part II: evaporative conditions. Atomization Sprays 2005;15:517–43.

[35] Desantes JM, Pastor JM, Martinez S, Riesco JM. Experimental characterization of the liquid phase penetration on evaporating diesel sprays (SAE Paper 2005-01-2095). 2005.

[36] Martinez-Martinez S, Sanchez-Cruz FA, Riesco-Avila JM, Gallegos-Munoz A, Aceves SM. Liquid penetration length in direct diesel fuel injection. Appl Therm Eng 2008;28:1756–62.

[37] Ochoterena R, Larsson M, Andersson S, Denbratt I. Optical studies of spray development and combustion characterization of oxygenated and Fischer–Tropsch fuels (SAE Paper 2008-01-1393). 2008.

[38] Takeda, Y., Keiichi, N., and Keiichi, N., 1996, "Emission Characteristics of Premixed Lean Diesel Combustion with Extremely Early Staged Fuel Injection," SAE Paper 961163, SAE Trans. 105(4), pp. 938-947.

[39] Drake, M.C., Fansler, T.D., Solomon, A.S., and Szekely, G.A., 2003, "Piston Fuel Films as a Source of Smoke and Hydrocarbon Emissions from a Wall-Controlled Spark-Ignited Direct-Injection Engine," SAE Paper 2003-01-0547, SAE Trans. 112(3), pp. 762-783.

[40] Mueller, C.J., Martin, G.C., Briggs, T.E., and Duffy, K.P., 2004, "An Experimental Investigation of in-Cylinder Processes under Dual-Injection Conditions in a Di Diesel Engine," SAE Paper 2004-01-1843, SAE Trans. 113(3), pp. 1146-1164.

[41] Hardy, W.L. and Reitz, R.D., 2006, "A Study of the Effect of High EGR, High Equivalence Ratio, and Mixing Time on Emissions Levels in a Heavy-Duty Diesel Engine for Pcci Combustion," SAE Paper 2006-01-0026.

[42] Kashdan, J.T., Mendez, S., and Bruneaux, G., 2007, "On the Origin of Unburned Hydrocarbon Emissions in a Wall-Guided, Low NOx Diesel Combustion System," SAE Paper 2007-01-1836, SAE Trans. 116(4), pp. 234-257.

[43] Opat, R., Ra, Y., Gonzalez, M.A., Krieger, R., Reitz, R.D., Foster, D.E., Durret, R.P., and Siewert, R.M., 2007, "Investigation of Mixing and Temperature Effects on HC/CO Emissions for Highly Dilute Low Temperature Combustion in a Light-Duty Diesel Engine," SAE Paper 2007-01-0193.

[44] Martin, G.C., Mueller, C.J., Milam, D.M., Radovanovic, M.S., and Gehrke, C.R., 2008, "Early Direct-Injection, Low-Temperature Combustion of Diesel Fuel in an Optical Engine Utilizing a 15-Hole, Dual-Row, Narrow-Included-Angle Nozzle," SAE Paper 2008-01-2400, SAE Int. J. Engines 1(1), pp. 1057-1082.

[45] Martinez-Martinez, S., Sanchez-Cruz, F.A., Riesco-Avila, J.M., Gallegos-Munoz, A., and Aceves, S.M., 2008, "Liquid Penetration Length in Direct Diesel Fuel Injection," Appl. Therm. Eng., 28(14-15), pp. 1756-1762.

[46] Dec, J.E. and Tree, D.R., 2001, "Diffusion-Flame / Wall Interactions in a Heavy-Duty DI Diesel Engine," SAE Paper 2001-01-1295, SAE Trans. 110(3), pp. 1618-1634. 10 Copyright © 2013 by ASME

[48] "Engine Combustion Network," Online source, http://www.sandia.gov/ecn/index.php. Accessed Apr. 7, 2013.

[49] Yokota H, Kudo Y, Nakajima H, Kakegawa T, Suzuki T. A new concept for low emission diesel combustion (SAE Paper 970891). SAE Trans 1997;106:1479–90.

[50] Klingbeil AE, Juneja H, Ra R, Reitz RD. Premixed diesel combustion analysis in a heavy-duty diesel engine (SAE Paper 2003-01-0341). SAE Trans 2003;112:445–59.

[51] Mueller CJ, Martin GC, Briggs TE, Duffy KP. An experimental investigation of in cylinder processes under dual-injection conditions in a DI diesel engine (SAE Paper 2004-01-1843). SAE Trans 2004;113:1146–64.

[52] Kanda T, Hakozaki T, Uchimoto T, Hatano J, Kitayama N, Sono H. PCCI operation with early injection of conventional diesel fuel (SAE Paper 2005-01-0378). SAE Trans 2005;114:584–93.

[53] Kook S, Bae C, Miles PC, Choi D, Pickett LM. The influence of charge dilution and injection timing on low-temperature diesel combustion and emissions (SAE Paper 2005-01-3837). SAE Trans 2005;114:1575–95.

[54] Lechner GA, Jacobs TJ, Chryssakis CA, Assanis D, Siewert RM. Evaluation of a narrow spray cone angle, advanced injection timing strategy to achieve partially premixed compression ignition combustion in a diesel engine (SAE Paper 2005-01-0167). SAE Trans 2005;114:394–404.

[55] Minato A, Tanaka T, Nishimura T. Investigation of premixed lean diesel combustion with ultra high pressure injection (SAE Paper 2005-01-0914). SAE Trans 2005;114:756–64.

[56] Musculus MPB. Multiple simultaneous optical diagnostic imaging of early injection low-temperature combustion in a heavy-duty diesel engine (SAE Paper 2006-01-0079). SAE Trans 2006;115:83–110.

[57] Fang T, Coverdill RE, Lee CF, White RA. Smokeless combustion within a small bore HSDI diesel engine using a narrow angle injector (SAE Paper 2007-01-0203). SAE Trans 2007;116:255–70.

[58] Musculus, M.P.B., "Multiple Simultaneous Optical Diagnostic Imaging of Early-Injection, Low-Temperature Combustion in a Heavy-Duty Diesel Engine". 2006-01-0079, SAE TRANSACTIONS 115(3): p. 83, 2006

[59] Martin, G.C., Mueller, C.J., Milam, D.M., Radovanovic, M.S. et al., "Early Direct-Injection, Low-Temperature Combustion of Diesel Fuel in an Optical Engine Utilizing a Hole, Dual-Row, Narrow-Included-Angle Nozzle," *SAE Int. J.* Engines 1(1):1057-1082, 2008.

[60] Pickett, L.M., Kook, S., and Williams, T.C., "Transient Liquid Penetration of Early-Injection Diesel Spray," *SAE Int.J.* Engines 2(1): 785-804, 2009.

[61] Abraham J, Givler SD. Conditions in which vaporizing fuel drops reach a critical state in a diesel engine (SAE Paper 1999-01-0511). SAE Trans 1999;108: 601–12.

[62] Ricart LM, Reitz RD, Dec JE. Comparisons of diesel spray liquid penetration and vapor fuel distributions with in-cylinder optical measurements. Trans ASME 2000;122:588–95.

[63] Post S, Abraham J. A computational study of the processes that affect the steady liquid penetration in full-cone diesel sprays. Combust Sci Technol 2001;165:1–40.

[64] Rotondi R. Simulation of liquid phase penetration in diesel spray (SAE Paper 2001-01-3229). 2001.

[65] Senecal PK, Pomraning E, Richards KJ, Briggs TE, Choi CY, McDavid RM, et al. Multi-dimensional modeling of direct-injection diesel spray liquid length and flame lift-off length using CFD and parallel detailed chemistry (SAE Paper 2003-01-1043). SAE Trans 2003;112:1331–51.

[66] Desantes JM, Payri R, Salvador FJ, Gil A. Development and validation of a theoretical model for diesel spray penetration. Fuel 2006;85:910–7.

[67] Tonini S, Gavaises M, Arcoumanis C, Theodorakakos A. Prediction of liquid and vapor penetration of high pressure diesel sprays (SAE Paper 2006-01-0242).2006.

[68] Desantes JM, López JJ, García JM, Pastor JM. Evaporative diesel spray modeling. Atomization Sprays 2007;17:193–231.

[69] Siewert RM. A phenomenological engine model for direct injection of liquid fuels, spray penetration, vaporization, ignition delay, and combustion (SAE Paper 2007-01-0673). 2007.

[70] Pastor JV, López JJ, García JM, Pastor JM. A 1D model for the description of mixing-controlled inert diesel sprays. Fuel 2008;87:2871–85.

[71] Marques, O., " Practical Image and Video Processing Using Matlab," John Wiley & Sons, Inc., 2011. p. 389-391

APPENDICES

APPENDIX A. VESSEL ASSEMBLY AND INSTALLATION

This document gives a detailed explanation of assembly of the fuel-injection flow vessel, as well as the integration of pressure and temperature sensors.

A.1 PRIMARY ASSEMBLY PARTS

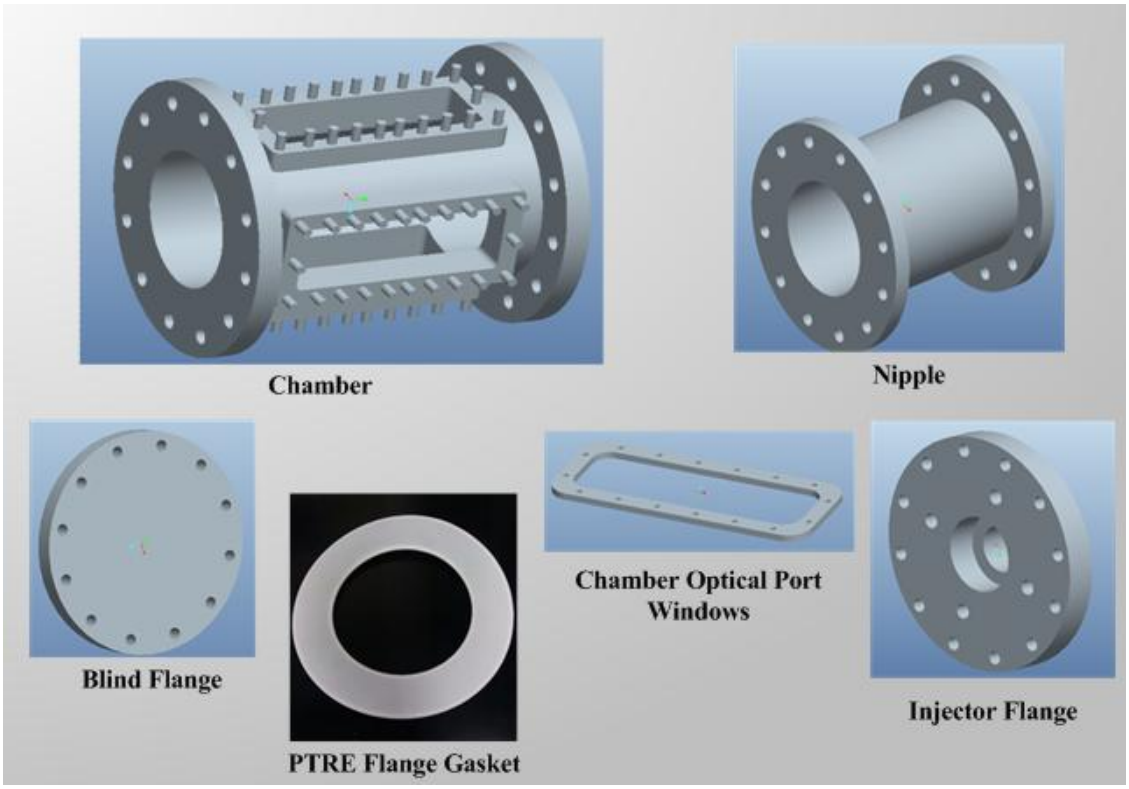

Figure A.1: The Major Components of the flow vessel to be assemble

A.2 PRE SETUP INSTRUCTIONS

1) First six optical posts, measuring 6” in length and ¼” in diameter, should be screwed in the optical work bench as illustrated in Figure A.2 below with the appropriate dimension spacing and location.

