2015

The Effect Of Biodiesel Blends On Particle Number Emissions From A Light Duty Diesel Engine

Tyler Samuel Feralio
University of Vermont

THE EFFECT OF BIODIESEL BLENDS ON PARTICLE NUMBER EMISSIONS FROM A LIGHT DUTY DIESEL ENGINE

A Dissertation Presented

by

Tyler Samuel Feralio

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Civil and Environmental Engineering

October, 2015

Defense Date: June 8, 2015
Dissertation Examination Committee:

Britt A. Holmén, Ph.D., Advisor
Robert G. Jenkins, Ph.D., Chairperson
Donna M. Rizzo, Ph.D.
Ruth M. Mickey, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

ABSTRACT

Numerous studies have shown that respirable particles contribute to adverse human health outcomes including discomfort in irritated airways, increased asthma attacks, irregular heartbeat, non-fatal heart attacks, and even death. Particle emissions from diesel vehicles are a major source of airborne particles in urban areas. In response to energy security and global climate regulations, the use of biodiesel as an alternative fuel for petrodiesel has significantly increased in recent years. Particle emissions from diesel engines are highly dependent on fuel composition and, as such, the increased use of biodiesel in diesel vehicles may potentially change the concentration, size, and composition of particles in respirable air. One indicator used to evaluate the potential health risk of these particles to humans is particle diameter (D_p). Ultrafine particles (UFPs, D_p<100nm) are of health concern because their increased mobility relative to larger particles allows penetration into the alveolar region of the human lung where they may subsequently pass directly into the cardiovascular system.

Current research in automotive emissions primarily focuses on particle emissions measured on a total particle mass (PM) basis from heavy-duty diesel vehicles. The nation's light-duty diesel fleet is, however, increasing; and because the mass of a UFP is much less than that of larger particles, the total PM metric is not sufficient for characterization of UFP emissions. As such, this research focuses on light-duty diesel engine transient UFP emissions, measured by particle number (PN), from petrodiesel, biodiesel, and blends thereof. The research objectives were to determine: 1) the difference in UFP emissions between petrodiesel and blends of waste vegetable oil-based biodiesel (WVO), 2) the differences between UFP emissions from blends of WVO and soybean oil-based biodiesel (SOY), and 3) the feasibility of using genetic programming (GP) to select the primary engine operating parameters needed to predict UFP emissions from different blends of biodiesel.

The results of this research are significant in that: 1) Total UFP number emission rates (ERs) exhibited a non-monotonic increasing trend relative to biodiesel content of the fuel for both WVO and SOY that is contrary to the majority of prior studies and suggests that certain intermediate biodiesel bends may produce lower UFP emissions than lower and higher blends, 2) The data collected corroborate reports in the literature that fuel consumption of diesel engines equipped with pump-line-nozzle fuel injection systems can increase with biodiesel content of the fuel without operational changes, 3) WVO biodiesel blends reduced the overall mean diameter of the particle distribution relative to petrodiesel more so than SOY biodiesel blends, and 4) Feature selection using genetic programming (GP) suggests that the primary model inputs needed to predict total UFP emissions are exhaust manifold temperature, intake manifold air temperature, mass air flow, and the percentage of biodiesel in the fuel; These are different than inputs typically used for emissions modeling such as engine speed, throttle position, and torque suggesting that UFP emissions modeling could be improved by using other commonly measured engine operating parameters.

ACKNOWLEDGEMENTS

This research would not have been possible without my advisor, Dr. Britt Holmén and the help of Jim Dunshee, Karen Sentoff, John Kasumba, Dave Wheeler, Daryl Deprey, Tom Davidson, John Nummy, and Brad Haire – all current or former members of the Holmén Group. Additionally, I would like to thank my defense committee, Dr. Robert Jenkins, Dr. Donna Rizzo, and Dr. Ruth Mickey for their support through this endeavor.

Thanks also to Dr. Richard Parnas and Iman Noshadi from The University of Connecticut for processing the biodiesel used for this research.

I'd also like to thank Earnie Cook, from Moeller Marine Products, who supplied fuel system components necessary for testing multiple blends of biodiesel to this project.

Funding for this research came from the U.S. Department of Transportation through the University Transportation Centers Program at the University of Vermont Transportation Research Center.

Finally, I'd like to thank my son, Tristan Feralio, who was three months old when I started, for putting up with all the hardships associated with having a parent getting their PhD.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Motivation

Particle emissions from combustion processes are a major source of airborne particles in urban areas [1,2]. Numerous studies have shown that airborne particulate matter contributes to adverse human and environmental health outcomes around the world [3–5]. Exposure to high levels of airborne particles can lead to a number of respiratory and cardiovascular problems including discomfort in irritated airways, increased asthma attacks, irregular heartbeat, non-fatal heart attacks, and even death [6].

Particle diameter (D_p) is an indicator used to evaluate the potential health risk of particles to humans. Particle mobility increases as D_p decreases, increasing the potential for deposition deeper within the human respiratory system. Ultrafine particles (UFP) (D_p<100nm) are of particular concern because of their increased mobility relative to larger particles. This allows them to penetrate into the alveolar region of the human lung, the interface between the respiratory and cardiovascular systems, where they may subsequently pass directly into the cardiovascular system [3,6].

The transportation sector currently runs on the process of combustion. As such, it is a major sink for petroleum-based fuels and a leading contributor to particle emissions in urban areas (>65% of particle emissions in some cities [2]) that effect both health and climate. Due to the political consequences associated with importing petroleum, legislation mandating the use of renewable fuels that can be produced domestically has come to fruition [7]. While this push has led to a surge in the use of new fuels to power

the fleet, relatively little is known about how the transition from conventional fuels to alternative fuels will affect emissions profiles and ambient air quality.

1.2. Biodiesel

Biodiesel is a drop-in renewable fuel for diesel engines that is increasingly used as an alternative to petroleum-based diesel fuel, hereinafter denoted as petrodiesel. In 2013, the U.S. biodiesel consumption was approximately 137 times that of 2001 (Figure 1.1) [8].

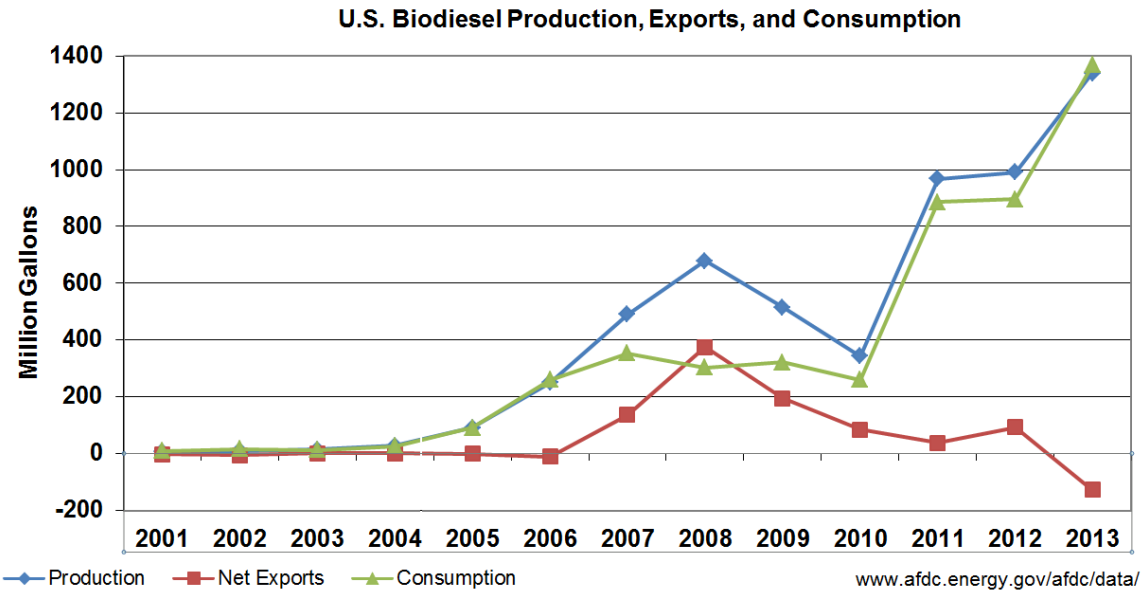


Figure 1.1: U.S. Biodiesel Production, Consumption, and Export. Generated from data in Table 10.4 of the EIA May 2015 Monthly Energy Review [8]

Biodiesel can be processed from a number of natural lipids such as plant oils, animal fats, or combinations thereof. The process used to generate biodiesel from a lipid is transesterification. This process entails a chemical reaction between the lipid and an alcohol in the presence of a base catalyst. Biodiesel currently being processed commercially in the U.S. is typically a mixture of fatty acid methyl esters (FAMEs)

meaning that the alcohol used in the transesterification process was methanol (CH_4O) [9]. Biodiesel can also be processed with ethanol (C_2H_6O), in which case it would be considered a mixture of fatty acid ethyl ester (FAEE). Typical catalysts used for the transesterification process are sodium hydroxide (NaOH) or potassium hydroxide (KOH). The regulations controlling the quality of neat biodiesel are EN14214 in Europe and ASTM D6751 in the U.S. [9]. A comparison between typical number 2 ultra-low sulfur diesel and pure biodiesel fuel properties are provided in Table 1.1.

Table 1.1: Select Properties of Typical No. 2 Ultra-Low Sulfur Diesel and Biodiesel Fuels [9]

Fuel Property	Diesel	Biodiesel
Fuel Standard	ASTM D975	ASTM D6751
Higher Heating Value, Btu/gal	~137,640	~127,042
Lower Heating Value, Btu/gal	~129,050	~118,170
Kinematic Viscosity, @ 40°C (104°F)	1.3–4.1	4.0–6.0
Specific Gravity kg/l @ 15.5°C (60°F)	0.85	0.88
Density, lb/gal @ 15.5°C (60°F)	7.1	7.3
Carbon, wt %	87	77
Hydrogen, wt %	13	12
Oxygen, by dif. wt %	0	11
Sulfur, wt %	0.0015 max	0.0–0.0024
Boiling Point, °C (°F)	180–340 (356–644)	315–350 (599–662)
Flash Point, °C (°F)	60–80 (140–176)	100–170 (212–338)
Cloud Point, °C (°F)	-35 to 5 (-31 to 41)	-3 to 15 (26 to 59)
Pour Point, °C (°F)	-35 to -15 (-31 to 5)	-5 to 10 (23 to 50)
Cetane Number	40–55	48–65

The differences in fuel properties between petrodiesel and biodiesel affect engine emissions through changes in both the fuel injection and combustion processes. As a

result, UFP emissions from a diesel engine fueled by petrodiesel are different than that fueled by biodiesel.

1.3. Research Objectives

The research discussed here set out with three objectives. Chapter 2 addresses the first research objective – determine the difference in UFP emissions from petrodiesel and blends of waste vegetable oil-based biodiesel. This Chapter will be submitted to SAE International Journal of Fuels and Lubricants for publication in the coming months.

Chapter 3 addresses the second research objective – determine the differences between UFP emissions from blends of waste vegetable oil-based biodiesel and soybean oil-based biodiesel. This chapter will also be submitted to SAE International Journal of Fuels and Lubricants for publication in the coming months.

Chapter 4 addresses the third research objective – determine the feasibility of using genetic programming to select engine operating parameters that are primary indicators of UFP emissions for emissions modeling. This chapter will be developed further and submitted for publication in the SAE International Journal of Engines.

CHAPTER 2: VARYING EFFECT OF WASTE VEGETABLE OIL-BASED BIODIESEL BLENDS ON TOTAL ULTRAFINE PARTICLE EMISSIONS FROM A DIESEL ENGINE

2.1. Abstract

To determine the effect of biodiesel fuel blends on engine-out particle emissions, a naturally aspirated, diesel engine with a pump-line-nozzle fuel injection system was fueled with neat ultra-low sulfur on-road diesel (B0), neat waste vegetable oil-based biodiesel (B100), and B10, B20, and B50 blends thereof (where XX in BXX refers to the percentage of biodiesel v/v in the blend). Particle number concentrations (#/cm^3) were collected at 1Hz with a TSI 3090 Engine Exhaust Particle Sizer (EEPS; 32 channels from 5.6-560nm) while the engine followed a transient drive cycle developed from on-road vehicle operation. Total ultrafine particle (TUFP; Dp<100nm) number emission rates (ERs; #/sec) exhibited a non-monotonic increasing trend relative to biodiesel content of the fuel that is contrary to the majority of prior studies. The ratios of transient TUFP ERs from B10, B20, B50, and B100 relative to B0 were 2.2, 0.9, 2.0, and 3.2, respectively. Additionally, although there were no statistically significant differences in throttle position, engine speed, and torque from test to test, fuel consumption increased with the percentage of biodiesel in the fuel. Other factors that may have contributed to the non-monotonic trend observed in TUFP ERs but that weren't measured here include: 1) advanced start of combustion (SOC) due to increased oxygen content and cetane number of biodiesel relative to petrodiesel, 2) a possible reduction of premixed combustion relative to diffusion combustion due to advanced SOC, and 3) an increase in particle

nucleation relative to condensation and adsorption, due to the increased oxygen content and decreased volatility of biodiesel relative to petrodiesel.

2.2. Introduction

2.2.1. Motivation

Particle emissions from combustion processes, specifically those from diesel on-road vehicles, are a major source of airborne particles in urban areas [1,2]. Numerous studies have shown that airborne particulate matter contributes to adverse human and environmental health outcomes worldwide [3–5]. Exposure to high levels of airborne particles can lead to a number of respiratory and cardiovascular problems including discomfort in irritated airways, increased asthma attacks, irregular heartbeat, non-fatal heart attacks, and even death [6].

2.2.2. Particle Emissions - Background

One indicator used to evaluate the potential health risk of particles to humans is particle diameter (D_p). As D_p decreases, particle mobility increases, increasing the potential for deposition deeper within the human respiratory system. Diesel engine exhaust particle size distributions (PSD) are defined by three distinct modes; the nuclei, accumulation, and coarse modes [10]. The D_p ranges of these modes are typically $5<D_p<50$nm, $50<D_p<1000$nm, and $D_p>1000$nm, respectively [10]. The smaller a particle is, the more mobile it is giving it the ability to bypass a humans natural defenses and deposit deep within the lung. Kittelson et al. [10] showed that ultrafine particles (UFP) ($D_p<100$nm) have the highest potential to deposit within the alveolar region of the lung where they may subsequently pass directly into the cardiovascular system [3,6].

Emissions from light-duty automobiles in the U.S. are currently regulated by the Tier 2 emissions standards, which limit tailpipe particulate emissions on a total particle mass (PM) per distance basis (g/mi) [11]. Measuring total particle mass, however, does not adequately characterize UFP emissions because the mass of a UFP is essentially negligible relative to larger particles. A more effective measurement used to characterize UFP emissions is particle number (PN) concentration, the number of particles per volume of air which is why PN emission regulations have been introduced for automotive emissions in the E.U. [10,12–15].

Particle emissions from diesel vehicles are highly dependent on fuel composition [16,17], which, has been continually evolving on a national scale in response to environmental and energy security regulations [7,18]. Between 2006 and 2010, on-road diesel fuel transitioned from low sulfur diesel (LSD; S content ≤ 500ppm) to ultra-low sulfur diesel (ULSD; S content ≤ 15ppm) [18,19] and was shown to reduce PM emissions by approximately 23% by reducing the concentration of particle across the entire PND range with the largest reductions below 30nm (within the UFP range) [20]. Concurrently, interest in energy independence and security led to legislation which mandates domestic use of renewable fuels [7]. Biodiesel is currently the primary renewable fuel used as a ‘drop-in’ alternative for petrodiesel [9]. Between 2012 and 2013 there was a 28% increase in the required production of biomass-based diesel (primarily biodiesel) in the U.S., adding it to the fuel supply available for use by the nation’s fleet [21]. Biodiesel can be processed from a variety of lipid feedstocks, such as plant oils, animal fats, or a combination thereof including recycled waste oils, resulting in variability in composition

within the biodiesel supply itself [9]. Furthermore, although it is possible to use neat biodiesel as an alternative for petrodiesel, it is typically blended with petrodiesel for use at levels ≤B20 (20% biodiesel; 80% petrodiesel v/v) because, among other reasons, higher blends of biodiesel can void vehicle warranties, can act as a solvent causing the fuel system of vehicles that primarily run on petrodiesel to clog, and can gel at warmer temperatures [9].

In concert with an evolving fuel supply, the diesel vehicle fleet has continually been changing in response to tightening tailpipe emission regulations and increased fuel costs. In 2007, after the adoption of ULSD and the EPA's introduction of more stringent exhaust emissions standards for diesel engines [11], the so called 'green diesel' vehicles became available, many of which were light-duty diesel (LDD) vehicles. These vehicles boasted not only superior fuel economy relative to their gasoline-powered counterparts, but much cleaner tailpipe emissions relative to their predecessors. The advances in tailpipe emissions were due, in part, to the utilization of ULSD which enable diesel particulate filter (PDF) control of particulate emissions, and selective catalyst reduction (SCR) to control NO_x emissions, among other technological advances [19,22,23]. The number of diesel passenger car and sport utility vehicle models available in the U.S. has increased from 3 to 22 between the years 2000 and 2014 [24]. This, and the fact that U.S. registration for these vehicles rose by 24% between 2010 and 2012 [25], indicates an increased demand among U.S. consumers for LDD vehicles.

2.2.3. Biodiesel and Particle Emissions

Although biodiesel emissions relative to petrodiesel emissions have been studied, the results are somewhat ambiguous. The overall trend shows a reduction in PM emissions when running biodiesel compared to petrodiesel [9,17,20,26–33]; however, some studies report the opposite [16,34,35]. Of the relatively fewer studies that report particle emissions on a PN basis, most report that the use of biodiesel increases nuclei mode particles (5 - 50nm) and decreases accumulation mode particles (50 – 1000nm) [16,17,27,28,36–38], although some studies found otherwise [35,39]. Additionally, studies have shown that total PN (TPN) can either increase [16,40] or decrease [27,30,32,39,40] with the use of biodiesel.

There are two main mechanisms through which biodiesel fuel affects engine emissions: 1) hydraulic – the differences in the way biodiesel behaves as a fluid relative to petrodiesel, and 2) chemical – the differences in the way biodiesel oxidizes (combusts) relative to petrodiesel. Hydraulically, biodiesel has a higher viscosity, density, and bulk modulus [9,41,42] than petrodiesel. These properties can all affect the performance of the fuel delivery system. For pump-line-nozzle type fuel injection systems, the injector pump is lubricated by the fuel. As such, tolerances between internal components of the pump allow some ‘leakage’ of the fuel for adequate lubrication (Figure A23). Because the viscosity of biodiesel (4.0-6.0 cSt) is higher than that of petrodiesel (1.3-4.1 cSt) [9] , less biodiesel fuel ‘leaks’ through the components of the injector pump resulting in a sharper, higher peak pressure at the injector pump outlet [27]. The speed of the pressure pulse in the fuel line between the injector pump and the fuel injector is also increased because biodiesel has a higher bulk modulus [30,41,42]. These two factors result in the

mechanical fuel injector ‘seeing’ a higher pressure sooner from biodiesel as compared to petrodiesel and can result in different fuel injection characteristics. First, because less fuel ‘leaks’ through the injection pump, more biodiesel fuel is available for injection into the combustion chamber. Second, because the pressure pulse reaches the injector faster, it is possible for the biodiesel start of injection (SOI) to be advanced relative to top dead center (TDC) of the piston [27]. Third, the injection duration and/or injection rate over the injection event can be different between the two fuels because of the difference in shape of the pressure pulse between the two fuels. Additionally, due to the differences in fuel density and viscosity, the fuel sprayed into the combustion chamber by the fuel injector may atomize differently, possibly changing the distribution of fuel into the combustion chamber and subsequently altering local stoichiometric conditions [43]. These effects are typically reported to increase with respect to blend level, showing the largest difference relative to petrodiesel when neat biodiesel is used. Because more recent common rail fuel injection systems (MY≥2008 for VW North America) utilize a fuel rail at constant pressure and electronically controlled fuel injectors, tailpipe emissions from more modern engines are less susceptible to differences in the hydraulic properties of the fuel. Additionally, some of the newest control technologies utilize in-cylinder pressure sensors to provide the engine control unit (ECU) with the feedback necessary to directly adjust SOC in an effort to more accurately control exhaust emissions [44].

Chemically, neat biodiesel is an oxygenated fuel containing ~11% oxygen by mass while the oxygen content of neat petrodiesel is negligible [9]. Because most engines are typically not modified to run on blends of biodiesel, there is an increase in excess

oxygen (more fuel-lean) in the combustion chamber when running on biodiesel compared to petrodiesel given the same amount of fuel injected into the combustion chamber. It has been shown that, in many cases, this leads to faster combustion rates and more complete combustion, especially in fuel-rich zones of the combustion chamber where fuel-borne oxygen is available [27]. Typically, biodiesel also has a higher cetane number than petrodiesel which can result in a reduction of ignition delay, the finite amount of time between start of injection (SOI) and start of combustion (SOC) [27]. Changes in ignition delay can affect pre-ignition charge mixing and, therefore, the ratio of premixed combustion to diffusion combustion which affects both particle and NOx emissions [27,45]. Additionally, with a boiling point between 599 and 662°C, the volatility of biodiesel can be less than that of petrodiesel (boiling point between 356 and 644°C) [9] making biodiesel atomization in the cylinder more difficult [43]. The unburned hydrocarbons in biodiesel exhaust may also have a lower volatility than those of petrodiesel exhaust causing them to more readily condense into the liquid phase, and increase particle emissions from biodiesel [46]. Biodiesel also has a lower heating value than petrodiesel which typically results in the need for more fuel to be consumed in order to generate an equivalent amount of power as from petrodiesel [9].

That being said, variables other than fuel composition can affect engine emissions such as: 1) engine technology (light-duty vs. heavy-duty; new technology vs. old) [23], 2) drive cycle [16,30] (steady-state vs. transient, degree of transient nature), and 3) dilution conditions [47,48] (dilution temperatures and residence times). Because these variables are not consistent between studies, reported results are often contradictory. Comparison

of biodiesel emissions studies must, therefore, take into account engine technology (specifically the type of fuel injection system employed), the properties of the baseline petrodiesel, the feedstock oil used to process the biodiesel, the biodiesel/ petrodiesel blend, and the dilution conditions used for the study because each of these factors has the potential to affect the particle number distribution (PND – number weighted PSD) and particle composition to some degree.

2.2.4. Study Objectives

The majority of prior studies on biodiesel vehicle/engine particle emissions examined heavy-duty diesel (HDD) engines, reported only PM emissions, and were executed with baseline petrodiesel fuels that had a higher sulfur content than ULSD. It is, however, important to recognize that the 1) LDD fleet is expanding [24,25], 2) UFP emissions measured on a PN basis are more relevant to human health than total PM emissions, and 3) transition to ULSD fuel reduced PN emissions considerably and, therefore, may have altered the way in which blending biodiesel with petrodiesel affects particle emissions. The objective of this study was to quantify the changes in transient engine-out UFP emissions from a LDD engine running on multiple blends of waste vegetable oil-based biodiesel (WVO) and ULSD. As such, this research was conducted with a transient drive cycle that simulated light-duty vehicle operation in an urban setting, where particle emissions are of particular concern.

2.3. Methodology

2.3.1. Engine and Fuel Specifications

The apparatus used to collect particle emissions consisted of a naturally aspirated, four cylinder Volkswagen 1.9L SDi engine with a pump-line-nozzle fuel injection system coupled to an Industrias Zelu, S.L. K-40 power absorber unit (eddy current dynamometer) (Table 2.1). Sold for industrial use, the engine conforms to emission certification EC 97/68 Stage IIIA; however, mechanically, the engine is similar to those in EURO II Volkswagen LDD automobiles. The engine was not equipped with an exhaust gas recirculation system or any exhaust aftertreatment devices – the emissions data reported are engine-out.

Table 2.1: Engine and Dynamometer Specifications

Engine	
Manufacturer:	Volkswagen
Identification Code:	ARD
Charge Air:	Naturally Aspirated
Capacity:	1896cm^3
Cylinders:	4
Bore:	79.5mm
Stroke:	95.5mm
Compression Ratio:	19.5:1
Nominal Output:	44 kW @ 3600 RPM
Max Torque:	130Nm @ 2000 - 2400 RPM
Minimum CN:	49
Control System:	Bosch EDC
Fuel Injection:	Bosch VE injection pump
EGR:	None
Power Absorption Unit/ Eddy Current Dynamometer	
Manufacturer:	Zelu/ Klam
Model Number:	K-40 PAU
Max Power:	60kW
Max Torque:	145Nm

The fuels used for this study were one lot of B0 (neat ULSD, Trono Fuels, Burlington, VT) and one lot of B100 (neat biodiesel, University of Connecticut BioFuel Consortium processed from waste vegetable oil using the methods documented in Pomykala et al. and Boucher et al. [49,50]), and B10, B20, and B50 blends thereof. B10 and B20 were selected because they are within the range of biodiesel blends typically sold for on-road use. B50 and B100 were also tested to provide data across the range as it is possible to use blends up to B100. The neat biodiesel was treated with an antioxidant (Chemtura Naugalube® 403; see 'Antioxidant Data Sheet' section in the Appendix for more detail) at 2000ppm (w/w). Testing performed by the University of Connecticut BioFuels Center for Environmental Sciences and Engineering confirmed that the B100 conformed to ASTM-6751-11b except for cold soak filtration and combined sodium and potassium (Table A1).

To ensure accuracy in blending, the density of each parent fuel (B0 and B100) was measured both physically and with a density meter. The masses associated with the correct volume of B0 and B100 needed for blending were calculated and subsequently measured using a laboratory scale. The B00 and B100 were then combined in a tank, mechanically mixed, and finally sealed in fuel containers (UN certified 5 gallon buckets from Letica Corp. with unvented lids) with nitrogen headspace to minimize fuel oxidation during storage. The fuel was stored in an environmental chamber at 13°C to simulate underground storage.

Blend ratios (vol % biodiesel) were confirmed using an IROX Diesel (IROX-D) Analyzer from Grabner Instruments (Vienna, Austria), a mid-FTIR analyzer dedicated to

diesel analysis. The IROX-D also measured fuel density with a built-in high accuracy density meter [51] (Table A2). The IROX-D analyzer is, however, only capable of measuring biodiesel blends only in the range of B0 – B40, therefore direct measurements of fuel blends based on FTIR methods were only accurate for the B0, B10, and B20 blends tested here. To verify the blend ratio of the B50 used, the 'as blended' sample was diluted with hexane. The resulting IROX-D BXX measurement was then used along with the known dilution ratio to back calculate the 'as blended' BXX value. The IROX-D results for all 'as blended' WVO samples (B0, B10, and B20) were within 0.2% of the expected value. The back calculated BXX value for the WVO B50 was within 0.8% of the expected value (Table A2). Because the IROX-D measures density with a density meter and not through FTIR, the density measurements are valid for all fuel blends.

2.3.2. Drive Cycle

To simulate real-world urban driving, a transient drive cycle was developed with OBD-II engine speed and throttle position data collected from a 2003 Volkswagen TDi Jetta sedan (ALH engine code) with an automatic transmission as it drove a predefined route through downtown Burlington, VT [52]. The TDi engine in this on-road vehicle is essentially a turbocharged version of the SDi test engine.

The decision to develop a new drive cycle was made for multiple reasons. First, all federally mandated engine dynamometer tests were designed for HDD engines where the prescribed parameters are typically % of rated revolutions per minute (RPM) and % of rated torque. These tests, however, are not equivalent to typical light-duty vehicle drive cycles performed on chassis dynamometers using vehicle speed-time traces. On-

road studies have shown that the majority of real-world driving occurs at less than ~40% load for a modern passenger car [52]. Because legislated *engine* dynamometer test cycles were designed for HDD engines, they simulate the operation of HDD vehicles such as city and municipal vehicles that operate much differently and regularly exceed 40% load. Second, between 1982 and 2004 the power output of the average passenger car has increased by ~4hp/year [22,53], which means the % torque required to power the vehicle on the road under normal driving conditions would decrease with newer model year. Testing based on % torque, therefore, would not be comparable between engines of different model years. Generation of a new drive cycle using on-road data from a modern vehicle was necessary to ensure a realistic loading profile for the engine being tested.

The developed drive cycle (Figure 2.1) contained a 60-minute transient portion (developed with the on-road VW Jetta data) and three 10-minute steady-state portions (defined by RPM). These are referred to as Phases 3, 5, 7, and 9 (P3, P5, P7, and P9) and have average nominal percent loads of 12, 5, 36, and 50% (while fueled with neat ULSD; calculated with the torque curve supplied by Volkswagen; see the 'Percent Load Calculation' section of the Appendix for more detail), respectively. P3 commenced after warming the engine up by running it at 3000RPM and 60% throttle until the coolant temperature stabilized at 92±2°C.

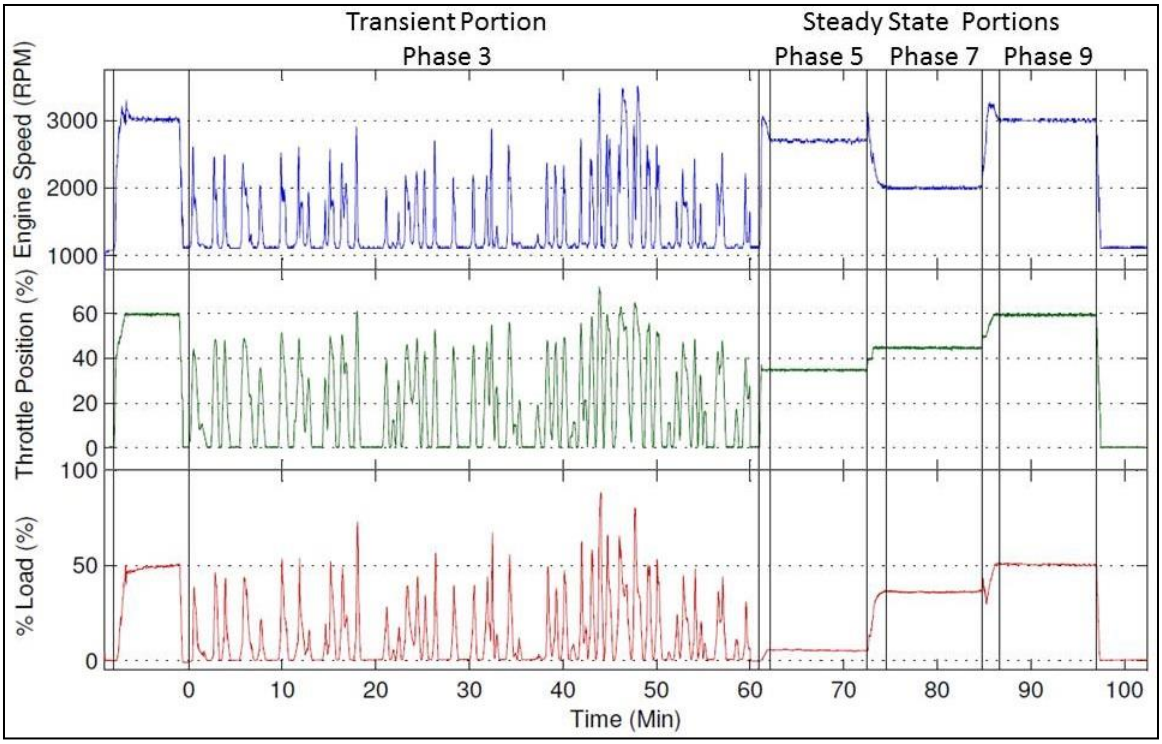


Figure 2.1: Drive cycle used for data collection. Transient and steady-state phases are indicated by vertical lines and phase numbers 3, 5, 7, and 9.

The ideal engine operation control for testing such as that performed is through specification of engine torque and engine speed as a function of time. Due to control software limitations, however, the control of the engine during the transient portion of the cycle was accomplished by specifying throttle position and dynamometer voltage supply, a surrogate for dynamometer load. For the steady-state portions of the cycle, a proportional-integral-derivative (PID) controller automatically adjusted the dynamometer supply voltage to maintain a set point engine speed while throttle position was held constant.

Prior to collecting data, the engine was run at 3300RPM and 85% throttle for two 10 minute periods to elevate exhaust temperatures enough to volatize any contaminants within the exhaust system. The engine oil was then changed to ensure that it did not

contaminate the emissions data. Data collection proceeded in order of increasing biodiesel blend volume to minimize biodiesel contamination of the engine oil. Triplicate tests were performed for each fuel blend.

2.3.3. Measurement Methodology

2.3.3.1. Exhaust Dilution

Dilution of the raw exhaust was necessary to simulate atmospheric dilution and for particle instrumentation measurement. A modified Dekati (Kangasala, Finland) ejector diluter designed to provide a constant dilution ratio (DR) of ~80 was used. Dilution air and exhaust sample temperatures were maintained at 30°C and 110°C, respectively, as they entered the ejector diluter. Table A3 lists the components of the dilution system shown in Figure A1, a schematic of the dilution system. More detail regarding the dilution system can be found in Holmén et al. [54].

2.3.3.2. Data Acquisition

Engine operating conditions, dilution conditions, and PN emissions were measured and recorded simultaneously at a sampling rate of ≥1 Hz. Engine conditions were recorded via a Ross-Tech VCDS scantool (ver. 11.11.6) from the engine control unit (ECU) and the engine/ dynamometer control software, Armfield ArmSoft (ver. 1.43), from auxiliary sensors. Additional engine and dilution system conditions were logged with a National Instruments data acquisition system (LabView, ver. 8.6.1). PN concentration (#/cm^3) data were collected at 1 Hz with a TSI Inc. (Shoreview, MN, USA) 3090 Engine Exhaust Particle Sizer (EEPS; 32 channels from 5.6-560nm).

To make sure all instruments were synchronous, all computers used for logging were connected to a local area network with one computer designated as a time server. The remaining computers synchronized their clocks with the time server every 16 seconds to minimize any asynchrony.

For data analysis, the data from all instruments were post-processed by applying calibration equations to raw data, where necessary, and by interpolating data logged at high frequencies to 1Hz as described in Holmén et al. [54]. The particle emissions data were also time aligned with the operational data to take into account the time needed for the exhaust sample to get from the sample port in the exhaust system to the measurement instrument (see the 'Temporal Alignment' section of the Appendix for more detail) .The post-processed data from each instrument were then concatenated into one file.

2.3.3.2.1. Particle Number Measurements

PN concentration (#/cm^3) data were collected at 1 Hz with a TSI Inc. (Shoreview, MN, USA) 3090 Engine Exhaust Particle Sizer (EEPS; 32 channels from 5.6-560nm). The serial number of the specific instrument used was 3001, it was operating with firmware version MCU:3.10,DSP:3.02, used the 'Default' inversion matrix, was pulling 9 LPM of aerosol sample, and was equipped with a 10LPM, 1µm cut inlet cyclone during data collection. The EEPS bins PN emissions data by D_p. The bounds and midpoint for each bin can be found in Table A4.

To verify that the EEPS was measuring correctly throughout the data collection sequence, both instrument and tunnel blanks were analyzed. The instrument blanks comprised 10 minutes of EEPS 1Hz data collected before each run with a HEPA filter

attached to the aerosol inlet. The tunnel blanks contained the same amount of data collected with the dilution system connected to the EEPS without the engine running. The average particle concentration for both the instrument blank and the tunnel brank were calculated for each EEPS bin and plotted against TSI's stated minimum detection limit (Figure A2). To account for differences in ambient particle concentrations from run to run, the EEPS concentration data (#/cm^3) were corrected with the tunnel blank data during post processing (see the 'PN Data and Blank Correction' section of the Appendix for more detail).

Because it took the aerosol sample a finite amount of time to travel from the sampling port in the exhaust pipe, through the dilution system, and to the EEPS, there was sampling lag associated with the EEPS data. The EEPS data were lag aligned to operational data during data post processing (see the 'Temporal Alignment' section of the Appendix for detail). Once lag aligned, raw exhaust particle concentrations were back calculated using the DR calculated on a second-by-second basis. Finally, the raw exhaust emissions rate (ER) was calculated from the raw exhaust particle concentration and exhaust flowrate. Although exhaust flowrate was directly measured with a pitot tube in the exhaust pipe, exhaust flowrate was modeled using exhaust temperature at the pitot tube and mass air flow (MAF) data. This was necessary because the pressure pluses within the exhaust system of this naturally aspirated engine caused excessive noise in the pitot data. EQ 2.1 was used to calculate exhaust flowrate from the exhaust temperature and MAF and EQ 2.2 was used to calculate ER. More information regarding these

calculations can be found in the 'Engine Exhaust Particle Sizer Post Processing' section of the Appendix.

$$Q_{exh} = \frac{(Q_{\text{int}})(T_{exh})}{293.15} \quad 2.1$$

Where:

Q_{exh} = Volumetric exhaust flowrate (LPM)
Q_{int} = MAF (SLPM – assumed to be approximately volumetric flowrate at ambient conditions)
T_{exh} = Exhaust temperature at the pitot tube (°C) (close to the sample port)

$$ER_P = PN_{CON} \times Q_{exh} \times \frac{1\text{min}}{60\,\text{sec}} \times \frac{1000cm^3}{1L} \quad 2.2$$

Where:

ER_P = particle emission rate (#/sec)
PN_{CON} = Dilution ratio corrected particle number concentration (#/cm^3)
Q_{exh} = Volumetric exhaust flowrate (LPM)

2.4. Results and Discussion

2.4.1. Experimental Control

2.4.1.1. Engine Operation

Cumulative distribution functions of torque, throttle position, and engine speed data from the transient portion of the drive cycle for all runs showed that all sampling events from test to test were comparable from an operational viewpoint (Figure A5). Both torque and throttle position were consistent from run to run, as expected for control variables. There was slight variation in engine speed, as expected for a response variable. Triplicate data for each fuel blend appear to group in the engine speed plot between ~1300 and 2500 RPM, with the B00 data on the upper portion of the curve and with B100 blends on the lower portion of the curve. Although this is an indication of a slight fuel

blend effect on engine speed, two-sample Kolmogorov–Smirnov (KS) tests between all 15 individual WVO runs failed to reject the null hypothesis that the curves being compared were from the same continuous distribution (Figure A5 – WVO runs only). The highest coefficient of variance (CV) value associated with non-idle engine speeds (Engine Speed > 1134 RPM) was 10.3% suggesting that engine speed was consistent between run.

2.4.1.2. Dilution Conditions

Although the DR was relatively constant during individual runs, there was an average drop of 4.4% in exhaust transfer line (tailpipe to diluter) flow rate from run to run; other measured dilution system parameters were constant. Across all runs and phases, the mean ± standard deviation of exhaust inlet temperature, dilution air inlet temperature, dilution air flow rate, and dilution air pressure were 107.7±0.44°C, 26.7±1.2 °C, 85.9±1.4 SLPM, and 29.75±0.39 PSIG, respectively. The measured exhaust inlet temperature and dilution air inlet temperature are ~3°C below the setpoint temperatures of 110°C and 30°C, respectively, because in both cases the thermocouple supplying temperature feedback to the temperature controller was different than the thermocouple providing temperature data to the data acquisition system. The reduction in exhaust transfer line flow rate was due to fouling of the flow control orifice within the transfer line. Rather than disassembling the transfer line to clear the control orifice and risk modifying the system's flow characteristics upon reassembly, the only de-fouling method used was to reverse the flow through the orifice to clear the accumulated particles. Although this helped de-foul the orifice, it did not prevent it completely. Thus, a

continually increasing DR was observed, starting at a low mean value of ~72 for the B0 data collection to a high of ~115 for the B100 data collection (Figure A6). The literature shows that nucleation mode (geometric number median diameter, DGN, in the 7-30nm range) concentrations can be sensitive to changes in DR below a DR of 40, while the accumulation mode (DGN 50-80nm range) remains relatively constant [48]. Above a DR of approximately 50, however, the saturation ratio of condensable hydrocarbons decreases substantially, suggesting that PNDs are less sensitive to variations in DR>50 [10]. Because the DR for this research was consistently >50 and the PN data were corrected with second-by-second DR, it was assumed that the effect of the ~60% change in DR on the raw exhaust PN concentration over all the WVO runs was minimal.

2.4.1.3. Ambient Conditions

Variation in ambient conditions, including ambient temperature, pressure, and absolute (ABS) humidity were examined to determine their effect, if any, on total UFP (TUFP) emissions (defined as the sum of the first 20 EEPS bins; $5.6<D_p<99.7$nm). Their mean ± standard deviations across all runs were 23.5±3.9°C, 991.2±4 mbar, and 11.9±4.6 $mg_{H20}/Liter_{AIR}$, respectively. With CV values across all WVO runs of less than 20%, ambient temperature and pressure were considered consistent. The maximum CV value for ambient absolute humidity across all WVO runs was, however, 38.9%. A scatter plot matrix of absolute humidity vs. TUFP ER by fuel blend and phase (Figure A12) showed that TUFP ERs decreased slightly as absolute humidity increased; however, variability in absolute humidity over the full WVO test sequence was not sufficient to determine a significant relationship between absolute humidity and TUFP emissions.

To examine whether there was a statistical difference in TUFP between replicate engine tests, the Tukey-Kramer method was employed (JMP Pro 10.0.2). As one might expect, given the power associated with large sample numbers (n=3600 for P3 and n=600 for P5, P7, and P9), the majority of the replicates were found to be *statistically* different (α=0.05) from one another [55]. The only replicates not found to be *statistically* different were runs 1 and 3 for B10 in P9, runs 2 and 3 for B20 in P3, and runs 1 and 2 for B20 in P9. Percent differences between the average TUFP ER of all three replicates and the average of each individual replicate were also calculated. The maximum percent difference across all blends and all Phases was 59.2% (for the third B50 run during P7). The average percent difference across all fuel blends and Phases was 24.0%.

Considering the relative consistency of the operational and dilution conditions, and the minimal, if any, effect of ambient conditions, it was determined that, operationally, all replicates could be fairly compared between all fuel blends. The triplicate PN data for each fuel blend were averaged for further analysis.

2.4.2. Total Ultrafine Particle Emission Rate

Figure 2.2 shows that the trend of mean TUFP emissions relative to biodiesel blend across all phases was non-monotonic; increasing relative to B0 for B10, decreasing for B20, and increasing again for B50 and B100. The same trend was seen for TPN emissions (summation of all EEPS bins - 5.6-560nm – Figure A15). The average TUFP ERs (and TUFP concentrations; Figure A14) during P3 were lower than those of P5, P7, and P9 possibly because the engine speeds associated with P5, P7, and P9 (2700, 2000, and 3000RPM, respectively) were higher than the average engine speed of P3

(1500RPM). As engine speed increases, the finite amount of time available for combustion decreases, likely increasing emissions of unburnt and partially burnt hydrocarbons, subsequently increasing particle concentrations. Additionally, as engine speed increases, so does exhaust flowrate, increasing ERs at high engine speed more so than at low engine speeds.

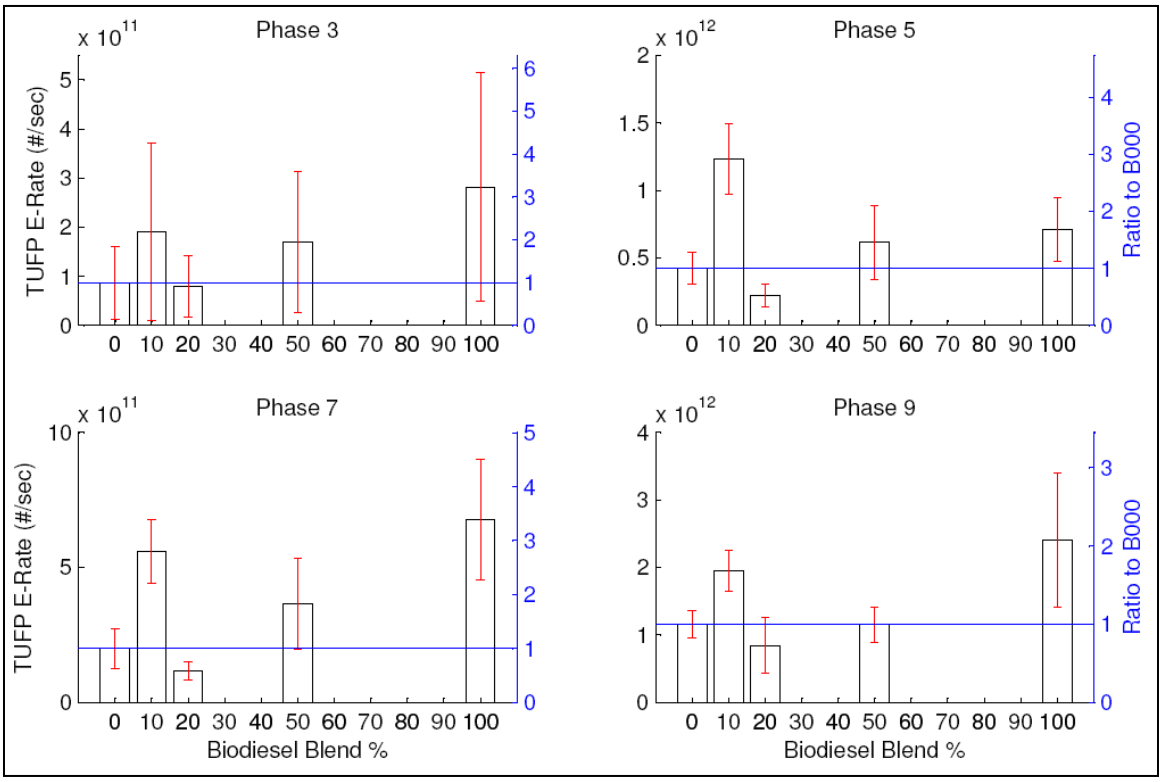


Figure 2.2: WVO biodiesel emissions by run phase and blend percentage. Left Axis = Mean TUFP ER (5.6nm ≤ D_p ≤ 99.7nm). Right Axis = Ratio of biodiesel TUFP emissions rate to that of neat petrodiesel (B00). Each column represents the mean of combined triplicate data for each fuel blend and error bars represent ± 1 standard deviation. N = 3600x3 for P3 and 600x3 for P5, P7, and P9. Note: Y-axes are scaled differently from plot to plot.

Although the majority of previous studies did not report specifically on TUFP emissions, many reported an increase in nuclei mode particle concentrations and a decrease in accumulation mode particle concentration suggesting that TUFP emissions could either increase or decrease depending on the proportion of change within the UFP

D_p range [16,17,27,28,30,36–38]. The majority of studies suggest a decrease in TUFP [27,30,32,39,40], while others suggest an increase [16,17,37,56] as the content of biodiesel in the fuel increases.

2.4.3. Particle Number Distribution (PND)

To further examine the non-monotonic trend in TUFP emissions relative to biodiesel blend reported here, the average PND for each set of triplicate data were compared (Figure 2.3). By convention, the plots depicted are log-log, which somewhat obscures the differences discussed. These differences are more apparent on log-linear plots (Figure A16).

The PNDs measured here (Figure 2.3 and Figure A16), were trimodal. Henceforth, these modes will be described in terms of modal diameter (D_{Mo}) as the small, middle, and large mode. These modes changed by fuel and phase in terms of both D_{Mo} (along the X axis) and in terms of ER (along the Y axis). The small mode D_{Mo} consistently fell in the 10.8nm EEPS bin (9.98 – 11.52nm) across all fuels and phases while the middle mode was always within 3 consecutive EEPS bins spanning 15.36 – 23.65nm. Both small and middle modes were consistently within the nuclei mode D_p range defined by Kittelson et al. ($5<D_p<50$nm). The large mode was always within 5 consecutive EEPS bins from 27.31 – 56.09nm and tended to be smaller than the defined minimum D_p of the accumulation mode ($50<D_p<1000$nm) [10].

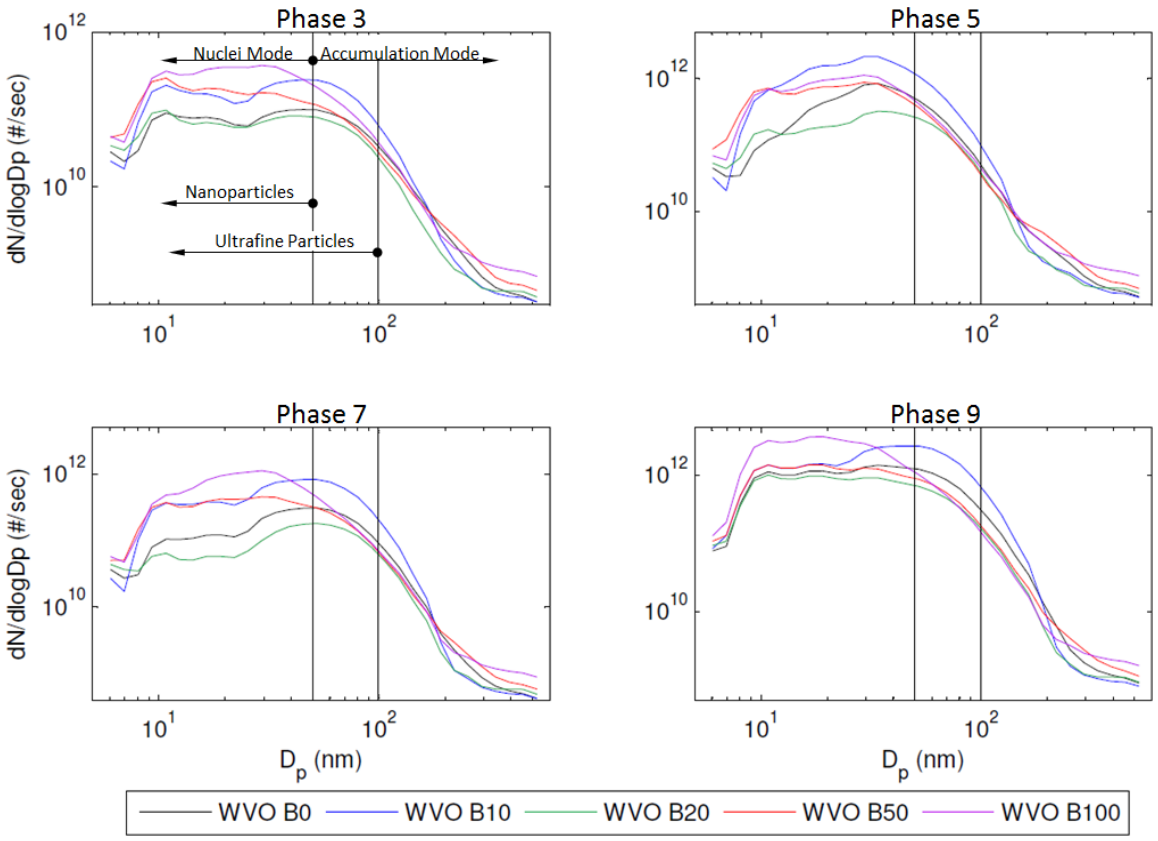


Figure 2.3: Average EEPS particle number distributions (PNDs) for each fuel blend.