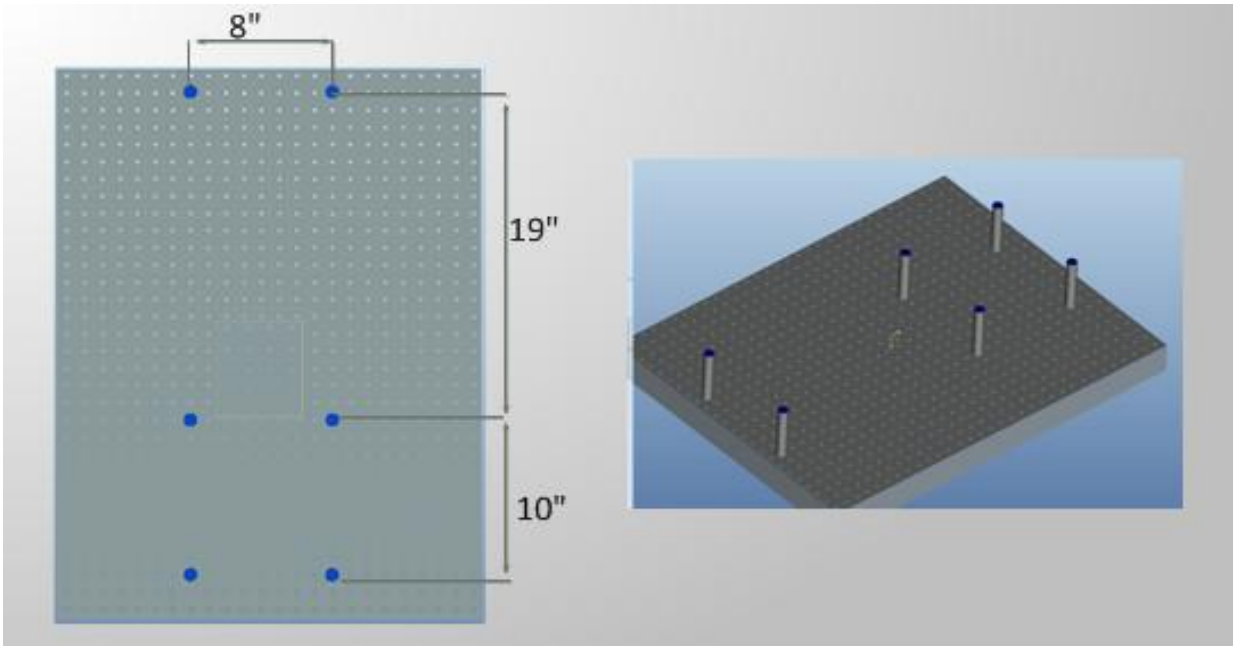

Figure A.2: Optical post locations.

2) After inserting the post, 3 chamber saddle plates should then be attached using 20 socket-head 1/4-20 cap screws.

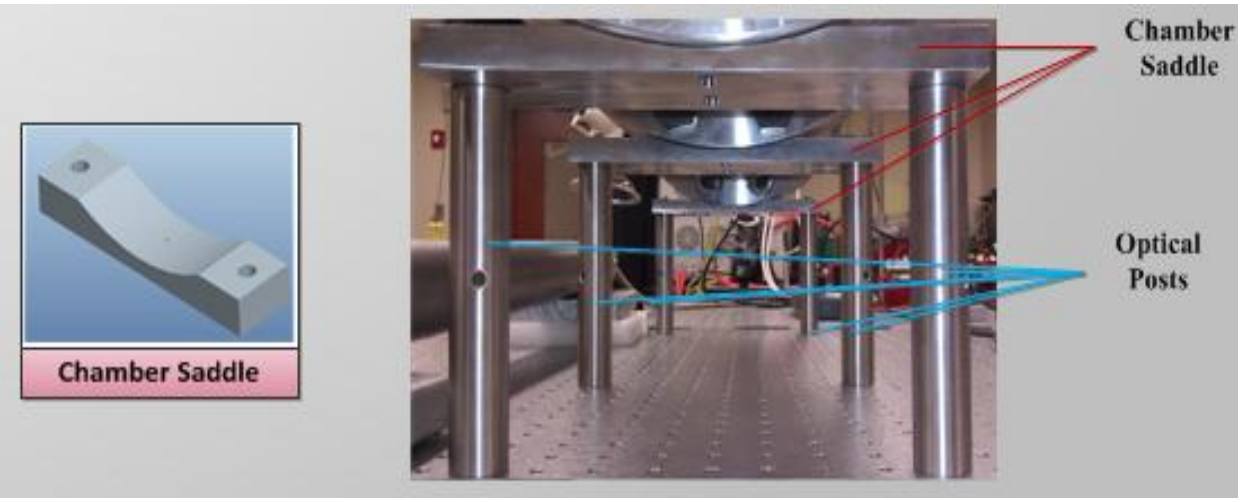

Figure A.3: Chamber saddle and assembly of 3 chamber saddles with 6 optical posts.

3) The chamber, which is one of the key pieces, should then be placed safely on the chamber saddle plate along with the nipple adjacent to it. Both components should then be oriented on the saddle plate until the flanges hoop threaded holes are aligned with the saddle slot.

4) Next, insert 20 socket-head 1/4-20 cap screws into each saddle slot and screw, thus connecting each saddle to the chamber or the nipple that lies on it.

A.3 CHAMBER SETUP INSTRUCTIONS

5) The chamber and the nipple are fastened together using 12 3/4-10 threaded hex-head bolts and nuts after placing a PTFE flange gasket with a 6” inner diameter between them

as illustrated in Figure A.4 along with the appropriate fastening pattern illustrated in Figure A.5.

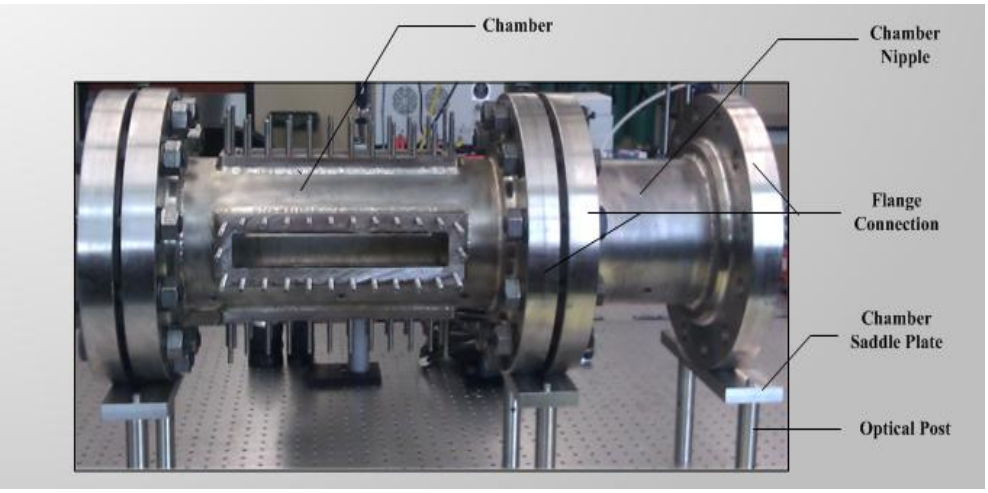

Figure A.4: Illustrates the chamber, chamber nipple and blind flange assembled on the saddle plate

6) When fastening the bolt ensure that a torque wrench is used to torque the hex bolts to 10, 25, 50, 100 and finally 200 ft-lb, using the appropriate sequence pattern seen in Figure A.5.

Figure A.5: The torque pattern for the hex head bolts for the orifice or injector flange

7) Repeat step 6 for the orifice flange.

A.4 CHAMBER WINDOWS INSTALLATION

8) Place the custom gasket over the protruding threaded studs of the chamber optical port.
9) Then carefully align the window and place it over the studs so that the gasket is sandwiched between the chamber and the high-pressure quartz window.
10) Following this, 1/4" washers and 1/4-28 hex nuts are place over the studs and fastened in the pattern shown below in Figure A.6. Begin first by tightening each nut 1/2 turn only and then move on to each subsequent nut and continue this process until a final torque of 4ft- lb (48 in-lb) is reached.

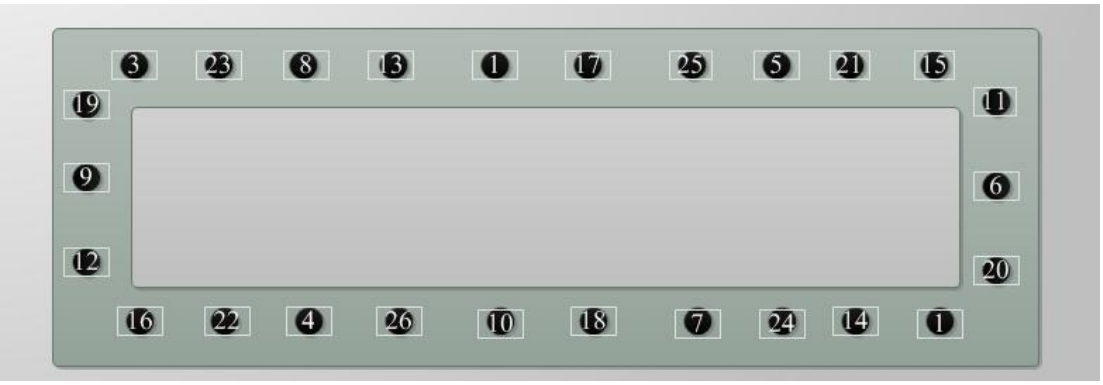

Figure A.6: Hex nuts tightening pattern

11) Repeat steps 9 through 11 for the remaining 3 window installations.

A.5 INJECTOR AND INJECTOR FLANGE INSTALLATION PROCESS

12) First the injector flange is placed on the flat surface allowing the injector holder to point upwards similar to Figure A.7. Once the flange is in this position, carefully center co-axially the first flow-conditioning plate on the injector holder and press down until it hits the stop.

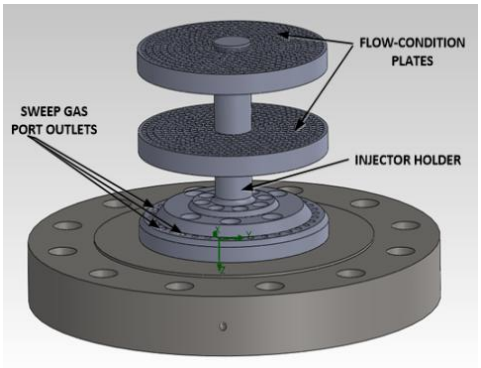

Figure A.7: Injector flange assembly.

13) Following this a separator is placed on the holder in a similar manner, followed by the second flow-conditioning plate and a snap ring at the end, as shown in Figure A.8.

14) Place a PTFE flange gasket concentrically on top of the injector flange

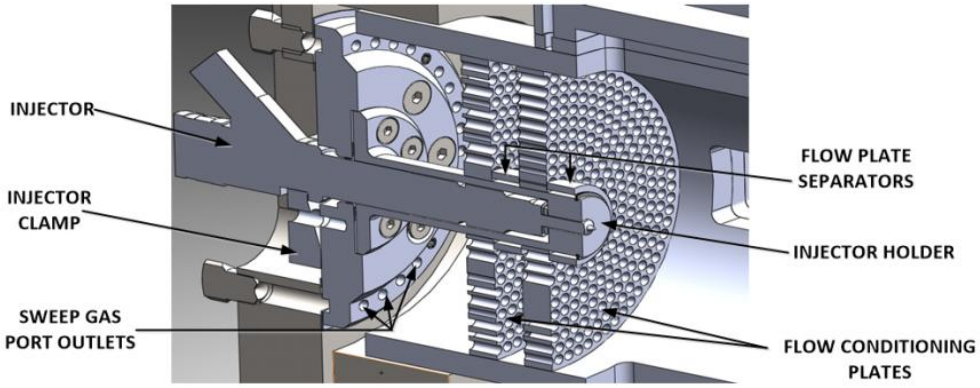

Figure A.8: Section view of primary vessel entry area

15) The injector flange with the flow-conditioning plates is now ready for its final assembly phase. This requires carefully inserting the injector holder with the plates inside the primary section of the vessel until the primary vessel flange and the injector flange are perfectly aligned with the PTFE flange gasket between them.
16) Once both flanges are perfectly aligned repeat step 6 to fasten the bolts.