Table 2.2 tabulates EEPS bin midpoint D_{Mo} by biodiesel blend and phase. For P3, the middle mode D_{Mo} fell within the 16.5nm EEPS bin for all fuels except for B100 where the D_{Mo} fell within the 22.1nm EEPS bin. For P5, P7, and P9, the middle mode D_{Mo} most often fell within the 19.1nm EEPS bin, decreasing to 16.5nm EEPS bin for B10 (P7) and B50 (P9) and increasing to 22.1nm EEPS bin for B100 (P7). The 16.5, 19.1, and 22.1nm midpoint EEPS bins are consecutive bins, therefore the middle mode D_{Mo} change may have been just outside of the bounds of the 19.1nm EEPS bin (17.74 – 20.48nm). The large mode D_{Mo} was the most variable, shifting to a smaller D_{Mo} as the percentage of biodiesel increased. The only exception to this was in P9, the high load steady-state

phase, when the largest D_{Mo} increased by 2 EEPS bins (increasing from the 34nm EEPS bin to the 45.3nm EEPS bin, skipping the 32.9nm EEPS bin) for B10 relative to B0.

Table 2.2: Modal Diameter (D_{Mo}; nm) and Mode Emission Rate Ratio relative to petrodiesel (B0). D_{Mo}/(ER_{BXX}/ER_{B0}) – Highlighted values indicate a reduction in ER relative to B0.

	BXX	B0	B10	B20	B50	B100
Phase 3	Small	10.8 / 1	10.8 / 2.3	10.8 / 1.1	10.8 / 2.8	10.8 / 3.5
	Middle	16.5 / 1	16.5 / 2	16.5 / 0.9	16.5 / 2.4	22.1 / 4.4
	Large	45.3 / 1	45.3 / 2.4	39.2 / 0.8	29.4 / 1.6	29.4 / 3.7
Phase 5	Small	10.8 / 1	10.8 / 5.6	10.8 / 1.4	10.8 / 6	10.8 / 5.8
	Middle	19.1 / 1	19.1 / 3.6	19.1 / 0.4	19.1 / 1.7	19.1 / 2.2
	Large	34 / 1	34 / 2.5	34 / 0.4	29.4 / 1	29.4 / 1.3
Phase 7	Small	10.8 / 1	10.8 / 3.5	10.8 / 0.6	10.8 / 3.5	10.8 / 4.6
	Middle	19.1 / 1	16.5 / 3.1	19.1 / 0.5	19.1 / 3.5	19.1 / 7.7
	Large	52.3 / 1	52.3 / 2.7	52.3 / 0.6	29.4 / 1.5	29.4 / 3.6
Phase 9	Small	10.8 / 1	10.8 / 1.3	10.8 / 0.9	10.8 / 1.3	10.8 / 2.9
	Middle	19.1 / 1	19.1 / 1.3	19.1 / 0.8	16.5 / 1.2	19.1 / 3.2
	Large	34 / 1	45.3 / 1.9	29.4 / 0.7	29.4 / 0.9	34 / 1.8
Average	Small	10.8 / 1	10.8 / 3.2	10.8 / 1	10.8 / 3.4	10.8 / 4.2
	Middle	18.5 / 1	17.8 / 2.5	18.5 / 0.7	17.8 / 2.2	19.9 / 4.4
	Large	41.4 / 1	44.2 / 2.4	38.7 / 0.6	29.4 / 1.3	30.6 / 2.6

Table 2.2 also tabulates the ratio of BXX ER (ER_{BXX}) to B0 ER (ER_{B0}) showing how the different blends increased or decreased the peak ER of the small, middle, and large modes relative to that of B0. Generally, B10, B50, and B100 increased peak modal ERs of all modes and phases relative to petrodiesel while B20 decreased them. Peak modal ERs for B10 were on average greater than 2 times those of B0, while B20 ERs were, on average, less than or equal to those of B0, consistent with the non-monotonic trend seen in the TUFP data.

The shifts observed in D_{Mo} and ER of the small, middle, and large modes resulted in a reduction in the mean diameter (MD) of the PND of up to 15.5 nm as the percent of biodiesel in the fuel increased (Table 2.3; for detail on calculation, see the 'Mean Diameter Calculation' section of the Appendix for more detail). This agrees with the literature which typically reports reductions in MD as the biodiesel content of the fuel increases [46,57]. It has been suggested that the reduction in MD is a result of the increased oxygen content (~11% O2 m/m for B100) of biodiesel blends relative to B0 [16]. Because diesel engines run fuel-lean, the primary path through which an oxygenated fuel, such as a biodiesel blend, affects emissions is by providing oxygen to fuel-rich zones within the combustion chamber. As a result, the number of solid particles generated from fuel-rich combustion, mainly in the accumulation mode, is reduced through soot particle oxidation [16,57–59]. As the number of solid particles generated in the combustion chamber decreases, so does the surface area available for subsequent adsorption and condensation of volatile gases within the exhaust, which, in turn promotes nucleation of particles, ultimately decreasing the overall MD of the PND [16]. This explanation, however, indicates that, typically, the ER of the large, or accumulation mode, would decrease as the percentage of biodiesel increased.

Table 2.3: Mean Diameter (nm) by Blend and Phase. $\Delta = MD_{BXX}-MD_{B0}$

		B0	Δ	B10	Δ	B20	Δ	B50	Δ	B100	Δ
WVO	P3	38.9	0	38.1	-0.802	35	-3.92	29	-9.97	29.7	-9.29
	P5	35.7	0	31.7	-4.06	35.2	-0.489	27	-8.77	27.8	-7.9
	P7	45.3	0	42.2	-3.1	46.4	1.07	32.2	-13.2	30.1	-15.3
	P9	35.8	0	40.6	4.81	31.3	-4.47	30	-5.79	24.6	-11.2

Although a reduction in the overall MD of the PND was observed here, the only fuel blend that generated a notable decrease in the large mode ER to B0 across all phases was B20 (Table 2.2 & Figure A16). The fact that the overall MD of the PND decreased while there was little loss in large mode particle ER for B50 and B100, coupled with the non-monotonic trend observed in TUFP emissions, suggest that there were both hydraulic and chemical mechanisms altering particle emissions as the biodiesel content of the fuel increased.

2.4.4. Fuel Injection

Because the engine used for this test employed a pump-line-nozzle fuel injection system, the hydraulic properties of the fuel likely affected fuel injection characteristics from blend to blend, potentially altering the PN emissions. Many researchers have indicated that, due to the higher viscosity of biodiesel relative to petrodiesel, less fuel 'leaks' through the injection pump for lubrication resulting in more fuel being injected into the combustion chamber [27,41,46,57,60–62]. Additionally, the higher bulk modulus of biodiesel relative to petrodiesel results in the fuel pressure pulse reaching the mechanical fuel injector sooner [9,41]. The net result of changes in these fuel properties indicate that more fuel can be injected into the combustion chamber sooner (SOI advance) when biodiesel is used relative to petrodiesel, advancing SOC. Biodiesel's higher viscosity and lower volatility relative to petrodiesel can also affect fuel spray duration, geometry, and atomization into the cylinder, potentially affecting combustion dynamics [43].

Here, fuel consumption was measured by weighing the fuel tank for the duration of each run. The scale data indicated that the fuel consumption rate increased with biodiesel blend in terms of mass (Figure 2.4A, see Table A7 for calculation details). Additionally, the volumetric fuel consumption rate was calculated using the scale data and the fuel densities in Table A2, and indicated that the volumetric fuel consumption rate also increased with biodiesel blend (Figure 2.4B). This, combined with analysis showing no statistical difference in throttle position, torque, and RPM from run to run, suggests that the increase in fuel consumption rate and, therefore, overall fuel consumption, was not the result of a change in operating conditions but likely the result of a change in fuel viscosity.

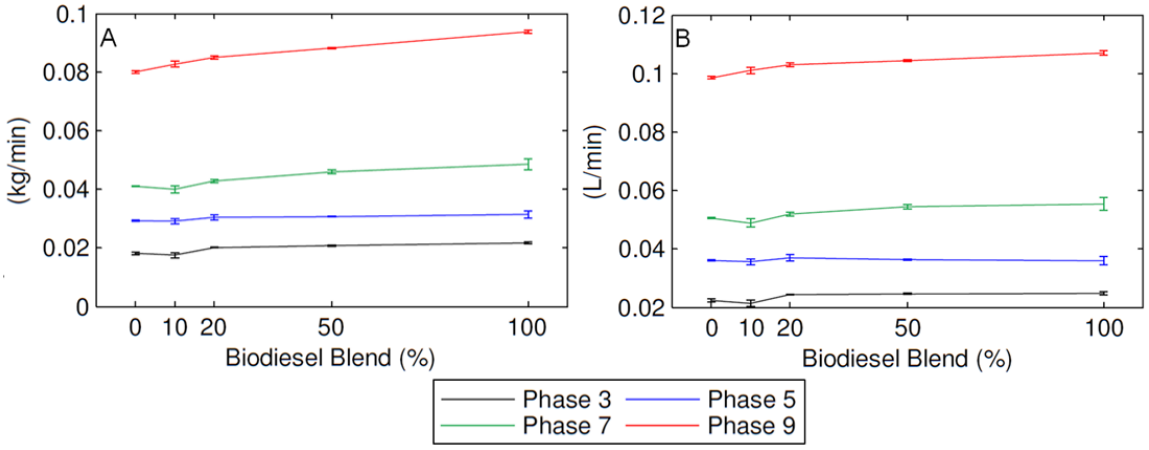


Figure 2.4: Average and standard deviation of fuel consumption rate by blend and phase from the fuel tank scale. Fuel consumption rate determined from each replicate. Average and standard deviation are of the three replicates: A) Fuel Consumption (mg/min); B) Fuel Consumption (L/min). (n=3)

'Fuel Injection Quantity' was also logged via the scantool. This parameter, reported in mg/stroke, is based on an empirical relationship between injection pump operation and the fluid properties of standard petrodiesel. Figure 2.5A shows the average and standard deviation reported 'Fuel Injection Quantity'. These data indicate that,

according to the ECU, the amount of fuel injected when using biodiesel blends was less than or equal to that of petrodiesel. The difference between the directly measured scale data and the empirically reported ECU data further support the finding that fuel consumption of engines with pump-line-nozzle fuel injection systems can increase without a change in operation (the ECU actually 'thinks' less fuel was being injected in some instances). This is likely due to the difference in viscosity between biodiesel blends and neat petrodiesel [27]. See the 'Injector Pump Operation' section in the Appendix for more detail.

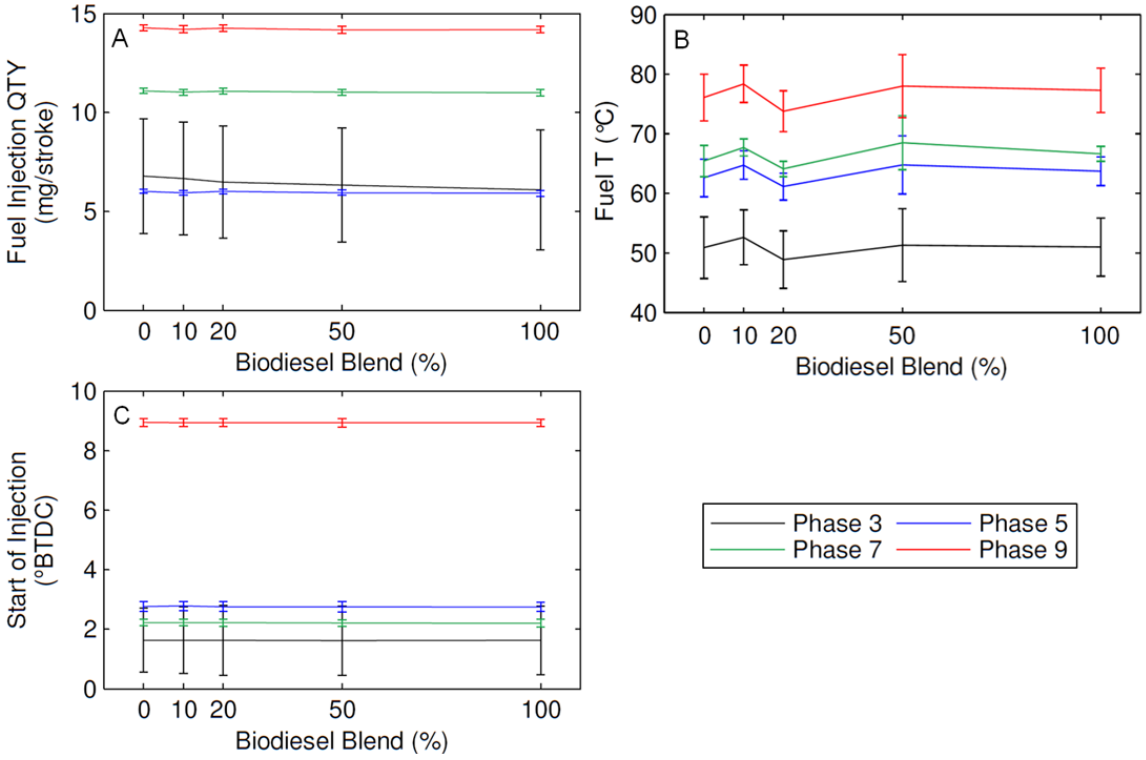


Figure 2.5: Average and standard deviation of scantool fueling properties by blend and phase: A) Fuel Injection QTY (mg/Stroke) B) Fuel Temperature (°C); C) Start of Injection (°BTDC). (n=3550 for 3 and n=600 for P5, P7, and P9)

Additionally, fuel viscosity is dependent on fuel temperature. As fuel temperature increases from 45°C to 55°C, the viscosity of petrodiesel would decrease by

approximately 0.28 cSt, while the viscosity of neat biodiesel would decrease by approximately 0.81 cSt [9]. As fuel temperature increases beyond 55°C, the change in viscosity diminishes [9]. Here, the average fuel temperature measured within the injector pump via the scantool (Figure 2.5B) indicates that the fuel temperature during the B10 data collection was slightly elevated (<5°C) relative to the other fuels which may have caused a slight decrease in viscosity. Depending on the sensitivity of the injector pump to fuel viscosity, this may have allowed more fuel to 'leak' through the injector pump contributing to the reduction in B10 fuel consumption relative to B0 (Figure 2.4). The relationship between the viscosity of B10 and fuel temperature is, however, closer to that of petrodiesel than neat biodiesel, therefore, the change in viscosity would have been minimal, suggesting the increase in fuel temperature may not have been the only cause for the decrease in B10 fuel consumption.

In addition to Injection Quantity and Fuel Temperature, the scantool also collected SOI data. These data (Figure 2.5C) show that average SOI remained consistent by phase throughout testing. It is hypothesized that SOI did not change by fuel blend because the engine used for this study was equipped with a needle lift sensor on the #3 fuel injector giving the ECU a measure of SOI. With the SOI feedback provided by this sensor, the ECU may have been able to adjust injector pump firing to maintain consistent SOI regardless of the fuel being used minimizing the effect of advanced SOI on SOC and, therefore, the effect on TUFP emissions.

2.4.5. Combustion and Particle Growth

Although the data here do not support an advance in SOC due to an advance in SOI, SOC may still have been advanced due to the oxygen content and the possible cetane number increase of the biodiesel blends relative to the baseline petrodiesel [9]. Advanced SOC, along with possible changes in fuel spray duration, geometry, and atomization, may have altered the proportion of premixed combustion to diffusion combustion which may have affected the size and number of particles emitted, contributing to the differences observed in PNDs and TUFP ERs from blend to blend [27,45].

The increased oxidation of solid particles in the combustion chamber due to the increased oxygen content of biodiesel is expected to decrease overall *solid* particle emissions and the overall MD of the PND relative to B0 [16,57–59]. Here, a decrease in overall MD was generally seen for all biodiesel blends relative to petrodiesel (Table 2.3) which is similar to other diesel fuel oxygenates [63]; however, a reduction in ERs was not observed (Table 2.2). The increases in ER was likely due to multiple factors: 1) the increase in fuel consumption indicates enriched combustion resulting in a larger proportion of diffusion combustion and, therefore, soot formation (particles typically in the accumulation mode) [45]; 2) and a possible increase soluble organic fraction (SOF) in biodiesel exhaust relative to that of petrodiesel that has been shown to results in more particle formation and potential growth relative to petrodiesel exhaust [16,30,35,37,58,64–67].

Additionally, solid particles formed in the combustion chamber provide the surface area needed for condensation and adsorption of gas or liquid phase constituents as

they travel through the exhaust system. Changes in the solid particle size distribution entering the exhaust system have the potential to affect the proportion of particle nucleation to particle growth through condensation and adsorption onto existing particles, further altering the PND measured at the sampling port.

Out of 12 studies that examined changes in both SOF and PM emissions from biodiesel relative to B0, all reported increases in SOF; 8 reported decreases in PM [30,37,58,66–70]; 2 reported PM increases [35,64]; and 2 reported PM decreases or increases depending on test conditions [16,65].

Fontaras et al. reported increases in both PN and PM resulting from increased SOF in biodiesel exhaust relative to petrodiesel [16]. The PNDs reported in Fontaras et al. were similar to those reported here in that biodiesel exhaust particle ERs increased across the majority of the D_p range measured. They, however, only tested B0, B50, and B100 [16]. Tinaut et al., on the other hand, measured PM emissions from two LDD vehicles fueled by B0, B5, B10, B20, B50, and B100 and reported a non-monotonic trend in PM emissions for both vehicles: B5 and B10 increased PM emissions relative to that of B0, B20, B50, and B100 [71].

In summary, while researchers have reported non-monotonic trends between low blends of biodiesel for gas-phase emissions, PM emissions, and thermal efficiency [61,62,71,72], this study is one of few to report non-monotonic trends for TUFP emissions.

2.5. Conclusion

On average, WVO B10 increased TUFP ERs to 2 times that of B0 while following a drive cycle that simulated light-duty vehicle urban driving conditions. This increase in TUFP was due to an increase in 5.61 to 100 nm diameter particles with an accumulation mode D_{Mo} between the 34 and 52.3nm EEPS bins. The accumulation mode ER was lower for B20 and shifted to smaller D_{Mo} for B50 and B100 relative to B0 (Figure 2.3 and Figure A16). The TUFP ER decreased slightly relative to B0 when using B20 while the ER observed for B50 and B100 increased to ~2 and 3 times that of B0. Unlike B10, however, the B50 and B100 TUFP increases were primarily due to increases in emissions of smaller diameter particles. Additionally, the data show that engine speed, more so than load, may be a better indicator of particle emission rates given that P5 and P9 (with load/engine speeds of 5%/2700RPM and 50%/3000RPM, respectively) generated the highest particle number emission rates of all four phases (Figure 2.2, Figure A14, & Figure A16).

The trend in TUFP data observed here is likely due to a combination of factors including fuel composition, engine type, and dilution conditions. Because there are a variety of engine designs in use, it is important to perform similar tests on multiple engines to get a clear view of the effect of biodiesel on fleet emissions. Dilution condition variation also complicates comparison of results across studies. If a standard dilution system capable of consistent UFP PND measurements was adopted, comparison across various studies could be more readily conducted. Particle measurement systems that conform to the E.U. Particle Measurement Programme (PMP) are a step in the right direction; however, they only measure solid particles above Dp~23nm [73] whereas

particles, both solid and liquid, smaller than 23nm may play an important role in human health.

CHAPTER 3: DIFFERENCES IN TOTAL ULTRAFINE PARTICLE EMISSIONS FROM WASTE VEGETABLE OIL-BASED BIODIESEL AND SOYBEAN OIL-BASED BIODIESEL FROM A DIESEL ENGINE

3.1. Abstract

To determine the effect of biodiesel feedstock on engine-out particle emissions, a naturally aspirated diesel engine was fueled with neat ultra-low sulfur on-road diesel (B0), two neat biodiesels (B100), and B10, B20, and B50 blends thereof (where XX in BXX refers to the percentage of biodiesel v/v in the blend). The two lots of biodiesel were neat waste vegetable oil-based biodiesel (WVO) and neat soybean oil-based biodiesel (SOY). The fatty acid methyl ester composition of the WVO biodiesel suggests that the waste vegetable oil was primarily used soybean oil. Particle number concentrations were collected at 1Hz with a TSI 3090 Engine Exhaust Particle Sizer, while the engine followed a drive cycle consisting of a transient portion developed from on-road vehicle operation and three steady-state modes. Total ultrafine particle (TUFP; Dp<100nm) number emission rates (ERs) for both biodiesel fuels exhibited a non-monotonic increasing trend – increasing relative to B0 with B10, decreasing for B20, and then increasing again for both B50 and B100. The ratios of B10, B20, B50, and B100 ERs to B0 during transient operation were 2.2, 0.9, 1.9, and 3.2 for the WVO blends and 1.2, 0.83, 1.1, and 2.5 for the SOY blends, respectively. Additionally, WVO biodiesel blends increased nucleation mode emissions relative to B0 more so than SOY biodiesel blends resulting in a larger reduction of the overall mean particle diameter for WVO blends relative to SOY blends. The data collected in this study suggest that the primary

cause for the differences observed in particle emissions between WVO and SOY was due to the heat cycling of the feedstock oil prior to biodiesel production and that TUFP emissions from WVO blends relative to petrodiesel are generally higher than those from SOY blends. This suggests that emissions from WVO blends may be more detrimental to human health than that from SOY blends.

3.2. Introduction

3.2.1. Background

In urban areas, motor vehicles account for a significant fraction of particle emissions [1,2]. Numerous studies have shown that airborne particle emissions contribute to adverse human and environmental health outcomes worldwide [3–5,74]. Exposure to high levels of airborne particles can lead to a number of respiratory and cardiovascular problems including discomfort in irritated airways, increased asthma attacks, irregular heartbeat, non-fatal heart attacks, and even death [6].

One indicator of a particles potential health threat is particle diameter (D_p). As particle size decreases, so does particle mass, resulting in lower inertia and higher mobility. The particle size distribution (PSD) found in the atmosphere is typically comprised of three modes: the coarse, accumulation, and nuclei modes [10]. Their modal diameters generally fall in the D_p>1000nm, 50<D_p<1000nm, and 5<D_p<50nm ranges, respectively [10]. Of these particles, those that have the highest potential to deposit in the alveolar region of the lung where they may subsequently pass directly into the cardiovascular system are ultrafine particles (UFP; D_p<100nm) [3,6,10]. Emissions from light-duty automobiles in the U.S. are currently regulated by the Tier 2 emissions standards, which limit tailpipe particulate emissions on a total particle mass (PM) per

distance basis (g/mi) [11]. Total PM, however, does not adequately characterize UFP emissions because the mass of a UFP is essentially negligible relative to larger particles. A more effective measurement used to characterize UFP emissions is particle number (PN) concentration, the number of particles per volume of air which is why PN emission regulations have been introduced for automotive emissions in the E.U. [10,12–15].

Research has shown that diesel vehicles are a major source of UFP in urban areas and that the particulate emissions from them are highly dependent on fuel composition. Recent interest in energy independence and security has led to legislation that mandates the use of renewable fuels in the U.S. leading to further expansion of domestic biodiesel fuel use [7]. This is evident by the 28% increase in the required production of biomass-based diesel (primarily biodiesel) between 2012 and 2013 [21]. This change in the fuel composition used by the nation's fleet potentially affects UFP concentrations in respirable air.

3.2.2. The Diesel Fleet

Although the U.S. diesel fleet is primarily heavy-duty diesel (HDD), there has been a surge in light-duty diesel (LDD) sales. Registration for LDD vehicles, including passenger cars and sport utility vehicles, rose by 24% between 2010 and 2012 [25]. Additionally, the number of diesel passenger car and sport utility vehicle models available in the U.S. has increased from 3 to 22 between the years 2000 and 2014 [24].

3.2.3. Variation in Biodiesel Properties

Biodiesel can be produced from a variety of lipid feedstocks such as vegetable oils, animal fats, or combinations thereof, resulting in variation between different

biodiesels. Biodiesels processed from different lipid feedstocks have different fatty acid methyl ester (FAME) profiles. The degree of unsaturation of the individual fatty acids is an indicator of their reactivity – the higher the degree of unsaturation, the more reactive the fatty acid is. There are 5 fatty acids that typically dominate the FAME profile of biodiesels derived from vegetable oils or animal fats. In order of increasing degree of unsaturation, these include palmitic acid, steric acid, oleic acid, linoleic acid, and linolenic acid [75]. The average unsaturation level of all the fatty acids that make up a particular biodiesel has been found to be highly correlated to fuel properties such as viscosity, specific gravity, cetane number, iodine value, and low temperature performance metrics [75].

Biodiesel can also be produced from fresh or used (heat cycled) lipids that can result in further variation. When cooking oil is heated, three basic types of reactions occur: thermolytic, oxidative, and hydrolytic reactions [76,77]. Thermolytic reactions occur at high temperatures in the absence of oxygen. Normal alkanes, alkenes, lower molecular weight fatty acids (fewer carbon atoms), symmetric ketones, oxopropyl esters, CO, and CO_2 can be produced from triglycerides that contain saturated fatty acids [76,77]. Dimers and trimers can also form through reactions of different unsaturated fatty acids [76,77]. Triglycerides that contain unsaturated fatty acids can form compounds such as dehydrodimers, saturated dimers, and polycyclic compounds [76,77]. Oxidative reactions occur with unsaturated fatty acids. Hydroperoxides are typically formed as a primary product of an oxidative reaction [76,77]. Hydroxy or keto derivatives can also be formed [76,77]. Additionally, free fatty acids, glycerol, monoglycerides, and diglycerides

are products of the hydrolysis of triglycerides [76,77]. Although feedstock oil is typically pre-processed prior to transesterification to purify the oil of impurities [50], some of the products of heat cycling can inevitably affect the properties of the biodiesel produced from that feedstock.

Finally, although biodiesel can be used as a diesel alternative in its neat form, it is typically blended with petrodiesel for on-road use at levels of ≤B20 (20% biodiesel; 80% petrodiesel, v/v). Given the multitude of fuel parameters that can affect emissions, it has been suggested that there may be interactions, or synergies, between different emissions formation mechanisms that result in non-monotonic trends in emissions as the biodiesel content in the fuel increases [61,62,71,72,78].

3.2.4. Fuel Properties and Emissions

Fuel properties can alter particle emissions by affecting injection, combustion, and nucleation characteristics. Compared to petrodiesel, biodiesel has a higher viscosity, density, and bulk modulus [9,41,42]. There is also variation in these fuel properties between biodiesels from different feedstocks and, of course, between biodiesel blends. Fuel lubricates the injection pump utilized in pump-line-nozzle fuel injection systems. To provide adequate lubrication, the tolerances between parts within the pump are designed to allow some fuel through, diverting it from the fuel injector and back through the pump. As the viscosity of the fuel increases, less fuel ‘leaks’ through the pump, providing more fuel to the combustion chamber. This results in a sharper rise in fuel pressure and a higher ultimate pressure at the outlet of the injector pump [27]. The speed of the pressure pulse through the fuel line between the injection pump and the injector is then dependent

on the bulk modulus of the fuel – the higher the bulk modulus, the faster the pressure pulse [30,41,42]. If a fuel has a higher viscosity and bulk modulus, the resulting injection event can inject more fuel sooner (advanced start of injection (SOI)) compared to a fuel with a lower viscosity and bulk modulus [27]. In some cases, engines are equipped with a sensor that can provide SOI feedback to the ECU which can then alter injection pump firing to minimize changes in SOI. Additionally, changes in pressure rise and the amount of fuel injected can alter the duration and rate of fuel injection into the combustion chamber, potentially changing the stoichiometric profile within the combustion chamber before and during combustion.

Fuel properties can also affect combustion. Fuels that are more oxygenated than others can result in leaner combustion for an equivalent injection volume. Even though diesel engines typically run fuel lean, oxygenated fuels provide fuel-borne oxygen to areas in the combustion chamber that tend to be locally rich [27]. This can result in faster combustion rates and more complete combustion. Cetane number, an indicator of ignition delay – the finite amount of time between SOI and start of combustion (SOC), can also vary from fuel to fuel. Fuels with a higher cetane number can advance SOC that can subsequently alter the amount of premixed combustion relative to diffusion combustion [27,45]. Additionally, atomization of lower volatility fuels (such as biodiesel relative to petrodiesel) within the combustion chamber can be more difficult, again, affecting the stoichiometric profile within the combustion chamber [43].

Low volatility fuels also tend to have low volatility exhaust gases which can affect emissions by altering the proportion of particle nucleation relative to condensation and adsorption as the exhaust gases cool within the exhaust pipe [46].

3.2.5. Objectives

Given that the diesel fleet in the U.S. is primarily HDD and that particulate emissions are currently regulated in terms of PM, the majority of prior studies on biodiesel vehicle/engine particle emissions examined HDD engines and typically report a reduction in PM emissions as the biodiesel content of the fuel increases [57]. It is, however, important to recognize that the 1) LDD fleet is expanding [24,25] 2) UFP emissions measured on a PN basis are more relevant to human health than total PM emissions, 3) biodiesel emissions generated from one feedstock may be different than those generated from another feedstock, and 4) processing of the feedstock, prior to biodiesel production, may have an effect on subsequent UFP emissions. The objective of this study was to compare the engine out UFP emissions of an engine similar to those in LDD vehicles while exercised through a transient drive cycle fueled by multiple biodiesel blends from two different feedstocks. The biodiesels used were waste vegetable oil-based biodiesel (WVO) processed from used soybean oil and soybean oil-based biodiesel (SOY) processed with fresh refined, bleached, and deodorized (RBD) soybean oil, both blended with ultra-low sulfur diesel (ULSD).

3.3. Methodology

3.3.1. Engine Specifications

PN emissions data for this study were collected from a 4 cylinder 1.9L Volkswagen SDi engine coupled to an eddy current dynamometer (Table 3.1). The engine is similar to those found in EURO II Volkswagen automobiles. Sold for industrial purposes, this engine conforms to emissions certification EC 97/68 Stage IIIA. It has no exhaust gas recirculation, selective catalyst reduction, diesel particulate filter, or catalytic converter – the emissions data reported are engine out.

Table 3.1: Engine and Dynamometer Specifications

Engine	
Manufacturer:	Volkswagen
Identification Code:	ARD
Charge Air:	Naturally Aspirated
Capacity:	1896cm^3
Cylinders:	4
Bore:	79.5mm
Stroke:	95.5mm
Compression Ratio:	19.5:1
Nominal Output:	44 kW @ 3600 RPM
Max Torque:	130Nm @ 2000 - 2400 RPM
Minimum CN:	49
Control System:	Bosch EDC
Fuel Injection:	Bosch VE injection pump
EGR:	None
Power Absorption Unit/ Eddy Current Dynamometer	
Manufacturer:	Zelu/ Klam
Model Number:	K-40 PAU
Max Power:	60kW
Max Torque:	145Nm

3.3.2. Fuel Specifications

The fuels used for this study were two lots of B0 (neat ULSD, Trono Fuel, Burlington, VT), one lot of WVO B100, and one lot of SOY B100. Both the neat WVO and neat SOY biodiesels were produced by the University of Connecticut (UCONN) BioFuel Consortium using the methods documented in Pomykala et al. and Boucher et al. [49,50]. The WVO was processed from waste vegetable oil from UCONN dining services and the SOY was processed from edible soybean oil sourced from Catania-Spagna Corporation. Both neat biodiesels were treated with antioxidant (Chemtura Naugalube® 403, see the 'Antioxidant Data Sheet' section of the Appendix for more detail) at 2000ppm (w/w). B10, B20, and B50 were blended from WVO and the first lot of B0 and from SOY and the second lot of B0. B10 and B20 were selected because they are within the range of biodiesel blends typically sold for on-road use. B50 and B100 were also tested to provide data across the range as it is possible to use blends up to B100. The UCONN BioFuels Center for Environmental Sciences and Engineering performed ASTM testing on all but the first lot of petrodiesel and confirmed that both biodiesels conformed to the ASTM standards with the exception of cold soak filtration and combined sodium and potassium of the WVO B100. ASTM test results can be found in Table A1. All fuels were tested in triplicate (30 tests in total).

An analysis of the FAME profiles of the WVO and SOY biodiesels was performed (Figure 3.1) [79]. Compared to the FAME compositional profile of the SOY biodiesel and to those found in Hoekman et al. [75], it was determined that the used cooking oil employed for production of the WVO biodiesel was likely to primarily be soybean oil [79].

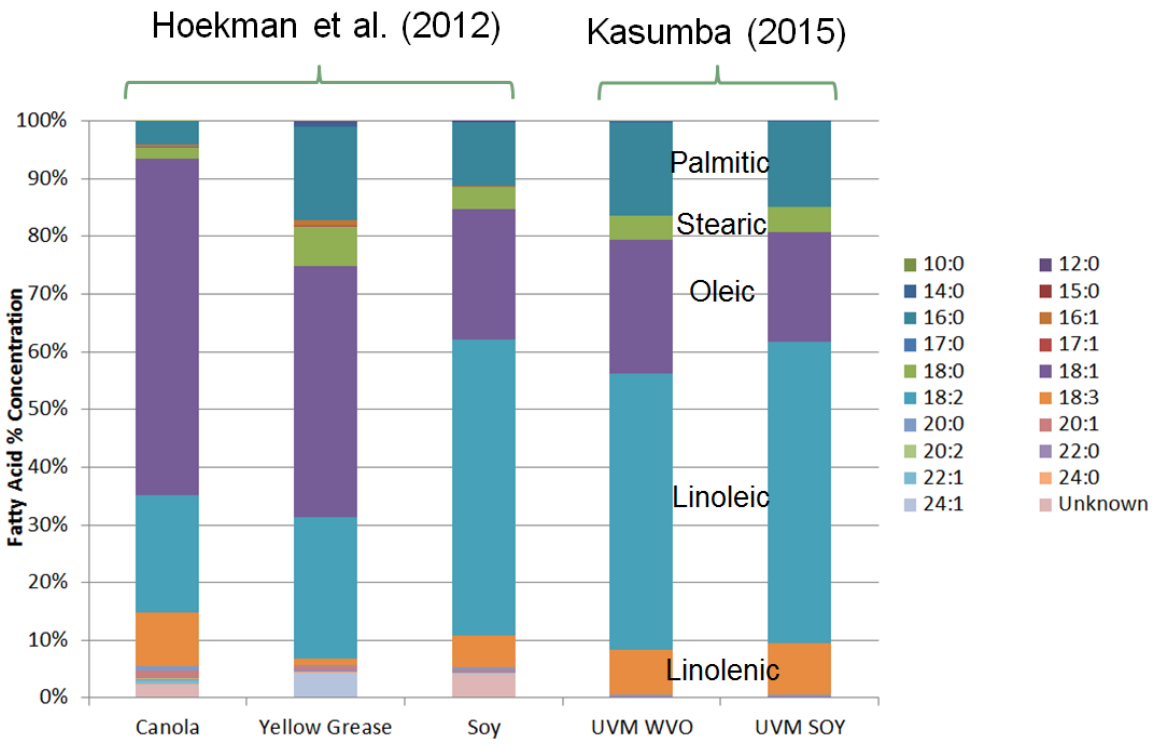


Figure 3.1: Percent composition of FAMEs for UVM WVO and UVM SOY (determined by PhD student, John Kasumba, [79] through GC-MS analysis) compared to those in Hoekman et al. [75].

Chemical analysis of the two lots of petrodiesel was also performed. The results indicate that, in terms of n-alkanes, the two lots of petrodiesel were nearly identical (Figure 3.2) [79].

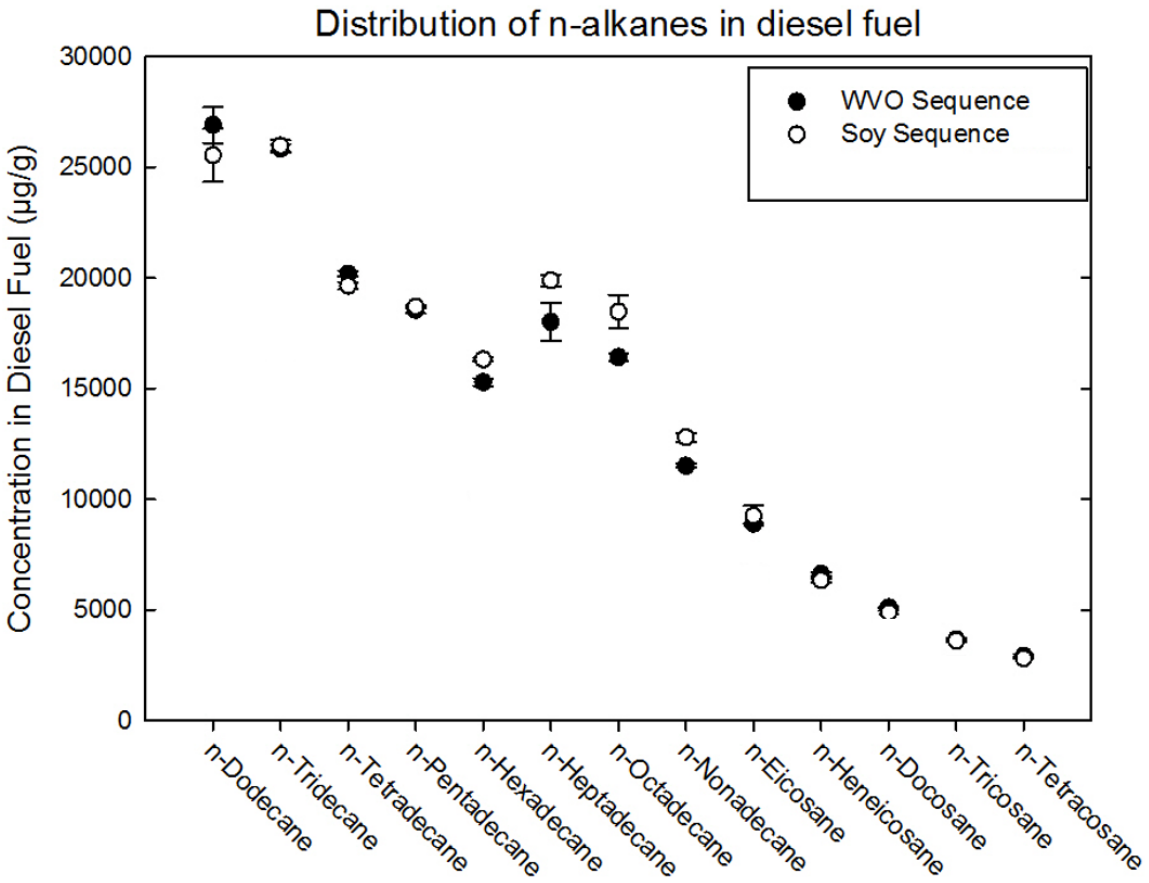


Figure 3.2: Distribution of n-alkanes in the petrodiesel fuel used to prepare the WVO and SOY biodiesel blends. Error bars refer to one standard deviation. n = 2. Determined by PhD student, John Kasumba, and found as Figure 3.12 in his dissertation [79]. Data from a Schauer et al. (1999) removed because they were unnecessary here.

The blended fuels were also analyzed for n-alkanes. Regression equations generated for total n-alkanes vs. BXX% by feedstock showed little difference between BXX blends further indicating that, in terms of n-alkane content, the differences between the two lots of petrodiesel and the different biodiesels were negligible (Figure 3.3) [79].

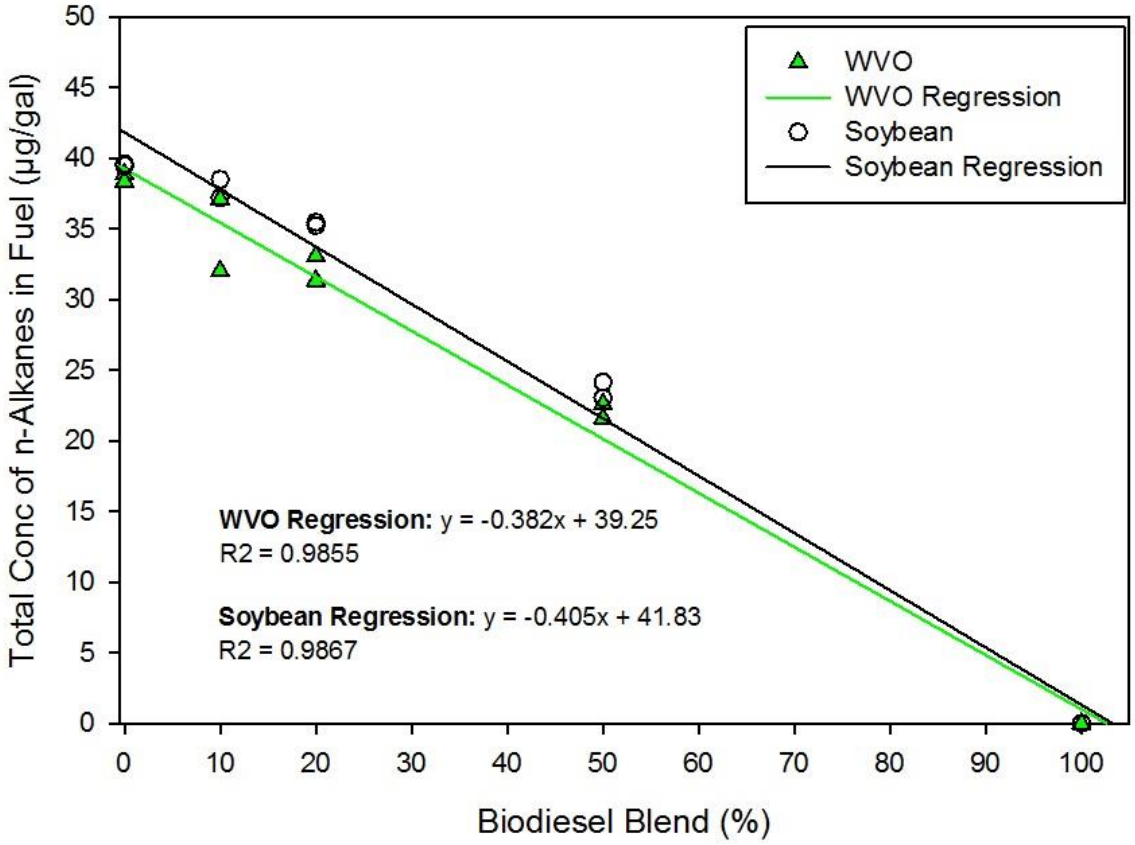


Figure 3.3: Concentration of n-alkanes (µg/gal) in diesel (B00) and biodiesel fuel blends from both feedstocks. Determined by PhD student, John Kasumba, and found as Figure 3.14 in his dissertation [79].

3.3.3. Drive Cycle

A portion of the drive cycle used to collect PN emissions for this study was developed from on road data collected via scantool from a 2003 Volkswagen TDi Jetta sedan (ALH engine code) with an automatic transmission along a predefined driving route through downtown Burlington, VT [52]. The TDi engine in this on-road vehicle was essentially a turbocharged version of the SDi test engine.

The drive cycle consisted of a 60-minute transient portion (developed with the on-road VW Jetta data) and three 10-minute steady-state portions (defined by RPM). These are referred to as Phases 3, 5, 7, and 9 (P3, P5, P7, and P9) and have average nominal %

loads of 12, 5, 36, and 50% (while fueled with neat ULSD; calculated with the torque curve supplied by Volkswagen; see the 'Percent Load Calculation' section of the Appendix for more detail), respectively. P3 commenced after warming the engine up by running it at 3000RPM, 60% throttle until the coolant temperature stabilized at 92±2ºC.

Additionally, prior to collecting data for this study and between collection of the WVO data and SOY data, the engine oil was changed, and the engine was run at 3300RPM and 85% throttle for two ten minute periods in order to volatilize any contaminants within the exhaust system. Data were then collected in triplicate for WVO B0, B10, B20, B50, and B100, followed by the same blends of SOY in the same order. More information regarding the decision to develop a new drive cycle and the drive cycle itself can be found in Chapter 2 and the 'Drive Cycle Development' section of the Appendix.

3.3.4. Measurement Methodology

3.3.4.1. Exhaust Dilution

A Dekati diluter (Kangasala, Finland) modified to provide a dilution ratio (DR) of approximately 80 was used to simulate atmospheric dilution and to facilitate sample measurement for this study. Inlet gas temperatures were maintained at ~30°C (dilution air) and ~110°C (raw exhaust sample) throughout sampling. Table A3 lists the components of the dilution system which are numbered to correspond to Figure A1, a schematic of the dilution system. More detail regarding the dilution system can be found in Holmén et al. [54].

3.3.4.2. Data Acquisition

Engine operating parameters, dilution system data, and PN emissions were collected simultaneously at a minimum of 1Hz for this study. Engine operating data were collected via a Ross-Tech VCDS scantool (ver. 11.11.6) from the engine control unit (ECU) and the engine/ dynamometer control software, Armfield ArmSoft (ver. 1.43) from auxiliary sensors. A National Instruments data acquisition system (Labview, ver. 8.6.1) collected additional engine operating parameters and dilution system conditions. A TSI Inc. (Shoreview, MN, USA) 3090 Engine Exhaust Particle Sizer (EEPS; 32 channels from 5.6-560nm) was used to collect PN concentration (#/cm^3) data. All computers used for data collection updated their clocks to a local time server at 16 second intervals to ensure all data were recorded relative to the same clock, minimizing the need for time alignment during post-processing. Data from the individual instruments were post processed using calibration equations and by interpolation to a common time stamp where necessary.

To verify the EEPS measurement accuracy throughout the data collection sequence, both instrument and tunnel blanks were analyzed. The instrument blanks comprised 10 minutes of EEPS 1Hz data collected before each run with a HEPA filter attached to the aerosol inlet. The tunnel blanks contained the same amount of data collected with the dilution system connected to the EEPS without the engine running. The average particle concentrations for both the instrument blank and the tunnel blank were calculated for each EEPS bin and plotted against TSI's stated minimum detection limit. To account for differences in ambient particle levels from run to run, the EEPS data were

corrected with the tunnel blank data during data post processing (see the 'PN Data and Blank Correction' section of the Appendix for more detail).

Once blanks were corrected, the EEPS data and the individual files collected by the other instruments were concatenated into one database. Even though each instrument logged data relative to the same clock, there was an inherent time lag between the operational data and the PN data associated with the time necessary for the exhaust sample to get from the sample port in the exhaust pipe, through the dilution system, and to the EEPS, where it was measured. To align the PN and operational data, the total PN (TPN) response measured by the EEPS was compared to the RPM response measured by ArmSoft at engine on and engine off. The time differences between TPN and RPM for these two events were measured for each run. The EEPS data for each run was then shifted by the average of these two measurements to align the data.

Once concatenated and aligned, the PN data were also corrected for DR with the dilution data collected via LabView. Finally, PN emissions rates (ERs) were calculated as stated in Chapter 2. More detail regarding data processing can be found in the 'Engine Exhaust Particle Sixer Post Processing' section of the Appendix.

3.4. Results and Discussion

3.4.1. Engine Operation

To verify that, operationally, the engine performed similarly for all 30 tests during P3, cumulative distribution functions (CDFs) were generated for torque, engine speed, and throttle position. The resulting CDFs for every run were then plotted on the same axes for comparison (Figure A5). These plots indicate that throttle position, a directly controlled parameter, remained consistent from run to run. The other directly controlled

parameter was % power to the dynamometer, a parameter closely related to torque which was also nearly equivalent. Lastly, engine speed did show some variability between runs. Upon further inspection it was apparent that the CDFs clustered by BXX blends with B00 blends on the upper portion of the curve and with B100 blends on the lower portion of the curve. This indicates that there may have been slight differences in throttle response related to fuel type. To determine if the observed variability within these parameters was statistically significant, two sample Kolmogorov–Smirnov (KS) tests were performed between all combinations of runs. None of the KS tests rejected the null hypothesis that the CDFs compared were from the same continuous distribution.

3.4.1.1. Dilution Conditions

Across all runs, including both the WVO and SOY sequences, dilution air temperature, exhaust sample transfer line temperature, dilution air flow, and dilution air pressure were consistent with mean ± standard deviation values of 26.7±0.93°C (CV = 3.5%), 107.8±0.47°C (CV = 0.44%), 83.6±2.7LPM (CV = 3.2%), 29.6±0.45PSIG (CV = 1.5%), respectively. There was, however, more variation in exhaust sample transfer line flowrate with a mean of 1.02±0.125LPM (CV = 12.2%). Because DR is sensitive to this flowrate, its mean across all runs was 84.2±11.3 (CV = 13.4%). DR box plots by run can be found in Figure A6. Abdul-Khalek et al. [48] found that the nucleation mode (geometric number median diameter, DGN, in the 7-30nm range) concentrations are sensitive to changes in DR below 40 while the accumulation mode (DGN 50-80nm range) remain relatively constant. Additionally, Kittelson et al. [10] found that, above a DR of approximately 50, the saturation ratio of condensable hydrocarbons decreases

substantially, suggesting that particle number distributions (PND – number weighted PSD) are less sensitive to variations in DR>50. Although the DR did vary during the test sequences, it was consistently >50 for this research and the PN data were corrected with second-by-second DR. With this in mind, it was assumed that the variation in dilution ratio had a negligible effect on the TUFP data.

3.4.2. Ambient Conditions

For this study, the engines intake air was ambient air (conditions uncontrolled) with the mean ± standard deviation values for ambient temperature, ambient pressure, and ambient absolute (ABS) humidity at 19.8±6.1C, 991±4.3mbar, and 9.9±4.6mg_{water}/$Liter_{air}$, respectively. These measurements are all within normal atmospheric conditions so any affects they may have had were indicative of real world variability. To investigate trends between these parameters and TUFP emissions, scatter plots of each parameter versus TUFP emissions by BXX blend and feedstock (Figure A8– Figure A13) were generated. A linear regression line was also included on each scatter plot to indicate if there was a positive or negative relationship between TUFP emissions and the parameter in question. The majority of the data show that TUFP ERs increase with an increase in ambient pressure, a decrease in ambient temperature, or a decrease in ambient ABS humidity. Some of this data, however, suggest the opposite; therefore, no definitive trend between these ambient parameters and TUFP emissions was found with this limited data set.