17) Place the o-ring for the injector in the designated area and then carefully insert the injector inside the injector holder until the tip is visible inside the chamber.

18) After the injector is properly placed in the holder such that the pressurized fuel inlet is pointing upwards use the injector clamp and torque to 30 ft-lb to firmly hold the injector.

APPENDIX B. EXPERIMENTAL APPARATUS CALIBRATION

B.1 PRESSURE SENSOR CALIBRATION

This document describes how to calibrate the pressure sensor PX309-200A5V Omega Engineering, etc. using a dead-weight tester and how to integrate the sensor with LabView to obtain the pressure readings. The readings recorded are then tabulated and plotted to obtain the appropriate calibration constants.

Apparatus/Equipment:

1) Omegadyne Inc. PX309-200A5V Pressure Sensor
2) Omega Engineering Inc. PSR-24S 24V Power Supply
3) Dead-weight tester with 195lbs of weights
4) NI Compact DaQ with National Instrument 2301 module
5) Computer with Microsoft Office Excel
6) Wires and Screw Drivers
7) LabView VI

Experimental Setup:

1) First the pressure sensor should be connected to the 24V power supply and then the pressure analogue signal wire (orange wire) was then connected to port 1 on the NI 2301 module.

2) Following this, the COM port of the 2301 module is then connected to the ground port of the power supply, thus allowing for the appropriate sensor readings to be obtained.

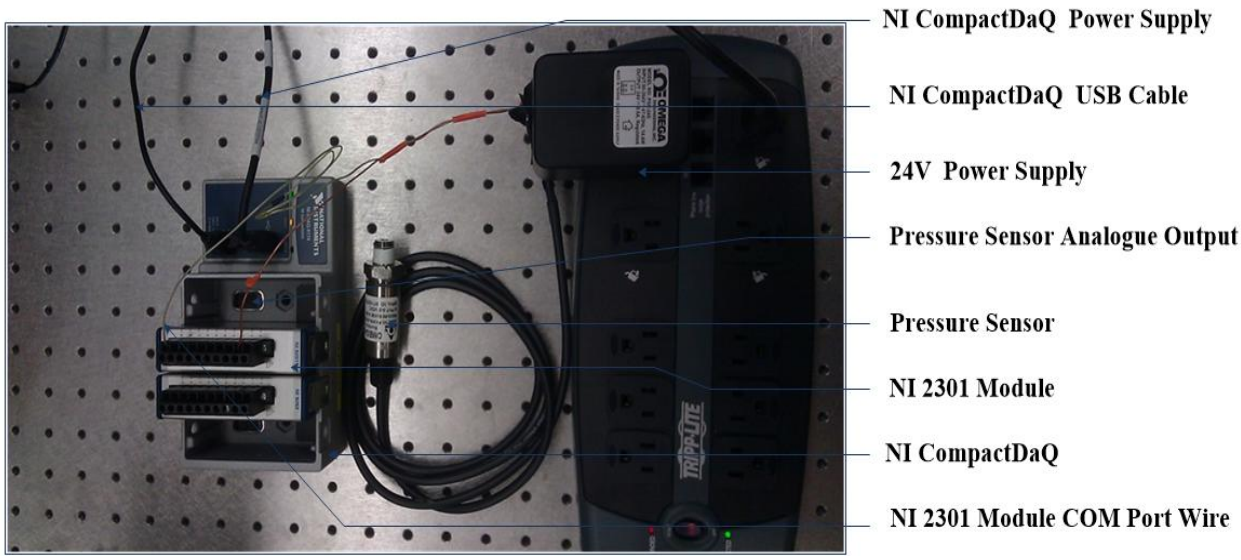

Figure B.1: Shows the pressure sensor integration is DAQ module

3) Before connecting the sensor to the dead weight tester, ensure that LabView is up and running.

   a. In the block diagram of the LabView, drag a DaQ Assistant onto the block diagram panel as illustrated in Figure B.2.

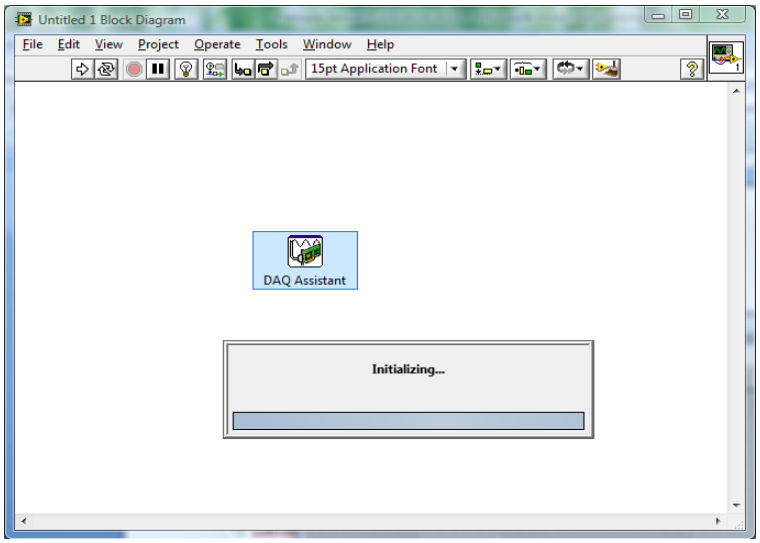

Figure B.2*:* Block Diagram of LabView with the DaQ Assistant initializing.

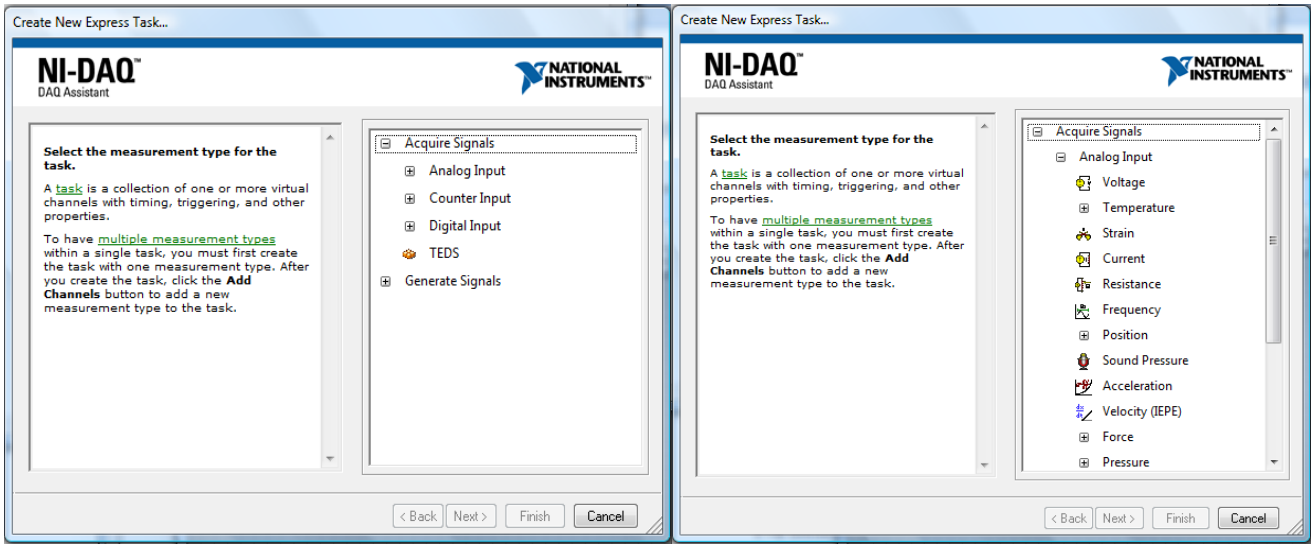

Figure B.3: NI-DAQ Assistant setup window.

b. The analogue input must be selected, followed by voltage as illustrated in Figure B.3. After the voltage selection, then 2301 module, this then displays all of the voltage ports. All ports can be deleted except input port 1, to which the sensor analogue output should be connected.

c. In the voltage range, the minimum and maximum voltage should be changed to 0V and 5V, respectively.

d. Change the sample rate to continuous and select the *Run* button for preliminary testing to see if there is a voltage input. The voltage input can be observed by turning on and off the 24V power supply for the sensor while watching the graph displayed on the VI seen below in Figure B.4.

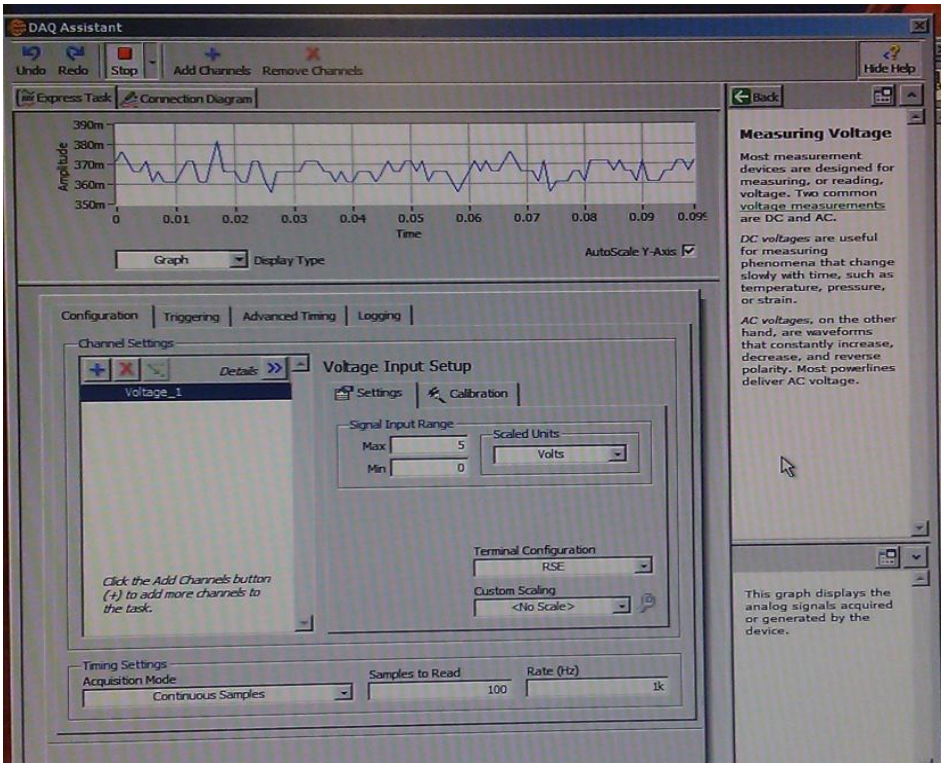

Figure B.4: DaQ Assistant VI with the appropriate settings and a graphical display of the input voltage.

4) Connect the sensor to the dead-weight tester using the adapter located on the dead-weight tester column illustrated in Figure B.5 below.

Figure B.5: Dead-weight tester.

5) Before adding weights, in order to apply different pressures to the sensor, ensure that the lead screw is fully extended, that the oil reservoir is half-way full after the lead screw is extended, that the valve is closed, and that the pressure sensor is properly connected to avoid oil leakage.

6) Add weights corresponding to 25 psig on the platform and turn the lead screw inwards until the platform rises.

7) Open the valve to ensure that the platform shoulder is above the piston and can spin freely.

8) Allow the platform to spin for at least 6 revolutions to ensure that it is totally free from friction effects that might alter the readings.

9) Once the platform is rotating freely, record the average voltage readings from the pressure sensor voltage output display.

10) Repeat Steps 8 and 9 for weights corresponding to 40, 80, 120,150 and 180 psig on the platform.

11) Repeat Steps 6 through 10 again, but this time randomizing the order of the weights.

12) After the final experiment, extend the lead screw, remove the weights, close the valve to isolate the oil flow, and then remove the sensor.

Results from previous calibration:

Table B.1: Data for the first experiment with the corresponding voltage for each psig (weight) added on the platform.

PSIG	PSI (Absolute)	Volts
25	39.7	0.985
40	54.7	1.36
80	94.7	2.36
120	134.7	3.36
150	164.7	4.1
180	194.7	4.85

Table B.2: Data for the second experiment with the corresponding voltage for each psig (weight) added on the platform.

PSIG	PSI (Absolute)	Volts
180	194.7	4.85
120	134.7	3.355
40	54.7	1.36
150	164.7	4.1025
25	39.7	0.985
80	94.7	2.3555

Discussion and Conclusion:

The data obtained from both tables were graphed, in order to obtain a linear fit that will give the necessary information to obtain athe relationship between the pressure and voltage.

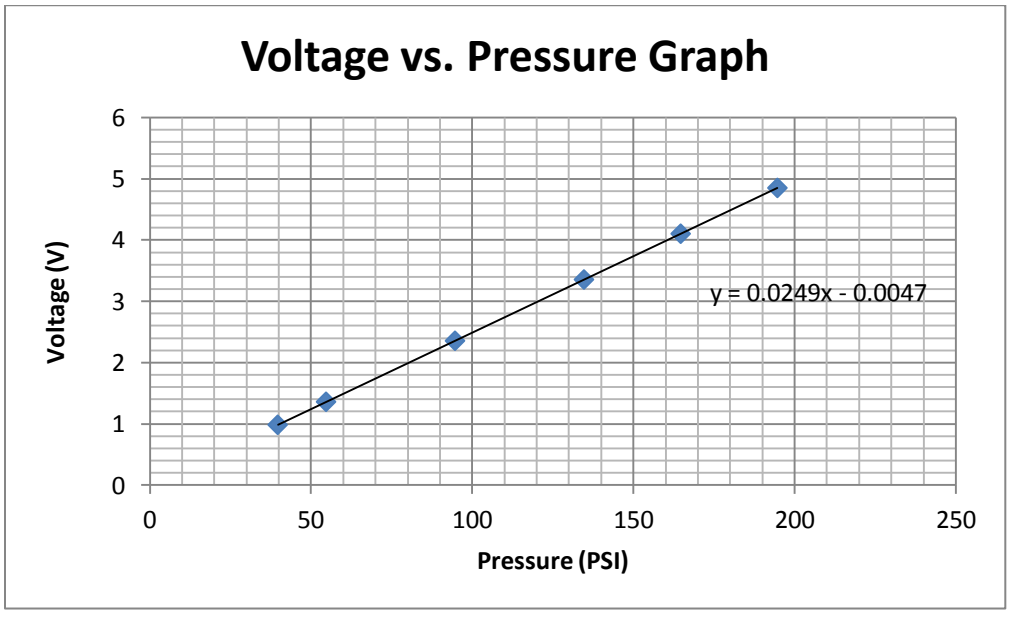

Figure B.6: Voltage vs. pressure for the data obtained for the first experiment.

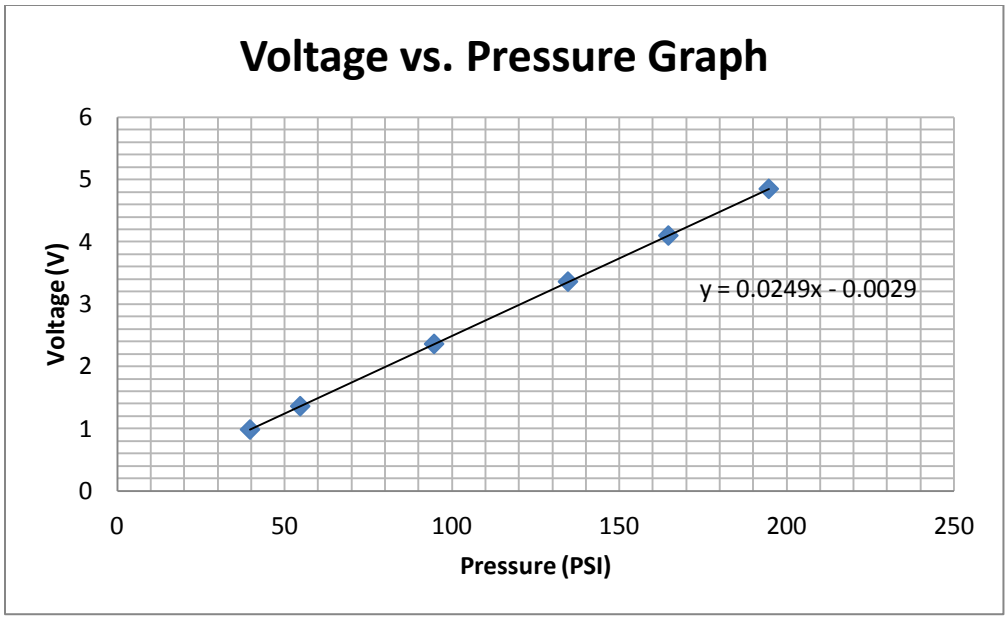

Figure B.7: Voltage vs. pressure for the data obtained for the second experiment.

From the linear fit data illustrated in both graphs, we can extrapolate that the average slope and intercept for both experiments was 0.0249 and -0.0038, respectively.

Final Calibration Stage:

13) After obtaining the slope a new scale was created, so that the DaQ Assistant would display the appropriate amplitude corresponding to the pressure instead on the voltage. This was done by first selecting the tool icon adjacent to custom scaling box.

14) A new window then pops up, and in the scaling parameters region seen below in Figure 8, change the slope to (1/0.0249), which is approximately 40 while leaving the y-intercept unchanged since 0.0038 is approximately equal to zero.

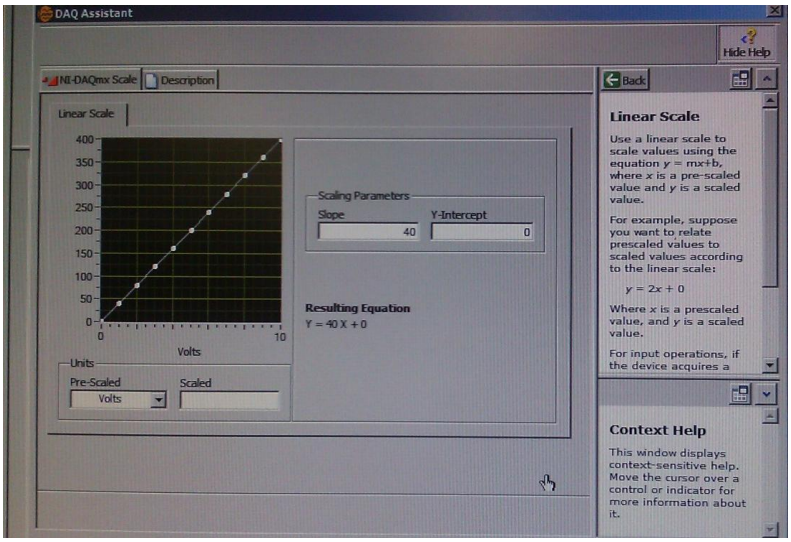

Figure B.8: Custom scaling window.

15) After the appropriate changes were made the “ok” button was selected

16) The DaQ Assistant VI was rerun to see if the calibrations constants were appropriate to measure the atmospheric pressure seen below in Figure A.17.

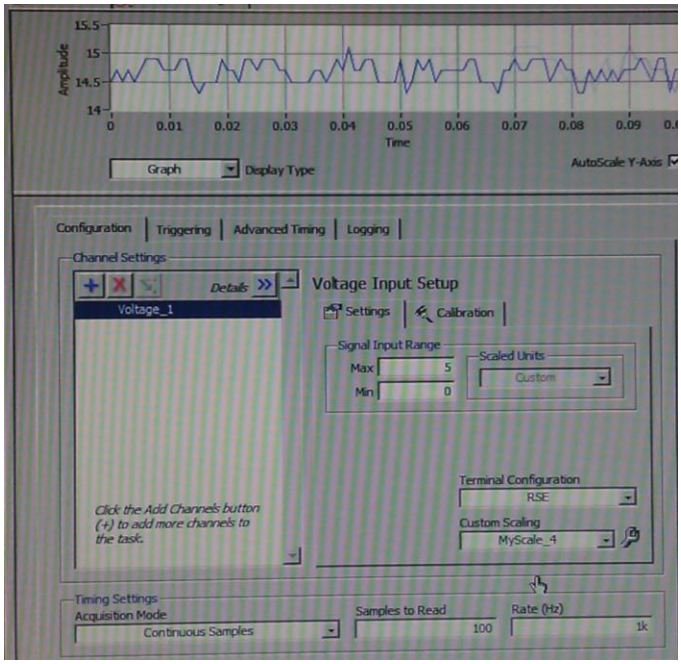

Figure B.9: Scaled amplitude corresponding to the pressure.

9.2.2 B.2 FUEL INJECTOR CALIBRATION FOR N-HEPTANE

This section focuses on the injector calibration of *n*-heptane using the Drivven SADI Driver. In order to start this calibration process, it is necessary that the injection hardware system is set up as seen in Figure B.10 below.

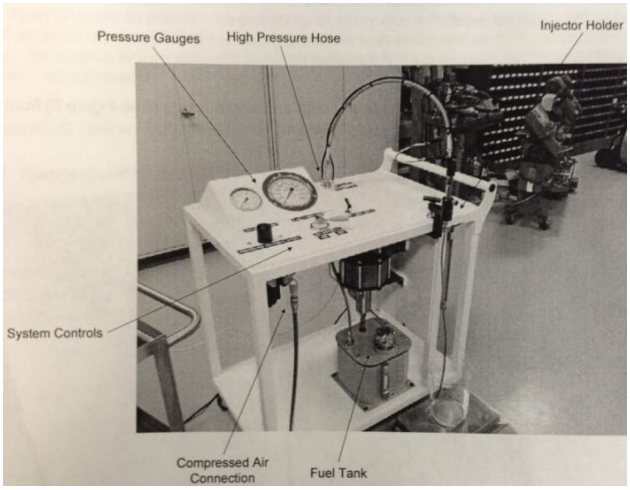

Figure B.10: Injection hardware system setup.

1. Once all of the components are connected and torque meets the appropriate requirements (see Exergy Pressure Amplified Fuel Cart Manual), the CRio DAQ must to be set up appropriately allowing Calview, the Drivven software, to drive the injector. The necessary connections are seen in Figure B.11 below.

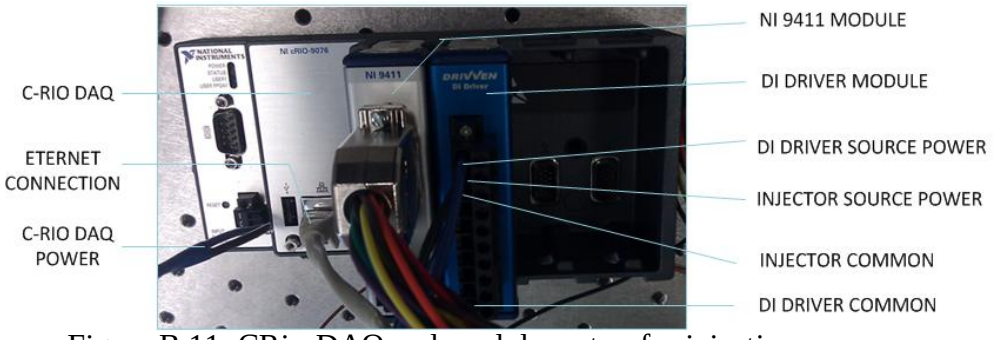

Figure B.11: CRio-DAQ and modules setup for injection.