3.4.3. Engine Exhaust Particle Sizer Blank Verification

An analysis of the average particle concentration for the instrument blanks collected before each run indicated that, throughout both the WVO and SOY sequences, the operation of the EEPS was consistent (Figure A2A). The tunnel blank data, however, show that background particle concentrations were elevated for the WVO sequence relative to the SOY sequence (Figure A2B). The difference in background concentration was accounted for by correcting the EEPS data with the tunnel blank data during data post processing (see the 'PN Data and Blank Correction' section of the Appendix for more detail).

3.4.4. TUFP Emission Rate

Figure 3.4 shows the TUFP ERs measured for each fuel tested by phase. Here, TUFP ER refers to the summation of the ERs from the first 20 EEPS bins; $5.6<D_p<99.7$nm. All phases show a non-monotonic trend in TUFP ERs relative to biodiesel blend for both WVO and SOY blends. TUFP ERs measured during the SOY sequence were higher than those measured during the WVO sequence. This included the TUFP ERs measured for the baseline petrodiesels, which, in terms of n-alkanes, were found to be very similar. In Chapter 2, it was suggested that the dependency of fuel viscosity on fuel temperature may affect fuel consumption and, subsequently, TUFP emissions. To this end, fuel temperature, as measured by the scantool, and fuel consumption rate, as measured by LabView via a fuel tank scale, were plotted (Figure 3.5). Although there was, on average, an approximate 10°C difference in B0 fuel temperature from the WVO sequence to the SOY sequence as seen in Figure 3.5A, there

was not a significant change in fuel consumption rate between the two fuels (Figure 3.5B & C).

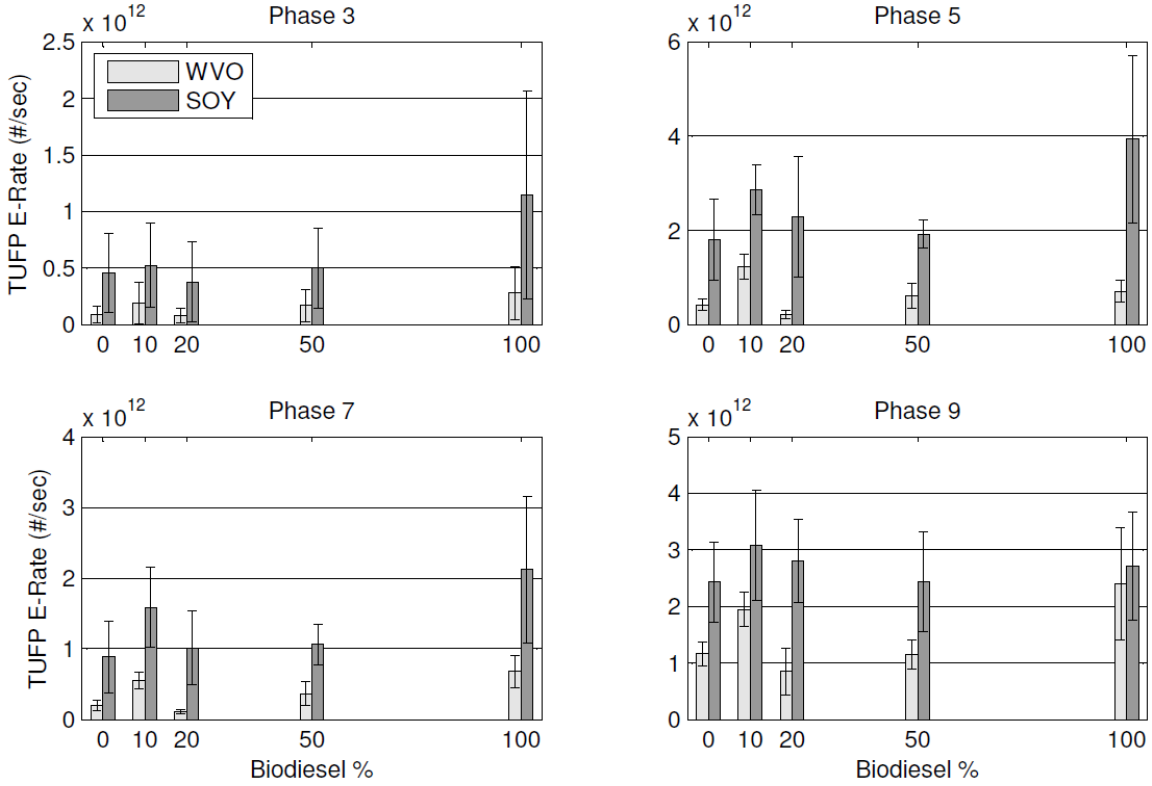


Figure 3.4: Mean TUFP ER ($5.6nm \leq D_p \leq 99.7nm$) by run phase and biodiesel blend percentage. Each column represents the mean of combined triplicate data for each fuel blend and feedstock. Error bars represent ± 1 standard deviation. N = 3600x3 for P3 and 600x3 for P5, P7, and P9. Note: Y-axes are scaled differently from plot to plot.

Although the data collected from the WVO and SOY sequences show consistency across fuels, fuel consumption rate, engine and dilution system operation, and ambient conditions, the difference in TUFP emissions, specifically those between the petrodiesels, suggest an underlying difference between the test sequences that was not captured in these measurements. For this reason, TUFP ERs were not directly compared between the feedstocks. Comparison between feedstocks was done on a ratio basis - the ratio between

the TUFP ER of each biodiesel blend (B10, B20, B50, and B100) and the TUFP ER of the associated petrodiesel (B0) (Table 3.2).

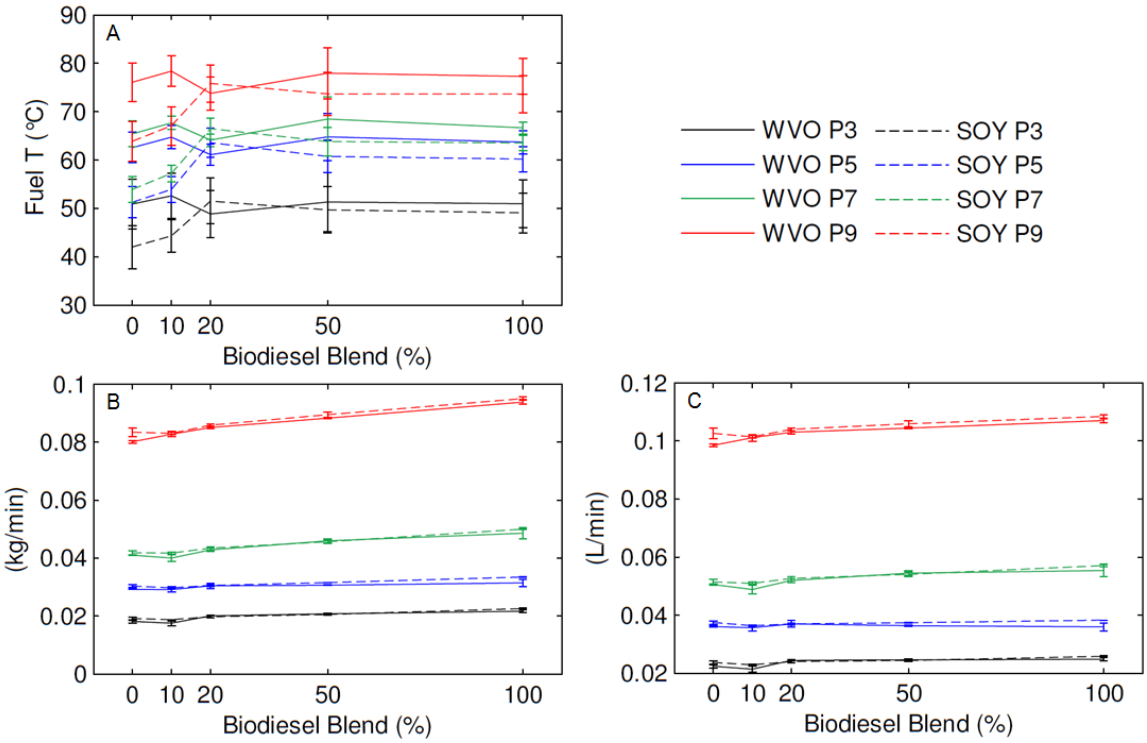


Figure 3.5: (A) Fuel Temperature, (B) Gravimetric fuel consumption, (C) Volumetric Fuel Consumption

Both Figure 3.4 and Table 3.2 indicate a non-monotonic trend between biodiesel content in the fuel and TUFP ER for both feedstocks similar to those found by Tinaut et al. for PM emissions [71] and by Surawski et al. [78] for PN emissions. The Perkins engine used by Surawski et al. was a 4 cylinder naturally aspirated engine with a pump-line-nozzle fuel injection system similar to the one used here. The test vehicles in Tinaut et al. were a Renault Laguna 2 1D and a Renault 19 1.9D. The engines in these vehicles were likely 4 cylinder engines as well and, given that the paper was published in 2005, may have also had pump-line-nozzle fuel injection systems and been naturally aspirated.

The data collected for this study show that the TUFP ER increased for B10 relative to B0, decreased for B20, and increased for B50 and B100 (Figure 3.4). Similar trends were found for TPN ERs (summation of all EEPS Data from 5.6-560nm –Figure A15). Additionally, this pattern was consistent through all 4 drive cycle phases. Table 3.2 shows that the non-monotonic trend was more pronounced for the WVO data in that the ER ratios relative to B0 deviated from one with greater magnitude than the SOY ER ratios.

Table 3.2: Ratio of Biodiesel Blend TUFP emissions to that of the Baseline Petrodiesel.

		B0	B10	B20	B50	B100
WVO	P3	1	2.2	0.9	1.9	3.2
	P5	1	2.9	0.52	1.5	1.7
	P7	1	2.8	0.58	1.8	3.4
	P9	1	1.7	0.73	1	2.1
	Average	1	2.4	0.7	1.6	2.6
SOY	P3	1	1.2	0.83	1.1	2.5
	P5	1	1.6	1.3	1.1	2.2
	P7	1	1.8	1.1	1.2	2.4
	P9	1	1.3	1.2	1	1.1
	Average	1	1.5	1.1	1.1	2.1

From a phase-to-phase perspective, P5 and P9, the light load and high load steady-state phases, respectively, produced the highest TUFP ERs (Figure 3.4) and TUFP concentrations (Figure A14). This is likely due in part to engine speed. The average engine speeds P3, P5, P7, and P9 were 1500, 2700, 2000, and 3000RPM, respectively. Since P5 and P9 had the highest average engine speeds, the finite amount of time available for the expansion (or power) stroke was, on average, less than that of P3 and P7 (the expansion stroke for P9 was on average ~1/2 that of P3). This likely led to more

unburnt and partially burnt hydrocarbons (fuel) passing through the combustion chamber. These hydrocarbons may have led to the formation of particles in the exhaust system as the gases cooled, increasing particle concentrations and ERs relative to lower engine speeds.

The majority of the literature reviewed did not specifically report on TUFP emissions; however, many showed that biodiesel increases nuclei mode particle emissions and decreases accumulation mode emissions relative to B0 [16,17,27,28,30,36–38]. This suggests that, depending on the proportion of change within the UFP range, TUFP emissions could either increase or decrease for biodiesel blends relative to B0. Most of the studies reviewed suggested that TUFP emissions decreased relative to B0 [27,30,32,39,40], while others suggested an increase with an increase in the proportion of biodiesel in the fuel [16,17,37,56,78].

3.4.5. Particle Number Distribution

In order to determine if certain particle sizes within the UFP range were causing the observed differences in TUFP emissions between the two feedstocks, the average PND for each fuel blend was plotted (n=3). Figure 3.6 depicts the average PNDs for both the WVO and SOY biodiesel blends by phase. These plots are log – log by convention, which veils the differences between blends. These differences are more apparent on log – linear plots (Figure A16 and Figure A17).

All PNDs reported are trimodal. These modes are described in terms of modal diameter (D_{Mo}) as the small, middle, and large mode. These modes change between fuel blends and phases relative to D_{Mo} (along the X axis) and ER (along the Y axis).

Throughout all testing, the D_{Mo} for the small mode consistently fell within the 10.8nm EEPS bin (9.98 – 11.52nm) regardless of fuel. The D_{Mo} of the middle mode ranged from 15.4 – 20.5nm (a span of 2 EEPS bins) for all but WVO-B100 during Phase 3, which had a D_{Mo} that fell within the 22.1nm EEPS bin (20.5 – 23.7nm). This shift (Table A6) was more dependent on drive cycle phase than fuel. The D_{Mo} of the large mode ranged from 27.3 – 56.1nm (a span of 5 EEPS bins) for the WVO dataset and from 27.3 – 48.6nm (a span of 4 EEPS bins) for the SOY dataset (Table A6). As discussed in Chapter 2, the large mode D_{Mo} for WVO-B0, WVO-B10, and WVO-B20 were typically the same while the large mode shifted to a smaller D_{Mo} for WVO-B50 and WVO-B100. The SOY PNDs, however, do not show as significant of a shift to smaller D_{Mo}. For P3, SOY-B0 and SOY-B10 exhibited a smaller D_{Mo} in the large mode than SOY-B20, SOY-B50, and SOY-B100. For P5, SOY-B0 had the highest large mode D_{Mo} in the 39.2nm EEPS bin, which dropped to the 34nm EEPS bin for B10, down to the 29.4nm EEPS bin for B20, and back up to the 34nm EEPS bin for B50 and B100. The large mode D_{Mo} for P7 and P9 were relatively consistent at 34nm except for SOY-B20, which generated a large mode D_{Mo} of 45.3nm in both phases.

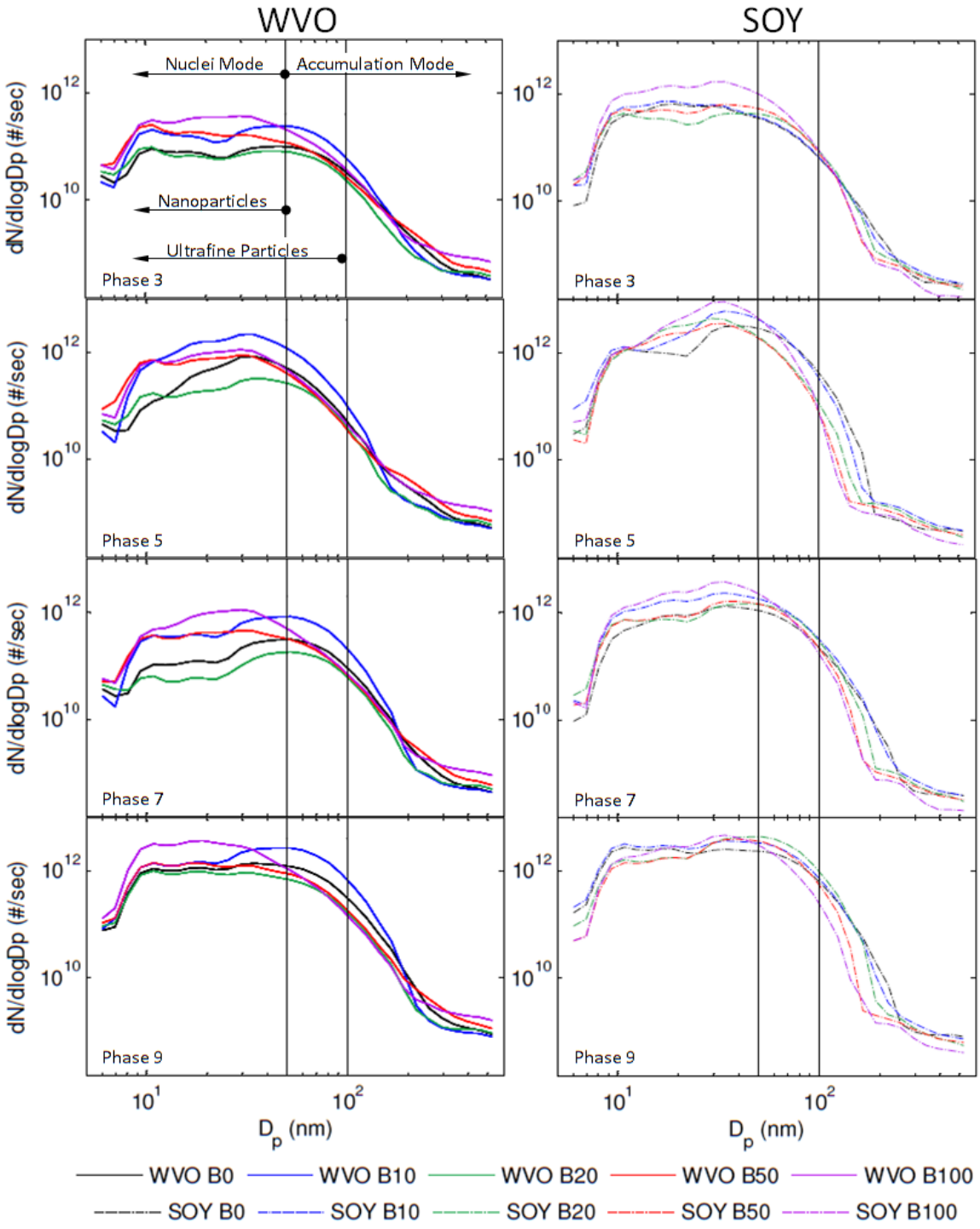


Figure 3.6: Average particle number distributions by biodiesel blend and drive cycle phase. Log - Log.

Regarding the ratio of D_{Mo} ER_{BXX}s to their associated ER_{B0}, the WVO ratios deviate from one more than those of the SOY (Table A6). The individual P3 D_{Mo} ERs for WVO-B10, WVO-B50, and WVO-B100 were on average more than 2 times that of WVO-B0 (Table A6). In contrast, all of the SOY blend D_{Mo} ERs except SOY-B100 were within 2 times that of SOY-B0 (Table A6). Additionally, SOY-B10 did not significantly increase the large mode ER relative to SOY-B0, as was the case for WVO-B10 compared to WVO-B0 (Table A6). The SOY PNDs for P5, P7, and P9 also showed more consistent shapes than the WVO PNDs (Figure A16 & Figure A17). All phases also demonstrate that SOY-B20 and SOY-B50 emissions were similar in terms of both PND shape and ER (Figure A17). Interestingly, Figure A17 shows that, during P9, the large mode ER for SOY steadily increased with biodiesel blend while the small and middle modes steadily decreased – the opposite of what has been reported in the literature [16,57–59]. This is supported by Figure 3.7, which shows the average ER_{BXX}/ ER_{B0} ratios across all phases for all three modes and each biodiesel blend. Here (Figure 3.7), one can see that, for the WVO biodiesel, the small and middle mode ERs increased more than the large mode ERs as the percentage of biodiesel in the fuel increased. Conversely, the middle and large mode ERs increased more than the small mode ERs for the SOY biodiesel blends. This suggests that WVO increased the concentration of nuclei particles relative to B0 more than SOY did.

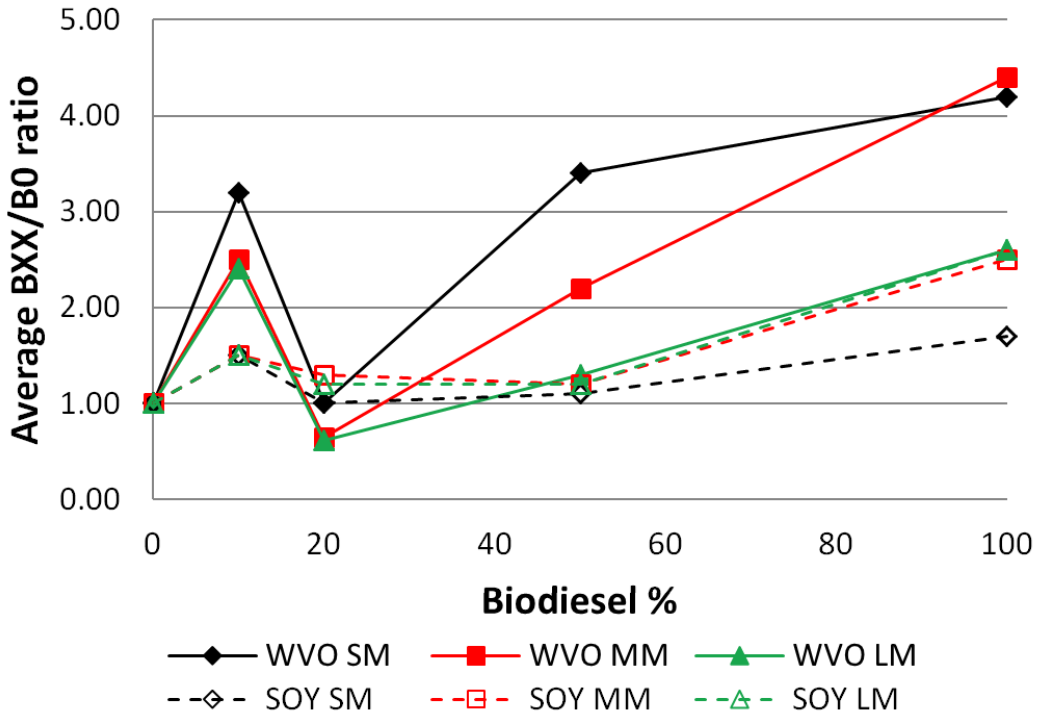


Figure 3.7: ER_{BXX}/ ER_{B0} ratio averaged over all phases by biodiesel blend and mode size (SM = Small Mode; MM = Middle Mode; LM = Large Mode)

The changes in D_{Mo} and peak modal ERs caused the overall mean diameter (MD) of particles emitted from the biodiesel blends to decrease relative their respective petrodiesel baseline in most cases which is similar to reports in the literature (Table 3.3) [46,57]. In general, the overall MD for WVO blends decreased more than that of SOY blends suggesting that, given that same particle composition, the TUFP emissions associated with WVO blends could be more detrimental to human health than those from SOY blends because more particles would be inhaled.

Table 3.3: Mean Diameter (nm) by Blend and Phase. $\Delta = MD_{BXX}-MD_{B0}$

		B0	Δ	B10	Δ	B20	Δ	B50	Δ	B100	Δ
WVO	P3	38.9	0	38.1	-0.802	35	-3.92	29	-9.97	29.7	-9.29
	P5	35.7	0	31.7	-4.06	35.2	-0.489	27	-8.77	27.8	-7.9
	P7	45.3	0	42.2	-3.1	46.4	1.07	32.2	-13.2	30.1	-15.3
	P9	35.8	0	40.6	4.81	31.3	-4.47	30	-5.79	24.6	-11.2
SOY	P3	30.6	0	29	-1.58	34.8	4.16	32.8	2.24	29.8	-0.744
	P5	40.1	0	37.3	-2.79	30.7	-9.44	31	-9.13	33	-7.07
	P7	38	0	35.5	-2.49	39.6	1.57	37.3	-0.692	32.2	-5.75
	P9	33.9	0	34.7	0.715	42.5	8.53	39.1	5.11	32.7	-1.2

The differences in TUFP emissions between feedstocks are more subtle than the differences observed between WVO biodiesel and petrodiesel in Chapter 2. The chemical testing performed on the fuels suggests the two lots of petrodiesel were very similar and that both the WVO and the SOY biodiesel were made from a soybean oil feedstock. It is hypothesized that the reactions that can occur during heat cycling of the feedstock oil as described in Section 3.2.3 may have been a main factor leading to the differences observed between WVO and SOY TUFP ERs and PNDs relative to their baseline petrodiesels.

Mittelbach et al. (1999) showed that polymer and polar compounds in rapeseed oil increased over time as they simulated cooking use by heating at 180°C. As a result, the content of dimeric fatty acid methyl esters in the biodiesel produced from the heated rapeseed oil increased [80]. Additionally, the viscosity and Conradson carbon residue (a test used as an indicator of a fuel's coke-forming propensity) of the biodiesel increased and the volatility decreased relative to the amount of time that the rapeseed oil was heated [80].

The neat WVO used for this study had a higher viscosity and carbon residue than the neat SOY used for this study (4.354 mm^2/sec and 0.050% mass compared to 4.166 mm^2/sec and 0.033% mass, respectively; Table A1), which is similar to the findings in Mittelbach et al. The higher carbon residue percentage for the WVO suggests that it had a higher propensity for generating particles, which could explain why the WVO ER_{BXX}/ ER_{B0} ratios were, in general, higher than the SOY ER_{BXX}/ ER_{B0} ratios (Table A6). Additionally, even if the differences in viscosity between the WVO blends and the SOY blends were not sufficient enough to cause a change in fuel consumption, it could have potentially caused a difference in fuel injection dynamics as well as fuel spray atomization between the two feedstocks [42]. It has been suggested that these changes can increase particle emissions [81]. Both of these points support the hypothesis that increase in the TUFP emissions for WVO blends relative to baseline petrodiesel, in contrast to SOY blends, could have resulted in part due to the heat cycling of the feedstock oil prior to being processed into biodiesel.

Additionally, it is assumed that the fatty acids in the WVO biodiesel feedstock oil may have polymerized during heat cycling resulting in higher molecular weight FAMEs in the WVO biodiesel than in the SOY biodiesel. If this was the case, there could have been an increase in adsorption and condensation within the WVO exhaust gases relative to the SOY exhaust gases. This could have been the cause of the higher ER_{BXX}s/ER_{B0} ratios observed.

It is assumed that the primary mechanisms responsible for the non-monotonic TUFP ER trends and the differences in PN emissions between the two feedstocks were

due to changes in the combustion process including, but not limited to, a change in the SOC, a change in fuel injection spray patterns, and a change in the proportion of diffusion combustion to premixed combustion. Given that the engine used for this study was not equipped with combustion diagnostics instrumentation such as an in-cylinder pressure sensor or a fuel pressure sensor, it is difficult to provide further insight.

The data presented here suggest that TUFP emissions associated with WVO blends may be more detrimental to human health compared to SOY blends because 1) the WVO blend ER_{BXX}/ER_{B0} ratios were typically higher than those of the SOY blends and 2) WVO blends tended to reduce the MD of the PND more than the SOY blends. Additionally, the non-monotonic trends in TUFP emissions observed here suggest that there may be an optimal blend ratio (in regards to TUFP emissions) between B10 and B50.

These data in conjunction with the findings of Mittelbach et al.(1999) [80] also suggest that the differences observed between the emissions of WVO biodiesel blends and those of SOY biodiesel blends relative to their respective baseline petrodiesels may have been a result of heat cycling the WVO biodiesel feedstock prior to transesterification, however, further study is required to evaluate this. Ideally, one lot of cooking oil and one lot of petrodiesel would be acquired. Half of the cooking oil would then be used for cooking. The used and unused cooking oils would then be processed into biodiesel and blended separately with the one lot of petrodiesel. Comparison of the emissions from the resulting fuels would provide more clarity on the effect of heat cycling the feedstock oil prior to biodiesel production and combustion. The data acquired

from such a test could then be used to inform the development of feedstock pretreatment techniques needed to produce higher quality biodiesels.

CHAPTER 4: DETERMINING THE PRIMARY ENGINE OPERATING PARAMETERS NEEDED TO MODEL TRANSIENT ULTRAFINE PARTICLE EMISSIONS IN REAL-TIME FROM A DIESEL ENGINE RUNNING ON BLENDS OF BIODIESEL

4.1. Abstract

Mandated increase in the domestic production and use of biodiesel as an alternative fuel for diesel vehicles, despite limited understanding of its impacts on human health and the environment, may alter the concentration and composition of particles in respirable air. To reduce total ultrafine particles (TUFP; particle diameter (D_p) < 100nm) emissions, better engine and emission control will need to be implemented. To do this, a model predicting TUFP emissions in real-time could act as a virtual sensor to provide feedback for control systems. To predict TUFP emissions in real time, the model would need to be efficient, using the minimum number of inputs to accurately predict TUFP. Traditional emissions models typically utilize inputs such as engine torque, engine speed, and throttle position; however, these were likely selected because they are some of the original engine operating parameters measured by engine control units (ECUs) and are, therefore, typically available from any ECU or based on the modelers intuition. To select input parameters from the full suite of engine operating parameters currently available from a typical ECU in an unbiased manner, this research leverages a genetic programming (GP) algorithm to perform feature selection for the prediction of TUFP emissions from a diesel engine running on different blends of petroleum-based diesel (petrodiesel) and waste vegetable oil (WVO) biodiesel. The feature selection performed

here suggests that exhaust manifold temperature, intake manifold air temperature, mass air flow, and the percentage of biodiesel in the fuel are the four primary model inputs needed to predict transient TUFP emissions. This is significant because it suggests that typical input parameters may not be as powerful as other commonly measured engine operating parameters when it comes to predicting TUFP emissions.

4.2. Introduction

Particulate emissions from combustion processes, specifically those from diesel on-road vehicles, are a major source of particulate emissions in urban areas [1,2]. Numerous studies have shown that airborne particulate matter contribute to adverse human and environmental health outcomes worldwide [3–5]. Exposure to high levels of airborne particles can lead to a number of respiratory and cardiovascular problems including discomfort in irritated airways, increased asthma attacks, irregular heartbeat, non-fatal heart attacks, and even death [6].

Particulate emissions from diesel vehicles are highly dependent on fuel composition [16,17]. Interest in energy independence and security led to legislation which mandates domestic use of renewable fuels resulting in an increased use of biodiesel as an alternative fuel for diesel vehicles [82]. The use of biodiesel also has the potential to change the concentration, size, and composition of particle emissions in respirable air.

In addition to a potential biodiesel effect on particulate emissions, automotive emissions regulations continue to tighten, requiring that vehicles emit fewer particles. As time goes on, engine and emission control will have to advance to keep pace with the

regulations. As such, the systems that control engine operation and active emissions control devises would benefit from a real-time feedback of the particle emissions being produced by the engine. This could be accomplished through sensor development and implementation, or a model that utilizes standard engine operating parameters already measured by the engine control unit (ECU) that accurately predicts particle emissions in real-time acting as a virtual sensor as described by Atkinson et al. [83].

Traditional diesel engine emissions models typically use either all parameters available as model inputs or model inputs selected via intuition [83–88]. Typical input parameters include engine speed, torque, and throttle position. In the case of biodiesel emissions modeling the percentage of biodiesel in the fuel is also used. The majority of these models were developed, however, for steady-state rather than transient operation, which is not indicative of real-word engine operation. Because many engine-operating parameters available from a modern ECU are correlated, there is redundancy across these possible model input parameters. For this reason, it would be ideal to "optimally" select those input parameters that best predict the desired output. For example, engine speed and mass airflow are highly correlated parameters so it would be a violation of the underlying assumptions for most traditional statistical techniques to use both of them. Rather than determining which parameters are best, one might use principal component analysis to convert all available inputs into independent principal components that could subsequently be used in statistical analysis modeling. This, however, would not necessarily reduce the number of measured parameters needed and could increase computational time due to the principal component calculation. Multiple models could be

analyzed, changing one input parameter at a time to determine which is best, but this would be a time-intensive process. Genetic algorithms, however, have been developed specifically for feature selection of multi-dimensional, nonlinear problems such as this. Genetic programming (GP), a genetic algorithm first introduced by John Koza in 1990, is particularly suited for feature selection of multidimensional, highly non-linear relationships [89,90]. The GP algorithm is presented with all possible model inputs and a variety of mathematical operators. The GP algorithm then combines a subset of the original input parameters and operators, generating an equation, or model that optimizes some user-defined outcome (or fitness function – i.e., minimizing the mean square error between the predicted output and the measured output) in the form of a tree structure. It then determines the fitness of that model using the measured output. The population of models, represented as tree structures, is evolved utilizing crossover functions (i.e., switching branches between models at some user-defined rate) and mutation (i.e., randomly altering branches in individual models at some user-defined rate). The fitness of the 'new' models relative to the measured output is evaluated over time as the algorithm converges to a hopefully "optimally" fit solution. The GP algorithm repeats this process 'finding' models of better fitness while keeping track of the branches (combination of inputs and operators) that occur more often in models of higher fitness. A Pareto front is generated by plotting complexity rating (i.e., a number that grows with the number and complexity of individual operators) against the fitness of each individual model [91]. The model with sufficient fitness (where sufficiency is pre-defined by the

user) and a minimal complexity rating is then selected as the best predictor of the observed output.

Given the applicability of GP to multidimensional, non-linear problems such as engine emissions modeling, this research utilized *Eureqa* (ver. 1.08.2 Beta (build 7500)), a software package developed by Nutonian, Inc. (Sommerville, MA) that utilizes a GP algorithm. The main objective was to find the most important and minimum number of variables needed to model transient TUFP emissions from a diesel engine running on different blends of biodiesel. The second objective of this paper was to evaluate two approaches to feature selection using *Eureqa* based on computational time, fitness, and problem insight.

4.3. Method

4.3.1. Data

The 1Hz data collected for this feature selection analysis were from a Volkswagen diesel engine as it followed a simulation of transient on-road urban operation fueled with petrodiesel (B0), waste vegetable oil-based biodiesel (WVO B100), and B10, B20, and B50 blends thereof. These blends were selected for higher resolution in the range typically sold for on-road use (B0 – B20) and to capture data across the full range of usable biodiesel blends (B0 – B100). Data for each fuel was collected in triplicate for a total of 15 engine runs of 3600 seconds each. The data are comprised of TUFP emissions, engine operating data (typical of ECU measurements), and the blend percentage of the biodiesel being used. Parameters 1 through 16 in Table 4.1 were used as model inputs while parameter 17 was used as the model output. Parameters 1 through 15 were dynamic during each engine run while parameter 16 remained static during individual runs.

Table 4.1: Parameters

Par #	Description	Acronym	Unit	MIN	MAX
1	Intake Manifold Air Pressure	MAP	kPa	0.63	2.17
2	Exhaust Manifold Temperature	EMT	°C	99.6	436.0
3	Torque	TOR	Nm	-1	120
4	Engine Speed	RPM	RPM	1081	3801
5	Start of Injection	SOI	°BTDC	-0.45	9.56
6	Throttle Position	TP	%	0	72
7	Injection Quantity	INJQ	mg/stroke	2.6	23.7
8	Atmospheric Pressure	AP	mbar	980.3	1000.0
9	Coolant Temperature	CT	°C	89.8	94.0
10	Intake Manifold Air Temperature	MAT	°C	20.6	36.3
11	Fuel Temperature	FT	°C	39.3	66.8
12	Absolute Humidity	ABSH	mg_{H2O} / $Liter_{air}$	5.3	21.5
13	Downstream Exhaust Temperature	DET	°C	77.7	203.1
14	Δ Exhaust Temperature	ΔET	°C	-50.3	288.2
15	Mass Air Flow	MAF	SLPM	759	3451
16	BIO %	BIO	%	0	100
17	Total UFP Emissions	TUFP	#/sec	1.7E+10	2.3E+12

Cross-semi-variograms were generated between each of the dynamic input parameters (1-15 in Table 4.1) and the output parameter (TUFP emissions) to determine if there was a temporal lag or correlation with any of the input parameters. The cross-semi-variograms identified up to a 25-second lag between the input and output parameters suggesting that presenting up to 25 seconds of previous input data to the model could improve its predictive capability. Adding 25 seconds of prior data for each dynamic input parameter, however, would have expanded the number of input parameters from 16 to 391, significantly increasing the computational time needed for convergence.

In order to include some temporal history for each dynamic input parameter and keep the computational time within reason, six derivatives (or slopes) associated with parameters 1 through 15 were included. The derivatives covered different time spans (described below) resulting in a total of 106 possible model inputs.

First, instead of dividing 25 seconds into 6 equal time spans, it was assumed that temporal history closer to the current time was more important that temporal history further away in time. With this in mind, the six derivatives chosen were Δ2, Δ5, Δ8, Δ12, Δ17, and Δ23, seconds which were calculated as shown in EQ 1.

$$X_{\Delta n}(t) = \frac{X(t) - X(t - n)}{n} \qquad 1$$

Where:

t = current time step
n = number of seconds prior to the current time step
X = the parameter in question
$X_{\Delta n}$ = the derivative of the parameter in question across the last n seconds

Once the derivatives were added to the data set, all dynamic model inputs were normalized so that the model would not bias parameters with higher numerical values. To ensure that data for all fuels were normalized over the same range, normalization was performed across the entire data set rather than for data pertaining to one fuel at a time. The values from all of the dynamic input parameters in Table 4.1 associated with time *t* were normalized between 0 and 1 as shown in EQ 2. The derivatives, however, were normalized between -0.5 and 0.5 to retain information regarding whether the value associated with the parameter was increasing (positive slope) or decreasing (negative

slope) prior to time *t*. Normalization of the derivatives was calculated as shown in EQ 3 so that the normalized value for a slope of zero would still be zero and to ensure equal weighting of all input parameters throughout the feature selection process.

$$X_n(\mathrm{t}) = \frac{\mathrm{X(t)} - \min(\mathrm{X})}{\max(\mathrm{X}) - \min(\mathrm{X})} \qquad 2$$

$$X_{\Delta n_n}(\mathrm{t}) = \frac{\mathrm{X}_{\Delta n}(\mathrm{t})}{\max(|\ \mathrm{X}_{\Delta n}\ |) \times 2} \qquad 3$$

Where:

X = the parameter in question

t = time

X_n = normalized instantaneous parameter

$\mathrm{X}_{\Delta n}$ = the derivative in question

$X_{\Delta n_n}$ = normalized derivative

4.3.2. *Eureqa* Setup

In *Eureqa*, the user controls the initial set of input and output parameters as well as the GP operators, error metrics, and how much data to use for training and model validation. Table 4.2 contains the operators used for this research.

Table 4.2: Operators used to initialize *Eureqa*

Operators	
Constant	Sine
Input Variable	Cosine
Addition	Exponential
Subtraction	Natural Logarithm
Multiplication	Power
Division	Logistic Function

A mean squared error (MSE) metric was selected to compare the results of the individual models, and 70% of the data were reserved for training and another 30% for validation. This error metric and the percentage of training data to validation data are typical for this type of modeling.

4.3.3. Feature Selection

The first feature selection approach was the *tournament selection approach*. In this approach, multiple *Eureqa* simulations were performed (Figure 4.1). *Eureqa* was initialized with all 106 possible inputs parameters and the data associated with each of the individual fuels for initial feature selection. Three simulations were run for each of the individual fuels to ensure that *Eureqa* converged similarly. Due to the limitations of using *Eureqa*, simulation duration was controlled by the number of generations. Simulations using data for the individual fuels were run for 100,000 generations. The 12 features selected from the initial 15 individual *Eureqa* simulations plus Biodiesel % (for a total of 13 inputs) were then used to initialize three additional simulations which were presented with data from all of the fuels combined to perform a set of "overall" feature selection simulations that used just over 3 million generations each. The 15 initial simulations were

run fewer generations than the last 3 to prevent minimization of the search space prior to performing the "overall" feature selection.

The second feature selection approach was the *single GP setup approach*. Instead of performing GP simulations for each fuel and then using the resulting data to perform GP simulations for all of the fuels, 3 replicate *Eureqa* simulations were initialized using all 106 possible inputs and presented with the data associated with all of the fuels. These simulations were run for 5 million generations due to the increase in search space. These two approaches to feature selection were then evaluated and compared based on computation time, features selected, and modeling insight.

4.4. Results and Discussion

4.4.1. Tournament Selection Approach Results

The tournament approach to feature selection followed the bracket depicted in Figure 4.1. The minimum MSE metrics as well as the maximum R^2 values for each *Eureqa* simulation are noted. A summary of the results for all GP models run is presented in Table A8. The MSE (R^2) values for each of the initial 15 simulations (3 simulations for each of the fuel blends) ranged between 2.3E-3 (0.60) and 1.4E-4 (0.84). Of the original 106 inputs presented to the initial 15 simulations, features selected for each of the fuel blends B0, B10, B20, B50, and B100 resulted in 8, 4, 6, 4, and 6 features respectively. The union of these feature sets resulted in 12 unique features; and these 12 features, plus the percentage of biodiesel in the fuel (13 features in all), were then used to initialize the subsequent *Eureqa* simulations for final feature selection.

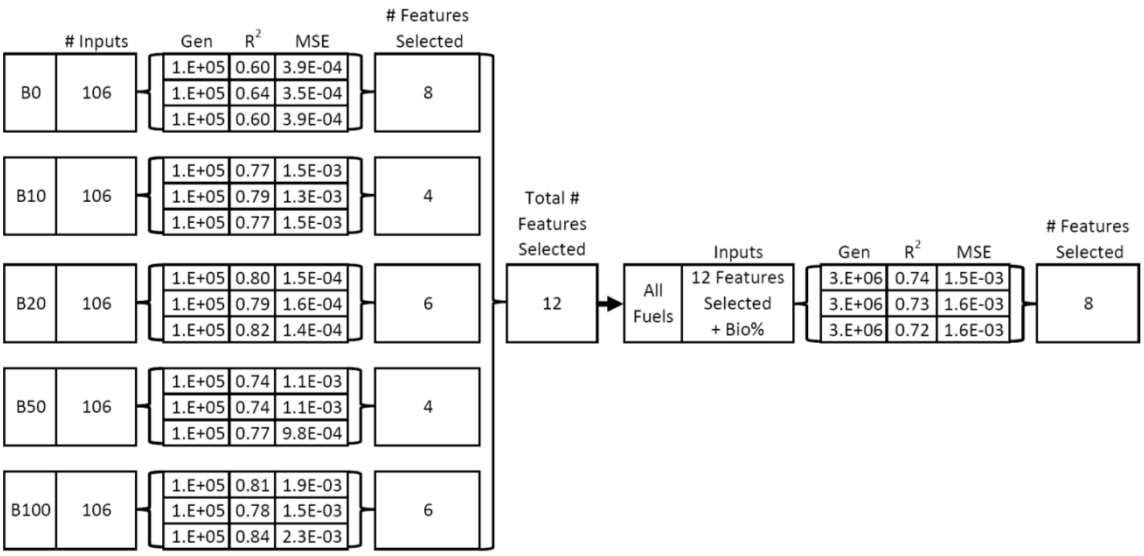


Figure 4.1: Bracket depicting the tournament selection approach to feature selection. The left side of shows the results of each of the three replicate simulations for each of the individual fuels. The right side shows the results of the three replicate simulations that used all of the data (all fuels combined).

Of the 13 inputs used to initialize the final three *Eureqa* simulations that utilized data from all of the fuels, 8 features were selected. Table 4.3 shows that none of the *Eureqa* simulations identify the exact same set of features. This suggests that: 1) the importance of parameters differs by fuel type, and 2) the differences in ambient conditions from one test to another may have affected the importance of some parameters. Another interesting thing to note from Table 4.3 is that the parameters typically used for modeling, such as engine speed, throttle position, and torque, were not selected suggesting there was enough redundancy between typical modeling inputs parameters and those selected to effectively characterize the processes that led to TUFP emissions. For example, none of the fueling parameters were selected implying that the fueling map (the algorithm used by the engine control unit to determine how much fuel to inject) designed for this engine was inferred through the parameters that were selected such as MAF or MAP, indicators of engine speed, and EMT, an indicator of load. It is

also interesting to note that only Δ23 derivatives were selected in some of the initial 15 *Eureqa* simulation while none of the derivatives were selected during the final three *Eureqa* simulations. Given that one would expect temporal information closer to time *t*, this suggests that the cross-semi-variograms may have been picking up the cyclic nature of the drive cycle itself and not a temporal relationship between the input parameters and the output parameters.

Table 4.3: List of all possible features at time *t* and those derivatives that were selected at least once. Features selected by each *Eureqa* simulation indicated by an X.

All Features Selected	Features Selected from Individual Simulations																	
	B0			B10			B20			B50			B100			All Fuels		
MAP	X	X					X	X	X	X	X	X						X
MAP Δ23	X																	
EMT		X		X	X	X	X						X	X	X	X	X	X
TOR																		
RPM																		
SOI																		
TP																		
INJQ																		
INJQ Δ23		X																
AP	X	X	X		X		X	X								X		
CT		X										X				X		X
MAT							X			X	X	X	X	X	X	X	X	X
FT													X	X	X			
ABSH									X	X	X		X	X	X			X
DET			X												X			
ΔET																		
ΔET Δ23				X	X	X												
MAF		X	X	X	X	X	X	X	X				X	X	X	X	X	X
BIO																X	X	X

The results for the final three *Eureqa* simulations show that EMT, MAT, MAF, and BIO were consistently selected indicating they are primary inputs needed to model

TUFP emissions. The best-fit model resulting from the final three simulations had a MSE of 0.0015 and an R^2 of 0.74. The equation generated is EQ 4:

$$
\begin{aligned}
&TUFP = \\
&0.115 * AP * CT \\
&+ 0.003 * BIO * EMT \\
&+ 0.322 * MAF^{2} \\
&- 1.017 * MAT \\
&- 0.092 * MAF * \sin(BIO + MAT) \\
&- 0.005 * BIO * EMT * \sin(BIO + MAT)
\end{aligned}
\qquad 4
$$

Figure 4.2 through Figure 4.4 show the predicted TUFP emissions (red) against observation data (light cyan = training data; dark cyan = validation data) for the three final simulations used for "overall" feature selection (top) and the Pareto front (bottom).

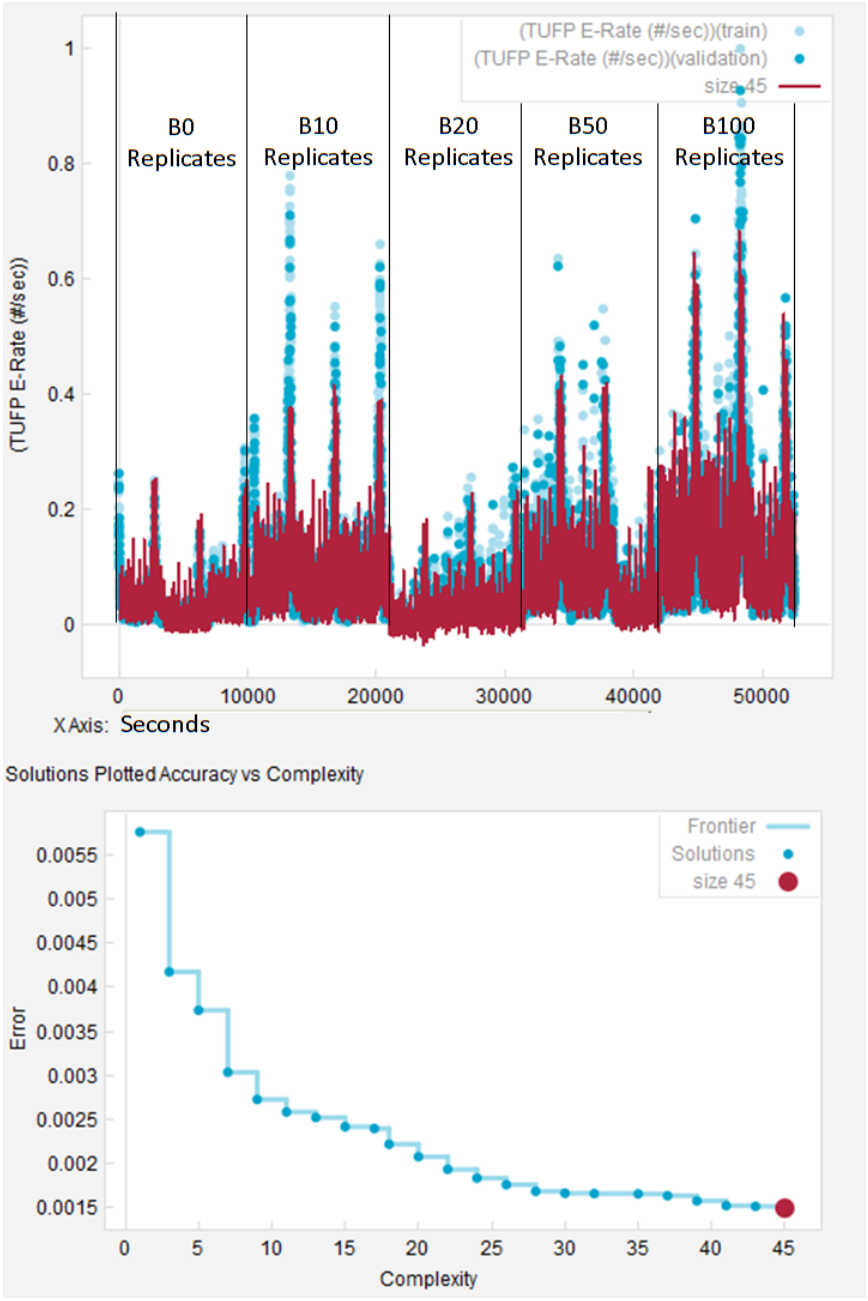


Figure 4.2: Results from replicate 1 of the tournament approach. Note: the model with a complexity of 30 at the Pareto point selected the same features as the model with a complexity of 45. The model with a complexity of 45 had an R^2 of 0.74.