2. The NI 9411 module must be placed in the first slot and the DI Driver in the second slot of the CRio DAQ; failure to do so will result in the software error or Calview will be unable to complete the injection process.

3. Once the hardware and electronics are set up, it is time to transition to running or calibrating the injector in Calview. To initiate Calview simply select the Calview icon on the desktop and the following window will pop up as seen in Figure B.12.

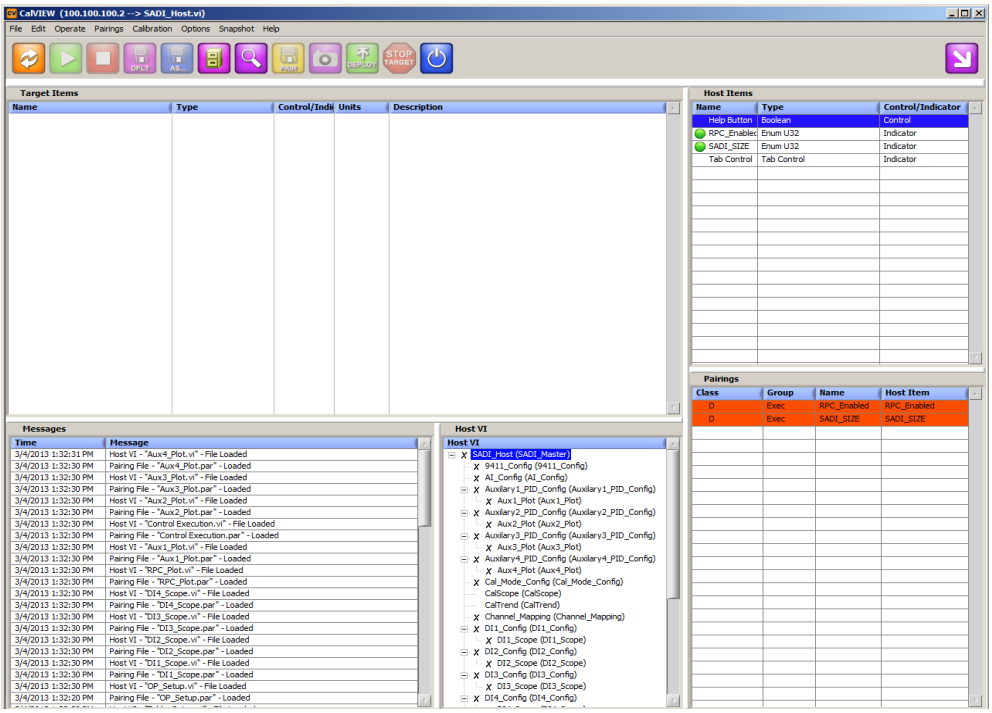

Figure B.12: Calview popup window.

4. If there are any connection errors, they will be displayed in the Target and Host Item panel. However, if all connections are successful the target items will appear without errors and the SADI Driver popup window will also be displayed similar to Figure B.13.

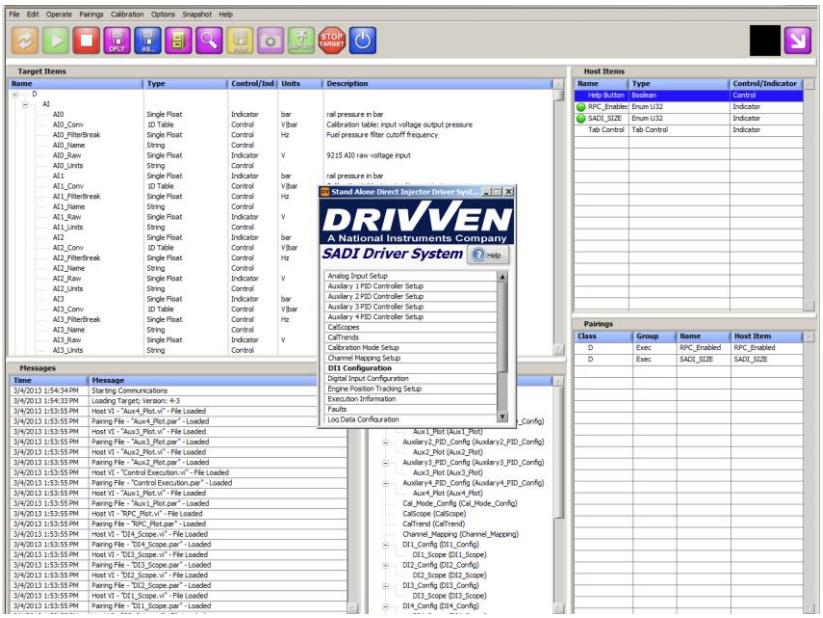

Figure B.13: CalView communicating with all of the target items and display of the SADI driver system.

5. Once the SADI Driver system popup window appears, see Figure B.14 for a closer view, there are multiple options available. However, before doing anything the injector current profile must be entered correctly in DI1 Configuration section.

Figure B.14: Expanded view of the SADI driver system popup window.

6. For this calibration process, and in subsequent experiments, a Bosh CRIN3 solenoid injector, with a single on-axis orifice with 100-μm diameter, was used. The current profile for this injector illustrated in Figure B.15; for each variable value, see Table A.3 were entered in the *DI1-Configuration* popup window shown in Figure B.16.

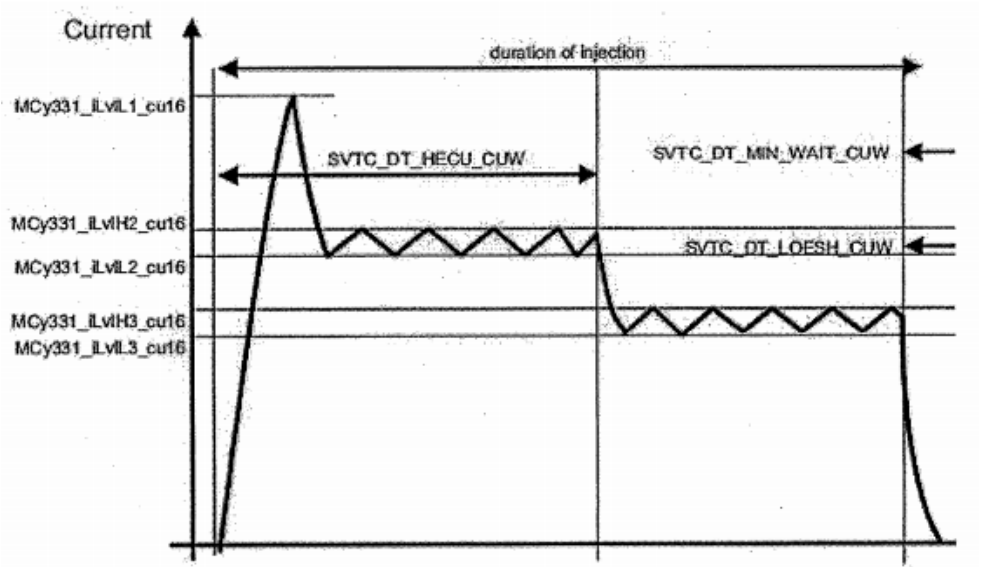

Figure B.15: Electric current profile for the injector.

Table B.3: Shows the values necessary to interpret Figure B.15.

	Requirement	**Description**
MCy331_iLvIL1_cu16	24.8 A	Initial Peak
MCy331_iLvIH2_cu16	16.0 A	First hold stage high limit
MCy331_iLvIL2_cu16	14.2 A	First hold stage low limit
MCy331_iLvIH3_cu16	10.1	Second hold stage high limit
MCy331_iLvIL3_cu16	11.9 A	Second hold stage low limit
SVTC_DT_HECU_CUW	450 μs	First hold stage duration
Firing voltage	48V	62 max , 38 min

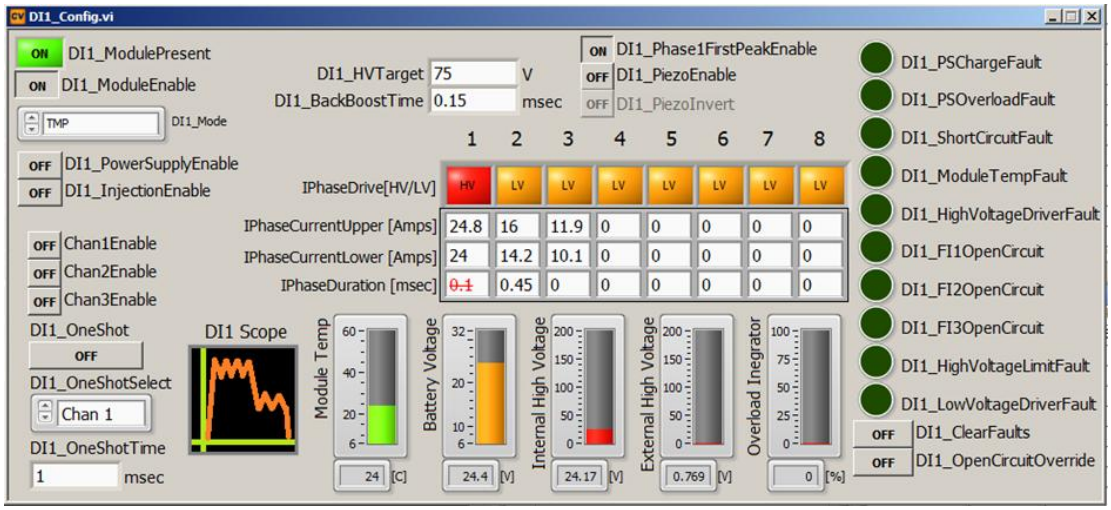

Figure B.16: DI1-Configuration VI.

7. The first phase the current needs to instantaneously hit the peak current. This can be achieved by selecting the *DI1_Phase1FirstPeakEnable* push button followed by entering the peak current of 24.8 in the first row and column of the current profile description table. Since this is an immediate peak the *IPhaseCurrentLower* value of 24 is not required.

8. The last row for the current profile description table is the *IPhaseDuration* which as the name suggests allows for that current phase duration during the injection. The second hold stage value of 0.45 ms was entered as indicated.

9. For the final phase of any current profile, it is recommended to leave blank.

Please Note: *For example, if the entire injection profile is 1 ms and the second stage hold is 0.45 ms, there is only 0.55 ms remaining for the current profile to hit peak current and execute the final phase. However since the time is usually unknown for the first phase (when it hits peak current) it is best to leave it blank and the software will do the iteration and internally adjust both the first and final stage to obtain an injection duration of 1 ms.*

10. After completing the current profile table setup, on the left side of the *DI1-Configuration vi*, the *DI1_Module Present and EnableI* was turn on and withiin the DI1_Mode dropdown box the *Cal* option was selected which basically sets the SADI Driver in calibration mode.

11. Following this, after ensuring that the buttons for *DI1_PowerSupplyEnable* and *DI1_InjectionE* are enabled, the SADI Driven System popup window was reopened and the Calibration Mode Setup option was selected.

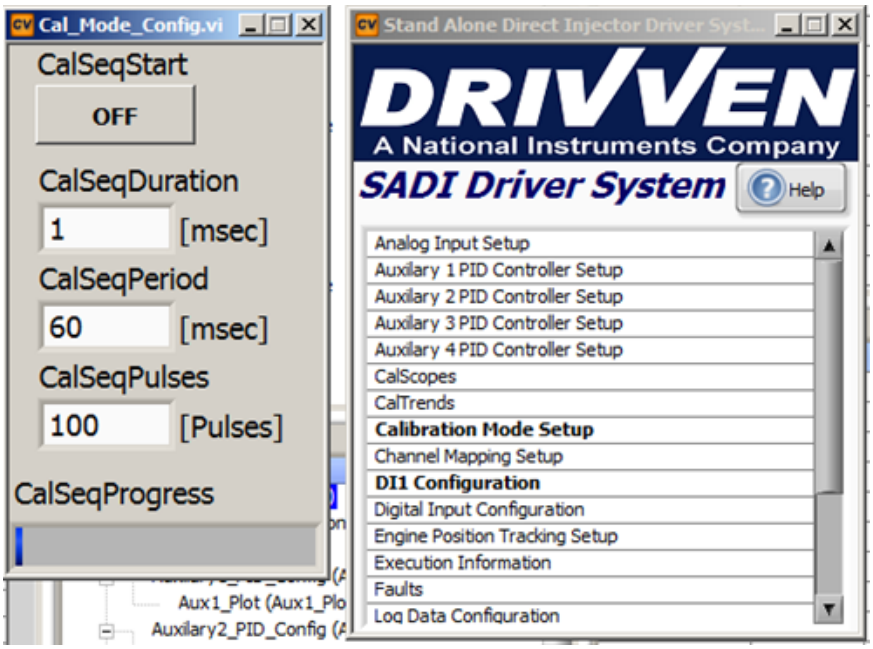

Figure B.17: SADI driver system and the calibration mode popup windows.

12. Once the desired values were entered in the Calibration mode configuration VI, the *CalSeqStart* button was selected to initialize the injection process. For this experiment the duration and the period between injections was specified at 1 and 600 ms respectively.

13. In order to calibrate the injector for n-heptane, various numbers of injections were run and the amount injected for each run was weighed. The result of each experiment corresponding to the number of injections and weighed data are shown in Table B.4.

Table B.4: Number of injections and weighted data for replicate injector calibration experiments.

Exp	Number of Injections	Weight
1	100	0.603
2	200	1.360
3	150	1.010
4	100	0.8196
5	250	1.697
6	200	1.334
7	150	1.000
8	250	1.566
9	100	0.620
10	250	1.583
11	150	0.950
12	200	1.130

Plotting the data obtained from Table B.4 leads to the graph displayed in Figure B.18.

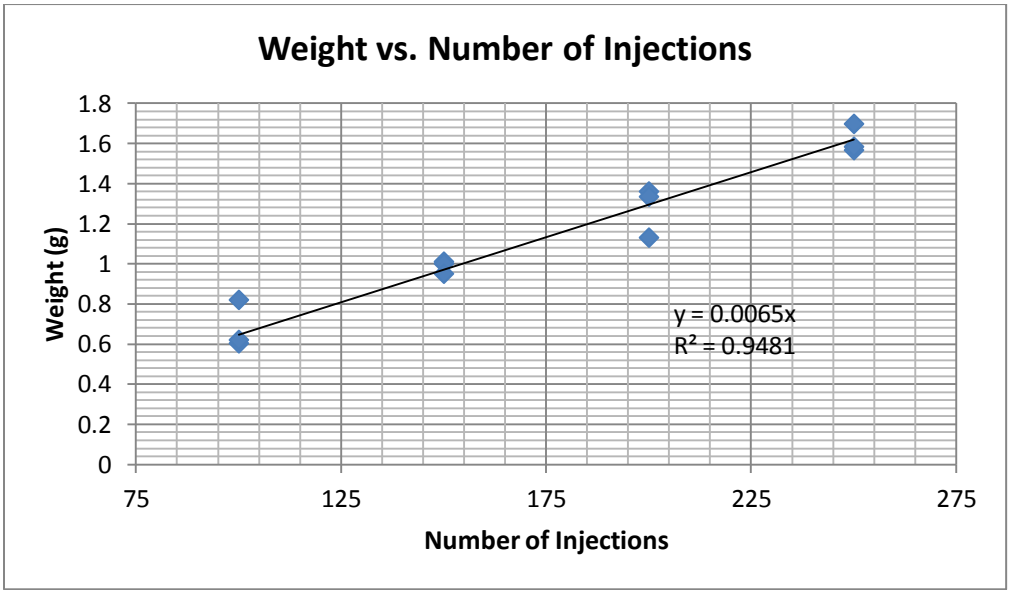

Figure B.18: Corresponding weight for each number of injections.

14. A linear regression fit for the data was perform, which provided a relationship between the number of injections and the associated weight; hence the injector was calibrated for the n-heptane. The r-square value from the graph is 0.9481, which means that approximately 94.81% of the variation in the weight data is due to the number of injections, hence it is fair to conclude that the relationship between both variables are linear.

If outliers (data points that fall outside the linear regression fit) are disregarded and the data are plotted again,Figure B,18 modifies to Figure B.19 illustrated below.

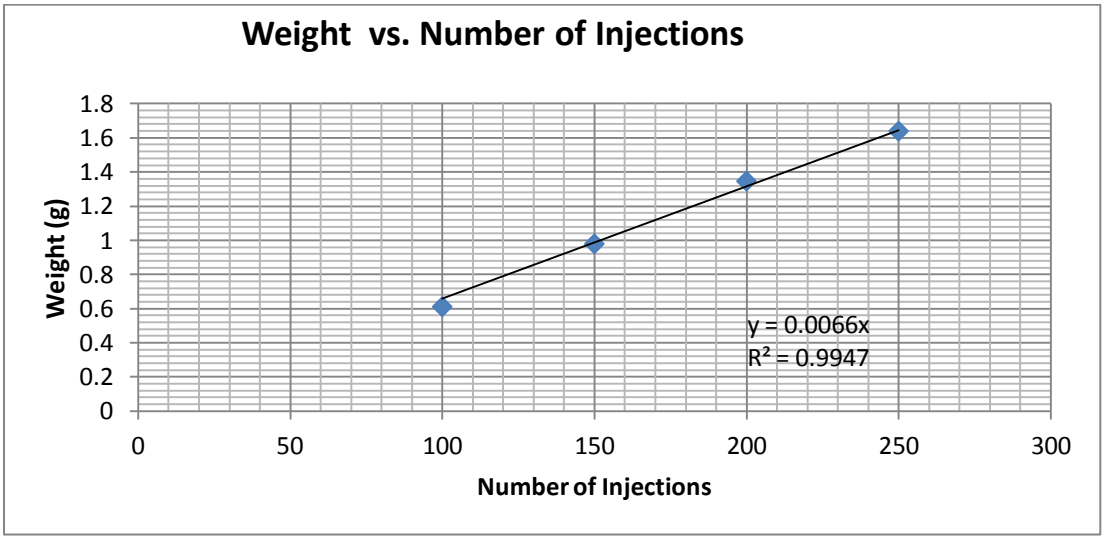

Figure B.19: Modified plot for weight against the number of injections.

The data obtained from the plot in Figure B.19 clearly show that removing the outliers results in 99.47% of the variation of the data that is due to the number of injections. Measured data showed that each injection of *n*-heptane, for the tested conditions, yielded 0.0066 grams (6.6 mg) of fuel.