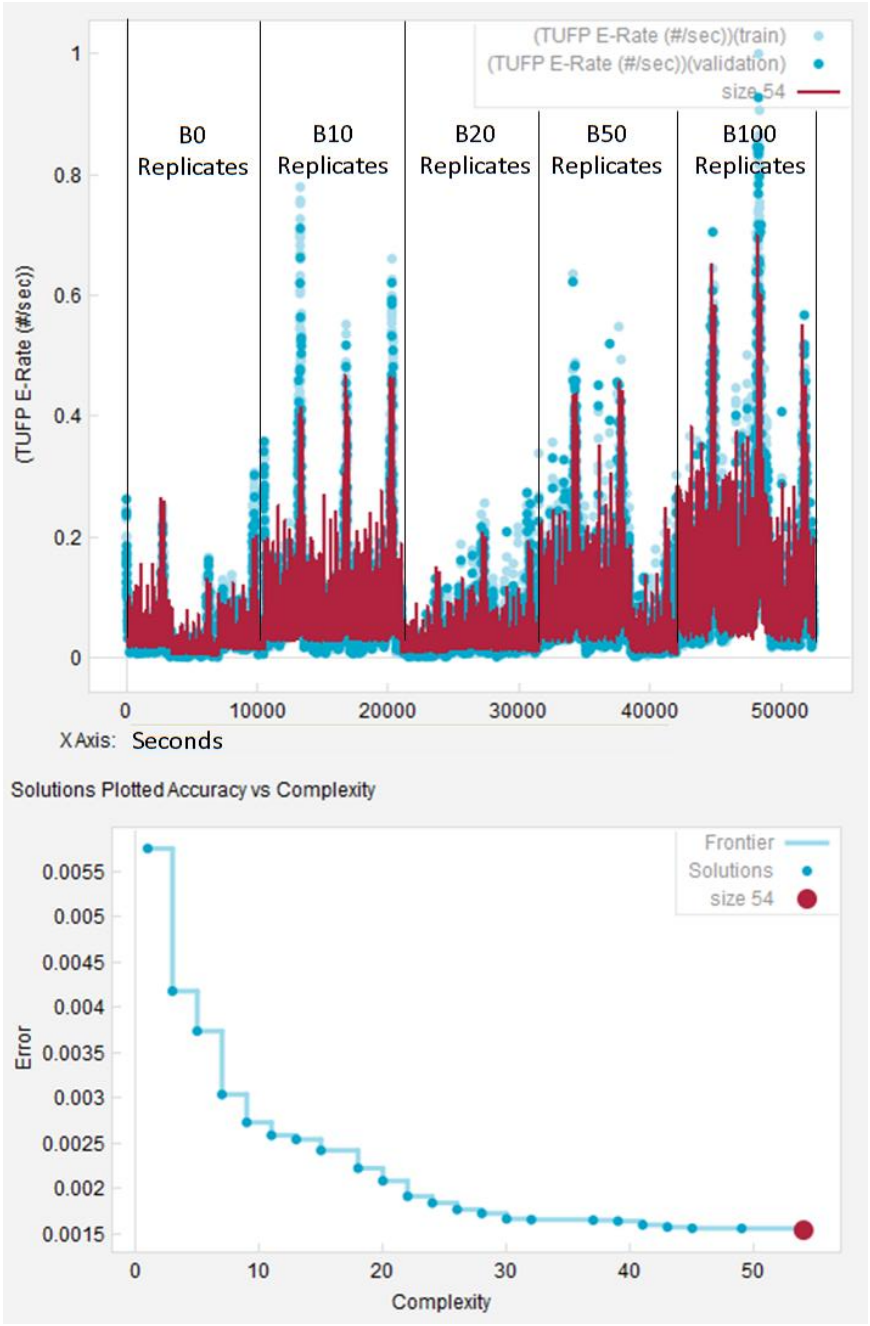


Figure 4.3: Results from replicate 2 of the tournament approach. Note: the model with a complexity of 30 at the Pareto point selected the same features as the model with a complexity of 54. The model with a complexity of 54 had an R^2 of 0.73.

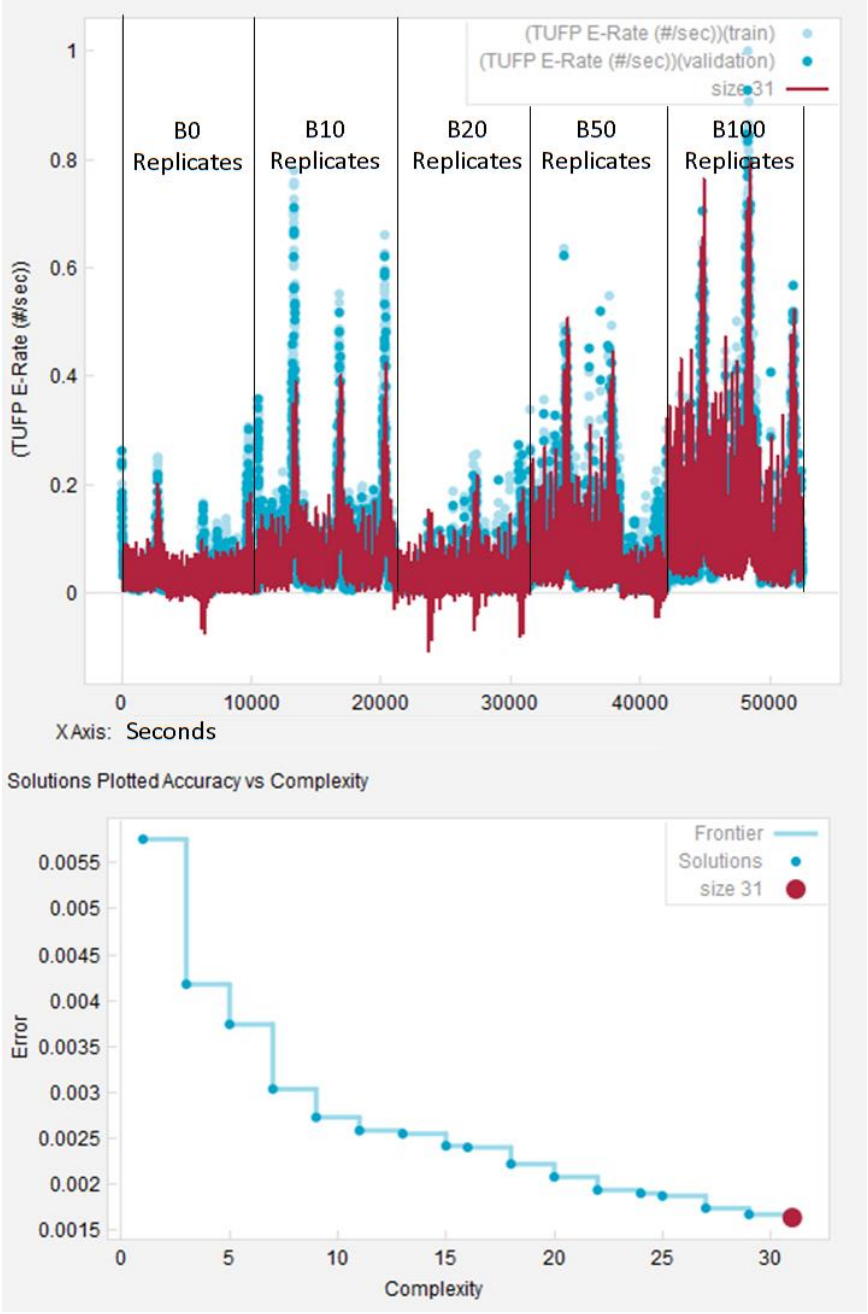


Figure 4.4: Results from replicate 3 of the tournament approach. Note: the Pareto point was not reached. The model with a complexity of 31 had an R^2 of 0.72.

4.4.2. Single GP Setup Approach Results

The single GP setup approach to feature selection was then performed to determine whether the tournament selection approach was beneficial. Again, three *Eureqa* simulations (replicates) were performed to verify convergence. These simulations

were initialized with all 106 possible input features and presented with all of the data from all of the fuels. Because of the increased search space, these simulations were run for 5 million generations. The results for each simulation (Table 4.4) indicate that EMT, MAT, MAF, and BIO were consistently selected as features in the best-fit models similarly to the tournament selection approach. The remaining three features selected with this approach (EMT Δ23, FT, and DET) were selected in only one of the three simulations.

Table 4.4: Features selected from the single GP setup approach presented with all possible inputs

Replicate	**1**	**2**	**3**	**Union**
R^2	0.74	0.75	0.67	
MSE	1.5E-03	1.5E-03	1.9E-03	
MAP				
EMT	X	X	X	X
EMT Δ23			X	X
TOR				
RPM				
SOI				
TP				
INJQ				
AP				
CT				
MAT	X	X	X	X
FT		X		X
ABSH				
DET	X			X
ΔET				
MAF	X	X	X	X
BIO	X	X	X	X

The best-fit model across all three simulations had a MSE of 0.0015 and an R^2 of 0.75 which is very close to the best-fit model that came from the tournament approach that had a MSE of 0.0015 and an R^2 of 0.74. The resulting equation (EQ 5) is as follows:

$$
\begin{aligned}
&TUFP = \\
&0.019 \\
&+ 0.005 * BIO * EMT \\
&+ 0.861 * FT * MAF^{2} \\
&- 0.006 * BIO * EMT * MAT \\
&- 0.381 * FT * MAF * \sin(BIO) \\
&- 0.824 * MAT * FT * MAF^{2}
\end{aligned}
\qquad 5
$$

Figure 4.5 through Figure 4.7 show the predicted TUFP emissions (red) against observation data (light cyan = training data; dark cyan = validation data) for the three simulation replicates of the single GP setup approach (top) and the Pareto front (bottom).

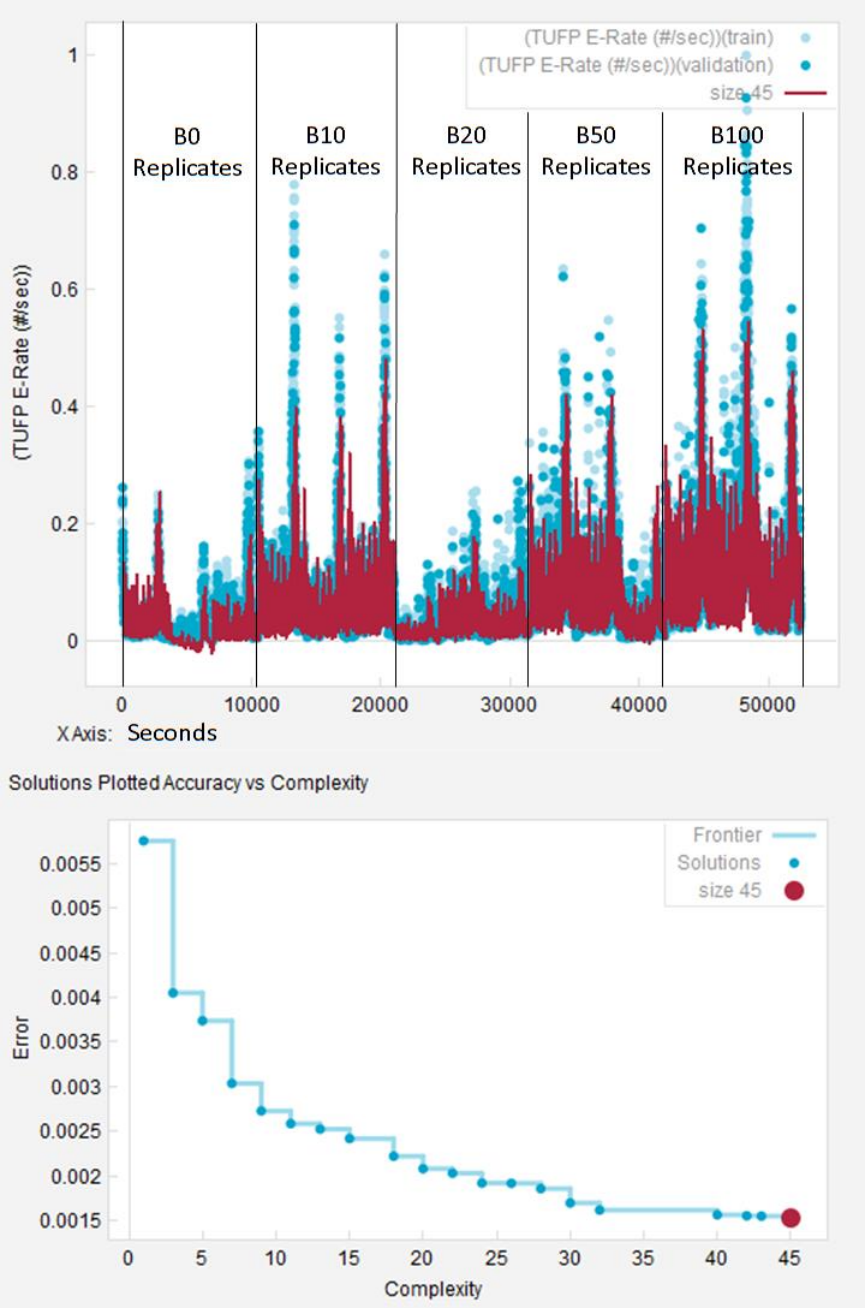


Figure 4.5: Results from replicate 1 of the single model approach. Note: the model with a complexity of 32 at the Pareto point selected the same features as the model with a complexity of 45. The model with a complexity of 45 had an R^2 of 0.74.

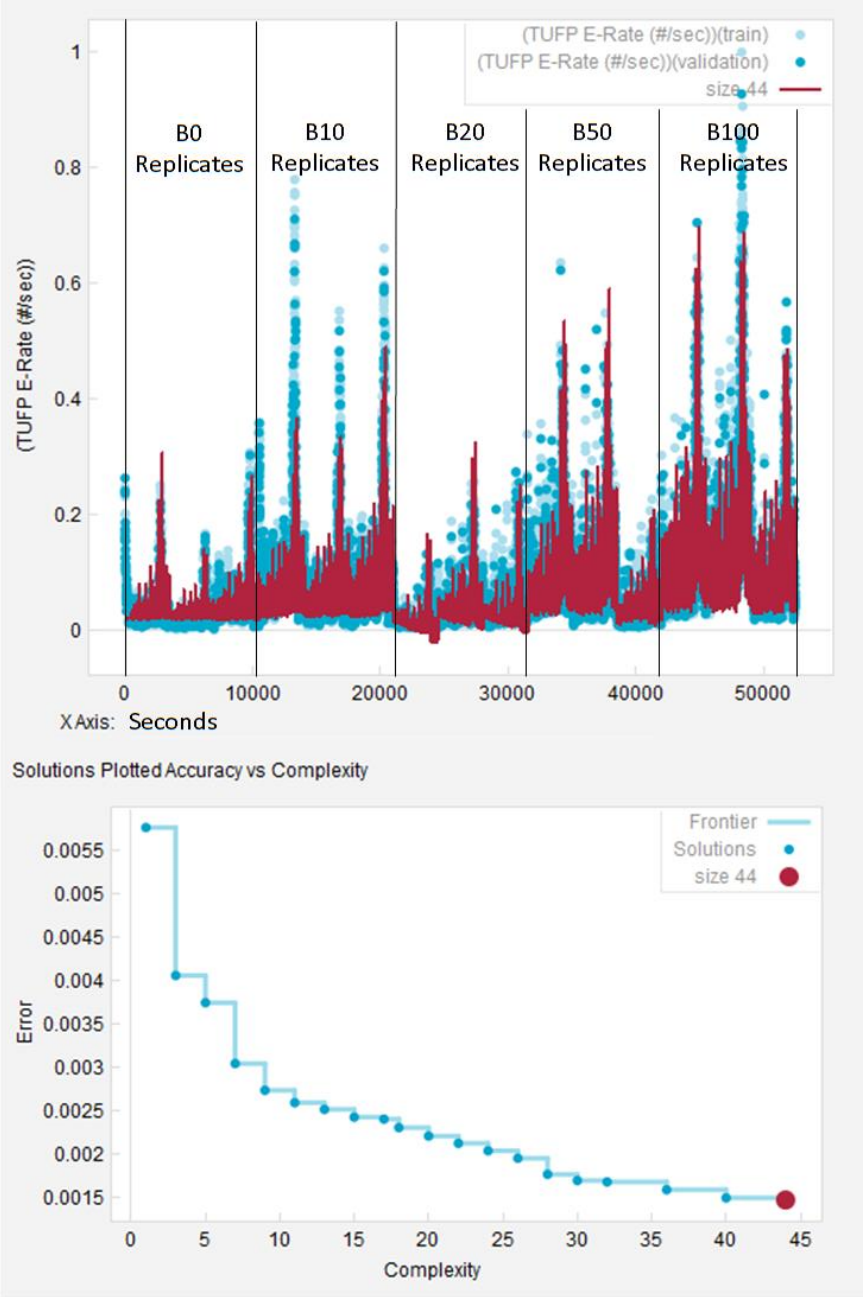


Figure 4.6: Results from replicate 2 of the single model approach. Note: the Pareto point was not reached. The model with a complexity of 44 had an R^2 of 0.75.

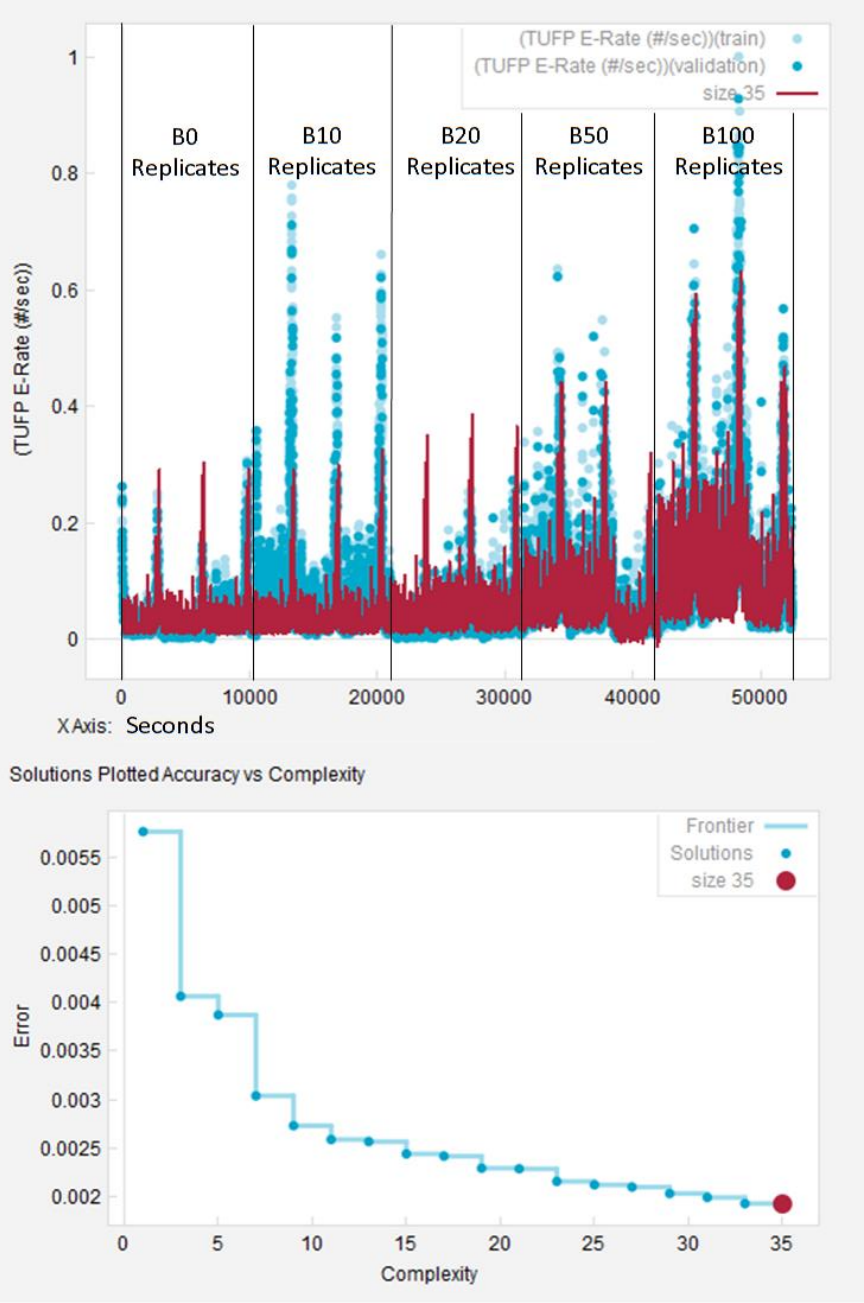


Figure 4.7: Results from replicate 3 of the single model approach. Note: the Pareto point was not reached. The model with a complexity of 35 had an R^2 of 0.67.

4.4.3. Feature Selection Approach Comparison

Both the tournament and single GP setup approach to feature selection yielded the same four primary input features needed for TUFP emissions modeling - EMT, MAT, MAF, and BIO. The tournament approach took approximately 13.7 hours of computation

time, while the single model approach took approximately 28.5 hours indicating that the tournament approach was more efficient regarding convergence of model fitness. Additionally, by analyzing the individual fuels first and seeing what features GP selected for the individual fuels, the tournament approach allowed the user to identify parameters that might be more important for modeling the TUFP emissions from one fuel relative to those of another.

The tournament and the single GP setup approach yielded MSE/R^2 values of ($1.5e^{-3}$/0.74, $1.6e^{-3}$/0.73, $1.6e^{-3}$/0.72) and ($1.5e^{-3}$/0.74, $1.5e^{-3}$/0.75, $1.9e^{-3}$/0.67) respectively. Comparison of EQ4 and EQ5 shows that, in addition to the 4 primary features selected by both approaches, the best-fit equation from the tournament approach added AP and CT, while that of the single GP setup approach included FT. In addition, both equations included a term of the form Constant*BIO*EMT. To determine if these additional parameters are truly important, a sensitivity analysis could be performed by initializing additional simulations with all possible input parameters other than the primary parameters selected here. The parameters selected with the additional simulations would then provide more insight on the underlying physics of TUFP emissions.

In the literature, particle emissions are typically reported to increase or decrease monotonically as the percentage of biodiesel in the fuel increases. The data used for this feature selection process, however, were somewhat unique in that there was a non-monotonic TUFP emissions trend – increasing relative to B0 for B10, decreasing for B20, and increasing again for B50 and B100. Analysis of the different models generated by the

GP along the Pareto front suggests that the sin(BIO) term enables the models to capture the non-monotonic trend in the data used. Although this does not provide insight into the cause of the non-monotonic trend, it shows the GP is capable of discovering solutions for this nonlinear relationship.

Additionally, the R^2 results depicted in Figure 4.1 indicates that models for smaller fuel ranges may more accurately predict TUFP emissions, suggesting an advantage to implementing multiple models throughout the range of blends (i.e., one model for B0 to B5, B5 to B10, and so on) rather than a single TUFP model for all fuel blends (the R^2 values for the individual models were, in general, higher than those of the all fuels models). The appropriate model could then be selected using a lookup table according to what fuel is being used. While the percentage of biodiesel in the fuel is not typically monitored by a modern ECU, a sensor has been developed to measure it which could be implemented [92].

Relative to more traditional emissions modeling, neither feature selection approach employed here selected throttle position, engine speed, or torque as model inputs. This is significant because it suggests that these parameters may not be as powerful as other commonly measured engine operating parameters when it comes to predicting TUFP emissions.

In conclusion, this research indicated that 1) there may be more powerful inputs available to predict TUFP emissions than those typically used for emissions modeling, and 2) implementing a tournament approach to feature selection not only reduces the

convergence time needed to identify features and associated model, but also allows the analyst to infer modeling differences from one fuel to another during the process.

CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS

5.1. Conclusion

The research presented here discusses: 1) the differences in TUFP emissions between petrodiesel (B0) and blends of WVO biodiesel, 2) the differences in TUFP emissions between blends of WVO biodiesel and blends of SOY biodiesel, and 3) the feasibility of using GP (specifically *Eureqa*) as a feature selection tool for modeling TUFP emissions.

An overall increasing non-monotonic trend in TUFP emissions was observed as the content of both WVO and SOY biodiesel increased in the fuel. The data collected suggest that the primary reason for the overall increase in TUFP emissions may have been due to an observed increase in fuel consumption. Further analysis corroborated reports in the literature that fuel consumption increases when fueling with biodiesel due to increased fluid viscosity of biodiesel blends relative to petrodiesel on engines equipped with pump-line-nozzle fuel injection systems. Newer common rail fuel injection systems operate with a fuel rail at constant fuel pressure and electronically controlled fuel injectors that are much less susceptible to this phenomenon.

As noted in Chapters 2 and 3, the WVO biodiesel did not pass the ‘cold soak filtration’ or ‘sodium and potassium’ tests in the ASTM specification for B100 (ASTM-D6751). One might question whether these could have caused the non-monotonic trend observed in the WVO data, however, because the non-monotonic trend was observed in both the WVO and the SOY data, it is not likely. The fact that both WVO and SOY blends produced the non-monotonic trend in TUFP emissions, along with data found in

the literature, suggests that the cause of the non-monotonic trend may have been related to engine technology (type of fuel injection system and air induction). The main difference between TUFP emissions from the two feedstocks was a larger decrease in overall mean diameter (MD) with increasing biodiesel content with WVO blends than with SOY blends (up to a 12nm MD difference; Table 3.3) which suggests that the consequences of using WVO biodiesel compared to SOY biodiesel may be greater relative to human health.

The differences in emission rate observed when comparing TUFP emissions between biodiesels and the non-monotonic trend in TUFP emissions of both WVO and SOY biodiesel relative to petrodiesel were likely due to differences in combustion dynamics related to differences in how the fuels were injected into the combustion chamber and how they combusted.

The use of *Eureqa* to identify the most important operational parameters in predicting TUFP emissions proved beneficial in that the models developed here predicted transient TUFP emissions with an R^2 of 0.75. Additionally, a tournament selection approach to feature selection identified similar features for comparable TUFP emissions modeling as a single GP setup approach to feature selection in approximately half the time. Although typical model inputs such as engine speed, throttle position, and torque were presented as potential input parameters, the features selected by both approaches (i.e., exhaust manifold temperature, intake manifold air temperature, mass air flow, and biodiesel percentage) did not include these more typical model inputs. All features selected, with the exception of biodiesel percentage, are commonly accessible through an

engine control unit. This suggests that the current state of modeling might be improved through better input selection. Additionally, a sensor that can provide biodiesel percentage feedback has been developed [92]. If implemented, the resulting data in conjunction with existing ECU data could be used to model TUFP emissions on-road in real-time as a virtual sensor. Because TUFP emissions are sensitive to fuel composition, it may be beneficial to develop different models for specific ranges of biodiesel blends, and subsequently use a lookup table to select the appropriate model for the fuel being used.

5.2. Future Recommendations

5.2.1. Measurement Equipment

To improve future research outcomes, some of the data collection equipment should be updated. First, the engine used for data collection is now relatively old technology. To collect data more relevant to current automotive technology, a more modern light-duty diesel automotive engine with common rail fuel injection should be acquired. Because combustion dynamics play a large role in particle formation, the engine should be instrumented with combustion diagnostics equipment. This would include a shaft encoder to know precisely how close the piston is to top dead center, an in-cylinder pressure sensor to indicate of start of combustion, and a fuel pressure sensor to give an indication of possible changes in fuel spray.

In addition, a more robust dynamometer control system that can more accurately simulate real-world driving should be developed. This dynamometer system would ideally also be capable of motoring the engine (spin the engine without it running). This would allow simulation of engine braking events (down shifts or coasting) as well as

provide in-cylinder pressure data without combustion needed as a baseline to analyze in-cylinder pressure data collected while the engine was running.

Acquiring a particle measurement system that is comparable to the E.U. Particle Measurement Programmes (PMPs) 'gold standard' for particle measurement, such as the HORIBA MEXA-1000 SPCS, would be beneficial alongside the current particle measurement system. This would allow clearer comparison to other research that utilizes PMP approved equipment and provide a check for the particle measurement system already in place that collects data pertaining to particles with diameters below 23nm unlike the PMP equipment.

5.2.2. Measurement Methods

The measurement methods used for this research could also be improved with the instruments already at hand. The Scanning Mobility Particle Sizer (SMPS) is the 'gold standard' when it comes to PN measurement; however, it takes too long to measure a full PND to be used for transient emissions testing. This is why the EEPS, an instrument capable of 10Hz PND measurements, was utilized for this research. Although the EEPS operation was verified with the SMPS prior to both the WVO and SOY data collection sequences, it would be best to do this for each run. In order to ensure that transient EEPS data is consistent from run to run, both the EEPS and the SMPS should be used to measure steady-state particle emissions before or after any transient cycle. The steady-state EEPS data could then be verified against the steady-state SMPS data for every test and corrected along with the transient EEPS data if need be. Although there may be some error associated with correcting transient EEPS data with a steady-state EEPS to SMPS

relationship, it would be consistent for every test and, therefore, provide a more robust data set.

To determine the amount of SOF in exhaust particles, it would also be beneficial in the future to collect particles on a filter after a thermodenuder to compare to particles collected on a filter without a thermodenuder. If a thermodenuder is not available, filter samples could also be sent to a lab for elemental carbon/ organic carbon analysis to determine the SOF content of exhaust particles.

Real world engine operating data should also be used to determine steady-state test points. To do this, histograms of real world transient data should be generated. These histograms should then be used to determine steady-state operating points that occur frequently. Those should be the operating points of focus.

5.2.3. Fuel

Although not originally intended, the biodiesels used for this testing were blended with two different lots of petrodiesel complicating comparison between the two feedstocks. In the future, it would be beneficial to use one lot of petrodiesel for all fuel blends. Additionally, both biodiesels used were produced in a small scale reactor therefore they may not have been representative of commercially available biodiesels. For this reason, future work should either utilize neat biodiesels sourced from commercial suppliers or ensure the process used during small-scale production is similar to that of commercial scale production.

The ASTM testing of the fuel should also be performed prior to blending to ensure it is within specification. Additionally, due to the non-monotonic trend observed

in TUFP emissions as the amount of biodiesel in the fuel increased, in the future, it would be ideal to have all blends used, even B0, tested to the ASTM-D6751, 'Standard Specification for Biodiesel Fuel Blend Stock (B100) for Middle Distillate Fuels' [93], standard. The baseline petrodiesel should be tested to the ASTM-D975, 'Standard Specification for Diesel Fuel Oils' [94], standards to ensure that it is also representative. Fuels could also be tested to the ASTM-D7467, 'Standard Specification for Diesel Fuel Oil, Biodiesel Blend (B6 to B20)' [95] standard for additional information. In addition, the distillation and viscosity curves for all fuels should be tested to better understand their relative volatilities and injection behaviors. If these tests are performed prior to future testing, they may help explain differences in the TUFP emissions between the different blends.

5.2.4. Modeling

The feature selection presented here shows much promise. Since artificial neural networks (ANNs) are also used for multidimensional nonlinear problem solving and have been used for engine emissions previously [84,85,87,88], one could be developed using the features selected in this research and compared against another using more typical model inputs to determine if the features selected using *Eureqa* do indeed provide better insight into TUFP emissions.

The purpose of the modeling presented here was to determine if GP could be used to select ECU parameters to model TUFP emissions for real-time on-road use. Since there is no dilution system on-vehicle, dilution system parameters measured during testing were not presented to the simulations. Similarly, fuel consumption data from the

fuel tank scale was not presented as it did not have the resolution necessary to be considered a robust second-by-second measurement. If desired, however, GP could be used as a quality control tool. If the features selected from a simulation that was initialized with engine operating and dilution system parameters included dilution system parameters, it would be an indication that variable dilution system parameters were affecting the measured TUFP emissions. This would suggest that TUFP measurements taken at different dilution conditions would not be comparable.

REFERENCES CITED

1. Sawyer, R.., Harley, R.., Cadle, S.., Norbeck, J.., Slott, R., and Bravo, H.., "Mobile sources critical review: 1998 NARSTO assessment," *Atmos. Environ.* 34(12–14):2161–2181, 2000, doi:10.1016/S1352-2310(99)00463-X.

2. Colvile, R.N., Hutchinson, E.J., Mindell, J.S., and Warren, R.F., "The transport sector as a source of air pollution," *Atmos. Environ.* 35(9):1537–1565, 2001, doi:10.1016/S1352-2310(00)00551-3.

3. Brook, R.D., Franklin, B., Cascio, W., Hong, Y., Howard, G., Lipsett, M., Luepker, R., Mittleman, M., Samet, J., Smith, S.C., and Tager, I., "Air Pollution and Cardiovascular Disease A Statement for Healthcare Professionals From the Expert Panel on Population and Prevention Science of the American Heart Association," *Circulation* 109(21):2655–2671, 2004, doi:10.1161/01.CIR.0000128587.30041.C8.

4. Gauderman, W.J., Avol, E., Gilliland, F., Vora, H., Thomas, D., Berhane, K., McConnell, R., Kuenzli, N., Lurmann, F., Rappaport, E., Margolis, H., Bates, D., and Peters, J., "The Effect of Air Pollution on Lung Development from 10 to 18 Years of Age," *N Engl J Med* 351(11):1057–1067, 2004, doi:10.1056/NEJMoa040610.

5. Li, N., Harkema, J.R., Lewandowski, R.P., Wang, M., Bramble, L.A., Gookin, G.R., Ning, Z., Kleinman, M.T., Sioutas, C., and Nel, A.E., "Ambient ultrafine particles provide a strong adjuvant effect in the secondary immune response: implication for traffic-related asthma flares," *Am. J. Physiol. - Lung Cell. Mol. Physiol.* 299(3):L374–L383, 2010, doi:10.1152/ajplung.00115.2010.

6. Wichmann, H., Spix, C., Tuch, T., Wölke, G., Peters, A., Heinrich, J., Kreyling, W., and Heyder, J., "Daily mortality and fine and ultrafine particles in Erfurt, Germany part I: role of particle number and particle mass.," *Res Rep Health Eff Inst* (98):5–86; discussion 87–94, 2000.

7. Energy Independence and Security Act of 2007, 2007.

8. U.S. Energy Information Administration, "Monthly Energy Report May 2015," DOE/EIA-0035(2015/05).

9. NREL, "Biodiesel Handling and Use Guide: Fourth Edition (Revised)," DOE/GO-102008-2658, NREL/TP-540-43672, 938562, National Renewable Energy Laboratory, Golden, CO., 2009.

10. Kittelson, D.B., Arnold, M., and Watts, W.F., "Review of Diesel Particulate Matter Sampling Methods: Final Report," University of Minnesota, Department of Mechanical Engineering, Center for Diesel Research, Minneapolis, MN., 1999.

11. Emission Standards: USA: Cars and Light-Duty Trucks—Tier 2, http://www.dieselnet.com/standards/us/ld_t2.php, Oct. 2013.

12. HEI Air Toxics Review Panel, "Mobile-Source Air Toxics: A Critical Review of the Literature on Exposure and Health Effects," HEI Special Report 16, Health Effects Institute, Boston, Mass, 2007.

13. Pope, C.A., 3rd, Renlund, D.G., Kfoury, A.G., May, H.T., and Horne, B.D., "Relation of heart failure hospitalization to exposure to fine particulate air pollution," *Am. J. Cardiol.* 102(9):1230–1234, 2008, doi:10.1016/j.amjcard.2008.06.044.

14. US EPA National Center for Environmental Assessment, R.T.P.N. and Stanek, L., "Integrated Science Assessment for Particulate Matter (Final Report)," DOCUMENT, http://cfpub.epa.gov/ncea/cfm/recordisplay.cfm?deid=216546, Aug. 2012.

15. Kittelson, D.B., "Engines and nanoparticles," *J. Aerosol Sci.* 29(5-6):575–588, 1998, doi:10.1016/S0021-8502(97)10037-4.

16. Fontaras, G., Karavalakis, G., Kousoulidou, M., Tzamkiozis, T., Ntziachristos, L., Bakeas, E., Stournas, S., and Samaras, Z., "Effects of biodiesel on passenger car fuel consumption, regulated and non-regulated pollutant emissions over legislated and real-world driving cycles," *Fuel* 88(9):1608–1617, 2009, doi:10.1016/j.fuel.2009.02.011.

17. Krahl, J., Munack, A., Schröder, O., Stein, H., Herbst, L., Kaufmann, A., and Bünger, J., "Fuel Design as Constructional Element with the Example of Biogenic and Fossil Diesel Fuels," *CIGR J.* VII(Manuscript EE 04 008), 2005.

18. U.S. Environmental Protection Agency, 40 CFR Part 80 subpart I - Motor Vehicle Diesel Fuel; Nonroad, Locomotive, and Marine Diesel Fuel; and ECA Marine Fuel, *Electron. Code Fed. Regul.*, Dec. 2014.

19. Diesel Fuel | Fuels and Fuel Additives | US EPA, http://www.epa.gov/otaq/fuels/dieselfuels/, 2011.

20. Aakko, P., Nylund, N.-O., Westerholm, M., Marjamäki, M., Moisio, M., Hillamo, R., and Mäkelä, T., "Emissions from heavy-duty engine with and without aftertreatment using selected biofuels," *FISITA 2002 World automotive congress proceedings*, 2–7, 2002.

21. U.S. EPA Office of Transportation and Air Quality, "EPA Finalizes 2013 Biomass-Based Diesel Volume," Regulatory Announcement EPA-420-F-12-059, Environmental Protection Agency, U.S., 2012.

22. Davis, S.C., Diegel, S.W., Boundy, R.G., and Moore, S., "2013 Vehicle Technologies Market Report," ORNL/TM-2014/58.

23. Hesterberg, T.W., Long, C.M., Sax, S.N., Lapin, C.A., McClellan, R.O., Bunn, W.B., and Valberg, P.A., "Particulate matter in new technology diesel exhaust (NTDE) is quantitatively and qualitatively very different from that found in traditional diesel exhaust (TDE)," *J. Air Waste Manag. Assoc.* 61(9):894–913, 2011.

24. Find and Compare Cars, http://www.fueleconomy.gov/feg/findacar.shtml, Jun. 2014.

25. U.S. Diesel Car Registrations Increase By 24%, Hybrids Up 33%; Total Car Market Registrations Increase Just 2.7% Since 2010, http://www.dieselforum.org/index.cfm?furl=news/u-s-diesel-car-registrations-increase-by-24-hybrids-up-33-total-car-market-registrations-increase-just-2-7-since-2010, Jun. 2014.

26. Boehman, A.L., Song, J., and Alam, M., "Impact of Biodiesel Blending on Diesel Soot and the Regeneration of Particulate Filters," *Energy Fuels* 19(5):1857–1864, 2005, doi:10.1021/ef0500585.

27. Czerwinski, J., Zimmerli, Y., Neubert, T., Heitzer, A., and Kasper, M., "Injection, Combustion and (Nano) Particle Emissions of a Modern HD-Diesel Engine With GTL, RME & amp; ROR," 2007-01-2015, SAE International, Warrendale, PA, 2007.

28. Krahl, J., Munack, A., Schröder, O., Stein, H., Herbst, L., Kaufmann, A., and Bünger, J., "The Influence of Fuel Design on the Exhaust Gas Emissions and Health Effects," 2005-01-3772, SAE International, Warrendale, PA, 2005.

29. Li, H., Lea-Langton, A., Andrews, G.E., Thompson, M., and Musungo, C., "Comparison of Exhaust Emissions and Particulate Size Distribution for Diesel, Biodiesel and Cooking Oil from a Heavy Duty DI Diesel Engine," 2008-01-0076, SAE International, Warrendale, PA, 2008.

30. Tinsdale, M., Price, P., and Chen, R., "The Impact of Biodiesel on Particle Number, Size and Mass Emissions from a Euro4 Diesel Vehicle," 2010-01-0796, SAE International, Warrendale, PA, 2010.

31. Bünger, J., Krahl, J., Baum, K., Schröder, O., Müller, M., Westphal, G., Ruhnau, P., Schulz, T.G., and Hallier, E., "Cytotoxic and mutagenic effects, particle size and concentration analysis of diesel engine emissions using biodiesel and petrol diesel as fuel," *Arch. Toxicol.* 74(8):490–498, 2000.

32. Chen, Y.-C. and Wu, C.-H., "Emissions of submicron particles from a direct injection diesel engine by using biodiesel," *J. Environ. Sci. Health Part A Tox. Hazard. Subst. Environ. Eng.* 37(5):829–843, 2002.

33. Koo-Oshima, S., Hahn, N., and Gerpen, J. Van, "Comprehensive Health and Environmental Effects of Biodiesel as an Alternative Fuel," 1998.

34. Krahl, J., Bünger, J., Schröder, O., Munack, A., and Knothe, G., "Exhaust emissions and health effects of particulate matter from agricultural tractors operating on rapeseed oil methyl ester," *J. Am. Oil Chem. Soc.* 79(7):717–724, 2002, doi:10.1007/s11746-002-0548-9.

35. Hansen, K.F. and Jensen, M.G., "Chemical and Biological Characteristics of Exhaust Emissions from a DI Diesel Engine Fuelled With Rapeseed Oil Methyl Ester (RME)," 971689, SAE International, Warrendale, PA, 1997.

36. Bertola, A., Schubiger, R., Kasper, A., Matter, U., Forss, A.M., Mohr, M., Boulouchos, K., and Lutz, T., "Characterization of Diesel Particulate Emissions in Heavy-Duty DI-Diesel Engines with Common Rail Fuel Injection Influence of Injection Parameters and Fuel Composition," SAE Technical Paper 2001-01-3573, SAE International, Warrendale, PA, 2001.

37. Krahl, J., Munack, A., Schröder, O., Stein, H., and Bünger, J., "Influence of Biodiesel and Different Designed Diesel Fuels on the Exhaust Gas Emissions and Health Effects," 2003-01-3199, SAE International, Warrendale, PA, 2003.

38. Jung, H., Kittelson, D.B., and Zachariah, M.R., "Characteristics of SME Biodiesel-Fueled Diesel Particle Emissions and the Kinetics of Oxidation," *Environ. Sci. Technol.* 40(16):4949–4955, 2006, doi:10.1021/es0515452.

39. Trapel, E., Mayer, C., Schulz, C., and Roth, P., "Effects of Bio Diesel Injection in a DI Diesel Engine on Gaseous and Particulate Emission," SAE Technical Paper 2005-01-2204, SAE International, Warrendale, PA, 2005.

40. Aakko, P. and Nylund, N.-O., "Particle Emissions at Moderate and Cold Temperatures Using Different Fuels," SAE Technical Paper 2003-01-3285, SAE International, Warrendale, PA, 2003.

41. Boehman, A.L., Morris, D., Szybist, J., and Esen, E., "The Impact of the Bulk Modulus of Diesel Fuels on Fuel Injection Timing," *Energy Fuels* 18(6):1877–1882, 2004, doi:10.1021/ef049880j.

42. Szybist, J.P. and Boehman, A.L., "Behavior of a Diesel Injection System with Biodiesel Fuel," 2003-01-1039, SAE International, Warrendale, PA, 2003.

43. Rakopoulos, C.D., "Diesel engine transient operation: principles of operation and simulation analysis," Springer, London, ISBN 978-1-84882-374-7, 2009.

44. Mass-produced cleanliness: BERU Pressure Sensor Glow Plug reduces the emission of pollutants in the new VW 2.0 liter TDI | BorgWarner BERU Systems GmbH, http://www.beru.com/bw/mass-produced-cleanliness-beru-pressure-sensor-glow-plug-reduces-the-emission-of-pollutants-in-the-n, Jan. 2014.

45. Flynn, P.F., Durrett, R.P., Hunter, G.L., Loye, A.O. zur, Akinyemi, O.C., Dec, J.E., and Westbrook, C.K., "Diesel Combustion: An Integrated View Combining Laser Diagnostics, Chemical Kinetics, And Empirical Validation," SAE Technical Paper 1999-01-0509, SAE International, Warrendale, PA, 1999.

46. Lapuerta, M., Armas, O., and Rodriguezfernandez, J., "Effect of biodiesel fuels on diesel engine emissions," *Prog. Energy Combust. Sci.* 34(2):198–223, 2008, doi:10.1016/j.pecs.2007.07.001.

47. Abdul-Khalek, I.S., Kittelson, D.B., Graskow, B.R., Wei, Q., and Bear, F., "Diesel Exhaust Particle Size: Measurement Issues and Trends," 980525, SAE International, Warrendale, PA, 1998.

48. Abdul-Khalek, I., Kittelson, D., and Brear, F., "The Influence of Dilution Conditions on Diesel Exhaust Particle Size Distribution Measurements," 1999-01-1142, SAE International, Warrendale, PA, 1999.

49. Pomykala, M., Stuart, J.D., Noshadi, I., and Parnas, R.S., "The interplay of phase equilibria and chemical kinetics in a liquid/liquid multiphase biodiesel reactor," *Fuel* 107:623–632, 2013, doi:10.1016/j.fuel.2012.12.017.

50. Boucher, M.B., Unker, S.A., Hawley, K.R., Wilhite, B.A., Stuart, J.D., and Parnas, R.S., "Variables affecting homogeneous acid catalyst recoverability and reuse after esterification of concentrated omega-9 polyunsaturated fatty acids in vegetable oil triglycerides," *Green Chem.* 10(12):1331–1336, 2008, doi:10.1039/B810225B.

51. Grabner Instruments, IROX DIESEL Operation Manual M-V1.07 SW-V3.18, 2008.

52. Holmén, B.A., Robinson, M., Sentoff, K., Montane, P., and Hathaway, K., "The On-Board Tailpipe Emissions Measurement System (TOTEMS): Proof-of-Concept," 2009.

53. NHTSA, "Passenger Car and Light Truck Fleets Characteristics," http://www.nhtsa.gov/cars/rules/cafe/NewPassengerCarFleet.htm, 2013.

54. Holmén, B.A., Feralio, T., Dunshee, J., and Sentoff, K., "Tailpipe Emissions and Engine Performance of a Light-Duty Diesel Engine Operating on Petro- and Bio-

diesel Fuel Blends," TRC Report 14-008, The University of Vermont, Transportation Research Center, 2014.

55. Howell, D.C., "Statistical methods for psychology," 5th ed, Duxbury/Thomson Learning, Pacific Grove, CA, ISBN 0-534-37770-X, 2002.

56. Munack, A., Schroeder, O., Krahl, J., and Buenger, J., "Comparison of Relevant Exhaust Gas Emissions from Biodiesel and Fossil Diesel Fuel," 2001.

57. Giakoumis, E.G., Rakopoulos, C.D., Dimaratos, A.M., and Rakopoulos, D.C., "Exhaust emissions of diesel engines operating under transient conditions with biodiesel fuel blends," *Prog. Energy Combust. Sci.* 38(5):691–715, 2012, doi:10.1016/j.pecs.2012.05.002.

58. Lapuerta, M., Armas, O., and Ballesteros, R., "Diesel Particulate Emissions from Biofuels Derived from Spanish Vegetable Oils," SAE Technical Paper 2002-01-1657, SAE International, Warrendale, PA, 2002.

59. Wang, W.G., Lyons, D.W., Clark, N.N., Gautam, M., and Norton, P.M., "Emissions from Nine Heavy Trucks Fueled by Diesel and Biodiesel Blend without Engine Modification," *Environ. Sci. Technol.* 34(6):933–939, 2000, doi:10.1021/es981329b.

60. Tat, M.E. and Gerpen, J.H. Van, "Measurement of biodiesel speed of sound and its impact on injection timing," *Natl. Renew. Energy Lab. NRELSR-510-31462*, 2003.

61. Ramadhas, A.S., Muraleedharan, C., and Jayaraj, S., "Performance and emission evaluation of a diesel engine fueled with methyl esters of rubber seed oil," *Renew. Energy* 30(12):1789–1800, 2005, doi:10.1016/j.renene.2005.01.009.

62. Labeckas, G. and Slavinskas, S., "The effect of rapeseed oil methyl ester on direct injection Diesel engine performance and exhaust emissions," *Energy Convers. Manag.* 47(13–14):1954–1967, 2006, doi:10.1016/j.enconman.2005.09.003.

63. Barrios, C.C., Martín, C., Domínguez-Sáez, A., Álvarez, P., Pujadas, M., and Casanova, J., "Effects of the addition of oxygenated fuels as additives on combustion characteristics and particle number and size distribution emissions of a TDI diesel

engine," *Fuel* 132:93–100, 2014, doi:10.1016/j.fuel.2014.04.071.

64. Durbin, T.D., Collins, J.R., Norbeck, J.M., and Smith, M.R., "Effects of Biodiesel, Biodiesel Blends, and a Synthetic Diesel on Emissions from Light Heavy-Duty Diesel Vehicles," *Environ. Sci. Technol.* 34(3):349–355, 2000, doi:10.1021/es990543c.

65. Graboski, M.S. and McCormick, R.L., "Combustion of fat and vegetable oil derived fuels in diesel engines," *Prog. Energy Combust. Sci.* 24(2):125–164, 1998, doi:10.1016/S0360-1285(97)00034-8.

66. Knothe, G., Sharp, C.A., and Ryan, T.W., "Exhaust Emissions of Biodiesel, Petrodiesel, Neat Methyl Esters, and Alkanes in a New Technology Engine†," *Energy Fuels* 20(1):403–408, 2005, doi:10.1021/ef0502711.

67. Tsolakis, A., "Effects on Particle Size Distribution from the Diesel Engine Operating on RME-Biodiesel with EGR," *Energy Fuels* 20(4):1418–1424, 2006, doi:10.1021/ef050385c.

68. Graboski, M.S., McCormick, R.L., Alleman, T.L., and Herring, A.M., "Effect of Biodiesel Composition on Engine Emissions from a DDC Series 60 Diesel Engine: Final Report; Report 2 in a Series of 6," NREL/SR-510-31461, 15003583, 2003.

69. Karavalakis, G., Tzirakis, E., Zannikos, F., Stournas, S., Bakeas, E., Arapaki, N., and Spanos, A., "Diesel/Soy Methyl Ester Blends Emissions Profile from a Passenger Vehicle Operated on the European and the Athens Driving Cycles," SAE Technical Paper 2007-01-4043, SAE International, Warrendale, PA, 2007.

70. Schumacher, L.G., Marshall, W., Krahl, J., Wetherell, W.B., and Grabowski, M.S., "BIODIESEL EMISSIONS DATA FROM SERIES 60 DDC ENGINES," *Trans. ASAE* 44(6), 2001, doi:10.13031/2013.6999.

71. Tinaut, F., Melgar, A., Briceńo, Y., and Horrillo, A., "Performance of vegetable derived fuels in diesel engine vehicles," *PTNSS Silniki Spalinowe* (Nr 2/2005 (121)):56–69, 2005.

72. Neto da Silva, F., Salgado Prata, A., and Rocha Teixeira, J., "Technical feasibility assessment of oleic sunflower methyl ester utilisation in Diesel bus engines," *Energy Convers. Manag.* 44(18):2857–2878, 2003, doi:10.1016/S0196-8904(03)00067-0.

73. Giechaskiel, B., Dilara, P., Sandbach, E., and Andersson, J., "Particle measurement programme (PMP) light-duty inter-laboratory exercise: comparison of different particle number measurement systems," *Meas. Sci. Technol.* 19(9):095401, 2008, doi:10.1088/0957-0233/19/9/095401.

74. Lim, S.S., Vos, T., Flaxman, A.D., Danaei, G., Shibuya, K., Adair-Rohani, H., AlMazroa, M.A., Amann, M., Anderson, H.R., Andrews, K.G., Aryee, M., Atkinson, C., Bacchus, L.J., Bahalim, A.N., Balakrishnan, K., Balmes, J., Barker-Collo, S., Baxter, A., Bell, M.L., Blore, J.D., Blyth, F., Bonner, C., Borges, G., Bourne, R., Boussinesq, M., Brauer, M., Brooks, P., Bruce, N.G., Brunekreef, B., et al., "A comparative risk assessment of burden of disease and injury attributable to 67 risk factors and risk factor clusters in 21 regions, 1990–2010: a systematic analysis for the Global Burden of Disease Study 2010," *The Lancet* 380(9859):2224–2260, 2012, doi:10.1016/S0140-6736(12)61766-8.

75. Hoekman, S.K., Broch, A., Robbins, C., Ceniceros, E., and Natarajan, M., "Review of biodiesel composition, properties, and specifications," *Renew. Sustain. Energy Rev.* 16(1):143–169, 2012, doi:10.1016/j.rser.2011.07.143.

76. Lam, M.K., Lee, K.T., and Mohamed, A.R., "Homogeneous, heterogeneous and enzymatic catalysis for transesterification of high free fatty acid oil (waste cooking oil) to biodiesel: A review," *Biotechnol. Adv.* 28(4):500–518, 2010, doi:10.1016/j.biotechadv.2010.03.002.

77. Kulkarni, M.G. and Dalai, A.K., "Waste Cooking OilAn Economical Source for Biodiesel: A Review," *Ind. Eng. Chem. Res.* 45(9):2901–2913, 2006, doi:10.1021/ie0510526.

78. Surawski, N.C., Miljevic, B., Ayoko, G.A., Elbagir, S., Stevanovic, S., Fairfull-Smith, K.E., Bottle, S.E., and Ristovski, Z.D., "Physicochemical Characterization of Particulate Emissions from a Compression Ignition Engine: The Influence of Biodiesel Feedstock," *Environ. Sci. Technol.* 45(24):10337–10343, 2011, doi:10.1021/es2018797.

79. Kasumba, J., “Organic chemical characterization of primary and secondary biodiesel exhaust particulate matter,” Dissertation, The University of Vermont, Burlington, VT, 2015.

80. Mittelbach, M. and Enzelsberger, H., “Transesterification of heated rapeseed oil for extending diesel fuel,” *J. Am. Oil Chem. Soc.* 76(5):545–550, 1999, doi:10.1007/s11746-999-0002-x.

81. Rahman, M.M., Pourkhesalian, A.M., Jahirul, M.I., Stevanovic, S., Pham, P.X., Wang, H., Masri, A.R., Brown, R.J., and Ristovski, Z.D., “Particle emissions from biodiesels with different physical properties and chemical composition,” *Fuel* 134:201–208, 2014, doi:10.1016/j.fuel.2014.05.053.

82. Energy Independence and Security Act, http://energy.gov/eere/femp/energy-independence-and-security-act, May 2014.

83. Atkinson, C.M., Long, T.W., and Hanzevack, E.L., “Virtual Sensing: A Neural Network-based Intelligent Performance and Emissions Prediction System for On-Board Diagnostics and Engine Control,” 980516, SAE International, Warrendale, PA, 1998.

84. Arcaklioğlu, E. and Çelıkten, İ., “A diesel engine’s performance and exhaust emissions,” *Appl. Energy* 80(1):11–22, 2005, doi:10.1016/j.apenergy.2004.03.004.

85. Shivakumar, Srinivasa Pai, P., and Shrinivasa Rao, B.R., “Artificial Neural Network based prediction of performance and emission characteristics of a variable compression ratio CI engine using WCO as a biodiesel at different injection timings,” *Appl. Energy* 88(7):2344–2354, 2011, doi:10.1016/j.apenergy.2010.12.030.

86. Mohamed Ismail, H., Ng, H.K., Queck, C.W., and Gan, S., “Artificial neural networks modelling of engine-out responses for a light-duty diesel engine fuelled with biodiesel blends,” *Appl. Energy* 92:769–777, 2012, doi:10.1016/j.apenergy.2011.08.027.

87. Ghobadian, B., Rahimi, H., Nikbakht, A.M., Najafi, G., and Yusaf, T.F., “Diesel engine performance and exhaust emission analysis using waste cooking biodiesel fuel with an artificial neural network,” *Renew. Energy* 34(4):976–982, 2009,

doi:10.1016/j.renene.2008.08.008.

88. Thompson, G.J., Atkinson, C.M., Clark, N.N., Long, T.W., and Hanzevack, E., "Technical Note: Neural network modelling of the emissions and performance of a heavy-duty diesel engine," *Proc. Inst. Mech. Eng. Part J. Automob. Eng.* 214(2):111–126, 2000, doi:10.1177/095440700021400201.

89. Eiben, A.E. and Smith, J.E., "Introduction to evolutionary computing," Springer, New York, ISBN 3-540-40184-9, 2003.

90. Koza, J.R., "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," Stanford University, Stanford, CA, USA, 1990.

91. Viewing Model Results in Eureqa - Official User Guide | Nutonian, http://formulize.nutonian.com/documentation/eureqa/user-guide/view-results/, Jun. 2015.

92. Munack, A., Krahl, J., Speckmann, H., Marto, A., and Bantzhaff, R., "Development of a sensor for determination of biodiesel, fossil diesel fuel, and their blends," *V International Scientific Conference on Microprocessor Systems in Agriculture, Conference Proceedings, Płock, Poland*, 2004, 2004.

93. D02 Committee, "Specification for Biodiesel Fuel Blend Stock (B100) for Middle Distillate Fuels," ASTM International, 2015.

94. D02 Committee, "Specification for Diesel Fuel Oils," ASTM International, 2015.

95. D02 Committee, "Standard Specification for Diesel Fuel Oil, Biodiesel Blend (B6 to B20)," ASTM International, May 2015.

96. Çengel, Y.A., "Fundamentals of thermal-fluid sciences," McGraw-Hill, Boston, ISBN 0-07-239054-9, 2001.

97. TSI Inc., “Engine Exhaust Particle Sizer (EEPS) Spectrometer Model 3090/3090AK - Operation and Service Manual,” Revision J, 2015.

98. Dr.-Ing. Helmut Tschöke, “Diesel distributor fuel-injection pumps - Technical Instruction,” 4th edition, Robert Bosch GmbH, Germany, 1998.

APPENDICES

ASTM Fuel Testing

The University of Connecticut BioFuels Center for Environmental Sciences and Engineering performed ASTM testing on all but the first lot of petrodiesel (the petrodiesel blended with the WVO) and confirmed that the biodiesels tested conformed to the ASTM standards with the exception of cold soak filtration and combined sodium and potassium of the WVO B100. Dr. Parnas, the head of the University of Connecticut BioFuel Consortium, suggested that the quality of the wash water used to process the WVO B100 was likely the cause of the high sodium and potassium test results. A high content of sodium and potassium in biodiesel is an indicator of soaps in the fuel which can also lead to failures in the cold soak filtration test. The ASTM test results can be found in Table A1.

Table A1: Results of ASTM testing

Test Name	ASTM Test Method [A] Biodiesel / Petrodiesel	Limits Biodiesel / Petrodiesel (units)	WVO Measurement	SME Measurement	Petro Measurement[B]
Cloud Point	D2500	Report (°C)	-0.15	-1.2	NM
Cold Soak Filtration	D7501	360/NA MAX (sec)	2058 (F)	86 (P)	NA
Flash Point, closed cup	D93	93/38 MIN (°C)	175.6 (P)	167.4 (P)	45.6
Glycerin	-	-	-		-
1.Free Glycerin	D6584	0.020/NA MAX (% mass)	0.003 (P)	0.005 (P)	NA
2. total Glycerin	D6584	0.240/NA MAX (% mass)	0.049 (P)	0.072 (P)	NA
Oxidation Stability (110°C)	EN14112	3/NA MIN (hours)	11.49 (P)	6.28 (P)	NA
Total Acid Number (5g sample)	D664	0.50/NA MAX (mg KOH/g)	0.196 (P)	0.231 (P)	0.040 (P)
Water & Sediment	D2709	0.05/0.05 MAX (% vol)	0.00 (P)	0.010 (P)	0.000 (P)
Copper Strip Corrosion (3h/50°C)	D130	No. 3/ No. 3 MAX (rating)	1 A (P)	1 A (P)	1 A (P)
Kinematic Viscosity, 40°C	D445	1.9 - 6.0/1.7-4.1 (mm^2/sec)	4.354 (P)	4.166 (P)	NM
Sulfur; S 15 Grade	D4294/ **D5453**[C]	15/15 MAX (ppm)	2.5 (P)	<1.0 (P)	1.2 (P)
Phosphorus	D4951	0.001/NA (%)	<0.001 (P)	<0.001 (P)	NA
Calcium and Magnesium	EN14538	5/ NA (ppm), combined	0.02 (P)	<0.5 (P)	NA
Sodium and Potassium	EN14538	5/ NA (ppm), combined	30.1 (F)	<0.5 (P)	NA
Methanol %	EN 14110	0.2/ NA (% mass)	0.001 (P)	NM	NA
Distillation Temperature @ 90%	D1160/ D86	360/ 338 MAX (°C)	348.9 (P)	345.3	NM
Carbon residue (100%)	D4530/ **D524**[C]	0.050/ 0.15 MAX (% mass)	0.050 (P)	0.033 (P)	0.003 (P)
Cetane Number	D613	47/ 40 min.	NM	49.9 (P)	46.7 (P)

[A] ASTM - 6751-11b referenced for biodiesel and ASTM-975-11b referenced for petrodiesel.

[B] Measurements are from the lot of petrodiesel used for blending with the SME biodiesel which was purchased in MAR 2013. The lot of petrodiesel used for blending with the WVO biodeisel was purchased from the same supplier in JAN 2013. NM = not measured. NA = not applicable to ASTM-975-11b

[C] ASTM-975-11b calls for a different test method than ASTM-6751-11b, however, the test was performed using the test method deliniated in ASTM-6751-11b.

Antioxidant Data Sheet

Naugalube® 403

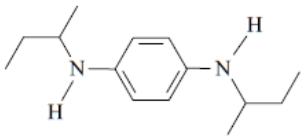
N,N'-di-sec-butyl-*p*-phenylene diamine

Issued: 1/2003
Rev. 1: 1/2010

DESCRIPTION

Naugalube® 403 provides excellent oxidation stability in a wide range of petroleum products, and is especially effective as an antioxidant in biodiesel. **Naugalube® 403** meets ASTM D 6751 & EN14214 biodiesel requirements, and is listed on the AGQM no harms list. It can also be used in petroleum diesel, motor gasoline (ASTM D 4814) and vegetable oils.
Naugalube® 403 as a lube antioxidant has excellent free radical and sulfur scavenging in a wide variety of lubricant base stocks including mineral, hydro-treated, synthetic, and vegetable oils.

N,N'-di-sec-butyl-p-phenylene diamine



TYPICAL BIODIESEL TREAT RATE MEETS EN14214 (6 Hr. Rancimat)		
	Soy-FAME (SME)	**400 ppm (w/w)**
	Rapeseed-FAME (RME)	**200 ppm (w/w)**
	Tallow- FAME	**25 ppm (w/w)**
	Palm-FAME	**10 ppm (w/w)**
	Used Soy Cooking Oil – FAME	**1,000 ppm (w/w)**

TYPICAL DATA

Properties	**Typical**
Appearance	Dark Red liquid
Specific Gravity (60/60°F)	0.94
Pounds per gallon	7.82
Viscosity	
Brookfield, cP @ 25°C	20
Kinematic, cSt @ 40°C	14.2
Flash Point (PMCC)	140°C

For more technical details please contact: Frank DeBlase Ph.D. (203-573-2795)

HANDLING INFORMATION

Contact of **Naugalube® 403** with skin and eyes may cause irritation. Avoid all personal contact. Observe good personal hygiene.

SAFETY INFORMATION

For more extensive information on the safe handling and use of this product, see the Material Safety Data Sheet.

SHIPPING INFORMATION

Tank cars, tank trucks, Isotanks, IBC and non-returnable 55 gallon drums.

Chemtura Corporation
Petroleum Additives
199 Benson Road
Middlebury, CT 06749

Technical Service 800.336.9315
Customer Service 800.325.6252
Fax 203.573.2324
www.chemtura.com

Fuel Blending and Analysis

To ensure accuracy in blending, the density of each parent fuel (B0 and B100) was measured both physically and with a density meter. The masses associated with the correct volume of B0 and B100 needed for blending were calculated and subsequently measured using a laboratory scale. The B00 and B100 were then combined in a tank, mechanically mixed, and finally sealed in fuel containers (UN certified 5 gallon buckets from Letica Corp. with unvented lids) with nitrogen headspace to minimize fuel oxidation during storage. The fuel was stored in an environmental chamber at 13°C to simulate underground storage.

Blend ratios (vol % biodiesel) were confirmed using an IROX Diesel (IROX-D) Analyzer from Grabner Instruments (Vienna, Austria), a mid-FTIR analyzer dedicated to diesel analysis with a built in high accuracy density meter [51] (Table A2). The FTIR based measurements from the IROX-D, however, were only accurate for B0 – B40 biodiesel blends, therefore direct measurements of fuel blends based on FTIR methods were only accurate for the B0, B10, and B20 blends tested here. To verify the blend ratio of the B50 used, the ‘as blended’ samples were diluted with hexane. The resulting IROX-D BXX measurements were then used along with the known dilution ratio to back calculate the ‘as blended’ BXX values.

The IROX-D results for all ‘as blended’ samples (WVO and SOY B0, B10, and B20) were within 0.5% of the expected value. The back calculated BXX value for the WVO and SOY B50 samples were within 5.6% of the expected value (Table A2).

Table A2: IROX-D test results for WVO and SOY sequences.

Feedstock	Blend	Density (g/cm^3)	BXX (Vol%)
WVO	B0	0.811	0
	B10	0.817	9.8
	B20	0.824	19.9
	B50	0.843	49.2*
	B100	0.876	100**
SOY	B0	0.81	0
	B10	0.816	10
	B20	0.822	19.5
	B50	0.842	44.4*
	B100	0.874	100**

* Values were back calculated from the IROX-D measurements of B50 samples diluted with hexane.

**Not measured

Drive Cycle Control

The ideal engine operation control for testing such as that performed here is through specification of engine torque and engine speed as a function of time. Due to ArmSoft control software limitations, however, the control of the engine during the transient portion of the cycle was accomplished by specifying throttle position and dynamometer voltage supply, a surrogate for dynamometer load. For the steady-state portions of the cycle, a proportional-integral-derivative (PID) controller automatically adjusted the dynamometer supply voltage to maintain a set point engine speed while throttle position was held constant.

Engine Oil

The only engine oil used was Castrol® Edge® with SPT (formerly called Castrol® Syntec®) SAE 5W-40 which is specifically formulated to meet or exceed

Volkswagen engine oil specifications VW 501 01, VW 502 00, and VW 505 00 which are the engine oil specifications listed in the VW workshop manual for this engine. The engine oil was changed prior to both the WVO and SOY data collection sequences.

Dilution System

The components of the dilution system are listed in Table A3 and are numbered corresponding to the numbered items in Figure A1, a schematic of the dilution system.

Table A3: Dilution system components. Numbers correspond to those in Figure A1

1	Compressor One	13	Critical Orifice (Flow Control)
2	Compressor Two	14	Orifice Flow Meter
3	Pressure Switch	15	Dekati Diluter
4	Coarse Dilution Air Pressure Regulator	16	Dilution Air Thermocouple
5	Condensation Drain Valve	17	Orifice Flow Meter
6	Condenser/ Expansion Tank in Ice Bath	18	Raw Exhaust Sample Thermocouple
7	Precision Air Pressure Regulator	19	Pinhole Orifice (Flow Control)
8	Silica Gel and Activated Carbon	20	Heat Cord (represented by red dots)
9	HEPA Filter	21	Perforated Sampling Probe
10	OMEGA Mass Air Flow Meter	22	Exhaust Temperature Thermocouple
11	Ice Bath	23	Exhaust Pipe Pitot Tube Flow Meter
12	Pressure Sensor		

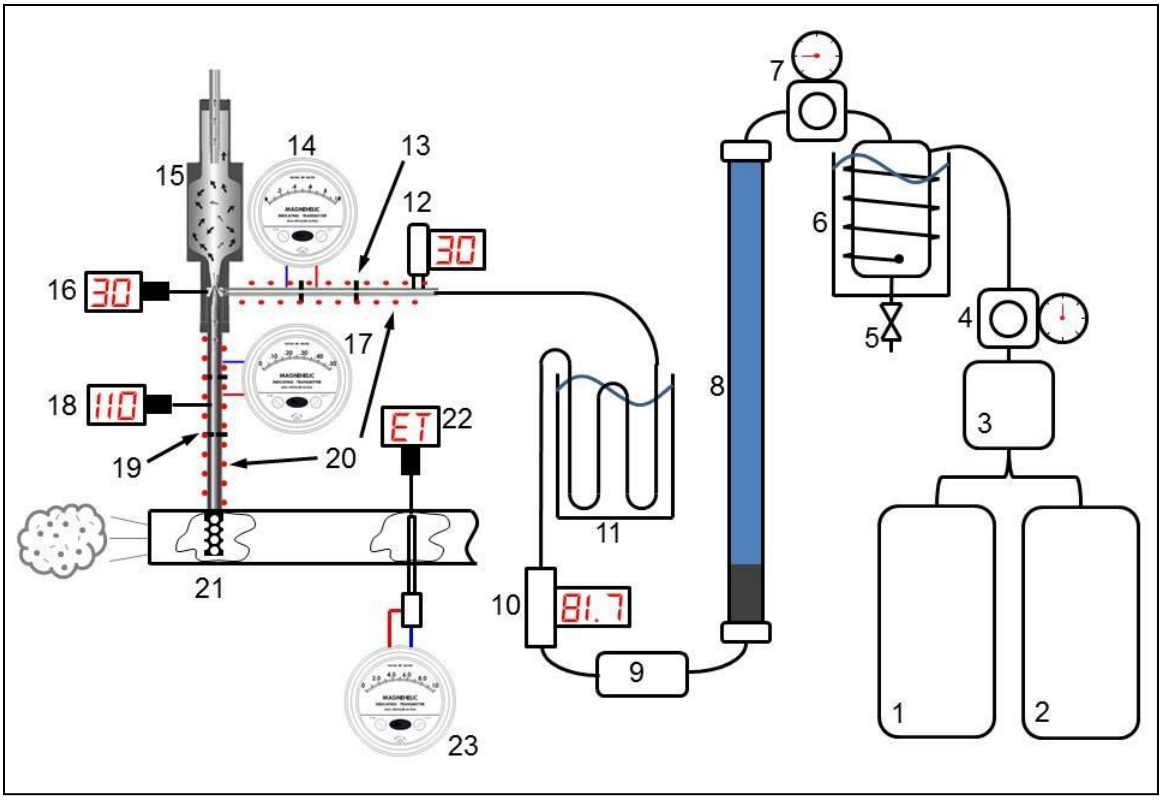


Figure A1: Engine Exhaust Dilution System Schematic

Absolute Humidity Calculation

Specific humidity (SH; mass water/mass air) was calculated on a second-by-second basis by plugging the ambient temperature and relative humidity measured in LabView and the atmospheric pressure measured by the scantool into EQ 4. EQ 4 was developed from EQ 1, 2, and 3 where EQ 3 is the Clausius-Clapeyron equation. EQ 5 was then developed from the relationship between air density, temperature, and pressure found in the appendix of Çengel et al. [96] and used to calculate absolute humidity (ABSH; mass water/ volume air) from SH using EQ 6 which contains the necessary unit conversions.

$$SH = 0.622\frac{e}{P} \quad [1]$$

Where:
SH = Specific Humidity (mass/mass; unitless)
e = vapor pressure (must be the same pressure unit as P)
P = Atmospheric Pressure (must be the same pressure unit as e)
(0.622 is a unitless constant)

$$e = (RH/100)e_{sat}(T) \quad [2]$$

Where:
e = vapor pressure (must be the same pressure unit as $e_{sat}(T)$)
RH = relative humidity
$e_{sat}(T)$ = saturation vapor pressure at temperature T
(must be the same pressure unit as e)

$$e_{sat}(T) = (6.11mbar)e^{\left\{5.423x10^3\left(\frac{1}{273.15K}-\frac{1}{T}\right)\right\}} \quad [3]$$

Where:
$e_{sat}(T)$ = saturation vapor pressure at temperature T (mbar)
T = temperature (K)

$$SH = \frac{0.622\left((RH/100)\left(6.11e^{\left\{5.423x10^3\left(\frac{1}{273.15K}-\frac{1}{T}\right)\right\}}\right)\right)}{P} \quad [4]$$

$$\rho_{AIR} = 355.42486(T)^{-1.00068}(P) \quad [5]$$

Where:
ρ_{AIR} = air density (kg/m^3)
T = temperature (K)
P = pressure (atm)

$$ABSH = SH\left(355.42486(T)^{-1.00068}(P/1013.25)\right)x\frac{1000g}{kg}x\frac{0.001m^3}{L} \quad [6]$$

Where:
$ABSH$ = absolute humidity (g/L)

Engine Exhaust Particle Sizer Bin Data

Table A4: Engine Exhaust Particle Sizer Bin Designations [97]

Bin Number	Bin Min D_p (nm)	Bin Midpoint D_p (nm)	Bin Max D_p (nm)
B1	5.61	6.04	6.48
B2	6.48	6.98	7.48
B3	7.48	8.06	8.64
B4	8.64	9.31	9.98
B5	9.98	10.75	11.52
B6	11.52	12.41	13.3
B7	13.3	14.33	15.36
B8	15.36	16.55	17.74
B9	17.74	19.11	20.48
B10	20.48	22.07	23.65
B11	23.65	25.48	27.31
B12	27.31	29.43	31.54
B13	31.54	33.98	36.42
B14	36.42	39.24	42.06
B15	42.06	45.32	48.57
B16	48.57	52.33	56.09
B17	56.09	60.43	64.77
B18	64.77	69.78	74.79
B19	74.79	80.58	86.37
B20	86.37	93.06	99.74
B21	99.74	107.46	115.18
B22	115.18	124.09	133
B23	133	143.3	153.59
B24	153.59	165.48	177.37
B25	177.37	191.1	204.82
B26	204.82	220.67	236.52
B27	236.52	254.83	273.13
B28	273.13	294.27	315.41
B29	315.41	339.82	364.23
B30	364.23	392.42	420.61
B31	420.61	453.16	485.71
B32	485.71	523.3	560.89

Temporal Alignment

Although the clocks on the computers used for data collection were synchronized, the time stamps associated with each instrument were not aligned perfectly. The raw data from Scantool, Armfield, Labview, and EEPS were aligned as follows.

First, because Scantool data were logged at a variable 'as fast as possible' rate, they were interpolated to the same frequency as the Armfield data, logged at 2Hz, using the Matlab function 'Interp1' and the 'linear' method. Since the data logged via the Armfield software did not have timestamps, the Armfield data were aligned to the Scantool data. This was done by performing Pearson's correlations between the throttle position data recorded via Scantool to throttle position data recorded via Armfield that had been shifted in intervals of one time step from $-t$ seconds to $+t$ seconds (where t is large enough to obtain a maximum correlation coefficient). The lag associated with the highest correlation coefficient was recorded and subsequently applied as the Armfield instrument offset. Once aligned, the Scantool and Armfield data were interpolated to the frequency of the data collected from the remaining instruments (1Hz).

Next, the Labview data were aligned with the Armfield/ Scantool data by correlating multiple time shifts of the Labview mass air flow (MAF) parameter to the Armfield intake air pressure; both measures of intake air flow. Again, the time offset associated with the highest correlation coefficient was selected and applied to time-align the Labview data set with Armfield/Scantool.

Because it took the exhaust sample a finite amount of time to travel from the sample port in the exhaust pipe, through the dilution system, and into the EEPS where it was measured, the EEPS data lagged the engine data. To account for this, Pearson's

correlations were run between the engine RPM and multiple versions of the EEPS TPN data shifted by consecutive time steps. In theory, the shifted data set with the highest correlation would be considered to have the correct lag adjustment. Engine speed, however, is more dynamic than TPN causing the RPM peaks to precede TPN peaks (TPN can still be increasing when RPM is decreasing). Because the Pearson's correlation improves as the TPN peaks center on the RPM peaks, the data set with the highest correlation actually showed TPN response prior to engine speed change. In order to ensure that the data reflected correct TPN response to changes in engine speed, engine start and engine stop RPM and TPN time series data for the individual runs were overlaid. The observed TPN lag relative to RPM was measured for these events and averaged for each run. The EEPS data set was then moved forward in time by the average of the observed start and stop lags to align the data of the associated run.

Engine Exhaust Particle Sizer Data Post Processing

Engine Exhaust Particle Sizer Blank Verification

To ensure consistency in EEPS operation and data collection, the following procedure was adhered to for each run. 1) While the EEPS was not in use a HEPA filter was attached to the sample inlet to avoid any contamination. 2) Prior to each run, with the HEPA filter attached, the EEPS electrometers were zeroed and the resulting offsets were verified to be within TSI specifications. 3) Ten minutes of EEPS data were recorded with the HEPA filter in place to provide an instrument blank. 4) With the HEPA filter removed, and with the sample line connected to the dilute exhaust sampling port, ten minutes of EEPS data were collected without the engine running to provide tunnel blank

data. Additionally, at the end of each run, post tunnel and instrument blanks were recorded in a similar fashion.

Analysis of the average particle concentration for the pre-instrument blanks collected before each run indicated that, throughout both the WVO and SOY sequences, the operation of the EEPS was consistent and that average measurements from each EEPS bin were below the minimum detection limit defined by TSI (Figure A2A). In other words, the measurements taken during the instrument blanks were lower than the expected noise for each EEPS bin. The pre-tunnel blank data, however, show that background particle concentrations were elevated for the WVO sequence relative to the SOY sequence (Figure A2B). The difference in background concentration was accounted for by correcting the EEPS data with the pre-tunnel blank data during data post processing.

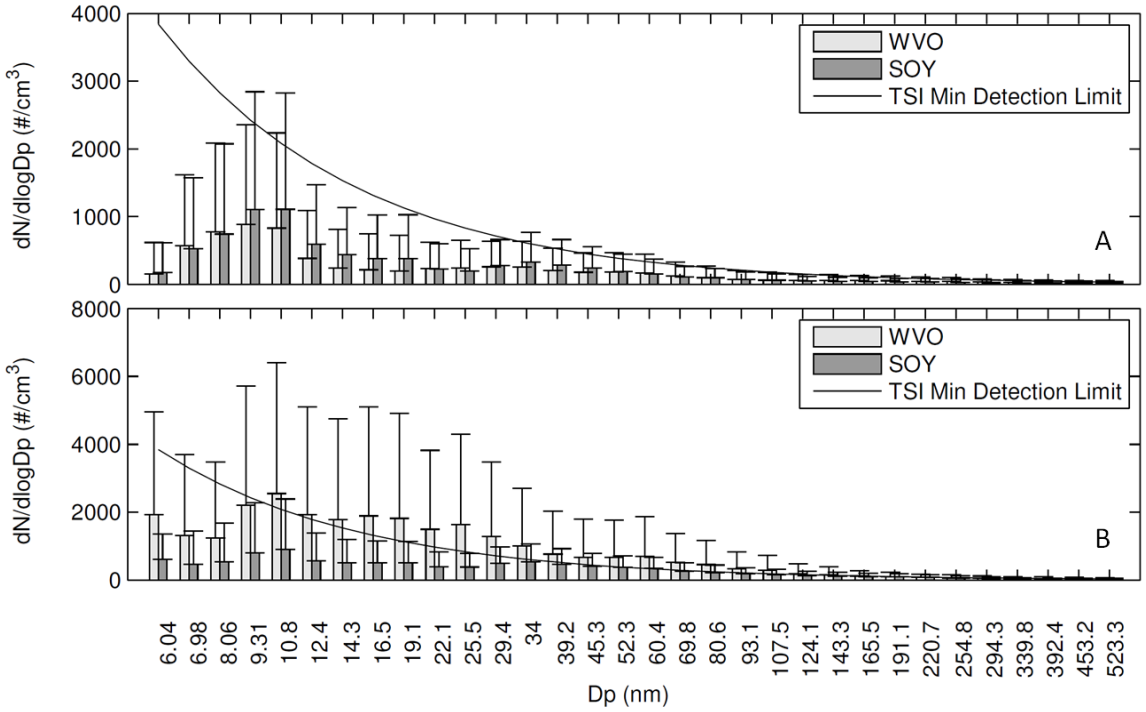


Figure A2: A) Average instrument blank measurements by feedstock. B) Average tunnel blank measurements by feedstock. Error bars represent + 1 StDev

PN Data and Blank Correction

Pre-tunnel blank data were used to background correct the engine test data for each engine test run, i, for each EEPS size bin, j using Equation [7].

$$C_{back(i,j)} = Ave\left(C_{TB(i.j)}\right) + 3 \times StDev\left(C_{TB(i.j)}\right) \quad [7]$$

Where:

$C_{back(i,j)}$ = background correction for run *i* EEPS bin *j*. (#/cm^3)

$C_{TB(i,j)}$ = tunnel blank particle concentration for run *i* EEPS bin *j*. (#/cm^3)

The average particle concentration of the tunnel blank represents the concentration of particles in the ambient air while the standard deviation of the tunnel blank represents the noise in the EEPS signal. Subtracting the run (i) bin (j) background concentration from the appropriate EEPS size bin (j) accounted for the concentration of particles in the ambient air. If the result of the subtraction was less than the correction

factor for that bin, then the concentration was set to the correction factor for that bin to provide a conservative particle concentration measure.

Raw Exhaust PN Concentration Calculation (DR Correction)

Tailpipe exhaust PN concentration was calculated by multiplying the background corrected EEPS concentration by the dilution ratio which was calculated on a second-by-second basis based with second-by-second diluter inlet flow measurements:

$$DR = \frac{Q_{dil,in} + Q_{exh,in}}{Q_{exh,in}} \quad [8]$$

Where:
DR = dilution ratio
$Q_{dil,in}$ = dilution air inlet flow
$Q_{exh,in}$ = exhaust sample inlet flow

Second-by-second diluter inlet flow rates were measured with custom inline orifice flow meters which consisted of Dwyer 605 transmitting Magnehelics® measuring the pressure difference across inline orifices. Given that the temperature and pressure of the dilution air and exhaust sample inlet gases were controlled, the recorded data are measures of mass flow rate.

Emission Rate Calculation

PN emission rate (ER; #/s) was computed by multiplying the DR-corrected PN concentration (#/cm^3) by exhaust flow rate (L/min) (EQ 9).

$$ER_P = PN_{CON} \times Q_{exh} \times \frac{1\text{min}}{60\text{sec}} \times \frac{1000cm^3}{1L} \quad [9]$$

Where:
ER_P = particle emission rate (#/sec)
PN_{CON} = DR corrected particle number concentration (#/cm^3)
Q_{exh} = Volumetric exhaust flowrate (LPM)

Exhaust Flow Rate Estimation Using Mass Air Flow (MAF)

Although exhaust flow rate was directly measured with a pitot tube (Dwyer DS-300 pitot tube connected to a Dwyer 605 transmitting Magnehelic®) in the tailpipe, the pressure pulses in the exhaust of the naturally aspirated engine generated significant noise in the pitot data. To provide a more consistent indication of exhaust volumetric flow rate, it was modeled as follows using mass air flow (MAF; VW/Bosch 037 906 461C/ 0 280 217 117) data measured in the air intake of the engine and temperature data measured at the pitot tube in the exhaust pipe close to the sample port.

$$\frac{(P_1)(V_1)}{T_1} = \frac{(P_2)(V_2)}{T_2} \quad [10]$$

Where:

P = pressure
V = volume
T = Temperature
Subscript refers to location, here 1 = intake and 2 = exhaust

Standard conditions:

T = 293.15 Kelvin
Pressure = 101325 Pa

Assumptions:

1. Pressure at the intake is the same as the pressure in the exhaust since they are both open to the atmosphere
2. The ambient conditions being ‘seen’ by the MAF sensor were very close to standard conditions (293.15K and 101325Pa) making this measure of SLPM (a mass flow rate) very close to LPM (a volumetric flow rate)
3. The same mass of air that is going into the engine is coming out, therefore, calculating a change in volume from one location to another is sufficient for estimating a change in volumetric flow rate.

$$\frac{V_1}{T_1} = \frac{V_2}{T_2} \quad [11]$$

Rearrange:

$$V_2 = \frac{(V_1)(T_2)}{T_1} = \frac{(V_1)(T_2)}{293.15} \quad [12]$$

Where:

V_2 = Volume of air in the exhaust
V_1 = Volume of air in the intake
T_2 = temperature at the pitot tube in the exhaust (close to the sample port)

The equation can then be rewritten and applied to volumetric flowrates:

$$Q_{exh} = \frac{(Q_{\text{int}})(T_{exh})}{293.15} \quad [13]$$

Where:

Q_{exh} = Volumetric exhaust flowrate
Q_{int} = MAF (assumed to be approximately volumetric flowrate at ambient conditions)
T_{exh} = Exhaust temperature at the pitot tube (close to the sample port)

The Pitot tube was calibrated at room temperature with the Sierra flow meter from 0 to 4000LPM. A pitot tube functions based on the differential between total pressure (the pressure developed due to fluid velocity over a specific cross sectional area) and static pressure (the pressure of the environment without taking into account fluid velocity. For this reason, it measures volumetric flow rate and has a relatively low sensitivity to fluid temperature therefore, it does not need to be temperature corrected to account for the difference between calibration conditions and 'in use' conditions.

Figure A3 shows the comparison of the estimated Exhaust flow rate (using the MAF and exhaust temperature) and the pitot tube exhaust flow rate measure.

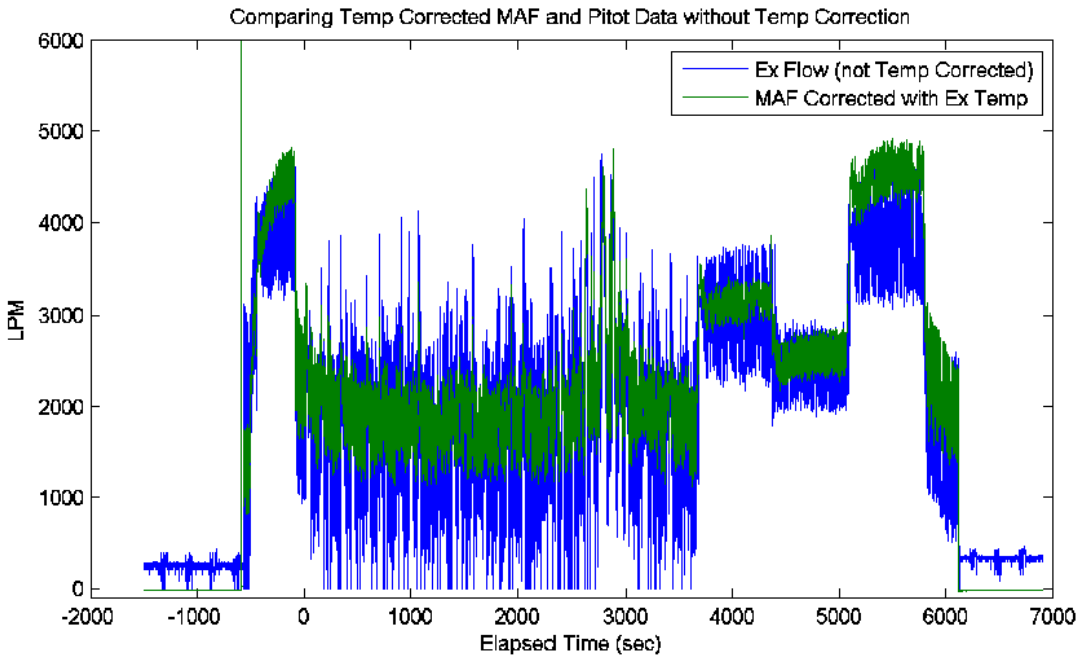


Figure A3: Pitot flow overlaid with temperature corrected MAF

A scatter plot was generated in Excel between estimated exhaust flow rate (using MAF and exhaust temp) and exhaust flow rate measured with the Pitot tube. The resulting linear regression equation was y=0.9503x-256.47. The fact that the slope is so close to 1 suggests that the estimated exhaust flow rate is close to actual exhaust flow rate and can be used as a surrogate for the direct measurement of exhaust flow.

Percent Load Calculation

The torque curve provided by Volkswagen for the SDi engine used for this research and the engine torque and engine speed measured by Armsoft were used to calculate % load. The torque curve provided by VW was interpolated into a piecewise function so that 'maximum torque' could be calculated across the RPM range (Figure A4).

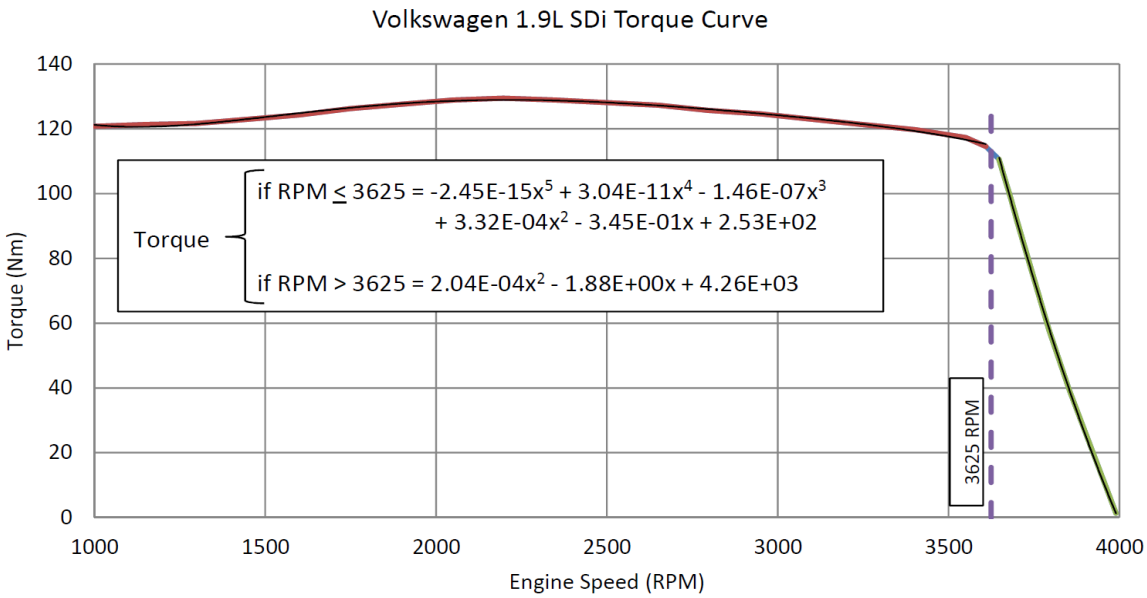


Figure A4: Volkswagen 1.9L SDi Torque Curve with polynomial trend lines

Percent load was then calculated by plugging the measured engine torque and the maximum calculated torque for the associated engine speed into EQ 14.

$$\%Laod = \frac{\tau_{act}}{\tau_{\max}} x100 \qquad [14]$$

Where:

τ_{act} = Measured engine torque (Nm)

τ_{max} = Maximum calculated engine torque (Nm)

Determining Consistency of Operation

Cumulative distribution functions (CDFs) were generated for torque, engine speed, and throttle position for all transient cycle runs of each fuel blend (n=30) (Figure A5) to determine if the engine operation was consistent across all runs. Two sample Kolmogorov–Smirnov (KS) test were then run on all possible combinations of the 30 CDFs. All KS tests failed to reject the null hypothesis that any combination of two of the 30 curves depicted were from the same continuous distribution.

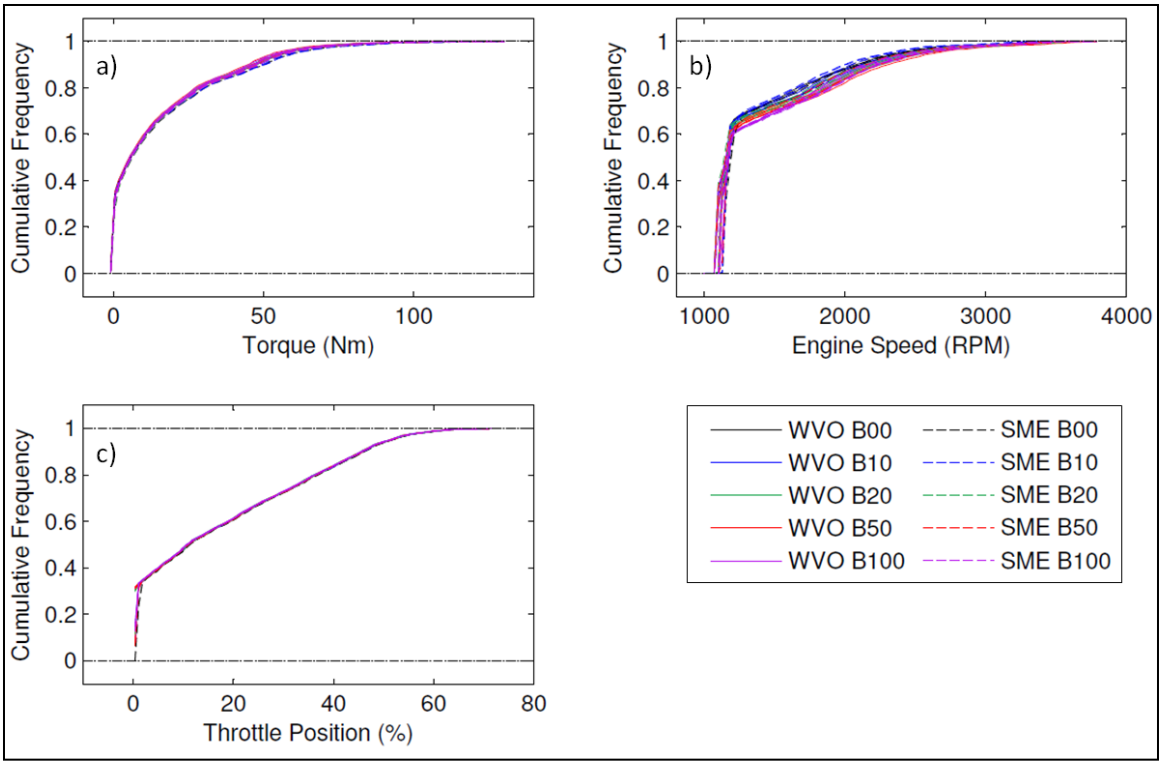


Figure A5: Overlaid cumulative distribution functions from all 30 runs for a) engine torque, b) engine speed, and c) throttle position

Dilution Ratio

The 1Hz dilution ratio data box plots are shown in Figure A6 for each individual run. There is a trend of increasing DR from run to run for the WVO sequence and for the first portion of the SOY sequence due to fouling of the orifice in the exhaust sample transfer line. For fear of altering dilution system characteristics, the dilution system was not disassembled for cleaning during the WVO sequence. Fouling during the SOY sequence was, however, more aggressive resulting in disassembly of the dilution system for cleaning before the 3rd SOY B20 replicate was run. Although there is inter-run DR variability, intra-run variability is relatively consistent. High DR outliers are present for most runs because there was, inevitably, some fouling of the control orifice during the

end of the drive cycle. Because all PN data was corrected with second-by-second DR, the effect of the DR variation is assumed to be minimal.

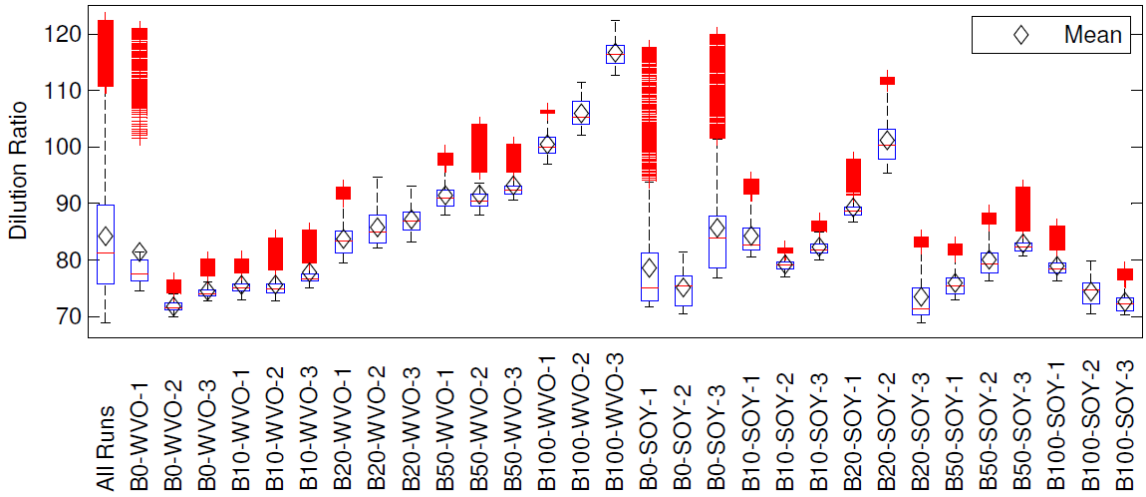


Figure A6: Dilution Ratio by Run

Ambient Conditions

Ambient conditions (temperature, pressure, and absolute humidity) were variable throughout testing (Figure A7). They were all, however, within normal atmospheric conditions so any affects they may have had were indicative of real world variability.

Ambient pressure measured by the scantool had a resolution of approximately 5mbar resulting in the majority of the raw data points consisting of up to about three values. The reasons for the outliers shown in the ambient pressure plot in Figure A7 are 1) one pressure value was primarily read and 2) interpolation of the raw data to common time stamps during data post processing resulted in values between those in the raw data.

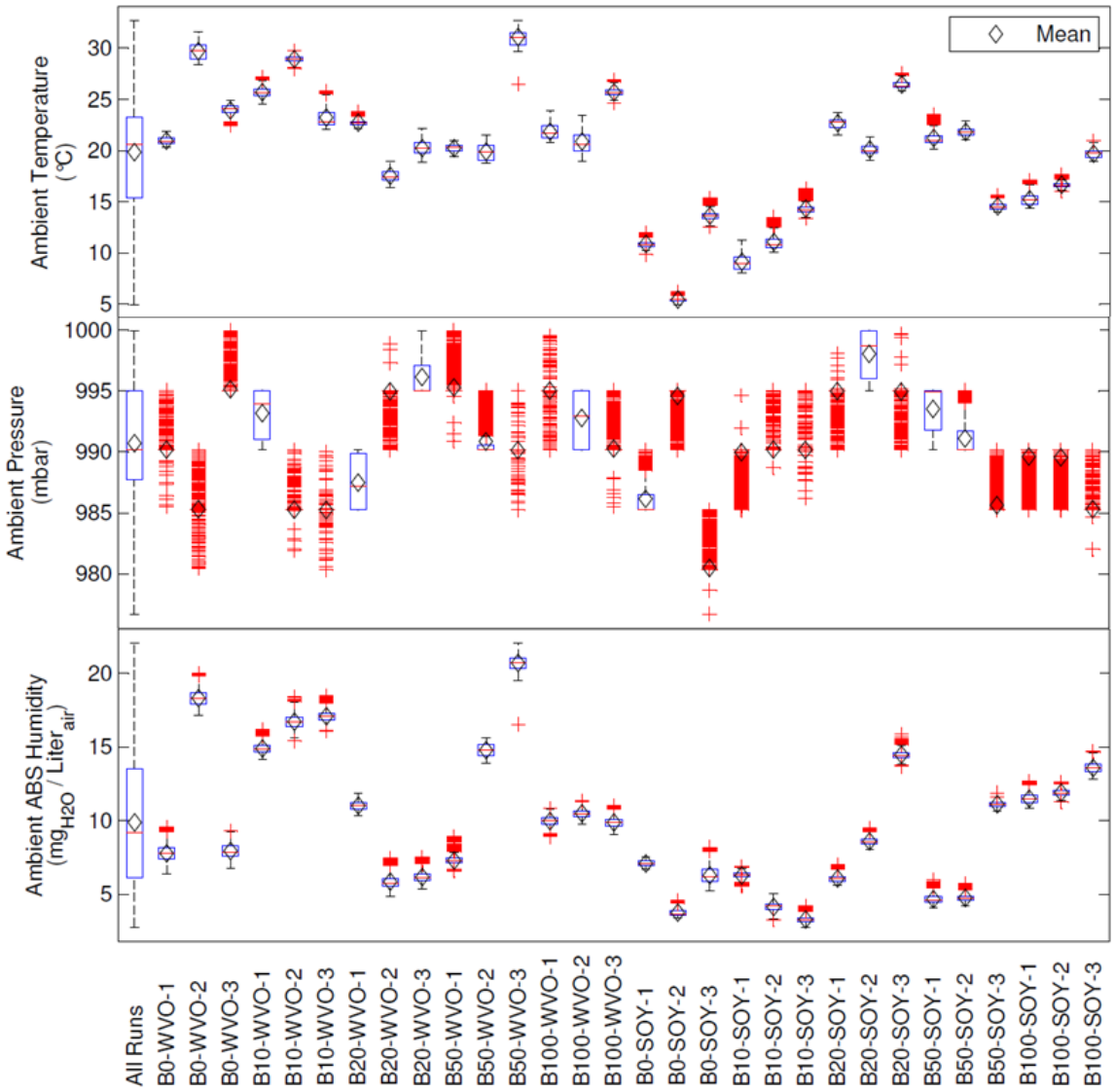


Figure A7: Ambient condition box plots by run

To investigate trends between ambient conditions (temperature, pressure, and absolute humidity) and TUFP emissions, scatter plots of each parameter versus TUFP emissions by BXX blend and feedstock were generated (Figure A8 – Figure A13). Additionally, the data in each scatter plot was used to generate a linear regression between the ambient condition parameter and TUFP emissions (blue line). These linear regressions indicate that the majority of the data suggest that TUFP ERs increase with an

increase in ambient pressure, a decrease in ambient temperature, or a decrease in ambient ABS humidity. Some of these data, however, suggest the opposite; therefore, no definitive trend between the ambient parameters and TUFP emissions could be found with this limited data set.