APPENDIX C. GUI AND M-FILES CODES

C.1 IAGUI CODES

```
function varargout = Image_Analysis_GUI(varargin)

%_____________________________________________________________________

% % Created by : Kemar James
% % From       : The University of Alabama, AL
% % Department : Mechanical Engineering
% %
% % Date Created : January 20, 2013
% % Modifications: June 1, 2013
_____________________________________________
% Discription: The Image Analysis Graphical User Interface (IAGUI) is an
essential tool
% that allows the user to extract information from a cine file. This
information varies
% from simple boundaries' isolation and extraction to acquiring numerical
information such
% as spray penetration length, conical angle, propagation rate, and origin.
% The IAGUI also allows the processing of jpeg images and has several built-
in tools to
%optimize large data processing and extraction

%_____________________________________________________________________


% IMAGE_ANALYSIS_GUI M-file for Image_Analysis_GUI.fig
%      IMAGE_ANALYSIS_GUI, by itself, creates a new IMAGE_ANALYSIS_GUI or
raises the existing
%      singleton*.
%
%      H = IMAGE_ANALYSIS_GUI returns the handle to a new IMAGE_ANALYSIS_GUI
or the handle to
%      the existing singleton*.
%
%      IMAGE_ANALYSIS_GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in IMAGE_ANALYSIS_GUI.M with the given input
arguments.
%
%      IMAGE_ANALYSIS_GUI('Property','Value',...) creates a new
IMAGE_ANALYSIS_GUI or raises the
```

```
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Image_Analysis_GUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Image_Analysis_GUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Image_Analysis_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Image_Analysis_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Image_Analysis_GUI is made visible.
function Image_Analysis_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Image_Analysis_GUI (see VARARGIN)

% Choose default command line output for Image_Analysis_GUI
handles.output = hObject;

%Initializing function tracking variables
handles.func=cell(7,1);
handles.func_order=0;

%Update handles structure
guidata(hObject, handles);

% UIWAIT makes Image_Analysis_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
```

```
function varargout = Image_Analysis_GUI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


 % --- Executes on button press in Flip_Image_pushbutton.
function Flip_Image_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Flip_Image_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  axes(handles.axes2)
  imshow(fliplr(handles.filename.current(:,:,1))) %flip image

  %track function execution
  handles.func_order=1+handles.func_order;
  handles.func{handles.func_order,1}='Flip_Image_pushbutton';


 %Update handles structure
  guidata(hObject,handles);

% --- Executes on button press in crop_pushbutton.
function crop_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to crop_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isempty(handles.filename.current)==1
axes(handles.axes1)
else
[handles.filename.current,handles.xycordinates] = imcrop; %cropping image
axes(handles.axes2)
image(handles.filename.current)
end
% set the rotate textbox to '0'
set(handles.Rotate_edit1,'String', num2str(0));

%track function execution
handles.func_order=1+handles.func_order;
handles.func{handles.func_order,1}='crop_pushbutton';

%Update handles structure
guidata(hObject,handles);

% --- Executes on button press in Reset_Display_pushbutton.
function Reset_Display_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Reset_Display_pushbutton (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.filename.current=[];
axes(handles.axes2)
imagesc(handles.filename.current)
axes(handles.axes1)

% reset handles.func_order to 0
handles.func_order=0;

%Update handles structure
guidata(hObject,handles)

    % --- Executes on button press in RGB_Popupmenu.
function RGB_Popupmenu_Callback(hObject, eventdata, handles)
% hObject    handle to RGB_Popupmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Determine the selected data set.

str = get(hObject, 'String');
val = get(hObject,'Value');

axes(handles.axes2);
%handles.filename.current=handles.filename.original;

switch str{val};

case 'Rgb to Gray'
   handles.filename.current=rgb2gray(handles.filename.current);
   imshow(handles.filename.current);

case 'Rgb to Index'
   [handles.filename.current, handles.map]=rgb2ind(handles.filename.current);
   imshow(handles.filename.current);

case 'Gray to Index'
   [handles.filename.current,handles.map]=gray2ind(handles.filename.current);
   imshow(handles.filename.current);

case 'Index to Gray'
    if isempty(handles.map)==1;
        msg = sprintf('You have not change format to obtain a color map');
     warndlg(msg,'Warning','modal');
    else
     handles.filename_current=ind2gray(handles.filename.current,handles.map);
     imshow(handles.filename.current);
    end
end
%Save the handles structure.

%track function execution
handles.func_order=1+handles.func_order;
```

```
handles.func{handles.func_order,1}='RGB_Popupmenu';
handles.func_rgb_str=str{val};

%Update handles structure
guidata(hObject,handles)


function Rotate_edit1_Callback(hObject, eventdata, handles)
% hObject    handle to Rotate_edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 v1= str2double(get(hObject,'String')); % returns contents of Rotate_edit1 as
double
 if isempty(v1)==1
 else

 % Rotate image
 handles.filename.current=imrotate(handles.filename.current,v1);

 %Display image on axes 2
 axes(handles.axes2)
 image(handles.filename.current)
 end

 %track function execution
 handles.func_order=1+handles.func_order;
 handles.func{handles.func_order,1}='Rotate_edit1';
 handles.func_rotate_v1=v1;

 %Update handles structure
 guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Rotate_edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Rotate_edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Open_Cine_pushbutton7.
function Open_Cine_pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to Open_Cine_pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

choice= questdlg('What file format do you wish to open ?','File
Format','Cine','Cin','Jpeg','Jpeg');
```

```
switch choice

    case 'Cine'
        [filename pathname]=uigetfile({'*.Cine'},'File Selector');
        handles.answer =str2double(inputdlg('How many image frames do you
wish to load ?','1'));
        handles.image_count=handles.answer;
        numimgs=[1:handles.answer];
        L=length(filename)-5;
        handles.filename=filename(1:L);
        [ImageData] = readcin(filename(1:L),numimgs);
        handles.ImageData=[ImageData];
        answer2 = str2double(inputdlg('Which image frame do you wish to view
?','1'));
        set(handles.Image_slider_edit,'string',num2str(answer2));
        set(handles.Image_slider1,'Value',floor(((answer2-1)/(handles.answer-
1))*25));
        set(handles.edit8,'string',handles.filename);


        %imagesc(ImageData.Images.RawImages(:,:,answer2))
        handles.filename.original =
imrotate(ImageData.Images.RawImages(:,:,answer2),90);
        handles.filename.current=
imrotate(ImageData.Images.RawImages(:,:,answer2),90);

        % Improve image with wiener restoration technique and custom 2.8%
intensity filter
        handles.filename.current=wiener2(handles.filename.current,[5 5]);

handles.filename.current(handles.filename.current<(0.028*max(handles.filename
.original(:))))=0;

        axes(handles.axes1);
        imagesc(handles.filename.original);

        axes(handles.axes2);
        imagesc(handles.filename.current);


    case 'Cin'
        [filename pathname]=uigetfile({'*.Cin'},'File Selector');
        handles.answer =str2double(inputdlg('How many image frames do you
wish to load ?','1'));
        numimgs=[1:handles.answer];
        L=length(filename)-5;
        [ImageData] = readcin(filename(1:L),numimgs);
        handles.fielname=filename(1:L);
        handles.ImageData=[ImageData];
        answer2 = str2double(inputdlg('Which image frame do you wish to view
?','1'));
        %imagesc(ImageData.Images.RawImages(:,:,answer2))
        set(handles.edit8,'string',handles.filename);
```

```
        handles.filename.original = ImageData.Images.RawImages(:,:,answer2);
        handles.filename.current= ImageData.Images.RawImages(:,:,answer2);
        axes(handles.axes1);
        imagesc(handles.filename.original);


    case 'Jpeg'
        [filename pathname]=uigetfile({'*.jpg'},'File Selector');
        handles.filename.original=imread(filename);
        handles.filename.current=imread(filename);
        axes(handles.axes1);
        imshow(handles.filename.original);
end

 %Update handles structure
 guidata(hObject, handles);


% --- Executes on button press in Edge_Detection_pushbutton.
function Edge_Detection_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Edge_Detection_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%choice= questdlg('Please select edge detection method ','Edge
Detection','sobel','prewitt','canny','canny');

str2 = get(hObject, 'String');
val2 = get(hObject,'Value');

% Set current data to the selected data set.

switch str2{val2};
    case 'Sobel'
       [handles.filename.current,t1]=edge(handles.filename.current, 'sobel');
% suggested threshold 0.2
       axes(handles.axes2)
       imshow(handles.filename.current)

    case 'Prewitt'
       [handles.filename.current,t1]=edge(handles.filename.current,
'prewitt');% suggested threshold 0.035 ??
       axes(handles.axes2)
       imshow(handles.filename.current)

    case 'Robert'
       [handles.filename.current,t1]=edge(handles.filename.current, 'robert',
0.2);
        axes(handles.axes2)
       imshow(handles.filename.current)

    case 'LOG'
       [handles.filename.current,t1]=edge(handles.filename.current, 'log');
        axes(handles.axes2)
        imshow(handles.filename.current)
```

```
    case 'Canny'
       [handles.filename.current,t1]=edge(handles.filename.current, 'canny',
0.5);
        axes(handles.axes2)
        imshow(handles.filename.current)
end

  %track function execution
  handles.func_order=1+handles.func_order;
  handles.func{handles.func_order,1}='Edge_Detection_pushbutton';
  handles.func_edge_str='str2{val2}';

  %Update handles structure
  guidata(hObject, handles);


function File_Save_As_Callback(hObject, eventdata, handles)
% hObject    handle to File_Save_As (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Acquire filename
handles.image_filename =char(inputdlg('Please enter a filename for the
modified image:','1'));

%Save image
imwrite(handles.filename.current,handles.image_filename,'jpg');


%Update handles structure
guidata(hObject, handles);


% --- Executes on button press in RGB_Gray_Pushbutton.
function RGB_Gray_Pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to RGB_Gray_Pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.filename.current=rgb2gray(handles.filename.current);
axes(handles.axes2)
imshow(handles.filename.current)

%track function execution
handles.func_order=1+handles.func_order;
handles.func{handles.func_order,1}='RGB_Gray_Pushbutton';

%Update handles structure
guidata(hObject, handles);


% --- Executes on button press in Spray_Measurements_Pushbutton.
function Spray_Measurements_Pushbutton_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Spray_Measurements_Pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%[handles.angle,handles.interception,
handles.pen_length]=angleoriginlength(handles.filename.current);
%[Fuel_length,angle,X_origin,Y_origin]=FLAO(handles.filename.current);
axes(handles.axes2);
[Fuel_length,angle,X_origin,Y_origin,Fuel_length2,angle2]=max_fuel_length_ang
le_meas(handles.filename.current);

axes(handles.axes1);
imagesc(handles.filename.original);

handles.angle = angle2;
handles.interception=[Y_origin,X_origin];
handles.pen_length =Fuel_length;

%%Data formating to 2 decimal place
handles.angle=sprintf('%0.2f',handles.angle);
handles.interception2=sprintf('%0.2f',handles.interception(2));
handles.interception1=sprintf('%0.2f',handles.interception(1));
handles.pen_length=sprintf('%0.2f',handles.pen_length);

%Display data in text boxes
set(handles.Spray_Angle_edit,'string',handles.angle);
b1=num2str(handles.interception2);
b2=num2str(handles.interception1);
bs=[b1 ' , ' b2];
set(handles.XY_Origin_Edit,'string',bs);
set(handles.Spray_Length_Edit,'string',handles.pen_length);

%track function execution
handles.func_order=1+handles.func_order;
handles.func{1,handles.func_order}='Spray_Measurements_Pushbutton';


%Update handles structure
guidata(hObject, handles);


function Spray_Angle_edit_Callback(hObject, eventdata, handles)
% hObject    handle to Spray_Angle_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Spray_Angle_edit as text
% str2double(get(hObject,'String')) returns contents of Spray_Angle_edit as a
double


%Update handles structure
guidata(hObject, handles);
```

```
 % --- Executes during object creation, after setting all properties.
function Spray_Angle_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Spray_Angle_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function XY_Origin_Edit_Callback(hObject, eventdata, handles)
% hObject    handle to XY_Origin_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Update handles structure
guidata(hObject, handles);



% Hints: get(hObject,'String') returns contents of XY_Origin_Edit as text
% str2double(get(hObject,'String')) returns contents of XY_Origin_Edit as a
double

 % --- Executes during object creation, after setting all properties.
function XY_Origin_Edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to XY_Origin_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Spray_Length_Edit_Callback(hObject, eventdata, handles)
% hObject    handle to Spray_Length_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Spray_Length_Edit as text
%        str2double(get(hObject,'String')) returns contents of
Spray_Length_Edit as a double

%Update handles structure
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function Spray_Length_Edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Spray_Length_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Apply_All_Creat_Spreedsheet.
function Apply_All_Creat_Spreedsheet_Callback(hObject, eventdata, handles)
% hObject    handle to Apply_All_Creat_Spreedsheet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
val3 = 0;
n=0;

choice= questdlg('Where do you wish to apply all operations ?','Operation
Path','Current Cine/Cin File','Selected Folder','Current Cine/Cin File');

switch choice
    case 'Current Cine/Cin File'
        number_of_images=handles.image_count;
        val3=1;

    case 'Selected Folder'
     h1= warndlg('Please select the first image in this folder you wish to
process','Instruction');
     pause(2.5);
     [Firts_mage_filename pathname]=uigetfile('File Selector');
     [pathstr, name, ext, versn] = fileparts(Firts_mage_filename);
     ext_2=['*',ext];
     h2= warndlg('Please select the last image in this folder you wish to
process','Instruction');
     pause(2.5);
     [Last_image_filename pathname]=uigetfile({ext_2},'File Selector');
     name_2=name(1:5);
     number_of_images =max(size(dir(ext_2)));
     indices=(1:1:number_of_images);

end

for jj=1:number_of_images

    % if cine or cin
    if val3==1
       n=n+1;

%handles.image=imread(sprintf([name_2,['(:,:,','%0.1d',')']],indices(jj)));
```

```
%handles.image=sprintf(handles.ImageData.Images.RawImages(:,:,'%0.1d'),indice
s(jj));
    handles.image=imrotate(handles.ImageData.Images.RawImages(:,:,n),90);

    % if jpeg or other format
    else
    handles.image=imread(sprintf([name_2,['%0.1d',ext]],indices(jj)));
    %handles.image=imread([name_2,'%0.1d.jpg'],indices(jj));
    %handles.image=imread(sprintf('spray%0.1d.jpg',indices(jj)));
    end

for ii=1:handles.func_order;

    if strcmp(handles.func(ii,1),'Edge_Detection_pushbutton');

       switch handles.func_edge_str;
            case 'Sobel'
               [handles.image,handles.image_t2]=edge(handles.image, 'sobel');
% suggested threshold 0.2

            case 'Prewitt'
               [handles.image,handles.image_t2]=edge(handles.image,
'prewitt');% suggested threshold 0.035 ??

            case 'Robert'
               [handles.image,handles.image_t2]=edge(handles.image, 'robert',
0.2);

            case 'LOG'
               [handles.image,handles.image_t2]=edge(handles.image, 'log');

            case 'Canny'
               [handles.image,handles.image_t2]=edge(handles.image, 'canny',
0.5);
       end


    elseif  strcmp(handles.func(ii,1),'Flip_Image_pushbutton');
             handles.image=fliplr(handles.image(:,:,1));

    elseif  strcmp(handles.func(ii,1),'crop_pushbutton');
            handles.image=imcrop(handles.image,handles.xycordinates);

    elseif  strcmp(handles.func(ii,1),'RGB_Popupmenu');
                switch handles.func_rgb_str
                case 'Rgb to Gray'
                   handles.image=rgb2gray(handles.image);

                case 'Rgb to Index'
                   [handles.image, handles.image_map]=rgb2ind(handles.image);

                case 'Gray to Index'
                   [handles.image,handles.image_map]=gray2ind(handles.image);
```

```
                case 'Index to Gray'
                    if isempty(handles.image_map)==1;
                        msg = sprintf('You have not change format to obtain a
color map');
                     warndlg(msg,'Warning','modal');
                    else
                     handles.image=ind2gray(handles.image,handles.image_map);

                    end
                end

          elseif  strcmp(handles.func(ii,1),'Rotate_edit1');

handles.image=imrotate(handles.image,handles.func_rotate_v1,'bilinear','crop'
);
   elseif  strcmp(handles.func(ii,1),'RGB_Gray_Pushbutton');
                   handles.image=rgb2gray(handles.image);

    else strcmp(handles.func(ii,1),'Spray_Measurements_Pushbutton');
         %[handles.angle_2,handles.interception_2,
handles.pen_length_2]=angleoriginlength(handles.image);

[Fuel_length,angle,Fuel_length2,angle2]=max_fuel_length_angle_meas2(handles.i
mage,handles.interception);

         handles.angle_2 = angle2;
         handles.pen_length_2 =Fuel_length;

         %Update handles structure
         guidata(hObject, handles);
    end

   %Update handles structure
   guidata(hObject, handles);
end
% normalized and assign the same image to the 3 different colormap
% handles.image = (handles.image(:)-min(handles.image(:))) /
(max(handles.image(:)-min(handles.image(:))));
  handles.image=handles.image/(max(handles.image(:)));
  Crgb(:,:,1) = handles.image;
  Crgb(:,:,2) = handles.image;
  Crgb(:,:,3) = handles.image;

  %Build an array of images
  handles.m(jj)=im2frame(Crgb);

  % Build an array of data
  handles.angle_spreadsheet(jj,1)=handles.angle_2;
%   handles.interception_x_spreadsheet(jj,1)=handles.interception_2(2);
%   handles.interception_y_spreadsheet(jj,1)=handles.interception_2(1);
  handles.interception_x_spreadsheet(jj,1)=handles.interception(2);
  handles.interception_y_spreadsheet(jj,1)=handles.interception(1);
  handles.pen_length_spreadsheet(jj,1)=handles.pen_length_2;
end
```

```
% write the array of data to spreadsheet
handles.info_array=[handles.angle_spreadsheet,
handles.interception_x_spreadsheet, handles.interception_y_spreadsheet,
handles.pen_length_spreadsheet];

%size(handles.info_array)
%handles.pen_length_spreadsheet(1:7,1)
%handles.pen_length_spreadsheet(7,1)

write2excel(handles.info_array);

%Update handles structure
guidata(hObject, handles);


function Video_Min_Index_edit_Callback(hObject, eventdata, handles)
% hObject    handle to Video_Min_Index_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Video_Min_Index_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Video_Min_Index_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Video_Max_Index_Edit_Callback(hObject, eventdata, handles)
% hObject    handle to Video_Max_Index_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function Video_Max_Index_Edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Video_Max_Index_Edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Create_Video_Pushbutton.
function Create_Video_Pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Create_Video_Pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
handles.video_name =inputdlg('Please enter a filename for the video:','',1);
%video_name = cat(2,video_name{1,1},'.avi')
%class(handles.video_name)

% Add the AVI file-extension to the filename
handles.video_name = cat(2,handles.video_name{1,1},'.avi');

%create AVI movie
movie2avi(handles.m,handles.video_name,'fps',10,'compression','None');

%Update handles structure
guidata(hObject,handles);

% --- Executes on button press in Play_Video_Pushbutton.
function Play_Video_Pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Play_Video_Pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Plays the AVI movie
implay(handles.video_name);

%[r c]=size(A);
%a=cat(1,'cos','cot','sin') input format for "A"
%a=sum(double('a yah suh nice'))

%Update handles structure
guidata(hObject, handles);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function File_Open_Jpeg_Callback(hObject, eventdata, handles)
% hObject    handle to File_Open_Jpeg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Acquire jpeg filename and path
[filename pathname]=uigetfile({'*.jpg'},'File Selector');

%Assign image to variables
handles.filename.original=imread(filename);
handles.filename.current=imread(filename);

%display image on axes 1
axes(handles.axes1);
imshow(handles.filename.original);

%Update handles structure
```

```
guidata(hObject, handles);

% --------------------------------------------------------------------

% --------------------------------------------------------------------
function I_Time_Callback(hObject, eventdata, handles)
% hObject    handle to I_Time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Reggae_Music_Callback(hObject, eventdata, handles)
% hObject    handle to Reggae_Music (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
web('http://www.reggae-vibes.com/');

 % --------------------------------------------------------------------
function Pandora_Callback(hObject, eventdata, handles)
% hObject    handle to Pandora (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
web('www.pandora.com');

% --------------------------------------------------------------------
function Tools_Callback(hObject, eventdata, handles)
% hObject    handle to Tools (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 % --------------------------------------------------------------------
function Manual_Angle_Measurement_Callback(hObject, eventdata, handles)
% hObject    handle to Manual_Angle_Measurement (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 my_angle_measurement_tool(handles.filename.current);

% --------------------------------------------------------------------
function Manual_Lenght_Measurement_Callback(hObject, eventdata, handles)
% hObject    handle to Manual_Lenght_Measurement (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

imtool(handles.filename.current);

 % --------------------------------------------------------------------
function Fast_Fourier_Transform_Callback(hObject, eventdata, handles)
% hObject    handle to Fast_Fourier_Transform (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure,imshow(fft2(handles.filename.current));


% --------------------------------------------------------------------
function BMT_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to BMT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%Another method for measuring spray angle using boundary trace and
%%polynomial fit

BW=im2bw(handles.filename.original);
[B,L,N,A] = bwboundaries(BW);

%extracting extreme pixel location
L2=max(B{1}(:,2));
L1=min(B{1}(:,2));

%Spray Length measurement
Fuel_length_bw= L2-L1;

%%region of interest for line fit (lower region)
%figure, imshow(I),plot(B{1}(1:250,2),B{1}(1:250,1), 'b','LineWidth', 1);
ab1=polyfit(B{1}(1:250,2),B{1}(1:250,1), 1);

% region of interst for line fit (upper region)
%figure,imshow(I),plot(B{1}(650:1000,2),B{1}(650:1000,1),'b','LineWidth',1);
ab2=polyfit(B{1}(650:1000,2),B{1}(650:1000,1),1);

% Find the Angle of interception
vect1 = [1 ab1(1)]; % create a vector based on the line equation
vect2 = [1 ab2(1)];
dp = dot(vect1, vect2);

% compute vector lengths
length1 = sqrt(sum(vect1.^2));
length2 = sqrt(sum(vect2.^2));

% obtain the larger angle of intersection in degrees
angle = acos(dp/(length1*length2))*180/pi;
intersection = [1 ,-ab1(1); 1, -ab2(1)] \ [ab1(2); ab2(2)];

%Spray Length measurement modified
FL= Fuel_length_bw -intersection(2);

%%Plotting the trace of the entire boundary
hold on
figure, imshow(BW),plot(B{1}(:,2),B{1}(:,1), 'b','LineWidth', 1);

% Display the angle, intersection and length in the title of the figure.
title(sprintf('(%0.2f) degrees   (%0.2f,%0.2f) Intersection   (%0.2f) Length
',angle,intersection(2),intersection(1),FL));


hold off

 % --------------------------------------------------------------------
function Line_Intensity_Plot_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Line_Intensity_Plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes2);
[cx,cy,c] = improfile;
figure,plot3(cx,cy,c),grid on;


% --- Executes on slider movement.
function Image_slider1_Callback(hObject, eventdata, handles)
% hObject    handle to Image_slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

v1=get(hObject,'Value');
v11=floor(v1*((handles.answer-1)/25)+1);
set(handles.Image_slider_edit,'string',num2str(v11));

        %imagesc(ImageData.Images.RawImages(:,:,answer2))
        handles.filename.original =
imrotate(handles.ImageData.Images.RawImages(:,:,v11),90);
        handles.filename.current=
imrotate(handles.ImageData.Images.RawImages(:,:,v11),90);

        % Improve image with wiener restoration technique and custom 2.8%
intensity filter
        handles.filename.current=wiener2(handles.filename.current,[5 5]);

handles.filename.current(handles.filename.current<(0.028*max(handles.filename
.original(:))))=0;

        axes(handles.axes1)
        image(handles.filename.original)
        axes(handles.axes2)
        image(handles.filename.current)

guidata(hObject, handles);

  % --- Executes during object creation, after setting all properties.
function Image_slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Image_slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function Image_slider_edit_Callback(hObject, eventdata, handles)
% hObject    handle to Image_slider_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Image_slider_edit as text
```

```
%        str2double(get(hObject,'String')) returns contents of
Image_slider_edit as a double
 v2=str2double(get(hObject,'String'));
 v22=floor(((v2-1)/(handles.answer-1))*25);
  if v2>=0 && v2<1;
     msg = sprintf('Max Height must be between 1 and
%s',num2str(handles.answer));
     warndlg(msg,'Warning','modal');
 else
set(handles.Image_slider1,'Value',v22);  %str2double(get(hObject,'String'))
returns contents of edit_slider_text as a double

        %imagesc(ImageData.Images.RawImages(:,:,answer2))
        handles.filename.original =
imrotate(handles.ImageData.Images.RawImages(:,:,v2),90);
        handles.filename.current=
imrotate(handles.ImageData.Images.RawImages(:,:,v2),90);

        % Improve image with wiener restoration technique and custom 2.8%
intensity filter
        handles.filename.current=wiener2(handles.filename.current,[5 5]);

handles.filename.current(handles.filename.current<(0.028*max(handles.filename
.original(:))))=0;

        axes(handles.axes1)
        imagesc(handles.filename.original)
        axes(handles.axes2)
        imagesc(handles.filename.current)
 end
 guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Image_slider_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Image_slider_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

C.2 GUI SUBFUCTION M-FILES

Finding the origin, angle and spray length function:

```
function
[Fuel_length,angle,Fuel_length2,angle2]=max_fuel_length_angle_meas2(I,YX)

ang_T=0;
ang_b=0;

%convert image to black and white
I=im2bw(I);
[row, col]=find(I);
max_col=max(col); % find the max colume with a nonzero pixel value
max_point=max_col/2;

%Tracing the boundary of the spray nd finding the max point
% [B,L,N,A] = bwboundaries(I);
% L2=max(B{1}(:,2));
% L1=min(B{1}(:,2));
% max_point=((L2-L1)/1.8);
L1=10;

%Illustrate the image and allow the user to select the spray origin
X_origin =floor(YX(2));
Y_origin= floor(YX(1));

if (L1-X_origin)< 15;

%Calculated multiple spray angles form a penetration length of 10 pixel about
50% of the spray length in each frame
for i=(10+ X_origin):5:max_point;
    n=0;
          [YT,XT] = find(I(:,i), 1, 'first');
          [YB,XB] = find(I(:,i), 1, 'last');
```

```
            ang_T(i)=180/pi*atan( (YT - Y_origin ) / ( XT - X_origin ) );
            ang_b(i)=180/pi*atan( (YB - Y_origin ) / ( XB - X_origin ) );

            Y1(i-(9+n*5))=YT;
            X1(i-(9+n*5))=XT;
            Y2(i-(9+n*5))=YB;
            X2(i-(9+n*5))=XB;

    n=n+1;
end

% Gives the average spray angle and the spray length
avgang=mean(ang_T(:));
avgang2=mean(ang_b(:));
angle=(avgang-avgang2)/4;
Fuel_length=max_col - X_origin;

% Fits two linear regression line to the data
ab1 = polyfit(X1,Y1,1);
ab2 = polyfit(X2,Y2,1);

% Create a vector based on the line equation
vect1 = [1 ab1(1)];
vect2 = [1 ab2(1)];
dp = dot(vect1, vect2);
% compute vector lengths
length1 = sqrt(sum(vect1.^2));
length2 = sqrt(sum(vect2.^2));

% obtain the larger angle of intersection in degrees
angle2 = cos(dp/(length1*length2))*45/pi;

intersection = [1 ,-ab1(1); 1, -ab2(1)] \ [ab1(2); ab2(2)];
% apply offsets in order to compute the location in the original,
% i.e. not cropped, image.
intersection = intersection;
inter_x = intersection(2);
inter_y = intersection(1);
Fuel_length2= max_col - inter_x;

else
% The spray length
Fuel_length=max_col - X_origin;
angle=0;
Fuel_length2=0;
angle2 =0;

end
end
```

Creating an excel files for the data extracted function

```
function write2excel(A)

[a b]=size(A);
if isempty(A)==1 || b~=4;
    msg = sprintf('Please enter a Nx4 array');
     warndlg(msg,'Warning','modal');
else

filename=inputdlg('Please enter the filename','',1);
filename=(filename{1,1});
d1 = {'Spray Angle','X-Origine','Y-Origine','Penetration Distance'};
d2=num2cell(A);
D=cat(1,d1,d2);

sheet = 1;
xlRange = 'B1';
xlswrite(filename,D,sheet,xlRange);
end
```