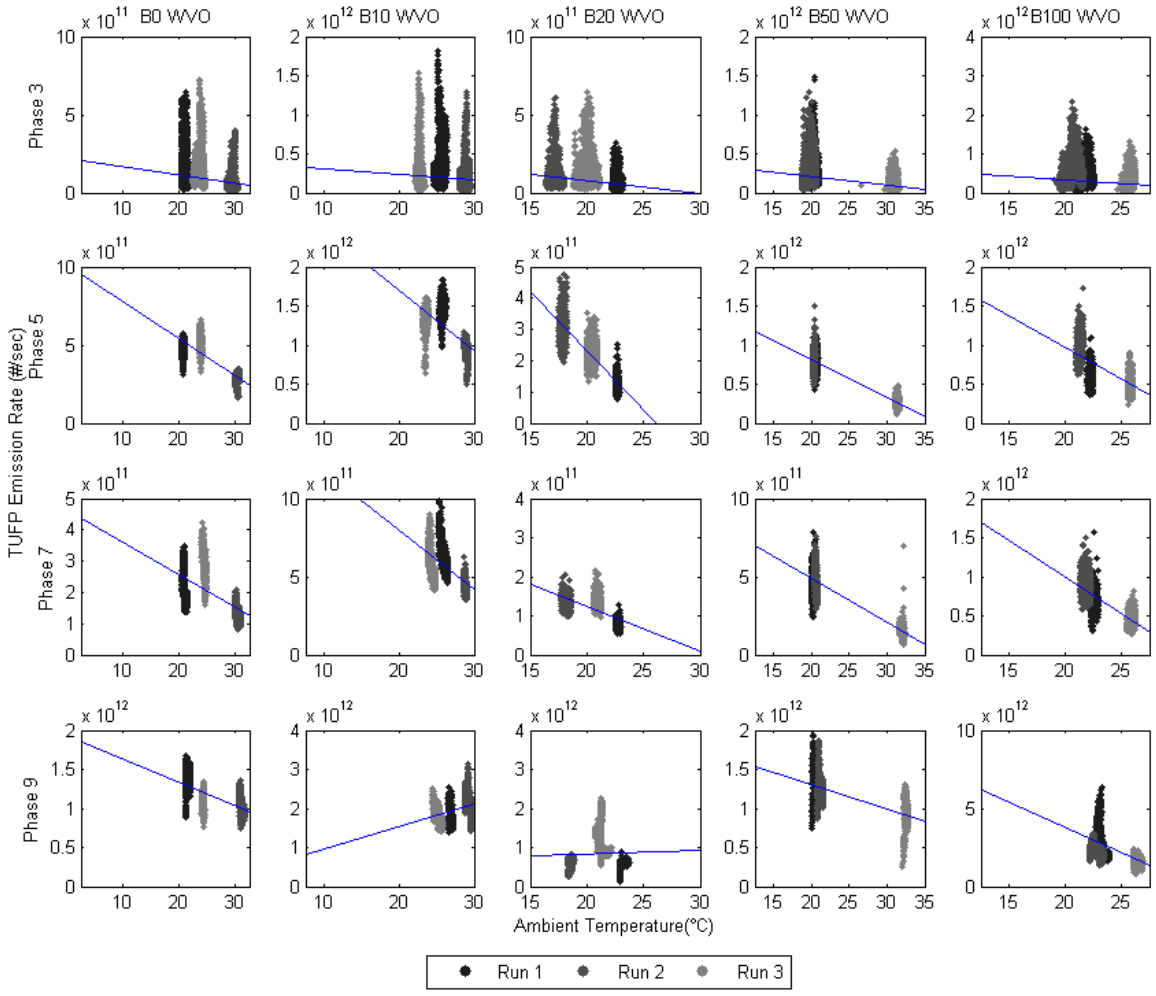


Figure A8: Scatter plots by WVO blend and phase of ambient temperature versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

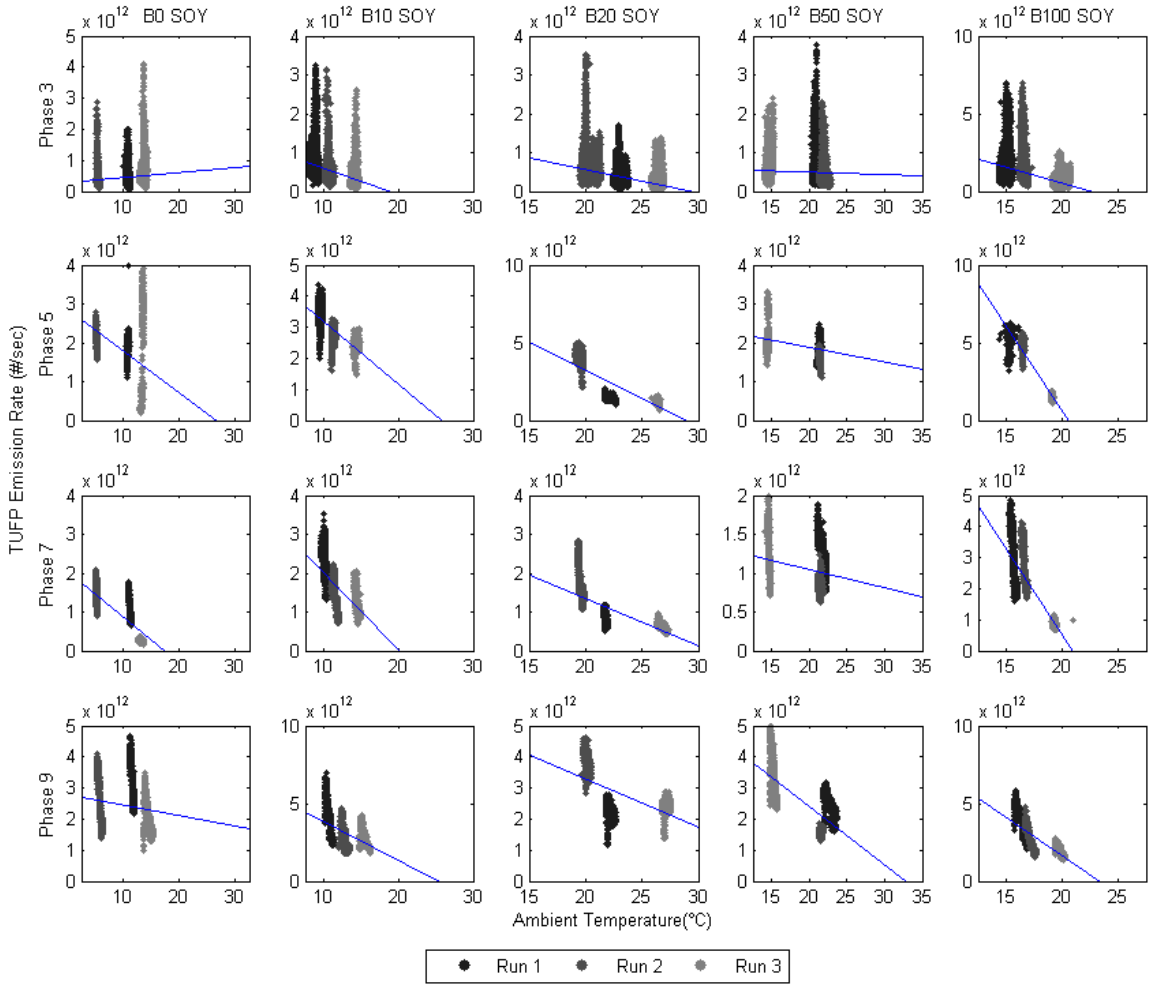


Figure A9: Scatter plots by SOY blend and phase of ambient temperature versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

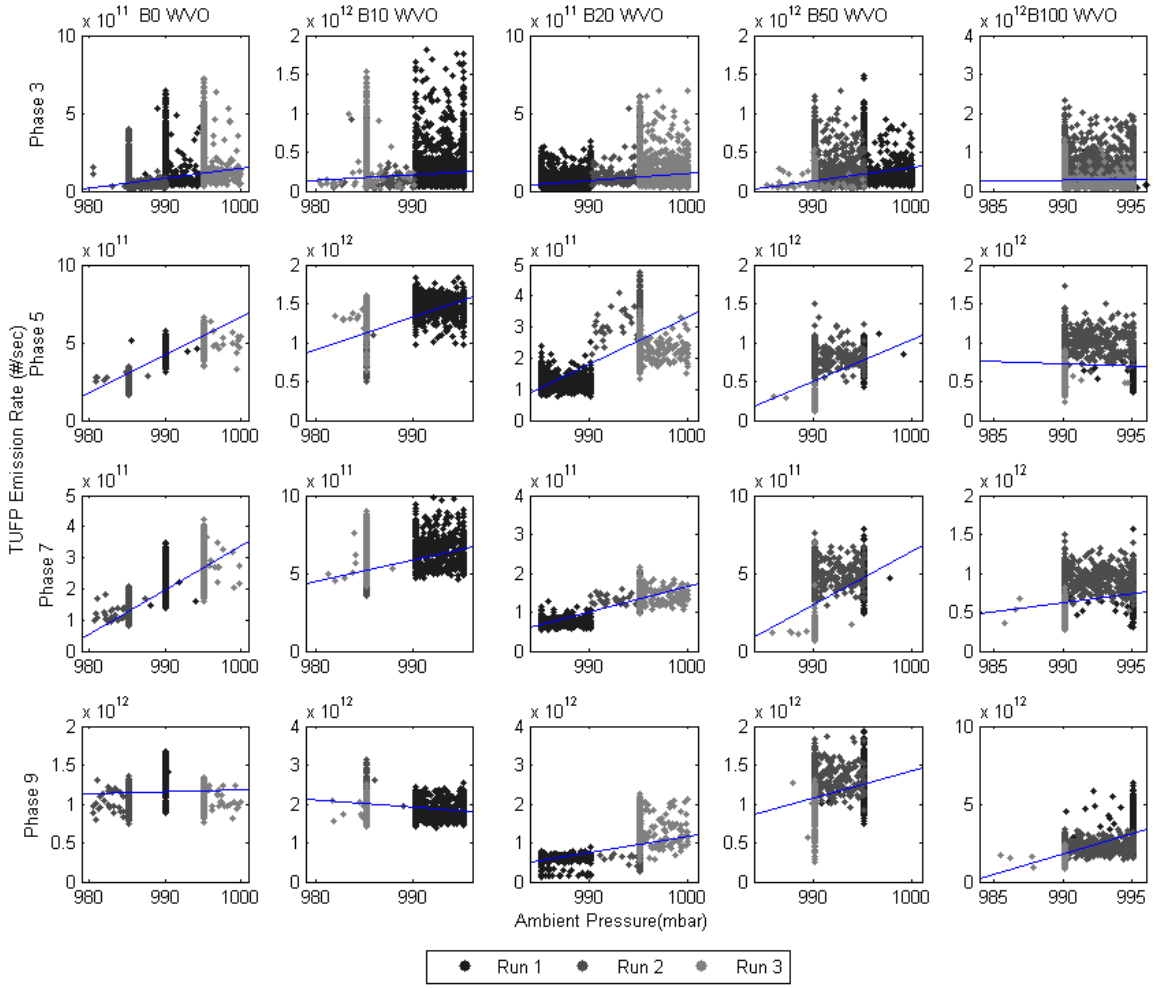


Figure A10: Scatter plots by WVO blend and phase of ambient pressure versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

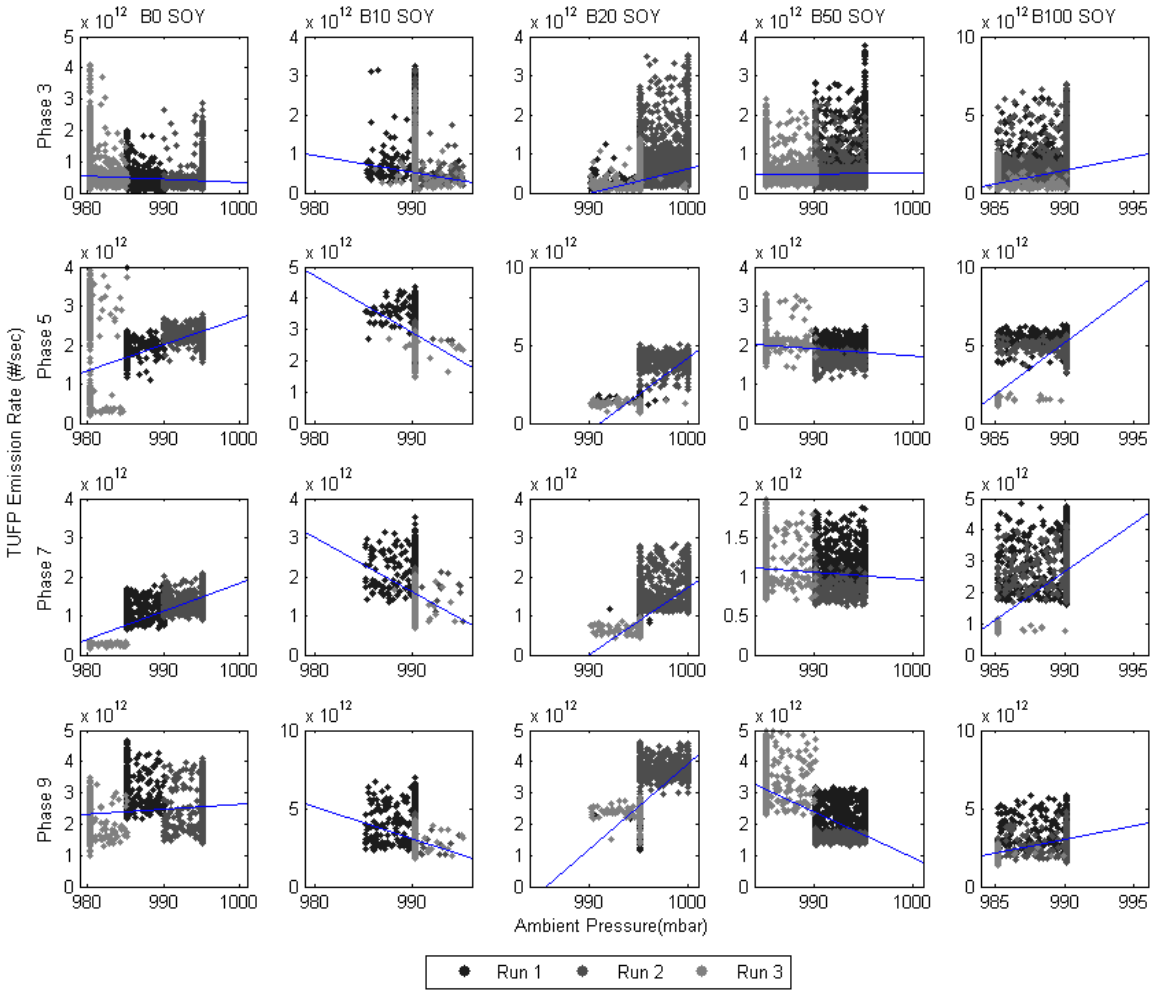


Figure A11: Scatter plots by SOY blend and phase of ambient pressure versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

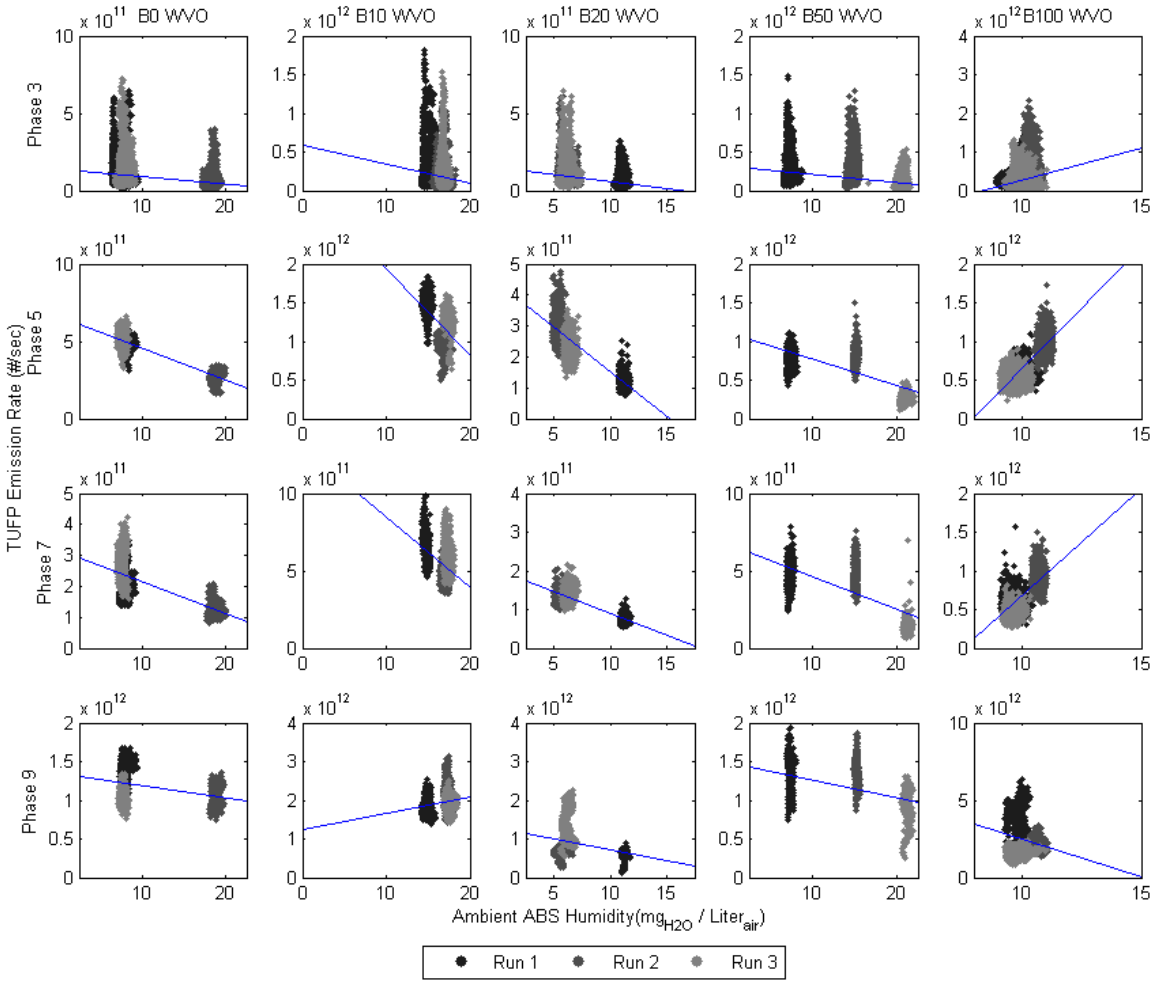


Figure A12: Scatter plots by WVO blend and phase of ambient absolute humidity versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

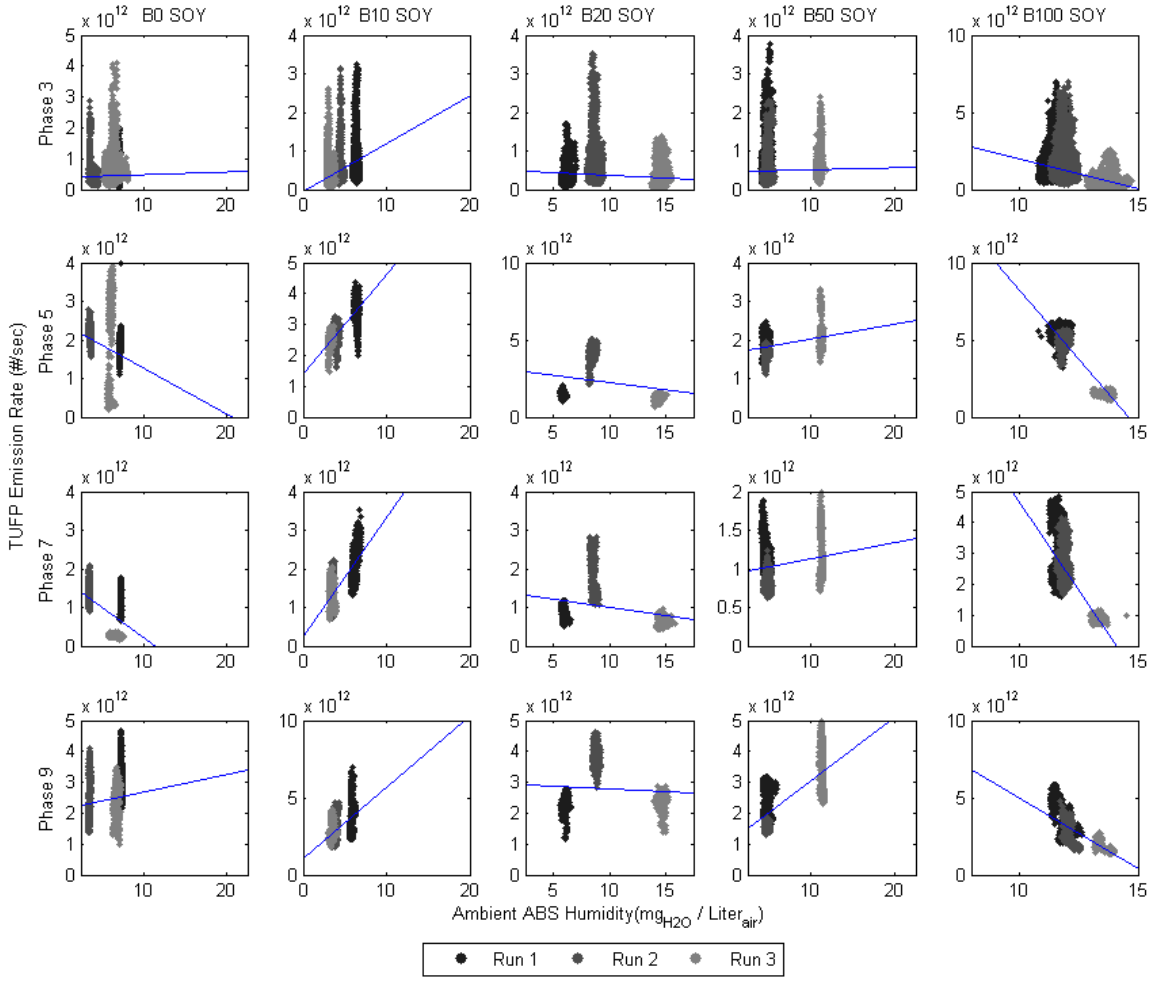


Figure A13: Scatter plots by SOY blend and phase of ambient absolute humidity versus TUFP emissions. Blue lines represent a linear regression of the scatterplot data.

TUFP Concentration (#/cm^3)

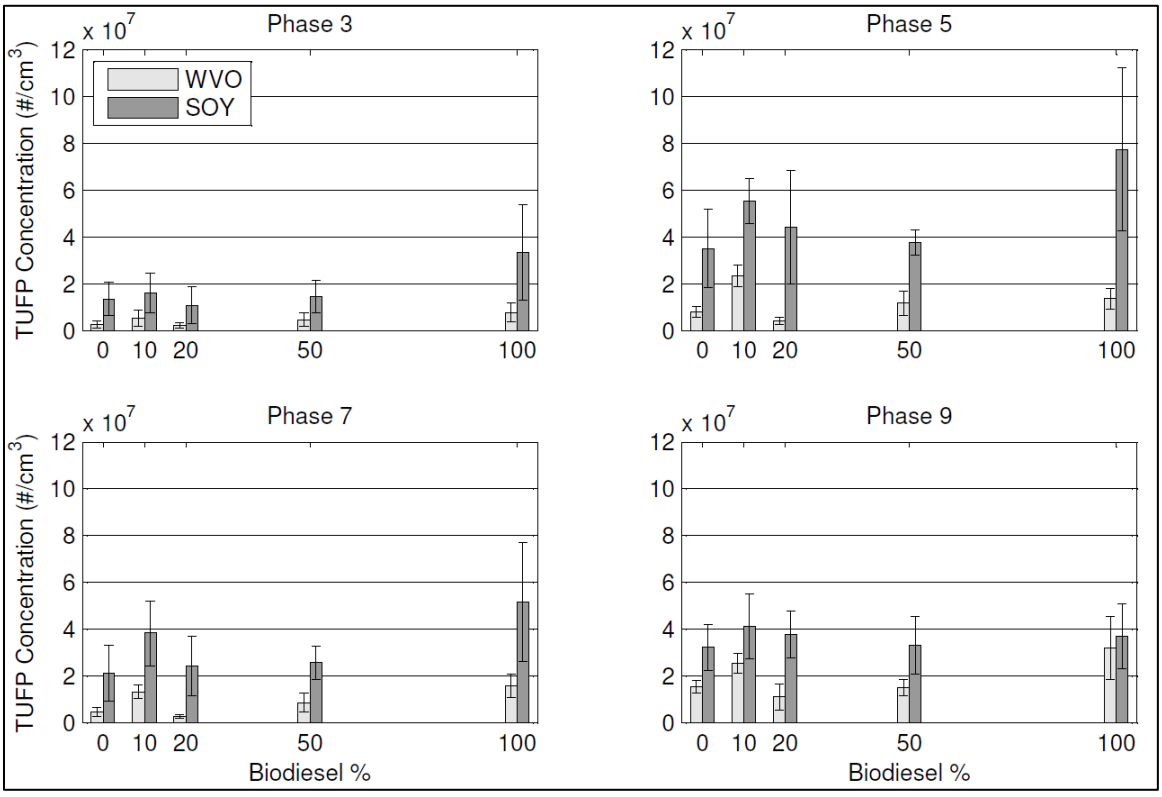


Figure A14: Mean TUFP concentration (5.6nm ≤ D_p ≤ 100nm) by run phase and biodiesel blend percentage. Each column represents the mean of combined triplicate data for each fuel blend and feedstock. Error bars represent ± 1 standard deviation. N = 3600x3 for P3 and 600x3 for P5, P7, and P9. Note: Y-axes are scaled differently from plot to plot.

TPN Emission Rate (#/sec)

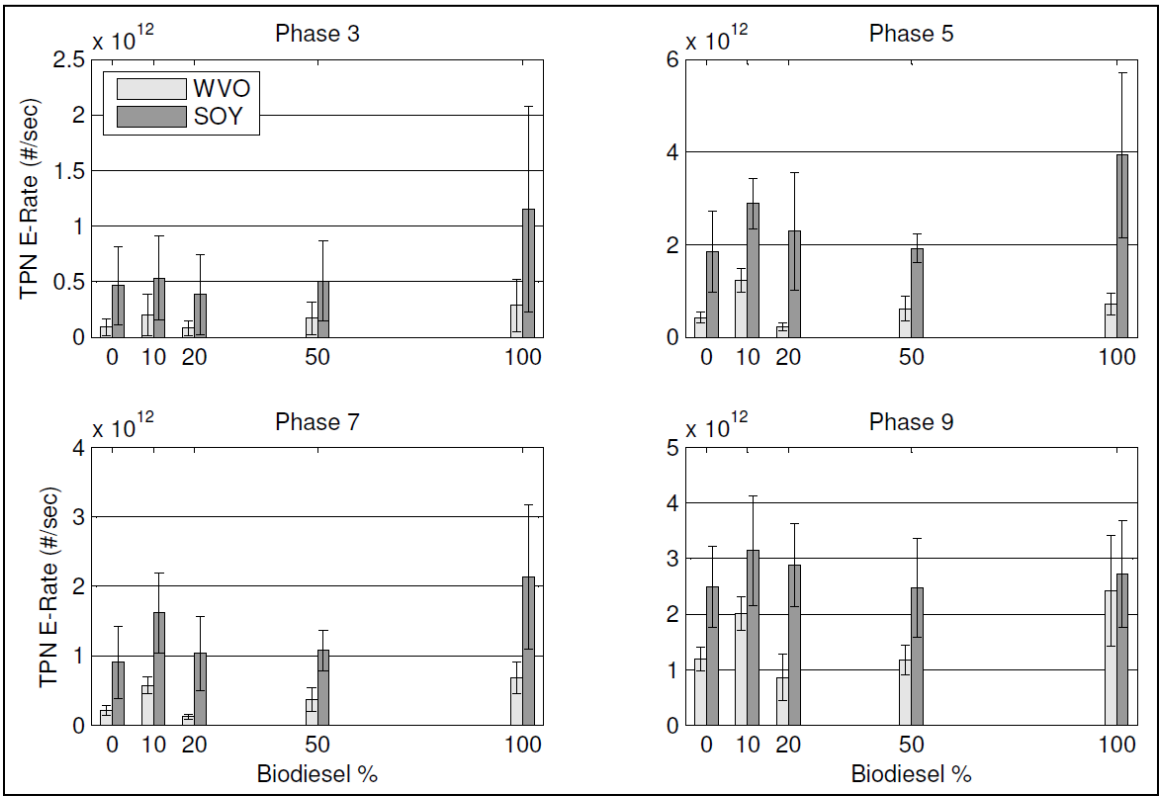


Figure A15: Mean TPN emission rates (5.6nm ≤ Dp ≤ 560nm) by run phase, biodiesel blend percentage, and biodiesel feedstock. Each column represents the mean of combined triplicate data for each fuel blend and error bars represent ± 1 standard deviation. N = 3600x3 for Phase 3 and 600x3 for Phases 5, 7, and 9. Note: Y-axes are scaled differently from plot to plot

Table A5: Run TUFP/ TPN Summary Table. Values are emission rates (x1e10) and rounded to 3 significant figures.

		Phase 3								Phase 5								Phase 7								Phase 9							
		TUFP ER (#/sec)			/	TPN ER (#/sec)				TUFP ER (#/sec)			/	TPN ER (#/sec)				TUFP ER (#/sec)			/	TPN ER (#/sec)				TUFP ER (#/sec)			/	TPN ER (#/sec)			
Run	Run ID Number	Mean	±	StDev	/	Mean	±	StDev		Mean	±	StDev	/	Mean	±	StDev		Mean	±	StDev	/	Mean	±	StDev		Mean	±	StDev	/	Mean	±	StDev	
B0-WVO-1	1-18JUN2013-B0-WVO	9.09	±	7.78	/	9.4	±	8.01		47.3	±	3.85	/	47.8	±	3.88		20.6	±	5.13	/	21.4	±	5.18		138	±	15	/	142	±	15	
B0-WVO-2	1-25JUN2013-B0-WVO	5.34	±	4.08	/	5.87	±	4.39		27.5	±	3.48	/	28.1	±	3.53		12.4	±	2.6	/	13.3	±	2.67		104	±	13.7	/	107	±	13.7	
B0-WVO-3	1-6AUG2013-B0-WVO	11.8	±	8.17	/	12.1	±	8.47		52.3	±	5.17	/	52.7	±	5.19		26.7	±	5.24	/	27.7	±	5.32		106	±	11.5	/	108	±	11.6	
B10-WVO-1	1-29AUG2013-B10-WVO	23.5	±	20.5	/	24.2	±	21.1		146	±	13.6	/	147	±	13.6		63.6	±	9.79	/	65.8	±	9.86		185	±	23.7	/	192	±	23.7	
B10-WVO-2	1-30AUG2013-B10-WVO	14.8	±	14.1	/	15.3	±	14.6		92.1	±	10.8	/	92.7	±	10.8		45.2	±	4.43	/	46.7	±	4.52		216	±	34.7	/	221	±	34.7	
B10-WVO-3	1-31AUG2013-B10-WVO	18.9	±	18	/	19.4	±	18.5		131	±	13.8	/	131	±	13.8		58.1	±	10.9	/	59.8	±	11		185	±	22.4	/	191	±	22.6	
B20-WVO-1	1-4SEP2013-B20-WVO	4.98	±	3.5	/	5.18	±	3.64		12.1	±	2.38	/	12.4	±	2.39		7.41	±	0.983	/	7.79	±	1.03		60.3	±	11.8	/	61.4	±	11.8	
B20-WVO-2	1-5SEP2013-B20-WVO	9.52	±	6.56	/	9.81	±	6.84		30.6	±	4.9	/	31.1	±	4.93		13.2	±	1.67	/	14.1	±	1.77		61.6	±	12.7	/	63.8	±	12.6	
B20-WVO-3	1-6SEP2013-B20-WVO	9.33	±	7.08	/	9.6	±	7.34		22.8	±	3.64	/	23.1	±	3.66		14.2	±	1.92	/	14.8	±	1.99		131	±	39.8	/	133	±	39.8	
B50-WVO-1	1-9SEP2013-B50-WVO	23.3	±	15.8	/	23.5	±	16		76.7	±	11.3	/	77	±	11.3		46.3	±	7.96	/	46.9	±	8		125	±	22.5	/	126	±	22.6	
B50-WVO-2	1-10SEP2013-B50-WVO	19.4	±	14.6	/	19.6	±	14.8		81.3	±	12.6	/	81.6	±	12.6		48.9	±	8.02	/	49.7	±	8.07		131	±	16.8	/	133	±	16.9	
B50-WVO-3	1-11SEP2013-B50-WVO	8.34	±	6.48	/	8.89	±	6.68		26.2	±	5.48	/	26.9	±	5.47		15	±	4.06	/	15.8	±	4.08		90.8	±	16.6	/	92.6	±	16.7	
B100-WVO-1	1-19SEP2013-B100-WVO	25.3	±	20.1	/	25.6	±	20.2		60.9	±	11.5	/	61.3	±	11.5		65.2	±	12.7	/	65.9	±	12.7		332	±	114	/	333	±	114	
B100-WVO-2	1-20SEP2013-B100-WVO	38	±	28.6	/	38.5	±	28.8		99.4	±	15	/	100	±	15		91.3	±	14.4	/	92.3	±	14.4		226	±	38.3	/	227	±	38.3	
B100-WVO-3	2-20SEP2013-B100-WVO	21.3	±	16.2	/	21.7	±	16.4		52.7	±	10.6	/	53.2	±	10.6		45.9	±	8.94	/	46.7	±	8.96		164	±	29.5	/	165	±	29.5	
B0-SOY-1	1-5DEC2013-B0-SOY	37.9	±	22.5	/	38.7	±	23.2		195	±	23.4	/	199	±	24.2		107	±	25.9	/	109	±	26.4		293	±	54.9	/	302	±	54.9	
B0-SOY-2	1-6DEC2013-B0-SOY	42	±	30.4	/	42.6	±	31		226	±	21.3	/	230	±	22		134	±	24.1	/	137	±	24.7		237	±	72.1	/	244	±	72.4	
B0-SOY-3	1-2MAY2014-B0-SOY	57.8	±	45.6	/	58.4	±	46.4		124	±	125	/	125	±	125		25.7	±	3.51	/	26.5	±	3.58		201	±	50.8	/	203	±	50.9	
B10-SOY-1	1-5MAY2014-B10-SOY	73.7	±	40	/	74.5	±	40.8		346	±	37.9	/	350	±	38.7		215	±	46.2	/	219	±	47		373	±	115	/	381	±	115	
B10-SOY-2	1-6MAY2014-B10-SOY	48.7	±	31.9	/	49.3	±	32.5		269	±	25.5	/	270	±	25.7		135	±	41.4	/	137	±	41.8		277	±	76.1	/	282	±	76.5	
B10-SOY-3	2-6MAY2014-B10-SOY	34.6	±	27.3	/	35.2	±	27.8		242	±	23	/	243	±	23.2		127	±	33	/	129	±	33.2		274	±	57.8	/	280	±	58.1	
B20-SOY-1	2-12MAY2014-B20-SOY	28.6	±	19.3	/	29.4	±	19.8		152	±	15.2	/	153	±	15.2		76.2	±	14.9	/	78.2	±	15		223	±	24.2	/	229	±	24.4	
B20-SOY-2	2-13MAY2014-B20-SOY	60.4	±	45.8	/	61.5	±	46.7		403	±	46	/	404	±	46		163	±	45.3	/	167	±	45.5		376	±	29.4	/	387	±	29.8	
B20-SOY-3	1-15MAY2014-B20-SOY	23.2	±	19.4	/	23.6	±	19.7		129	±	14.5	/	130	±	14.5		63.8	±	10.2	/	65	±	10.2		240	±	20.4	/	248	±	21	
B50-SOY-1	1-20MAY2014-B50-SOY	57.1	±	44	/	57.8	±	44.6		203	±	17.4	/	203	±	17.4		120	±	23	/	121	±	23.2		241	±	42.9	/	244	±	43.2	
B50-SOY-2	2-20MAY2014-B50-SOY	40.8	±	27.6	/	41.4	±	28		163	±	14.1	/	164	±	14.1		81.9	±	10.2	/	83.3	±	10.3		158	±	10.6	/	161	±	10.8	
B50-SOY-3	1-22MAY2014-B50-SOY	51.2	±	30	/	51.7	±	30.5		209	±	31.7	/	209	±	31.7		117	±	31.3	/	119	±	31.6		335	±	73.4	/	340	±	73.9	
B100-SOY-1	1-23MAY2014-B100-SOY	153	±	97.2	/	154	±	97.9		542	±	46.3	/	542	±	46.3		272	±	82.8	/	273	±	83.2		356	±	99.9	/	358	±	99.8	
B100-SOY-2	2-23MAY2014-B100-SOY	140	±	91.2	/	141	±	91.8		490	±	38.8	/	491	±	38.9		262	±	59.8	/	264	±	60		253	±	66.6	/	255	±	66.5	
B100-SOY-3	1-26MAY2014-B100-SOY	51	±	30.9	/	51.5	±	31.2		149	±	13.1	/	149	±	13.1		87	±	8.93	/	87.8	±	8.97		202	±	26.4	/	203	±	26.5	

Average Particle Number Distribution (PND) Emission Rate (Linear Scale)

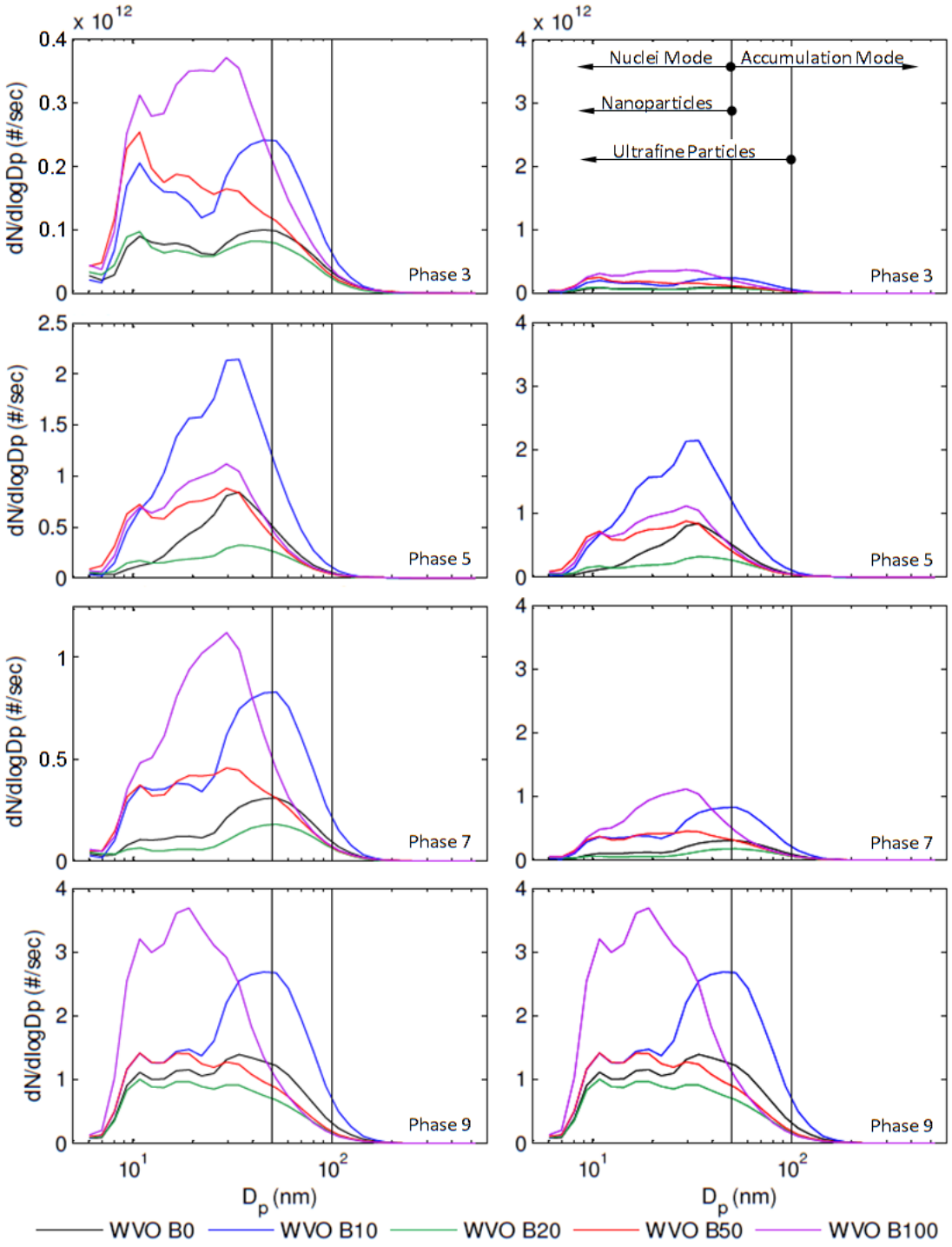


Figure A16: Average WVO particle number distributions by biodiesel blend and drive cycle phase. Log – Linear. Y-Scale limits are different on from plot to plot on the left side while the Y-Scale limits are the same from plot to plot on the right side.

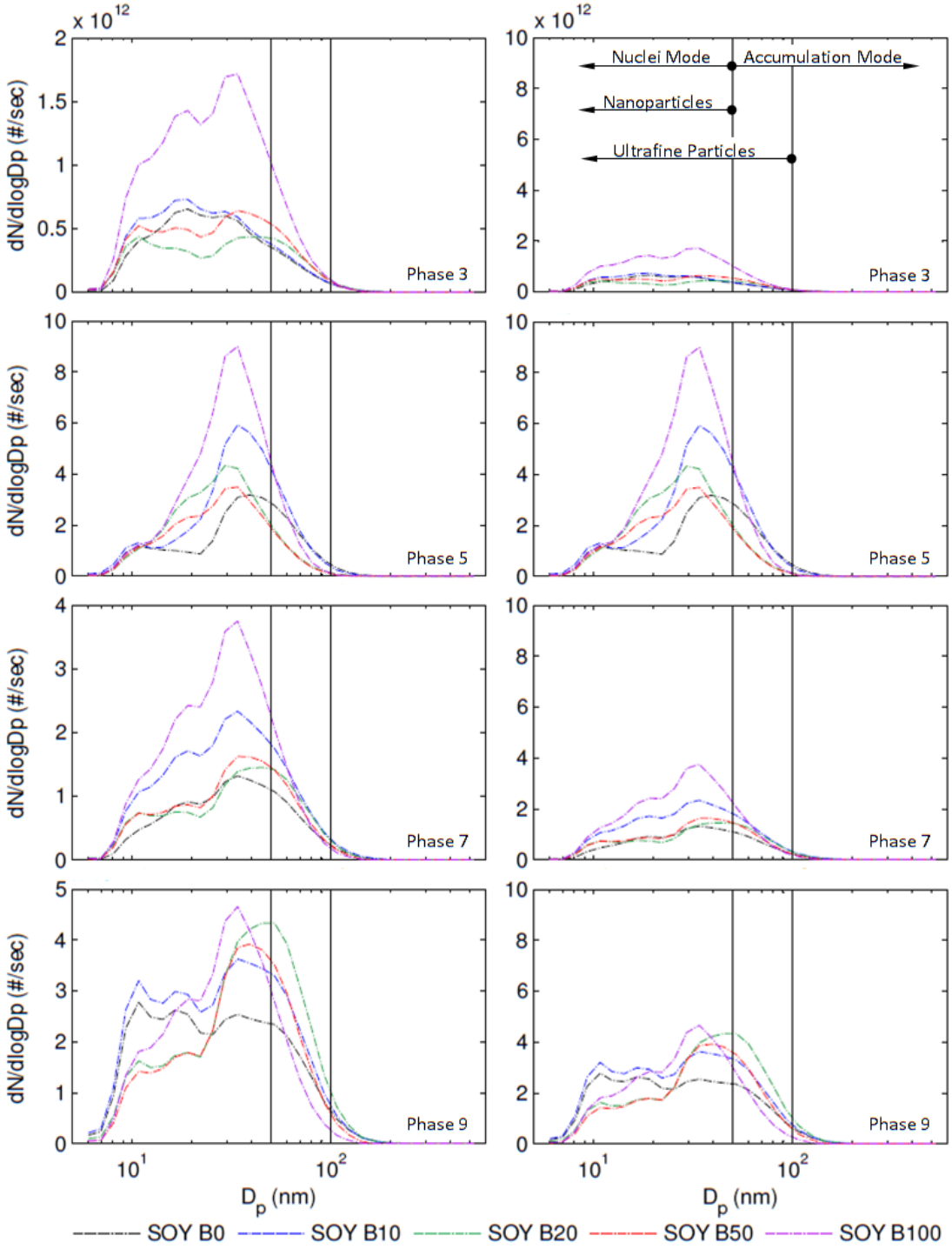


Figure A17: Average SOY particle number distributions by biodiesel blend and drive cycle phase. Log – Linear. Y-Scale limits are different on from plot to plot on the left side while the Y-Scale limits are the same from plot to plot on the right side.

Mean Diameter Calculation

The mean diameters (MDs) of particle number distributions were calculated as follows.

$$MD = \frac{\sum(D_{p,i} x ER_i)}{\sum ER_i} \quad [14]$$

Where:

MD = mean diameter (nm)

$\boldsymbol{D_{p,i}}$ **= EEPS $\mathbf{D_p}$ midpoint for bin *i* (nm) (Engine Exhaust Particle Sizer Bin Data**

Table A4)

ER_i = emission rate for EEPS bin *i* (#/sec)

Average Particle Number Distributions by Run, Phase, and Fuel

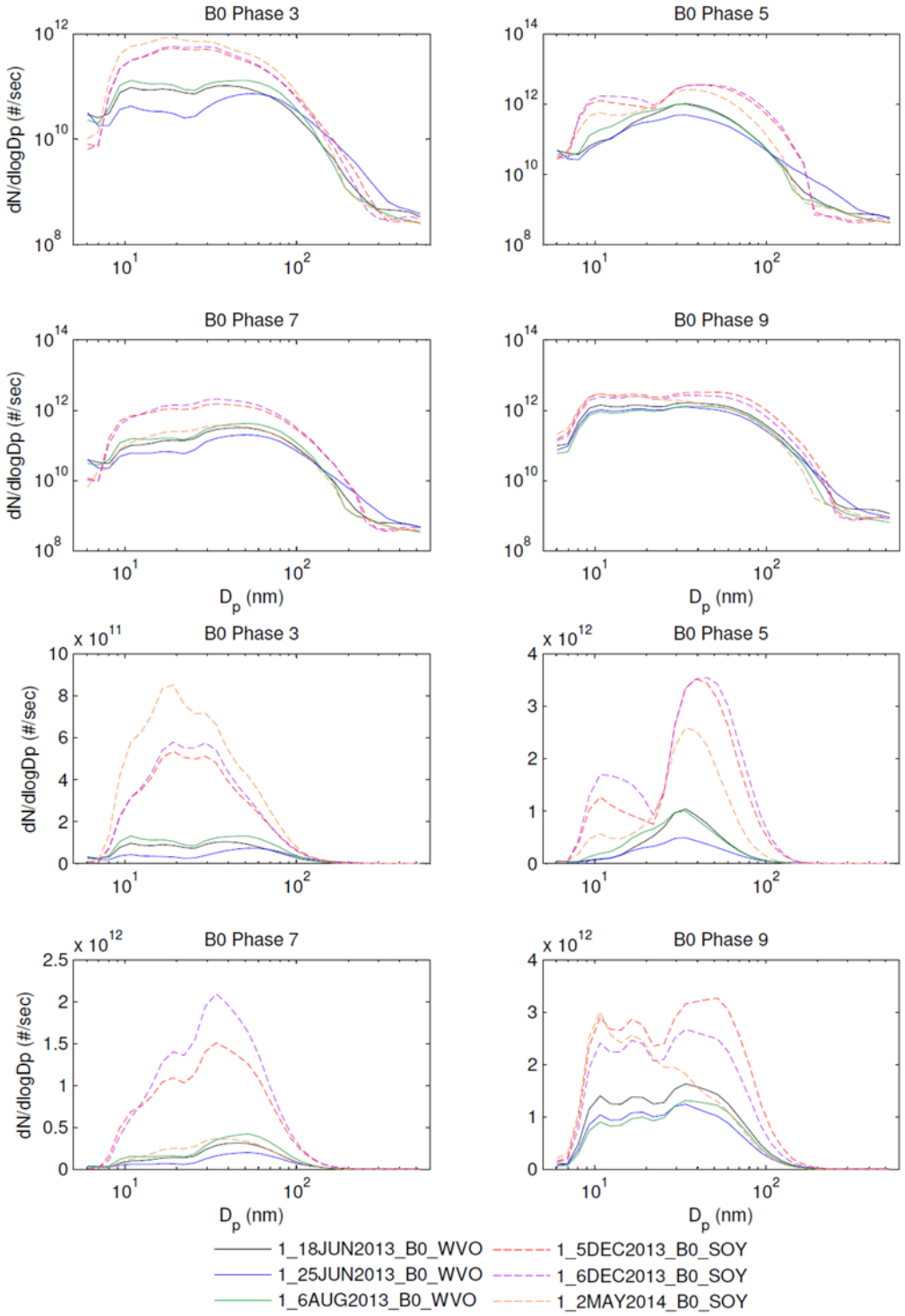


Figure A18: B0 run average particle number distributions. Top 4 panels are Log-Log; bottom 4 are Log-Linear.

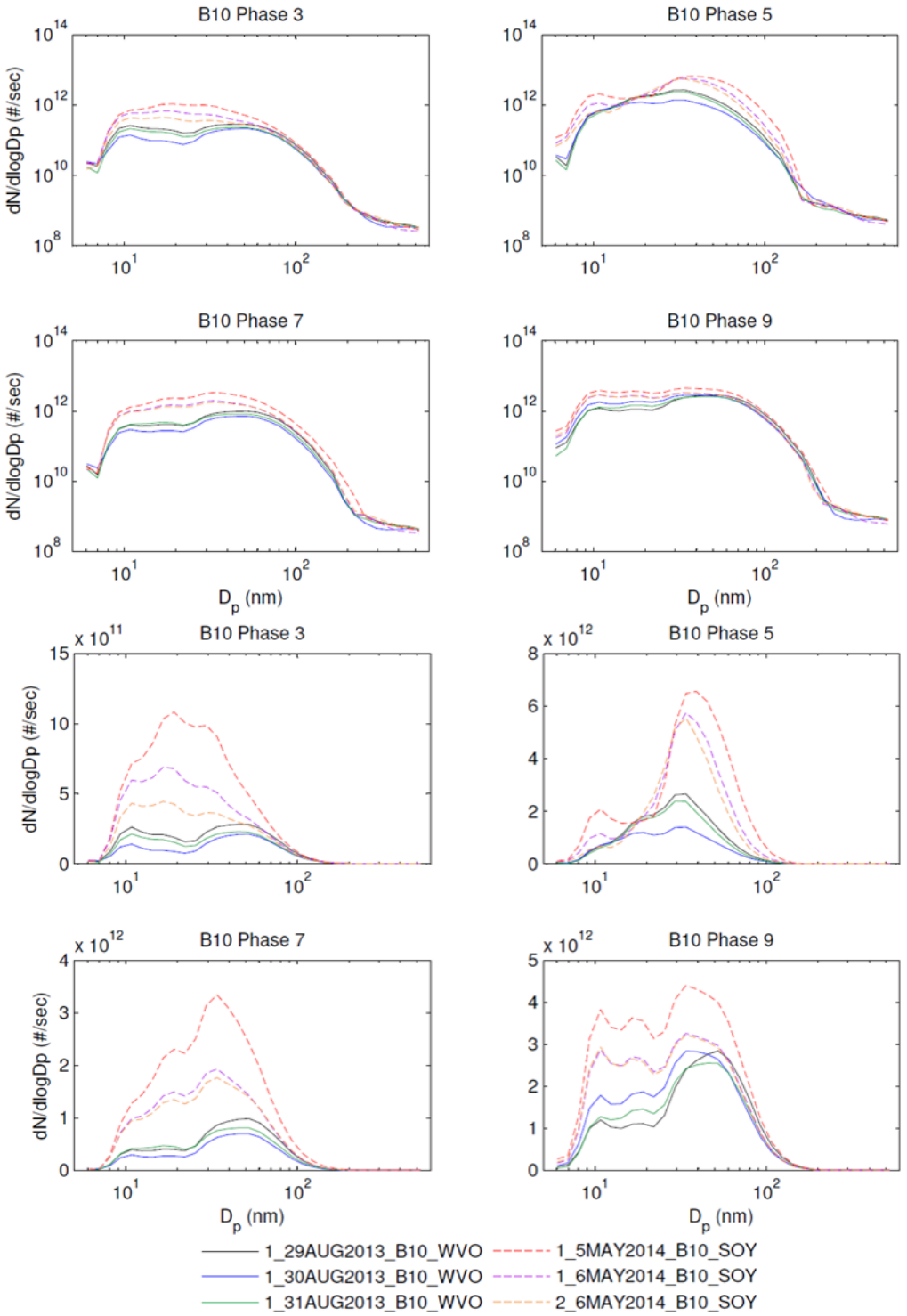


Figure A19: B10 run average particle number distributions. Top 4 panels are Log-Log; bottom 4 are Log-Linear.

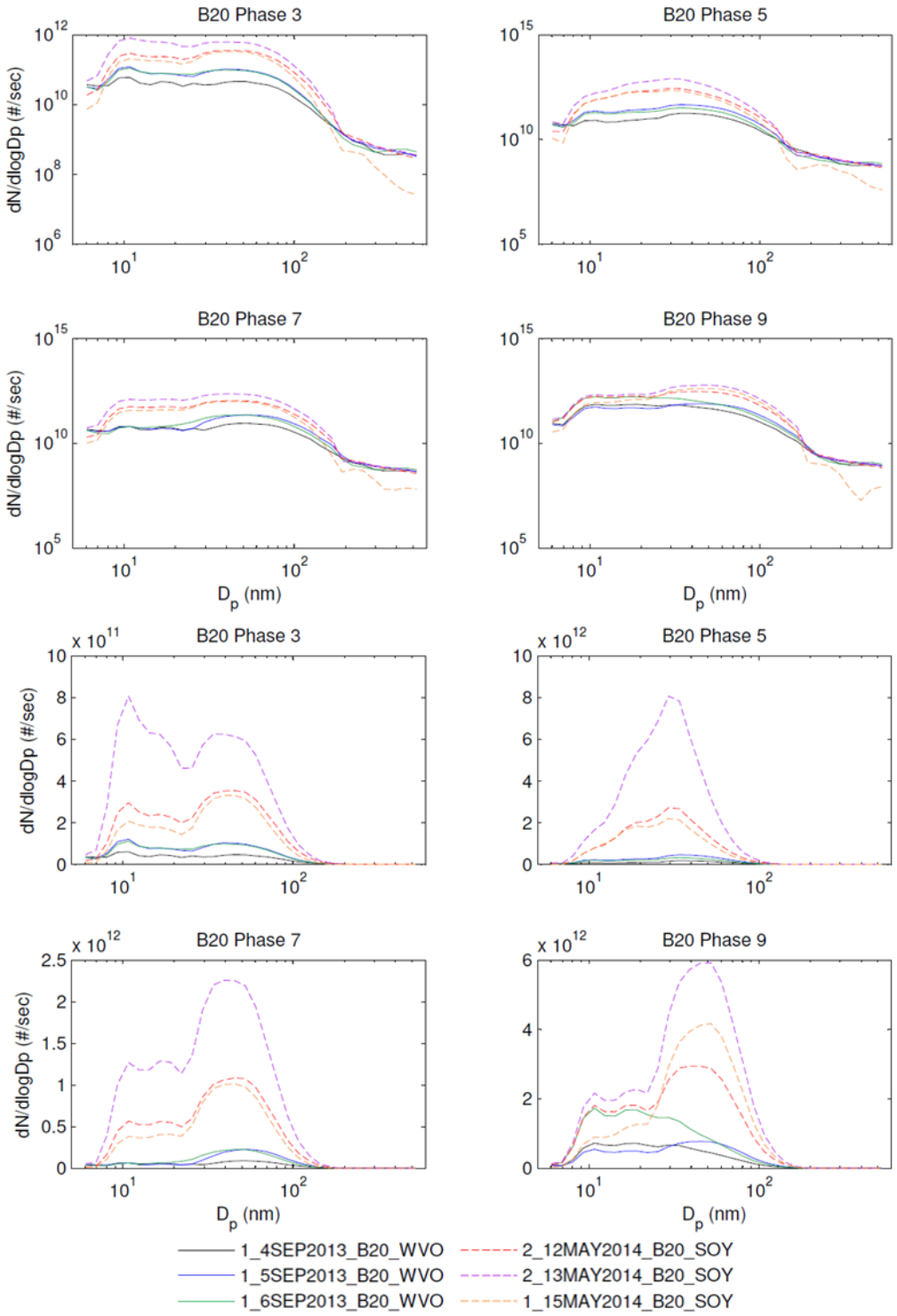


Figure A20: B20 run average particle number distributions. Top 4 panels are Log-Log; bottom 4 are Log-Linear.

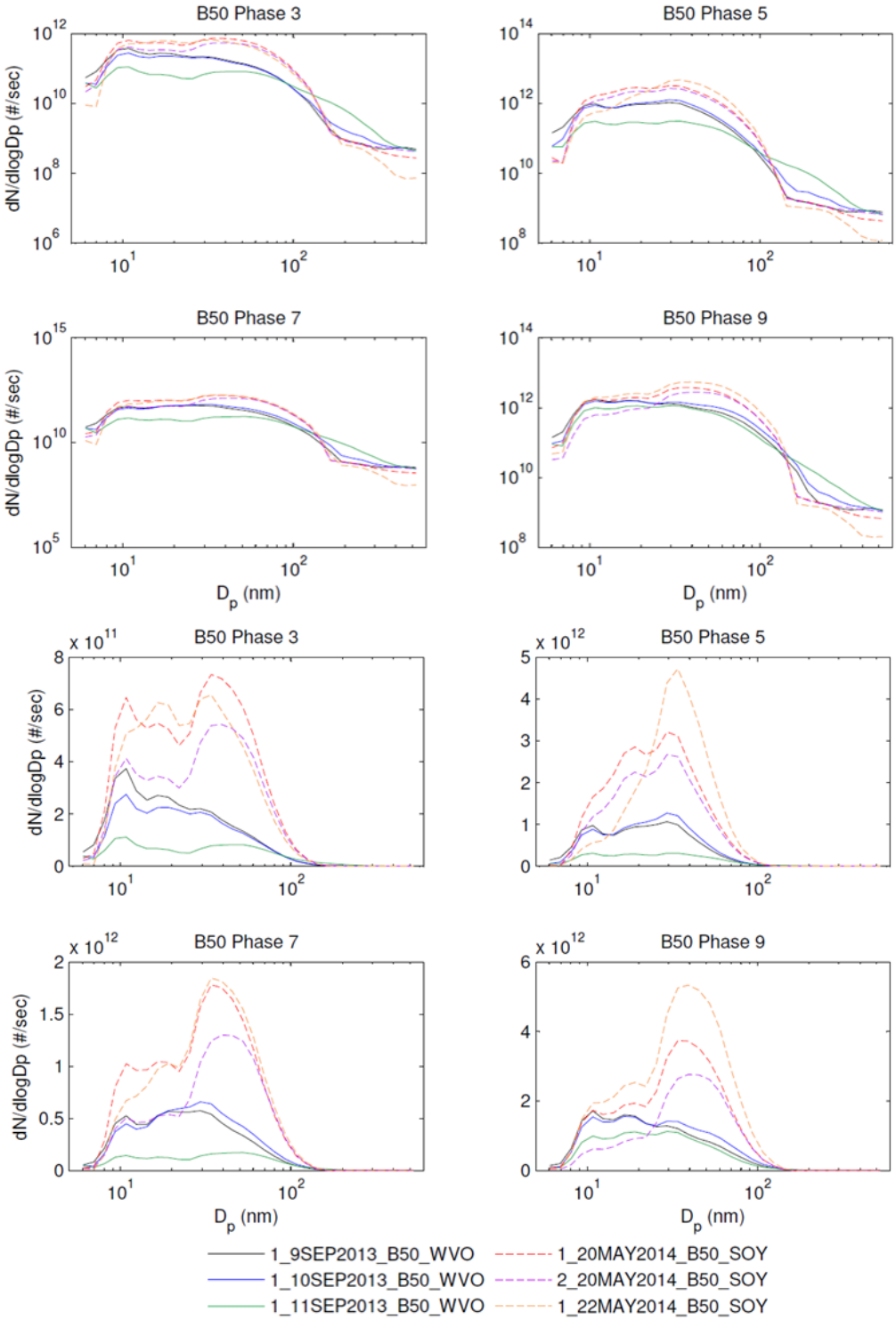


Figure A21: B50 run average particle number distributions. Top 4 panels are Log-Log; bottom 4 are Log-Linear.

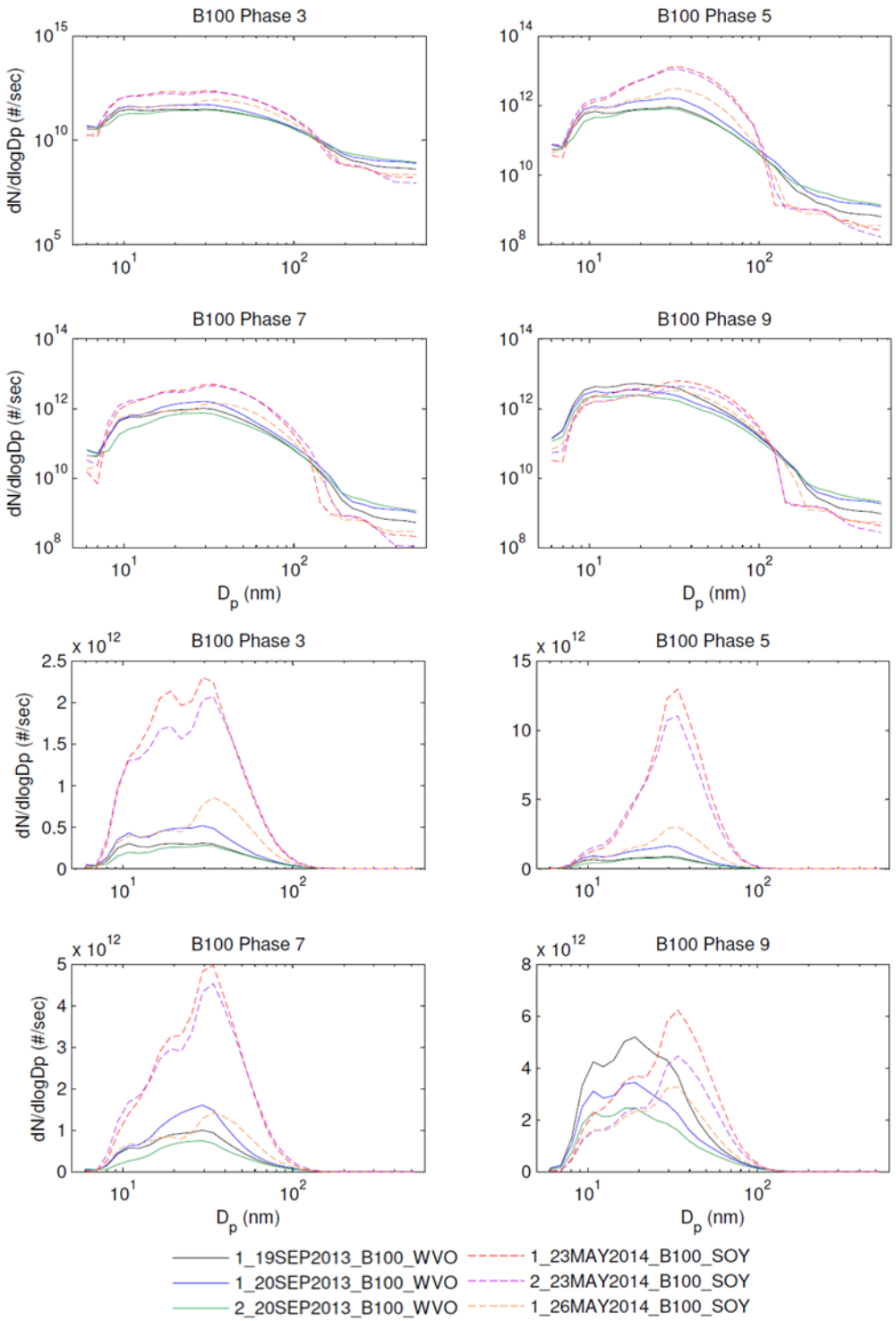


Figure A22: B100 run average particle number distributions. Top 4 panels are Log-Log; bottom 4 are Log-Linear.

Particle Number Distribution Modal Diameter Summary Table

Table A6: Particle number distribution mode D_{Mo}, ER, and ratio of blend emission rate to petrodiesel emission rate (ER_{BXX}/ ER_{B0}).

			Small Mode			Middle Mode			Large Mode		
		BXX	D_{Mo}	ER	ER_{BXX}/ ER_{B0}	D_{Mo}	ER	ER_{BXX}/ ER_{B0}	D_{Mo}	ER	ER_{BXX}/ ER_{B0}
WVO	Phase 3	B0	10.8	0.09	1.0	16.5	0.08	1.0	45.3	0.10	1.0
		B10	10.8	0.21	2.3	16.5	0.16	2.0	45.3	0.24	2.4
		B20	10.8	0.10	1.1	16.5	0.07	0.9	39.2	0.08	0.8
		B50	10.8	0.25	2.8	16.5	0.19	2.4	29.4	0.16	1.6
		B100	10.8	0.31	3.5	22.1	0.35	4.4	29.4	0.37	3.7
	Phase 5	B0	10.8	0.12	1.0	19.1	0.43	1.0	34	0.84	1.0
		B10	10.8	0.67	5.6	19.1	1.57	3.6	34	2.15	2.5
		B20	10.8	0.17	1.4	19.1	0.19	0.4	34	0.32	0.4
		B50	10.8	0.72	6.0	19.1	0.75	1.7	29.4	0.88	1.0
		B100	10.8	0.69	5.8	19.1	0.94	2.2	29.4	1.12	1.3
	Phase7	B0	10.8	0.11	1.0	19.1	0.12	1.0	52.3	0.31	1.0
		B10	10.8	0.37	3.5	16.5	0.38	3.1	52.3	0.83	2.7
		B20	10.8	0.06	0.6	19.1	0.06	0.5	52.3	0.18	0.6
		B50	10.8	0.37	3.5	19.1	0.42	3.5	29.4	0.46	1.5
		B100	10.8	0.48	4.6	19.1	0.94	7.7	29.4	1.12	3.6
	Phase 9	B0	10.8	1.12	1.0	19.1	1.15	1.0	34	1.39	1.0
		B10	10.8	1.41	1.3	19.1	1.47	1.3	45.3	2.69	1.9
		B20	10.8	1.01	0.9	19.1	0.97	0.8	29.4	0.92	0.7
		B50	10.8	1.42	1.3	16.5	1.42	1.2	29.4	1.27	0.9
		B100	10.8	3.21	2.9	19.1	3.69	3.2	34	2.50	1.8
SOY	Phase 3	B0	10.8	0.40	1.0	19.1	0.66	1.0	29.4	0.60	1.0
		B10	10.8	0.58	1.4	19.1	0.73	1.1	29.4	0.63	1.1
		B20	10.8	0.44	1.1	16.5	0.35	0.5	39.2	0.44	0.7
		B50	10.8	0.52	1.3	16.5	0.51	0.8	34	0.64	1.1
		B100	10.8	1.01	2.5	19.1	1.43	2.2	34	1.72	2.9
	Phase 5	B0	10.8	1.17	1.0	19.1	0.94	1.0	39.2	3.18	1.0
		B10	10.8	1.31	1.1	19.1	1.77	1.9	34	5.93	1.9
		B20	10.8	1.08	0.9	19.1	3.06	3.3	29.4	4.33	1.4
		B50	10.8	1.13	1.0	19.1	2.30	2.5	34	3.49	1.1
		B100	10.8	1.23	1.0	19.1	3.82	4.1	34	9.00	2.8
	Phase7	B0	10.8	0.47	1.0	19.1	0.91	1.0	34	1.32	1.0
		B10	10.8	1.07	2.3	19.1	1.71	1.9	34	2.34	1.8
		B20	10.8	0.74	1.6	16.5	0.75	0.8	45.3	1.45	1.1
		B50	10.8	0.73	1.6	19.1	0.87	1.0	34	1.62	1.2
		B100	10.8	1.26	2.7	19.1	2.43	2.7	34	3.75	2.8
	Phase 9	B0	10.8	2.78	1.0	16.5	2.63	1.0	34	2.54	1.0
		B10	10.8	3.20	1.2	16.5	2.99	1.1	34	3.63	1.4
		B20	10.8	1.63	0.6	19.1	1.79	0.7	45.3	4.34	1.7
		B50	10.8	1.42	0.5	19.1	1.78	0.7	39.2	3.92	1.5
		B100	10.8	1.81	0.7	19.1	2.83	1.1	34	4.66	1.8

Note: D_{Mo} is in nm and ER is in (#/sec) x1e12

Table A7: Fuel Consumption Summary Table (From Scale Data)

		Phase 3			Phase 5			Phase 7			Phase 9		
Run	Run ID Number	Fuel Consumption (kg/min)	Mean	StDev	Fuel Consumption (kg/min)	Mean	StDev	Fuel Consumption (kg/min)	Mean	StDev	Fuel Consumption (kg/min)	Mean	StDev
B0-WVO-1	1-18JUN2013-B0-WVO	0.0177			0.0292			0.0412			0.0805		
B0-WVO-2	1-25JUN2013-B0-WVO	0.0182	0.0181	4.04E-04	0.0292	0.0293	0.000173	0.0409	0.0411	1.53E-04	0.0797	0.0801	4.00E-04
B0-WVO-3	1-6AUG2013-B0-WVO	0.0185			0.0295			0.0411			0.0801		
B10-WVO-1	1-29AUG2013-B10-WVO	0.0176			0.0292			0.0394			0.0824		
B10-WVO-2	1-30AUG2013-B10-WVO	0.0184	0.0175	9.02E-04	0.0282	0.0291	0.000902	0.0414	0.04	1.18E-03	0.0821	0.0828	9.07E-04
B10-WVO-3	1-31AUG2013-B10-WVO	0.0166			0.03			0.0393			0.0838		
B20-WVO-1	1-4SEP2013-B20-WVO	0.0203			0.0308			0.0429			0.0856		
B20-WVO-2	1-5SEP2013-B20-WVO	0.0199	0.0201	2.08E-04	0.0312	0.0305	0.000889	0.0423	0.0429	6.00E-04	0.0847	0.085	5.20E-04
B20-WVO-3	1-6SEP2013-B20-WVO	0.0202			0.0295			0.0435			0.0847		
B50-WVO-1	1-9SEP2013-B50-WVO	0.021			0.0309			0.0453			0.0881		
B50-WVO-2	1-10SEP2013-B50-WVO	0.0205	0.0208	2.52E-04	0.0306	0.0307	0.000173	0.0466	0.046	6.56E-04	0.0881	0.0882	2.31E-04
B50-WVO-3	1-11SEP2013-B50-WVO	0.0208			0.0306			0.0461			0.0885		
B100-WVO-1	1-19SEP2013-B100-WVO	0.0213			0.032			0.0497			0.0944		
B100-WVO-2	1-20SEP2013-B100-WVO	0.0216	0.0217	4.58E-04	0.0323	0.0315	0.00119	0.0496	0.0485	1.93E-03	0.0939	0.0938	6.03E-04
B100-WVO-3	2-20SEP2013-B100-WVO	0.0222			0.0301			0.0463			0.0932		
B0-SOY-1	1-5DEC2013-B0-SOY	0.0196			0.0308			0.0424			0.0844		
B0-SOY-2	1-6DEC2013-B0-SOY	0.0192	0.0192	4.51E-04	0.0305	0.0304	0.000458	0.0418	0.0418	6.00E-04	0.0841	0.0834	1.48E-03
B0-SOY-3	1-2MAY2014-B0-SOY	0.0187			0.0299			0.0412			0.0817		
B10-SOY-1	1-5MAY2014-B10-SOY	0.0188			0.0296			0.0414			0.0833		
B10-SOY-2	1-6MAY2014-B10-SOY	0.0187	0.0187	5.77E-05	0.0298	0.0298	0.000153	0.042	0.0417	3.06E-04	0.0834	0.0831	4.36E-04
B10-SOY-3	2-6MAY2014-B10-SOY	0.0187			0.0299			0.0418			0.0826		
B20-SOY-1	2-12MAY2014-B20-SOY	0.0196			0.0304			0.0432			0.0854		
B20-SOY-2	2-13MAY2014-B20-SOY	0.0195	0.0197	3.21E-04	0.0303	0.0305	0.000265	0.043	0.0434	5.86E-04	0.0858	0.0858	4.51E-04
B20-SOY-3	1-15MAY2014-B20-SOY	0.0201			0.0308			0.0441			0.0863		
B50-SOY-1	1-20MAY2014-B50-SOY	0.0209			0.0315			0.0464			0.0906		
B50-SOY-2	2-20MAY2014-B50-SOY	0.0206	0.0206	2.52E-04	0.0315	0.0315	0.0000577	0.0454	0.0456	6.81E-04	0.0887	0.0895	1.00E-03
B50-SOY-3	1-22MAY2014-B50-SOY	0.0204			0.0316			0.0451			0.0891		
B100-SOY-1	1-23MAY2014-B100-SOY	0.0224			0.0336			0.0498			0.0951		
B100-SOY-2	2-23MAY2014-B100-SOY	0.0229	0.0226	2.52E-04	0.0335	0.0335	0.0001	0.0503	0.05	2.52E-04	0.0954	0.095	4.04E-04
B100-SOY-3	1-26MAY2014-B100-SOY	0.0226			0.0334			0.05			0.0946		

$$FuelCon = \frac{m_1 - m_2}{t}$$

Where:

m_1 = Average fuel tank weight for the first 10 seconds of the phase
m_2 = Average fuel tank weight for the last 10 seconds of the phase
t = time duration of the phase

Genetic Programming Summary Table

Table A8: Genetic program run summary

Tournament Method Results

	B0 Data			100,000 Generations
	Complexity	R^2	MSE	Solution
A	13	0.60	3.88E-04	TUFP = 0.225*MAP + 0.068*AP - 0.030 - 0.715*(MAP Δ23)
B	15	0.64	3.48E-04	TUFP = 0.034*AP + EMT*AP*CT*MAF - 0.250*(INJQ_Δ23)
C	11	0.60	3.90E-04	TUFP = 0.166*AP*MAF*exp(DET)

	B10 Data			100,000 Generations
	Complexity	R^2	MSE	Solution
A	13	0.77	1.46E-03	TUFP = 0.025 + 0.863*EMT*MAF - 3.958*(ΔET Δ23)*MAF
B	17	0.79	1.33E-03	TUFP = 0.064*AP + 0.816*EMT*MAF - (ΔET Δ23)*exp(MAF)
C	13	0.77	1.46E-03	TUFP = 0.023 + 0.863*EMT*MAF - 4.016*(ΔET Δ23)*MAF

	B20 Data			100,000 Generations
	Complexity	R^2	MSE	Solution
A	17	0.80	1.52E-04	TUFP = 0.172* MAP*EMT + 0.123*AP*MAF - 0.065*MAT*MAF
B	13	0.79	1.64E-04	TUFP = 0.097*AP*MAF + 0.220*MAP*AP*MAF
C	21	0.82	1.39E-04	TUFP = 0.013 + 0.048*MAF + 0.208*MAP*MAF - 0.030*ABSH - 0.283*MAP*ABSH

	B50 Data			100,000 Generations
	Complexity	R^2	MSE	Solution
A	15	0.74	1.07E-03	TUFP = 0.049 + 0.522*MAP - 0.042*ABSH - 0.437*MAP*MAT
B	15	0.74	1.08E-03	TUFP = 0.049 + 0.517*MAP - 0.042*ABSH - 0.428*MAP*MAT
C	19	0.77	9.76E-04	TUFP = 0.025 + 0.510*MAP + 0.072*CT - 0.058*MAT - 0.396*MAP*MAT

	B100 Data			100,000 Generations
	Complexity	R^2	MSE	Solution
A	23	0.81	2.26E-03	TUFP = 1.222*EMT*FT + 0.970*ABSH*MAF - 0.050*MAT - 1.608*EMT*MAT*FT
B	20	0.78	2.51E-03	TUFP = 2.067*EMT*FT*ABSH + 1.216*ABSH*MAF*cos(2.291*MAT)
C	25	0.84	1.85E-03	TUFP = 0.186*EMT + ABSH*MAF + 0.528*FT*MAF + 0.218*DET*MAF - 0.840*MAT*MAF

	All Data			3 million Generations
	Complexity	R^2	MSE	Solution
A	45	0.74	1.50E-03	TUFP = 0.115*AP*CT + 0.003*BIO*EMT + 0.322*MAF^2 - 0.017*MAT - 0.092*MAF*sin(BIO + MAT) - 0.005*BIO*EMT*sin(BIO + MAT)
B	54	0.73	1.54E-03	TUFP = 0.023 + 0.003*BIO*EMT + 0.378*MAF^2 - 0.120*MAT*MAF - 0.007*sin(BIO + MAT) - 0.003*BIO*EMT*sin(BIO + MAT) - 0.180*MAF^2*sin(BIO + MAT)
C	31	0.72	1.64E-03	TUFP = 0.376*CT*MAF + 0.012*BIO*MAP*cos(0.690 + MAT) - 0.372*EMT*CT*sin(BIO + ABSH)

Single Model Method Results

	All Data			5 million generations
	Complexity	R^2	MSE	Solution
A	45	0.73	1.53E-03	TUFP = 0.239*MAF + 0.230*DET*MAF + 0.002*BIO*EMT - 0.319*MAT*MAF - 0.113*MAF*sin(0.239 + BIO) - 0.198*DET*MAF*sin(0.239 + BIO)
B	44	0.75	1.46E-03	TUFP = 0.019 + 0.005*BIO*EMT + 0.861*FT*MAF^2 - 0.006*BIO*EMT*MAT - 0.381*FT*MAF*sin(BIO) - 0.824*MAT*FT*MAF^2
C	35	0.67	1.92E-03	TUFP = 0.010 + 0.463*EMT*MAF + 0.003*BIO*MAF + 2.073e-5*EMT*BIO^2 - 1.474*(EMT Δ23)*MAF - 0.005*BIO*MAT*MAF

Injector Pump Operation

The VW SDi engine used for this research was equipped with a pump-line-nozzle fuel injection system that utilized a Bosch VE injection pump. Figure A23 explains how the injector pump distributes fuel. In panel 'a', the high pressure chamber (cavity in white) is full of fuel when the outlet port opens. Then in panel 'b', the plunger is pushed to the right forcing the fuel into the outlet port and into the fuel line that leads to the mechanical fuel injector. The timing of when the plunger is pushed relative to TDC of the piston determines SOI relative to TDC. The amount of fuel injected is controlled by when the control collar is actuated to allow flow through the transverse cutoff bore in panel 'c'. In panel 'd', the transverse cutoff bore is again covered by the control collar, the outlet port is closed, and the plunger moves back to the left to allow the high pressure chamber to be filled with fuel for the next injection event. The fuel itself is the only lubrication for the moving parts within the injector pump. As such, the tolerances between the moving parts allow some fuel to 'leak' out of the fuel injection circuit. The amount of fuel that leaks is dependent on the viscosity of the fuel. This results in more fuel being injected for higher viscosity fuels (biodiesel blends) than for lower viscosity fuels (petrodiesel). This supports the data presented in Figure 2.4 and Figure 3.5 which show an increase in fuel consumption as the amount of biodiesel in the fuel increases.

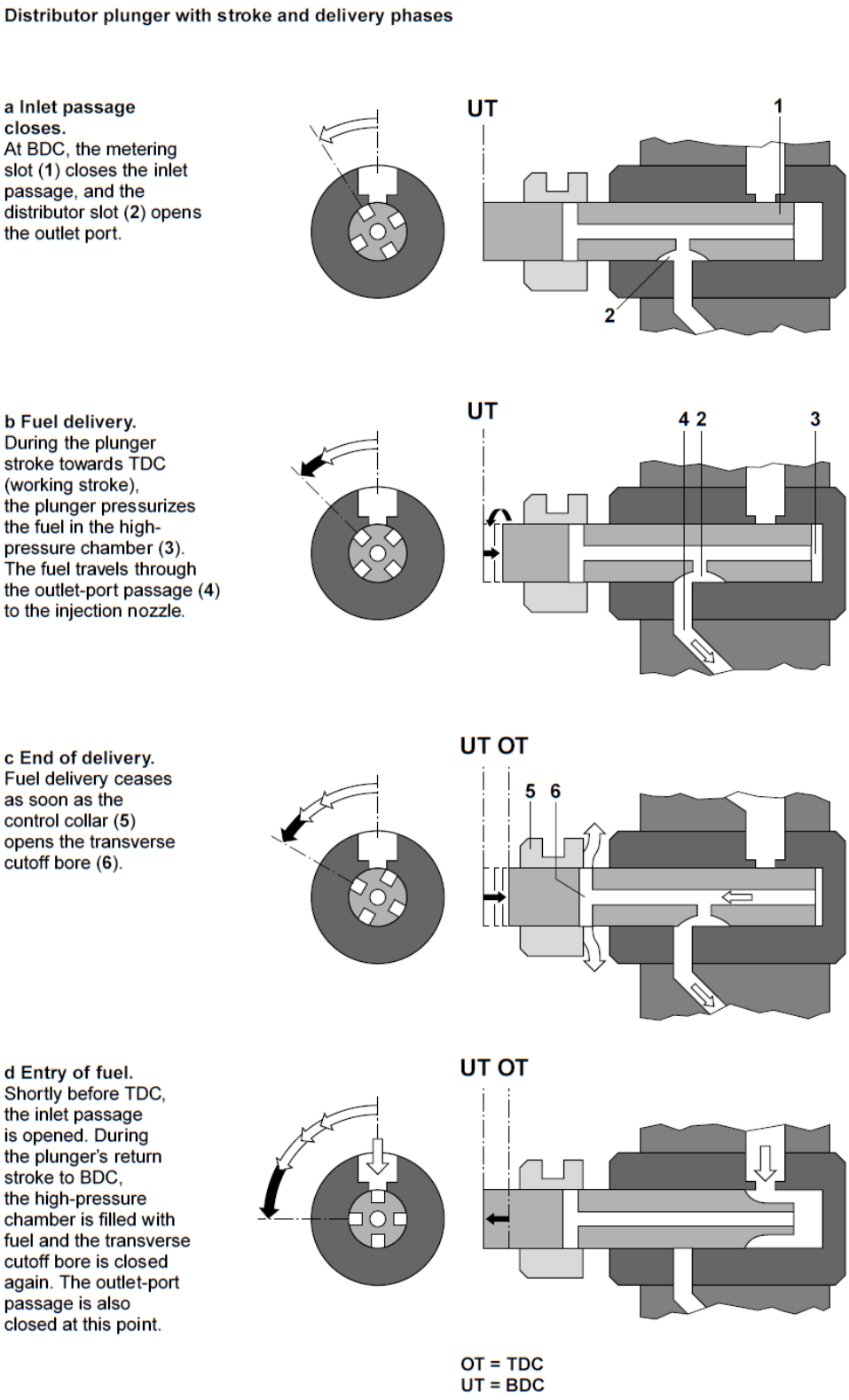


Figure A23: Description of injector pump operation from Bosch manual [98]

Drive Cycle Development

The following are the steps taken to develop the transient portion of the drive cycle.

1. Generate real world data with an on-road vehicle (includes Vehicle Speed, Engine Speed, Throttle Position, and Current Gear of Transmission).
2. Strip real world data of engine braking events using the 'no-load' RPM to throttle position relationship previously obtained from CM12 by mapping engine speed to throttle position without applying the brake (the dynamometer is only capable of slowing the engine down. It can't speed it up.)
3. Interpolate the 'corrected' real world data to a longer time scale so the PID controller of the CM12 can utilize the RPM and throttle position data gathered on the road to adjust the brake load accordingly. In this case the time scale was increased by a factor of 5 and the data was then interpolated to 2Hz.
4. Generate a RPM/ Throttle position scheduler file with the resulting throttle position and RPM setpoints.
5. Run the CM12 using the scheduler file while recording data at a frequency of 2Hz. Since this is a RPM/ Throttle Position scheduler file, PID control of the brake setting will be utilized (Data recorded during this run includes RPM, Throttle Position, and Brake Setting).
6. Using the data collected from the PID controlled run and the 'corrected' real world data, determine idle/ no load events and generate a new brake setting column for which the brake setting during identified idle/ no load events are 3% (this is not 0% because, when a vehicle is in gear, the engine is under slight load.

Only when a vehicle is in neutral or park is the engine not loaded). Here, idle/ no load events were determined by throttle position; when throttle position is zero, load is zero, and, therefore, brake setting was changed to 3% (Figure A24). In actuality, there would be instances when load would be negative (engine braking) when throttle position is zero, but because the CM12 cannot simulate engine braking, brake setting was simply set to 3%.

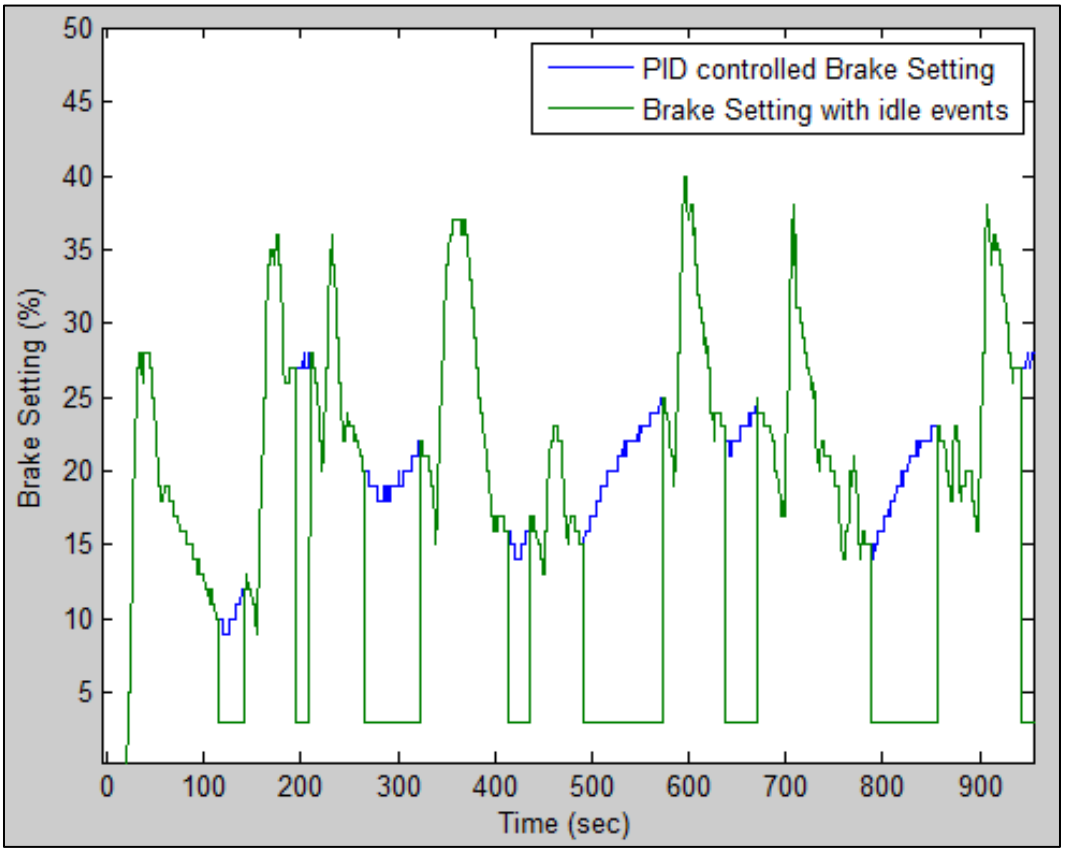


Figure A24: Comparison of PID controlled Brake Setting and intermediate step of Idle Adjustment

7. Overlay 'Brake setting with idle events' from Figure A24 above with Throttle Position to get an indication of what the brake setting should be doing. Using g-input in Matlab, identify start and finish points to connect the peaks in the brake setting data to the 3% load points (Figure A25).

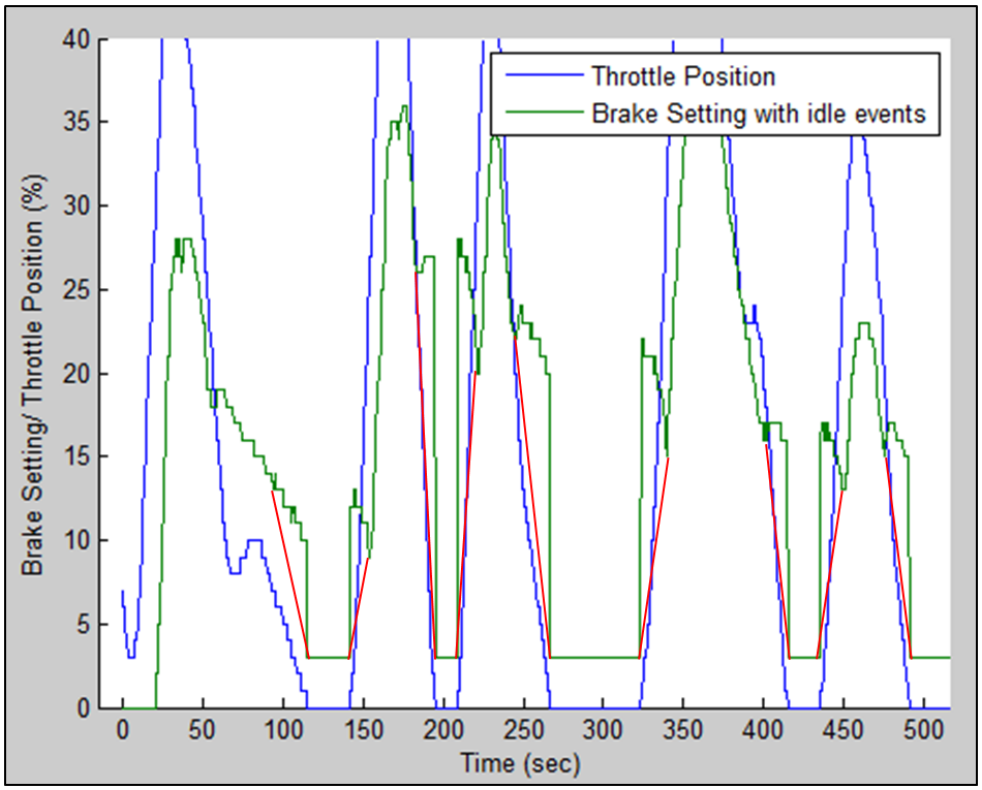


Figure A25: Connecting peaks to idle events. Endpoints of the red lines represent the points identified with g-input.

8. Interpolate between each line endpoint to generate 2Hz brake setting data for the transitions. Replace the transition sections of the green line in Figure A25 with the data generated from the red lines (Figure A26).

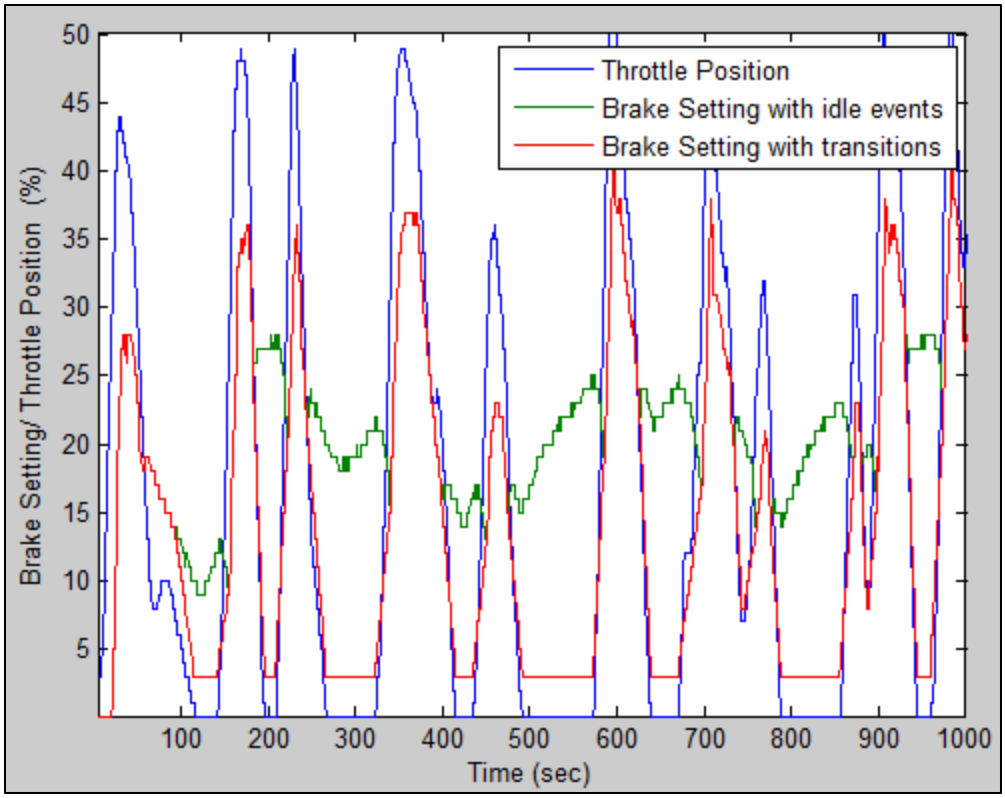


Figure A26: Brake Settings with complete idle event adjustment

9. Now, to focus on correcting the PID overshoot of the brake setting, overlay the drive cycle (desired) RPM and the measured (actual) RPM.

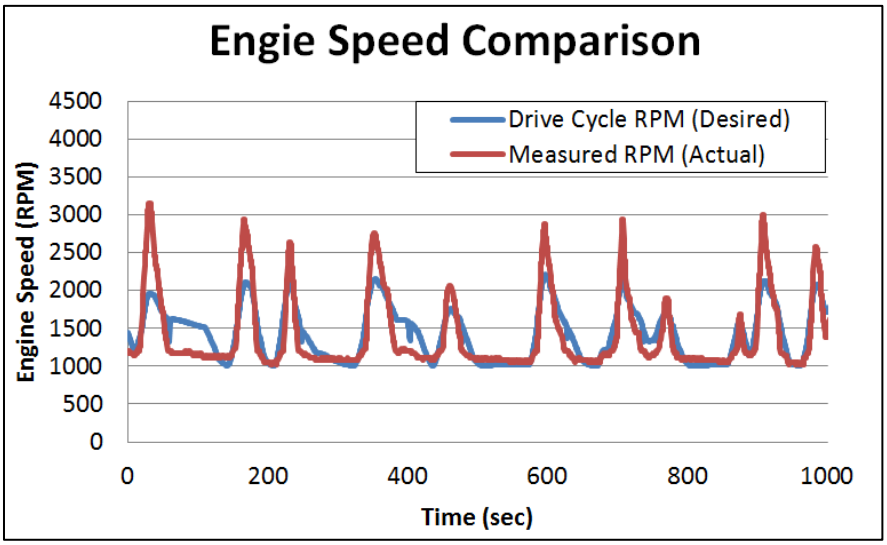


Figure A27: Comparison of desired RPM to measured RPM

10. Now, calculate the difference from actual RPM to desired RPM for those times that the brake setting has not already been adjusted – primarily the peaks of the brake setting data. Find the maximum RPM difference in this range.
11. Develop an algorithm to adjust the brake settings to align measured RPM with actual RPM. When measured RPM is greater than desired RPM, increase brake setting and vice versa. In this instance the maximum RPM difference was 1216. The maximum change in brake setting was set to 10%. The algorithm used was:

$$Amount\ to\ add\ to\ brake\ setting = round\left(\frac{\Delta RPM}{120} - 0.01\right)$$

12. To apply this algorithm, determine the maximum actual RPM value for the time steps that have already been adjusted for idle events. Apply this algorithm only to time steps with RPM values above this value.
13. Generate a new brake setting/ throttle position scheduler file with these modified brake settings and the original throttle positions and run it on the CM12.
14. Repeat steps 10 – 13 until the measured RPM is sufficiently close to the desired RPM.
15. Once engine speed match is complete, interpolate drive cycle back to a shorter time span to better simulate real-world driving.

MatLab Code

The MatLab code for this research was originally written by Tyler Feralio. Some of the code was subsequently modified by Karen Sentoff to incorporate FTIR data.

'Code_1_Raw_Processing_28JUL2014.m' retrieves the raw data from each instrument one run at a time, applies calibration equations where necessary, concatenates

blank and run data where necessary, and outputs new files for each instrument in a common format. 'CODE_2_Blank_Correction.m' retrieves the EEPS and FTIR data generated in Code 1 and performs the blank correction (see 'PN Data and Blank Correction' section in the Appendix for more detail). Once blank correction is complete, the code outputs blank corrected run (engine on) data files associated with both the EEPS and the FTIR. 'CODE_3_Time_Alignment_12JUN2014.m' retrieves the Armfield, VCDS, and Labview files output from Code 1 and the EEPS and FTIR data output from Code 2 for time alignment. See the 'Temporal Alignment' section of the Appendix for more detail on how this was performed. Once time alignment was complete, Code 3 generated one new file for each run that contained data from all instruments that share one time stamp. 'CODE_4_26JUN2014.m' retrieves all of the individual run files generated by Code 3, concatenates them into one large data set, and outputs the concatenated dataset to one large .txt file.

Code_1_Raw_Processing_28JUL2014.m


```
%Tyler Ferallo
%CM12 Engine Dyno Data compiler for 2013 transient / steady state test sequence
%Applies calibration EQs and concatonates raw data where necessary and ouputs
%raw physical values with formatted time stamps for the following instruments

%Armfield
%Scantool (VCDS)
%Labview
%EEPS
%FTIR

%Last Updated 03DEC2013 by TF
% KMS Updated 17JUN2014
% KMS Updated 9/2//2014

close all; clear all; clc;

%% Prompt user for run type:
% a. manual input of single run, or
% b. automated processing of full series

runType = input('Would you like to process a single run (type S) or the full series (type↙
F)? ','s');

% define inputs if manual input for a single run...
if strmatch('S',runType) == 1;
    DateStr = input('Please type the date string (ddmmmyyyy) for the desired run: ','s');
    DateStrAll = DateStr;
    FuelType = input('Please type the fuel type (B###) for the desired run: ','s');
    RunNum = input('Please type the run number of the day (1 or 2) for the desired run:↙
','s');
    FAstr = input('Please type the feedstock association for the desired run: ','s');
    % define inputs if full series to be processed automatically...
elseif strmatch('F',runType) == 1;
    % grab run info file data
    InfoNumCols = 5;
    InfoDelim = '\t';
    InfoFileID = fopen('Z:\SP2_ArmfieldEngine\CM12_data\2014\BiodieselData_RunInfo.↙
txt','r');
    InfoHeaders = textscan(InfoFileID,repmat('%s',1,InfoNumCols),1,'delimiter',↙
InfoDelim,'CollectOutput',1);
    InfoData = textscan(InfoFileID,repmat('%s',1,InfoNumCols),'delimiter',InfoDelim);
    % assign data to header variable names
    InfoHeaderVar = genvarname(InfoHeaders{1,1});
    for ind = 1:size(InfoData,2);
        eval([InfoHeaderVar{1,ind} '=InfoData{1,ind};'])
    end
    fclose(InfoFileID);
end

%% Loop through all runs or single run depending on user input
```



```
for indCur = 1:size(DateStrAll);
    if strmatch('F',runType) == 1;
        DateStr = DateStrAll(indCur);
        FuelType = FuelTypeAll(indCur);
        RunNum = RunNumAll(indCur);
        FAstr = FAstrAll (indCur);
    end

    %% Manual Input

    % % Input needed to build run ID
    % %Date data was collected
    % DateStr='15MAY2014';
    % %Fuel used for data collection
    % FuelType='B020';
    % %Run of the day
    % RunNum='1';
    % %Feedstock association -- 'WVO' or 'SOY'
    % FAstr = 'SOY';

    %% Armfield Sampling Rate
    %Hz at which Armfield is logging
    AHz = 2;

    %% Begin processing
    % Defining a Nominal Value to Feedstock Association
    if strcmp(FAstr,'WVO') == 1
        FAnom = 1;
    end

    if strcmp(FAstr,'SOY') == 1
        FAnom = 2;
    end

    % Percentage of Biodiesel in fuel
    BioPer = str2num(strrep(FuelType,'B',''));

    %Reference Date for second of the year calcualtion
    RefDateVec = [2013 1 1 0 0 0];
    % RefDateStr = datestr(RefDateVec,'mmmm dd, yyyy HH:MM:SS.FFF');
    RefDateNum = datenum(RefDateVec);

    %% Begin program

    % Generating datevec and datenum using input data
    DateVec=datevec(DateStr);
    %Retaining only the YYYY MM DD data
    DateVec=DateVec(:,1:3);

    %Using user input data to generate file names
    RunID=strcat(RunNum,'_',DateStr,'_',FuelType);
```

```
    %% Defining directory names to retrieve data from
    %Setting Directory that matlab files are in as the Parent Directory
    PD=cd;
    %Generating month portion of file path needed to retrieve data
    month=strcat(strrep(sprintf('%0.1f',(DateVec(1,2))/10),'.',''),'_',DateStr(1,3:5));
    %Defining directories:
    %Main Data Directory path
    Dir=strcat('H:\SP2_ArmfieldEngine\CM12_data\', num2str(DateVec(:,1)),'\', month,'\',↙
DateStr);
    %Checking if multipule runs were made in the same day by looking for
    %subfolders
    mr=exist(strcat(Dir,'\',RunID));
    %deciding which directory to pull data from
    if mr>0
        %Defining file paths if more than one run was collected in the same day
        TDataD=(strcat(Dir,'\',RunID,'\Instrument_Data'));
        EDataD=(strcat(Dir,'\',RunID,'\Instrument_Data\EEPS'));
        ADataD=(strcat(Dir,'\',RunID,'\Instrument_Data\Armfield'));
        SDataD=(strcat(Dir,'\',RunID,'\Instrument_Data\Scantool'));
        LDataD=(strcat(Dir,'\',RunID,'\Instrument_Data\Labview'));
        FDataD=(strcat(Dir,'\',RunID,'\Instrument_Data\FTIR'));
        EPlotD=(strcat(Dir,'\',RunID,'\Data_Plots\EEPS'));
        OPlotD=(strcat(Dir,'\',RunID,'\Data_Plots\Operational'));
        EOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\EEPS\Matlab_Output'));
        AOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Armfield\Matlab_Output'));
        SOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Scantool\Matlab_Output'));
        LOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Labview\Matlab_Output'));
        FOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\FTIR\Matlab_Output'));
    else
        %Defining file paths if only one run was collected in the same day
        TDataD=(strcat(Dir,'\Instrument_Data'));
        EDataD=(strcat(Dir,'\Instrument_Data\EEPS'));
        ADataD=(strcat(Dir,'\Instrument_Data\Armfield'));
        SDataD=(strcat(Dir,'\Instrument_Data\Scantool'));
        LDataD=(strcat(Dir,'\Instrument_Data\Labview'));
        FDataD=(strcat(Dir,'\Instrument_Data\FTIR'));
        EPlotD=(strcat(Dir,'\Data_Plots\EEPS'));
        OPlotD=(strcat(Dir,'\Data_Plots\Operational'));
        EOutputD=(strcat(Dir,'\Instrument_Data\EEPS\Matlab_Output'));
        AOutputD=(strcat(Dir,'\Instrument_Data\Armfield\Matlab_Output'));
        SOutputD=(strcat(Dir,'\Instrument_Data\Scantool\Matlab_Output'));
        LOutputD=(strcat(Dir,'\Instrument_Data\Labview\Matlab_Output'));
        FOutputD=(strcat(Dir,'\Instrument_Data\FTIR\Matlab_Output'));
    end


    % KMS added defining file paths if the FTIR data has been reprocessed
    FDataOrig = FDataD;
    FRepro = exist(strcat(FDataD,'\',RunID,'_FTIR_Reprocess'));
    FDataDRepro = strcat(FDataD,'\',RunID,'_FTIR_Reprocess');
```



```
    FRepro3 = exist(strcat(FDataDRepro,RunID,'_FTIR_1_Reprocess.prn'));
    if FRepro > 0 && FRepro3 ~= 0;
        FDataD = strcat(FDataD,'\',RunID,'_FTIR_Reprocess');
    end
    % FRepro3 = exist(strcat(FDataD,RunID,'_FTIR_1_Reprocess.prn'));

    % Checking to see if folders already exist to deposit Matlab output in.  If
    % they don't, Matlab will generate them.
    InstLet={'E' 'A' 'S' 'L' 'F'};
    for i=1:size(InstLet,2)
        var1=strcat(InstLet{i},'Out');
        var2=strcat(InstLet{i},'OutputD');
        %checking to see if 'Matlab_Output' folder exists
        eval(strcat(var1,'=exist(',var2,');'));
        %if it does not exist, create it
        if eval(strcat(var1,'==0;'));
            eval(strcat('mkdir(',var2,');'));
        else
        end
    end

    %% Construction of File Names to look for:
    % KMS moved down to determine if there was a reprocessing of the FTIR data
    % or the original data should be used

    % Timing
    TimeFile = strcat(RunID,'_Timing.xls');

    % Emissions Data
    % FTIR
    if FRepro > 0 && FRepro3 ~= 0;
        FTfilename = {strcat(RunID,'_FTIR_1_Reprocess.prn'), strcat↙
(RunID,'_FTIR_2_Reprocess.prn'), strcat(RunID,'_FTIR_3_Reprocess.prn'), strcat↙
(RunID,'_FTIR_4_Reprocess.prn'), strcat(RunID,'_FTIR_5_Reprocess.prn')};
        if FAnom == 1; % wvo
            FNumCols = 32;
        elseif FAnom == 2; %soy
            FNumCols = 33;
        end
    elseif FRepro > 0 && FRepro3 == 0;
        FTfilename = {strcat(RunID,'_FTIR_1.prn'), strcat(RunID,'_FTIR_2.prn'), strcat↙
(RunID,'_FTIR_3_Reprocess.prn'), strcat(RunID,'_FTIR_4.prn'), strcat(RunID,'_FTIR_5.↙
prn')};
        if FAnom == 1; % wvo
            FNumCols = 32;
            FNumColsQA = 43;
        elseif FAnom == 2; %soy
            FNumCols = 33;
            FNumColsQA = 44;
        end
    else
```

```
        FTfilename = {strcat(RunID,'_FTIR_1.prn'), strcat(RunID,'_FTIR_2.prn'), strcat
(RunID,'_FTIR_3.prn'), strcat(RunID,'_FTIR_4.prn'), strcat(RunID,'_FTIR_5.prn')};
        FNumCols = 43;
    end
    % EEPS
    EEPSfilename = {strcat(RunID,'_EEPS_1.txt'),strcat(RunID,'_EEPS_2.txt'),strcat
(RunID,'_EEPS_3a.txt'),strcat(RunID,'_EEPS_3b.txt'),strcat(RunID,'_EEPS_3c.txt'),strcat
(RunID,'_EEPS_3d.txt'),strcat(RunID,'_EEPS_4.txt'),strcat(RunID,'_EEPS_5.txt')};

    % Operational
    AFfilename=strcat(RunID,'_Armfield_ML.xls');
    LVfilename=strcat(RunID,'_Labview_ML.xls');
    STfilename=strcat(RunID,'_Scantool_ML.xls');

    %% Changing Directory to retrieve Timing Data
    cd(TDataD);
    % Reading Timing Data for the Run
    Timing = importdata(TimeFile,'\t');
    TimeStr = Timing.Sheet1(2:end,2);
    TimeVec = datevec(TimeStr);% Generating Time Vec for all timing events
    DateVecMesh = meshgrid(DateVec, 1:size(TimeStr,1));
    TimeVec(:,1:3) = DateVecMesh;% Appending the correct year-month-day to the Time
Vectors
    TimeNum = datenum(TimeVec); % Generating Time Numbers for all timing events
    for i=1:size(TimeStr,1)% Calculating the minutes from scheduler start for all timing
events (Elapse Time)
        TimeEla(i,1) = (60*TimeVec(i,4)+TimeVec(i,5)+TimeVec(i,6)/60)-(60*TimeVec(4,4)
+TimeVec(4,5)+TimeVec(4,6)/60);
    end

    % Counting the rows to each scan start
    ModeNum = size(TimeNum,1)-2;

    % Generating vector of mode bound times
    %datenum format for axes with time of day
    TickTime=TimeNum(2:end,1)';
    %elapse time elapse time axes needed for overlaying multiple runs
    ElapseTickTime = TimeEla(2:end,1)';

    %Hour Minute Second Header for data output
    hmsHeader = {'Hr', 'Min', 'Sec'};

    %Calculating Fuel Volume
    %Density (lbs/gal)
    B100Den = 7.314;
    B000Den = 6.779;
    Ratio = str2num(strrep(FuelType,'B',''))/100;
    Den = Ratio*B100Den + (1-Ratio)*B000Den;

    q = strcmp(FuelType, 'BurnOut');
    if q == 1
```



```
        Den = B000Den;
    end

    %% Retrieving Armfield data
    % Changing to the directory containing the Armfield data
    cd(ADataD);

    %list files in current directory
    DirCon=ls;
    %Checking for an Armfield File match
    AF=strmatch(AFfilename, DirCon,'exact');

    %If there is an Armfield File, retrieve and plot
    if AF>0
        %Read Armfield Data with 'Sample No' column formated to 'general'
        %-----------------------------------------------------------
        Armfield = importdata(AFfilename,'\t',1);
        %Set headers to a variable
        AFHeaders = {'DateNum','Elapse Time', 'SOY_AF', 'Run of the Day', 'Feedstock
Association', 'Bio %', 'Phase'  Armfield.textdata{1,4:5}, Armfield.textdata{1,7},
Armfield.textdata{1,9:10}, 'Water Inlet Temp', ' Water Outlet Temp', 'Exhaust Manifold
Temp', Armfield.textdata{1,14:15}, Armfield.textdata{1,18}, '% Load [%]' };
        %Removing Line Breaks
        AFHeaders = strrep(AFHeaders, char(010),'');
        %Determine number of rows in Armfield data
        AFNumR=sum(~isnan(Armfield.data(:,1)));

        % Generate an Armfield data matrix
        AFData = zeros(AFNumR, 12+7);
        %Set Armfield data to a variable
        AFData(:,7:end-1) = [Armfield.data(1:AFNumR,1), Armfield.data(1:AFNumR,3:4),
Armfield.data(1:AFNumR,6), Armfield.data(1:AFNumR,8:14), Armfield.data(1:AFNumR,17)] ;
        % Correcting Torque data by subtracting average torque measured during
        % initial idle period
        %------
        % Count the number of rows until throttle changes from zero
        t = find(AFData(:,8)<2);
        u=0;
        for i= 1:size(t,1)
            if t(i,1) == i
                u=u+1;
            end
        end
        %------
        %Averageing Torque values while engine is running above 900RPM and
        %throttle is zero during initial idle period
        v=1;
        for i=1:u
            if AFData(i,16)>900
                w(v,1)=AFData(i,17);
                v=v+1;
```

```
            end
        end
        % Subtracting Torque Measured during initial idle from torque column
        % (Taring)
        AFData(:,17) = AFData(:,17)- mean(w);

        %Generating date num time stamps - AFData(i,1) is sample number so it
        %is devided by AHz
        for i=1:AFNumR
            %Generating datenum time stamp
            AFData(i,1)=TimeNum(1,1)+datenum([0 0 0 0 0 AFData(i,7)/AHz]);
            %Calculating %Load
            if AFData(i,16) < 3624
                AFData(i,19) = 100*(AFData(i,17)/(-3.45924e-18 * AFData(i,16)^6 + 4.52412↙
e-14 * AFData(i,16)^5 - 2.35163e-10 * AFData(i,16)^4 + 6.16033e-7 * AFData(i,16)^3 -↙
8.53826e-4 * AFData(i,16)^2 + 5.99797e-1 * AFData(i,16) - 4.78081e1));
            else
                AFData(i,19) = 100*(AFData(i,17)/(1.27816e-4 * AFData(i,16)^2 - 1.28063e0↙
* AFData(i,16) + 3.07709e3));
            end
            %Generating Elapse Time Time Stamps (minutes from scheduler initiation)
            AFDateVec = datevec(AFData(i,1));
            AFData(i,2) = (3600*AFDateVec(1,4)+60*AFDateVec(1,5)+AFDateVec(1,6))-(3600↙
*TimeVec(4,4)+60*TimeVec(4,5)+TimeVec(4,6));
        end

        % Adding 'SOY' column
        AFData(:,3) = (AFData(:,1)-RefDateNum)*86400;
        % Run number column
        AFData(:,4) = str2num(RunNum);
        % Biodiesel association
        AFData(:,5) = FAnom;
        % Biodiesel percentage
        AFData(:,6) = BioPer;

        % Clearing the column that contained sample number and replacing it
        % with zeros so it can be filled with Phase info
        AFData(:,7) = 0;

        % Defining Mode Numbers - 0 = engine off; 1 = initial idel; 2 = warm up; 3 =
        % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
        % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
        % Stabilized; 10 = Cool down Idle
        PhaseNum = [1:size(Timing.Sheet1,1)-3];

        ARunPhase = zeros(AFNumR,1);
        for k=1:ModeNum+1
            n=0;
            for r=1:AFNumR
                if k < ModeNum+1
                    if AFData(r,1)>=TickTime(:,k) && AFData(r,1)<TickTime(:,k+1)
```



```
                        AFData(r,7) = PhaseNum(1,k);
                    end
                end
            end
        end



        %Generating a File of engine on data only
        v = 1;
        for i = 1:AFNumR
            if AFData(i,16)> 800
                AFEngOnData(v,:)=AFData(i,:);
                v = v+1;
            end
        end

        AFAve(1,:) = mean(AFEngOnData(:,8:19));
        AFAve(2,:) = std(AFEngOnData(:,8:19));



        %Printing Armfield commencement and termination times to command window
        ArmfieldCommence=datestr(TimeNum(1,1), 'HH:MM:SS.FFF')
        ArmfieldTerminate=datestr(AFData(end,1), 'HH:MM:SS.FFF')
        % % %     AFMINMAX(1,:) = min(AFData(:,:));
        % % %     AFMINMAX(2,:) = max(AFData(:,:));
        % % %     AFOffset(1,:) = .1*(AFMINMAX(2,:) - AFMINMAX(1,:));
        % % %     AFMINMAX(1,:) = AFMINMAX(1,:) - AFOffset(1,:);
        % % %     AFMINMAX(2,:) = AFMINMAX(2,:) + AFOffset(1,:);
        % % %
        % % %     %Specifying Limits for Exhaust Temp Plot
        % % %     AFMINMAX(:,15) = [0; 450];
        % % %     if q == 1
        % % %     AFMINMAX(:,15) = [0; 800];
        % % %     end
        % % %     %Specifying Limits for Engine Speed Plot
        % % %     AFMINMAX(:,16) = [750; 4000];
        % % %     %Specifying Limits for Torque Plot
        % % %     AFMINMAX(:,17) = [-5; 135];
        % % %     %Specifying Limits for % Load Plot
        % % %     AFMINMAX(:,19) = [-5; 120];
        % % %
        % % %
        cd(AOutputD);
        %Outputing Armfield Data
        ADataName=strcat(RunID,'_AFOut.xls');%%', upper(datestr(now, ' ddmmmyyyy')),'
        Ahms = [datevec(AFData(:,1))];

        %****Using the standard xlswrite in Matlab***
        xlswrite(ADataName,AFHeaders(1,8:19),'AveEngOnData','B1');
```

```
        xlswrite(ADataName,{'Ave'; 'StDev'},'AveEngOnData','A2');
        xlswrite(ADataName,AFAve,'AveEngOnData','B2');

        xlswrite(ADataName,['Yr', 'Mon' 'Day' 'Hr', 'Min', 'Sec', AFHeaders]↙
,'AllData','A1');
        xlswrite(ADataName,Ahms,'AllData','A2');
        xlswrite(ADataName,AFData,'AllData','G2');
    end


    %% Retrieving Scantool data
    % Changing to the directory containing the Scantool data
    cd(SDataD);

    %list files in current directory
    DirCon=ls;
    %Checking for a Scantool File match
    ST=strmatch(STfilename, DirCon,'exact');

    %If there is a Scantool File, retrieve and plot
    if ST>0
        %Retrieving Scantool Data
        Scantool = importdata(STfilename,'\t',7);
        %Defining Headers
        STHeaders = {'DateNum','Elapse Time', 'SOY_ST', 'Run of the Day', 'Feedstock↙
Association', 'Bio %', 'Phase', 'Engine Speed (RPM)', 'Start of Inj (°BTDC)', 'Throt Pos↙
(%)', 'Inj Qty (mg/stroke)', 'Atmos P (mbar)', 'Coolant T (°C)', 'Intake T (°C)', 'Fuel T↙
(°C)'};
        % Defining number of rows containing data (7 header lines...)
        STNumR = size(Scantool.data,1)-7;
        STData = zeros(STNumR, 9+6);

        %Defining a variable for the data
        STData(:,7:end)=[Scantool.data(8:end,1:5), Scantool.data(8:end,7:10)];

        %Converting Engine Speed from Hexidecimal to RPM
        STData(:,8)=STData(:,8).*21;
        %Converting SOI from Hexidecimal to °BTDC
        STData(:,9)=STData(:,9).*(0.08)-3.12;
        %Converting Throt Pos from Hexidecimal to %
        STData(:,10)=STData(:,10)./2.55;
        %Converting Inj Qty from Hexidecimal to mg/stroke
        STData(:,11)=STData(:,11).*(0.2);
        %Converting Atmos P from Hexidecimal to mbar
        STData(:,12)=STData(:,12).*(4.914)+7.3506;
        %Converting Coolant T from Hexidecimal to °C
        STData(:,13)=STData(:,13).*(-0.68878)+135.983;
        %Converting Intake T from Hexidecimal to °C
        STData(:,14)=STData(:,14).*(-0.68182)+135.191;
        %Converting Fuel T from Hexidecimal to °C
        STData(:,15)=STData(:,15).*(0.5625)-30.4875;
```



```
        %Pulling start time from scantool file
        TimeSStr=Scantool.textdata(1,5);
        %Correcting format of start time
        TimeSStr(1,9)='.';
        %generating date vec for start time
        TimeSVec=[DateVec, str2num(strrep(TimeSStr, ':', ' '))];
        %generating date num for start time
        TimeSNum=datenum(TimeSVec);
        %Calculating Date Num Time Stamps
        for i=1:size(STData,1)
            STData(i,1)=TimeSNum+datenum([0 0 0 0 0 STData(i,7)]);
            %Generating Elapse Time (minutes from scheduler initiation)
            STDateVec = datevec(STData(i,1));
            STData(i,2) = (3600*STDateVec(1,4)+60*STDateVec(1,5)+STDateVec(1,6))-(3600↙
*TimeVec(4,4)+60*TimeVec(4,5)+TimeVec(4,6));
        end

        % Adding 'SOY' column
        STData(:,3) = (STData(:,1)-RefDateNum)*86400;
        % Run number column
        STData(:,4) = str2num(RunNum);
        % Biodiesel association
        STData(:,5) = FAnom;
        % Biodiesel percentage
        STData(:,6) = BioPer;

        % Clearing the column that contained original scantool time and replacing it
        % with zeros so it can be filled with phase info
        STData(:,7) = 0;

        % Defining Mode Numbers - 0 = engine off; 1 = initial idel; 2 = warm up; 3 =
        % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
        % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
        % Stabilized; 10 = Cool down Idle
        PhaseNum = [1:size(Timing.Sheet1,1)-3];

        STNumR = size(STData,1);

        SRunPhase = zeros(STNumR,1);
        for k=1:ModeNum+1
            n=0;
            for r=1:STNumR
                if k < ModeNum+1
                    if STData(r,1)>=TickTime(:,k) && STData(r,1)<TickTime(:,k+1)
                        STData(r,7) = PhaseNum(1,k);
                    end
                end
            end
        end
```

```
        %Generating a File of engine on data only (scantool typically only
        %logs during engine on anyway but put this in just in case)
        v = 1;
        for i = 1:STNumR
            if STData(i,8)> 800
                STEngOnData(v,:)=STData(i,:);
                v = v+1;
            end
        end

        STAve(1,:) = mean(STEngOnData(:,8:15));
        STAve(2,:) = std(STEngOnData(:,8:15));


        %printing scantool start time to command window
        ScantoolCommence=datestr(TimeSNum, 'HH:MM:SS.FFF')
        %printing scantool termination time to command window
        ScantoolTerminate=datestr(STData(end,1), 'HH:MM:SS.FFF')
        % % %     %Scantool Min/Max Values
        % % %     STMINMAX(1,:) = min(STData(:,:));
        % % %     STMINMAX(2,:) = max(STData(:,:));
        % % %     STOffset(1,:) = .1*(STMINMAX(2,:) - STMINMAX(1,:));
        % % %     STMINMAX(1,:) = STMINMAX(1,:) - STOffset(1,:);
        % % %     STMINMAX(2,:) = STMINMAX(2,:) + STOffset(1,:);
        % % %
        % % %     %Specifying Limits for Atmospheric Pressure Plot
        % % %     STMINMAX(:,8) = [900; 1100];


        cd(SOutputD);
        %Outputing Armfield Data
        SDataName=strcat(RunID,'_STOut.xls');%%', upper(datestr(now, ' ddmmmyyyy')),'
        Shms = [datevec(STData(:,1))];

        %****Using the standard xlswrite in Matlab***
        xlswrite(SDataName,STHeaders(1,8:15),'AveEngOnData','B1');
        xlswrite(SDataName,{'Ave'; 'StDev'},'AveEngOnData','A2');
        xlswrite(SDataName,STAve,'AveEngOnData','B2');

        xlswrite(SDataName,['Yr', 'Mon' 'Day' 'Hr', 'Min', 'Sec', STHeaders]↙
,'AllData','A1');
        xlswrite(SDataName,Shms,'AllData','A2');
        xlswrite(SDataName,STData,'AllData','G2');
    end
    %---------------------------------------

    %% Retrieving Labview data
    % Changine to the directory containing the Labview data
    cd(LDataD);

    %list files in current directory
```



```
    DirCon=ls;
    %Checking for a Labview File match
    LV=strmatch(LVfilename, DirCon,'exact');

    %If there is an Armfield File, retreieve and plot
    if LV>0
        %Retrieving Labview Data
        Labview = importdata(LVfilename,'\t',5);
        %Define headers
        LVHeaders = repmat({'o'},1,3+15+3);
        LVHeaders =['DateNum', 'Elapse Time', 'SOY_LV', 'Run of the Day', 'Feedstock↙
Association', 'Bio %', 'Phase', strrep(Labview.textdata(4,2:15), char(095),char↙
(032)),'Totalizer P (PSIG)' 'OMEGA DR' 'MAG DR' 'Cum Fuel Con (Gallons)'];
        LVUnits = {'MatLab', 'min', 'sec', '-', '-', '%', '-', 'lbs', 'SLPM', 'SLPM',↙
'LPM', '°C', '%RH', 'PSIG', '°C', '°C', '°C', '°C', 'SLPM', 'SLPM', '°C', 'PSIG', '-',↙
'-', 'Gallons'};
        %Setting the number of rows of labview data to a variable
        LVNumR=size(Labview.data,1);
        %Predefining a matrix of the appropriate size for the labview data
        LVData=zeros(LVNumR,7+15+3);
        %Inserting sensor data
        LVData(:,8:22)=Labview.data(:,:);

        %Converting Dilution Air Flow from MAG Voltage to SLPM **Updated 25NOV2013
        %with Calibration performed on 14NOV2013**
        LVData(:,9)=LVData(:,9)*25.236+0.4925;
        %Converting Aerosol Inlet Flow from Voltage to SLPM **Updated 25NOV2013
        %with Calibration performed on 13NOV2013**
        LVData(:,10)=LVData(:,10)*(0.7381)-0.6433;
        %Converting Pitot Flow from Voltage to SLPM
        LVData(:,11)=real(sqrt(6888000*(LVData(:,11)-0.995)));
        %Converting Ambient Temp from Voltage to °C
        LVData(:,12)=LVData(:,12)*(18.863)-34.457;
        %Converting Ambient RH from Voltage to %
        LVData(:,13)=LVData(:,13)*(25)-25;
        %Converting Dilution Air P from Voltage to PSIG
        LVData(:,14)=LVData(:,14)*(25.0193)-25.167;
        %Converting Dilution Air T from Voltage to °C
        LVData(:,15)=LVData(:,15)*(94.753)+1.8464;
        %Converting Fuel T from Voltage to °C
        LVData(:,16)=LVData(:,16)*(94.753)+1.8464;
        %Converting Exhaust T from Voltage to °C
        LVData(:,17)=LVData(:,17)*(94.753)+1.8464;
        %Converting Sample T from Voltage to °C
        LVData(:,18)=LVData(:,18)*(94.753)+1.8464;
        %Converting OMEGA Dil Air Flow from Voltage to SLPM
        LVData(:,19)=LVData(:,19)*(20);
        %Converting Mass Air Flow (MAF) from Voltage to SLPM
        LVData(:,20)=53.135.*(LVData(:,20)).^3+104.38.*(LVData(:,20)).^2+56.724.*(LVData↙
(:,20))-13.702;
        %Converting Aerosol Inlet T from Voltage to °C
```

```
        LVData(:,21)=LVData(:,21)*(94.753)+1.8464;
        %Converting Raw Filter Line Pressure from Voltage to PSIG
        LVData(:,22)=LVData(:,22)*(7.5)-22.2;



        %Pulling commencment of logging from Labview File
        TimeLStr=Labview.textdata{3,2};
        %generating a date vec for the start time
        TimeLVec=[DateVec,str2num([strrep(TimeLStr,':',' ')])];
        %generating a date num for the start time
        TimeLNum=datenum(TimeLVec);

        %Creating Variables for engine on - 30 sec and engine off + 30 sec rows
        LVon = 0;
        LVoff = 0;
        for i=1:LVNumR
            %Generating date num time stamps
            LVElapse(i,:)=str2num([strrep(strrep(Labview.textdata{i+4, 1},'0  ',''),':','↙
')]);
            LVData(i,1)=TimeLNum+datenum([0,0,0,LVElapse(i,:)]);
            %Generating Elapse Time (minutes from scheduler initiation)
            LVDateVec = datevec(LVData(i,1));
            LVData(i,2) = (3600*LVDateVec(1,4)+60*LVDateVec(1,5)+LVDateVec(1,6))-(3600↙
*TimeVec(4,4)+60*TimeVec(4,5)+TimeVec(4,6));
            %Calculating dilution Ratio with OMEGA Mass Flow Data
            LVData(i,23) = (LVData(i,19) + LVData(i,10))/LVData(i,10);
            %Calculating dilution Ratio with MAG Mass Flow Data
            LVData(i,24) = (LVData(i,9) + LVData(i,10))/LVData(i,10);
            % Finding row numbers just before engine on and after engine off
            % (+/- 30 sec)
            if LVData(i,1) < (TimeNum(2,1) - datenum([0 0 0 0 0 30]))
                LVon = LVon+1;
            end
            if LVData(i,1) <= (TimeNum(end,1) + datenum([0 0 0 0 0 30]))
                LVoff = LVoff+1;
            end
        end

        for i = 1:LVNumR
            %Calculating Fuel consumption from Fuel Tank weight in gallons
            LVData(i,25) = (LVData(LVon,8) - LVData(i,8))/Den;
        end


        LVFileStart = 0;
        LVFileEnd = 0;
        for i = 1:LVNumR
            % Finding row numbers just before engine on and after engine off (+/- 30 min)
            if LVData(i,1) < (TimeNum(2,1) - datenum([0 0 0 0 30 0]))
                LVFileStart = LVFileStart+1;
```



```
            end
            if LVData(i,1) <= (TimeNum(end,1) + datenum([0 0 0 0 30 0]))
                LVFileEnd = LVFileEnd+1;
            end
        end

        % Adding 'SOY' column
        LVData(:,3) = (LVData(:,1)-RefDateNum)*86400;
        % Run number column
        LVData(:,4) = str2num(RunNum);
        % Biodiesel association
        LVData(:,5) = FAnom;
        % Biodiesel percentage
        LVData(:,6) = BioPer;


        % Defining Mode Numbers - 0 = engine off; 1 = initial idel; 2 = warm up; 3 =
        % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
        % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
        % Stabilized; 10 = Cool down Idle
        PhaseNum = [1:size(Timing.Sheet1,1)-3];

        LRunPhase = zeros(LVNumR,1);
        for k=1:ModeNum+1
            n=0;
            for r=1:LVNumR
                if k < ModeNum+1
                    if LVData(r,1)>=TickTime(:,k) && LVData(r,1)<TickTime(:,k+1)
                        LVData(r,7) = PhaseNum(1,k);
                    end
                end
            end
        end

        v = 1;
        for i = 1:LVNumR
            if LVData(i,20)> 200
                LVEngOnData(v,:)=LVData(i,:);
                v = v+1;
            end
        end

        LVAve(1,:) = mean(LVEngOnData(:,9:24));
        LVAve(2,:) = std(LVEngOnData(:,9:24));

        % %     %Calculating Average and StDev Dilution Ratio
        % %     AveOMEGADR = mean(LVData(LVon:LVoff,19))
        % %     StDevOMEGADR = std(LVData(LVon:LVoff,19))
        % %     AveMAGDR = mean(LVData(LVon:LVoff,20))
        % %     StDevMAGDR = std(LVData(LVon:LVoff,20))
        % %
```

```
% %     %Calculating Total Fuel Consumption
% %     FuelCon = LVData(LVoff,21) - LVData(LVon,21)

%Printing Commencment time to command window
LabviewCommence=datestr(TimeLNum, 'HH:MM:SS.FFF')
%Printing log termination time to command window
LabviewTerminate=datestr(LVData(end,1), 'HH:MM:SS.FFF')

% %     %Labview Min/Max Values
% %     LVMINMAX(1,:) = min(LVData(LVon:LVoff,:));
% %     LVMINMAX(2,:) = max(LVData(LVon:LVoff,:));
% %     LVOffset(1,:) = .1*(LVMINMAX(2,:) - LVMINMAX(1,:));
% %     LVMINMAX(1,:) = LVMINMAX(1,:) - LVOffset(1,:);
% %     LVMINMAX(2,:) = LVMINMAX(2,:) + LVOffset(1,:);
% %
% %     %Specifying Limits for Dilution Air Flow Plot
% %     LVMINMAX(:,5) = [75; 83];
% %     %Specifying Limits for Dilution Air Pressure Plot
% %     LVMINMAX(:,10) = [28; 32];
% %     %Specifying Limits for Exhaust Temperature Plot
% %     LVMINMAX(:,13) = [0; 275];
% %     if q == 1
% %         LVMINMAX(:,13) = [0; 500];
% %     end
% %     %Specifying Limits for OMEGA Dilution Air Flow Plot
% %     LVMINMAX(:,15) = [76; 90];
% %     %Specifying Limits for MAF Plot
% %     LVMINMAX(:,16) = [-20; 3500];
% %     %Specifying Limits for Aerosol Inlet Temp Plot
% %     LVMINMAX(:,17) = [104; 110];
% %     %Specifying Limits for Totalizer Pressure Plot
% %     LVMINMAX(:,18) = [-3; 0];



cd(LOutputD);
%Outputing Armfield Data
LDataName=strcat(RunID,'_LVOut.xls');%%', upper(datestr(now, ' ddmmmyyyy')),'
Lhms = [datevec(LVData(:,1))];

%****Using the standard xlswrite in Matlab***
xlswrite(LDataName,LVHeaders(1,9:24),'AveEngOnData','B1');
xlswrite(LDataName,LVUnits(1,9:24),'AveEngOnData','B2');
xlswrite(LDataName,{'Ave'; 'StDev'},'AveEngOnData','A3');
xlswrite(LDataName,LVAve,'AveEngOnData','B3');

xlswrite(LDataName,['Yr', 'Mon' 'Day' 'Hr', 'Min', 'Sec',↙
LVHeaders],'AllData','A1');
xlswrite(LDataName,Lhms(LVFileStart:LVFileEnd,:),'AllData','A2');
xlswrite(LDataName,LVData(LVFileStart:LVFileEnd,:),'AllData','G2');
```



```
    end

    %% Retrieving EEPS data
    % Changing to the directory containing the Labview data
    cd(EDataD);

    %list files in current directory
    DirCon=ls;
    %Checking for a Labview File match
    EE=strmatch(EEPSfilename{1,3}, DirCon,'exact');


    if EE>0
        for p = 1:8
            % Detrmining if file exists
            f(1,p) = exist(EEPSfilename{1,p}, 'file');
            if f(1,p) > 0
                fid = fopen(EEPSfilename{1,p});
                if p == 1
                    %Pulling Header from Prerun Instrument Blank File
                    EEtext_format1 = [' %s' repmat(' %*s',1,135) ' %s %s %*s %s %s %*s %s↙
%s %*s'];
                    EEHeader1 = textscan(fid, EEtext_format1,1,↙
'delimiter','\t','Headerlines',12);
                    EEtext_format2 = [' %*s' repmat(' %f',1,32)];
                    EEHeader2 = textscan(fid, EEtext_format2,↙
'delimiter','\t','Headerlines',1);
                    EEHeader2 = cellfun(@num2str,EEHeader2, 'uniformoutput', false);
                    for k = 1:size(EEHeader2,2)
                        EEHeader2{1,k} = [char(039), EEHeader2{1,k}];
                    end

                    % Defining format for input data
                    EEdata_format = [repmat(' %s',1,1) repmat(' %f',1,32) repmat([repmat↙
(' %*s',1,2) repmat(' %f',1,32)],1,3) repmat(' %*s %*s %f',1,3) ' %*s'];
                    z = 2;
                else
                    z = 15;
                end
                % Pulling Instrument data in and concatonating
                EEData1 = textscan(fid, EEdata_format, 'delimiter','\t','Headerlines',z↙
);
                fclose(fid);

                if p == 1
                    % Setting number of columns to a variable
                    EENumC = size(EEData1,2);
                end

                % Finding the minimum number of rows in the data file just in case the is↙
an anomalous 2nd header at the bottom of the file
```

```
                for i = 1:size(EEData1,2)
                    x = 0;
                    if i == 1;
                        EEmin = size(EEData1{1,i},1);
                    else
                        if size(EEData1{1,i},1) < EEmin
                            EEmin = size(EEData1{1,i},1);
                        end
                        %Determining if there are empty cells
                        for j = 1:EEmin
                            x = x + isnan(EEData1{1,i}(j,1));
                        end
                        EEmin = EEmin - x;
                    end
                end

                % Adding a column consisting of Quality Control ID (QC-ID) data -↙
basically the number associated with blank and run files
                phase = [1 2 3 3 3 3 4 5];
                EEData1{1,EENumC+1} = repmat(phase(1,p),EEmin,1);
                if p == 1 % Initiallizing concatination on first pass through loop.
                    % defining the variable to concatonate data under
                    EEData = zeros(EEmin,43);
                    % placing data in the desired columns
                    EEData(:,8) = EEData1{1,end}(1:EEmin,1); % QC-ID
                    for i = 1:32 % Pulling Bin Concentration Data
                        EEData(:,i+8)=EEData1{1,i+1}(1:EEmin,1);
                    end
                    for i = 1:3 % Pulling Tot Conc, Pressure, and Sample Temp Data
                        EEData(:,i+40)=EEData1{1,i+129}(1:EEmin,1);
                    end
                    EEtimeVec = datevec(EEData1{1,1}(1:EEmin,1)); % generating TimeVec

                else % Concatonating data to existing data
                    EEData2 = zeros(EEmin,43);% saving another variable to concatonate
                    EEData2(:,8) = EEData1{1,end}(1:EEmin,1); % QC-ID
                    for i = 1:32 % Bin Concentrations
                        EEData2(:,i+8)=EEData1{1,i+1}(1:EEmin,1);
                    end
                    for i = 1:3 % Pulling Tot Conc, Pressure, and Sample Temp Data
                        EEData2(:,i+40)=EEData1{1,i+129}(1:EEmin,1);
                    end
                    EEData = [EEData; EEData2]; % Conatinating Data
                    EEtimeVec = [EEtimeVec; datevec(EEData1{1,1}(1:EEmin,1))]; %↙
Conatinating TimeVec
                end
            end
        end
        % Apending to header
        EEHeader = ['DateNum', 'Elapse Time', 'SOY_EEPS','Run of the Day', 'Feedstock↙
Association', 'Bio %', 'Phase', 'QC_ID', EEHeader2, strrep(strcat(EEHeader1{1,2},{'_'},↙
```



```
EEHeader1{1,3}),'_',' '),strrep(strcat(EEHeader1{1,4},{'_'}, EEHeader1{1,5}),'_',' '),↙
strrep(strcat(EEHeader1{1,6},{'_'}, EEHeader1{1,7}),'_',' ') ];
        for i = 1: size(EEtimeVec,1)
            EEtimeVec(i,1:3) = DateVec(1,1:3);% Correcting Date portion of Time Vector
        end
        EEData(:,1) = datenum(EEtimeVec);% Saving DateNum

        %Setting number of rows as a variable
        EENumR = size(EEData,1);

        %Adding 'Elapse Time' Column
        for i = 1:EENumR
            %Generating Elapse Time (minutes from scheduler initiation)
            EEData(i,2) = (3600*EEtimeVec(i,4)+60*EEtimeVec(i,5)+EEtimeVec(i,6))-(3600↙
*TimeVec(4,4)+60*TimeVec(4,5)+TimeVec(4,6));
        end

        % Adding 'SOY' column
        EEData(:,3) = (EEData(:,1)-RefDateNum)*86400;
        % Run number column
        EEData(:,4) = str2num(RunNum);
        % Biodiesel association
        EEData(:,5) = FAnom;
        % Biodiesel percentage
        EEData(:,6) = BioPer;

        % Defining Mode Numbers - 0 = engine off; 1 = initial idle; 2 = warm up; 3 =
        % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
        % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
        % Stabilized; 10 = Cool down Idle
        PhaseNum = [1:size(Timing.Sheet1,1)-3];

        for k=1:ModeNum+1
            for r=1:EENumR
                if k < ModeNum+1
                    if EEData(r,1)>=TickTime(:,k) && EEData(r,1)<TickTime(:,k+1)
                        EEData(r,7) = PhaseNum(1,k);
                    end
                end
            end
        end


        cd(EOutputD);
        %Outputing Raw EEPS Data
        EDataName=strcat(RunID,'_EEPSOut.xls');%%', upper(datestr(now, ' ddmmmyyyy')),'
        Ehms = [datevec(EEData(:,1))];

        %****Using the standard xlswrite in Matlab***
        xlswrite(EDataName,['Yr', 'Mon' 'Day' 'Hr', 'Min', 'Sec',↙
EEHeader],'AllData','A1');
```

```
        xlswrite(EDataName,Ehms,'AllData','A2');
        xlswrite(EDataName,EEData,'AllData','G2');

    end


    %% Retrieving FTIR data
    % Changing to the directory containing the FTIR data
    cd(FDataD);

    %list files in current directory
    DirCon=ls;
    %Checking for a Labview File match
    FT=strmatch(FTfilename{1,1}, DirCon,'exact');

    %If there is an FTIR File, retrieve and plot
    if FT>0
        for p = 1:5
            fid = fopen(FTfilename{1,p});
            if fid == -1;
                cd(FDataDRepro)
                fid = fopen(FTfilename{1,p});
                cd(FDataD)
            end
            if p == 1
                if FRepro > 0 && FRepro3 == 0;
                    %Pulling Header from Prerun Instrument Blank File
                    FTtext_format = [repmat('%*s',1,2) repmat(' %s',1,FNumColsQA+1)];
                    FTHeader = textscan(fid, FTtext_format, 1, 'delimiter','\t');
                    % Defining format for input data
                    FTdata_format = [repmat(' %*s',1,2) ' %s' repmat(' %f',1,↙
FNumColsQA)];
                    % Setting number of columns to a variable
                    FTNumC = size(FTHeader,2);
                else
                    %Pulling Header from Prerun Instrument Blank File
                    FTtext_format = [repmat('%*s',1,2) repmat(' %s',1,FNumCols+1)];
                    FTHeader = textscan(fid, FTtext_format, 1, 'delimiter','\t');
                    % Defining format for input data
                    FTdata_format = [repmat(' %*s',1,2) ' %s' repmat(' %f',1,FNumCols)];
                    % Setting number of columns to a variable
                    FTNumC = size(FTHeader,2);
                end
            end
            if FRepro > 0 && FRepro3 == 0 && p ~= 3;
                FTdata_format3 = [repmat(' %*s',1,2) ' %s' repmat(' %f',1,FNumColsQA)↙
repmat(' %*f',1,FNumColsQA-FNumCols)];
                FTData1 = textscan(fid, FTdata_format3, 'delimiter','\t','Headerlines',1↙
);
            elseif FRepro > 0 && FRepro3 == 0 && p == 3;
                cd(FDataDRepro);
```



```
                % Pulling Instrument data in and concatenating
                FTData1 = textscan(fid, FTdata_format, 'delimiter','\t','Headerlines',1↙
);
            end
            fclose(fid);
            cd(FDataD);
            % Finding the minimum number of rows in the data file just in case the is an↙
anomalous 2nd header at the bottom of the file
            for i = 1:size(FTData1,2)
                if i == 1;
                    min = size(FTData1{1,i},1);
                else
                    if size(FTData1{1,i},1) < min
                        min = size(FTData1{1,i},1);
                    end
                end
            end
            % Adding a column consisting of Quality Control ID (QC-ID) data - basically↙
the number associated with blank and run files
            FTData1(1,FTNumC+1) = repmat(p,min,1);
            if p == 1 % Initializing concatenation on first pass through loop.
                % defining the variable to concatenate data under
                FTData = zeros(min,size(FTData1,2)+6);
                % placing data in the desired columns
                FTData(:,8) = FTData1(1,end)(1:min,1); % QC-ID
                for i = 2:size(FTData1,2)-1 % Pulling all other data
                    FTData(:,i+7)=FTData1{1,i}(1:min,1);
                end
                FTtimeVec = datevec(FTData1{1,1}(1:min,1)); % generating TimeVec
            else % Concatenating data to existing data
                FTData2 = zeros(min,size(FTData1,2)+6);% saving another variable to↙
concatonate
                FTData2(:,8) = FTData1{1,end}(1:min,1); % QC-ID
                for i = 2:size(FTData1,2)-1 % Pulling all other data
                    FTData2(:,i+7)=FTData1{1,i}(1:min,1);
                end
                FTData = [FTData; FTData2]; % Conatenating Data
                FTtimeVec = [FTtimeVec; datevec(FTData1{1,1}(1:min,1))]; % Concatenating↙
TimeVec
            end
        end
        % Appending to header
        FTHeader = ['DateNum',  'Elapse Time', 'SOY_FTIR','Run of the Day', 'Feedstock↙
Association', 'Bio %', 'Phase', 'QC_ID',FTHeader(1,2:end) ];
        for i = 1: size(FTtimeVec,1)
            FTtimeVec(i,1:3) = DateVec(1,1:3);% Correcting Date portion of Time Vector
        end
        FTData(:,1) = datenum(FTtimeVec);% Saving DateNum

        %Setting number of rows as a variable
        FTNumR = size(FTData,1);
```

```
        %Adding 'Elapse Time' Column
        for i = 1:FTNumR
            %Generating Elapse Time (minutes from scheduler initiation)
            FTData(i,2) = (3600*FTtimeVec(i,4)+60*FTtimeVec(i,5)+FTtimeVec(i,6))-(3600↙
*TimeVec(4,4)+60*TimeVec(4,5)+TimeVec(4,6));
        end
        % Adding 'SOY' column
        FTData(:,3) = (FTData(:,1)-RefDateNum)*86400;
        % Run number column
        FTData(:,4) = str2num(RunNum);
        % Biodiesel association
        FTData(:,5) = FAnom;
        % Biodiesel percentage
        FTData(:,6) = BioPer;

        % Defining Mode Numbers - 0 = engine off; 1 = initial idle; 2 = warm up; 3 =
        % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
        % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
        % Stabilized; 10 = Cool down Idle
        PhaseNum = [1:size(Timing.Sheet1,1)-3];

        for k=1:ModeNum+1
            for r=1:FTNumR
                if k < ModeNum+1
                    if FTData(r,1)>=TickTime(:,k) && FTData(r,1)<TickTime(:,k+1)
                        FTData(r,7) = PhaseNum(1,k);
                    end
                end
            end
        end


        cd(FOutputD);
        %Outputing FTIR Data
        FDataName=strcat(RunID,'_FTIROut.xls');%%', upper(datestr(now, ' ddmmmyyyy')),'
        Fhms = [datevec(FTData(:,1))];

        %****Using the standard xlswrite in Matlab***
        xlswrite(FDataName,['Yr', 'Mon' 'Day' 'Hr', 'Min', 'Sec', FTHeader↙
{1,:}],'AllData','A1');
        xlswrite(FDataName,Fhms,'AllData','A2');
        xlswrite(FDataName,FTData,'AllData','G2');

    end


    %% Final end for loop through all of the runs or single run depending on user input
end
```

CODE_2_Blank_Correction.m


```
6/16/15 9:24 PM  H:\GP2 Armfie...\CODE 2 Blank Correction.m  1 of 8

%Tyler Peralio
%CM12 Engine Dyno Data compiler for 2013 transient / steady state test sequence
%CODE 2: Retrieves emissions data output by CODE 1 blank corrects it and
%outputs a file with the blank corrected engine on data.

%EEPS
%FTIR

%Last Updated 03DEC2013 by TF
% Last updated 2SEP2014 by KMS (loop through all runs and fix)

close all; clear all; clc;
%% Prompt user for run type:
% a. manual input of single run, or
% b. automated processing of full series

runType = input('Would you like to process a single run (type S) or the full series (type↙
F)? ','s');

% define inputs if manual input for a single run...
if strmatch('S',runType) == 1;
    DateStr = input('Please type the date string (ddmmmyyyy) for the desired run: ','s');
    DateStrAll = DateStr;
    FuelType = input('Please type the fuel type (B###) for the desired run: ','s');
    RunNum = input('Please type the run number of the day (1 or 2) for the desired run:↙
','s');
    FAstr = input('Please type the feedstock association for the desired run: ','s');
    % define inputs if full series to be processed automatically...
elseif strmatch('F',runType) == 1;
    % grab run info file data
    InfoNumCols = 5;
    InfoDelim = '\t';
    InfoFileID = fopen('H:\GP2_ArmfieldEngine\CM12_data\2014\BiodieselData_RunInfo.↙
txt','r');
    InfoHeaders = textscan(InfoFileID,repmat('%s',1,InfoNumCols),1,'delimiter',↙
InfoDelim,'CollectOutput',1);
    InfoData = textscan(InfoFileID,repmat('%s',1,InfoNumCols),'delimiter',InfoDelim);
    % assign data to header variable names
    InfoHeaderVar = genvarname(InfoHeaders{1,1});
    for ind = 1:size(InfoData,2);
        eval([InfoHeaderVar{1,ind} '=InfoData{1,ind};'])
    end
    fclose(InfoFileID);
end

%% Loop through all runs or single run depending on user input
for indCur = 1:size(DateStrAll);
    if strmatch('F',runType) == 1;
        DateStr = DateStrAll(indCur);
        DateStr = DateStr{1,1};
        FuelType = FuelTypeAll(indCur);
        FuelType = FuelType{1,1};
    end
    RunNum = RunNumAll(indCur);
    RunNum = RunNum{1,1};
    FAstr = FAstrAll(indCur);
    FAstr = FAstr{1,1};
end

%% Manual input

% % Input needed to build run ID
% % Date data was collected
% DateStr='25JUN2013';
% % Fuel used for data collection
% FuelType='B000';
% % Run of the day
% RunNum='1';

%% Begin program
% Generating datevec and datenum using input data
DateVec=datevec(DateStr);
% Retaining only the YYYY MM DD data
DateVec=DateVec(:,1:3);

% Using user input data to generate file names
RunID=strcat(RunNum,'_',DateStr,'_',FuelType);

%% Construction File Names to look for
% Timing
TimeFile = strcat(RunID,'_Timing.xls');

% Instruments
Inst = {'EEPS', 'FTIR'};
for i = 1:size(Inst,2)
    Filename{1,i} = strcat(RunID,'_', Inst{1,i},'Out.xls');
end

%% Defining directories names to retrieve data from
% Setting Directory that matlab files are in as the Parent Directory
pd=cd;
% Generating month portion of file path needed to retrieve data
month=strcat(strrep(sprintf('%0.1f',(DateVec(1,2))/10),'.',''),'_',DateStr(1,3:5));
% Defining directories:
% Main Data Directory path
Dir=strcat('H:\GP2_ArmfieldEngine\CM12_data\', num2str(DateVec(:,1)),'\', month,'\',↙
DateStr);
% Checking if multiple runs were made in the same day by looking for
% subfolders
mr=exist(strcat(Dir,'\',RunID));
% deciding which directory to pull data from
if mr>0
    % Defining file paths if more then one run was collected in the same day
```

```
        TDataD=(strcat(Dir,'\',RunID,'\Instrument_Data'));
        EOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\EEPS\Matlab_Output'));
        FOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\FTIR\Matlab_Output'));
    else
        % Defining file paths if only one run was collected in the same day
        TDataD=(strcat(Dir,'\Instrument_Data'));
        EOutputD=(strcat(Dir,'\Instrument_Data\EEPS\Matlab_Output'));
        FOutputD=(strcat(Dir,'\Instrument_Data\FTIR\Matlab_Output'));
    end

    % Changing Diorectory to retrieve Timing Data
    cd(TDataD);
    % Reading Timing Data for the Run
    Timing = importdata(TimeFile,'\t');
    TimeStr = Timing.Sheet1(2:end,2);
    TimeVec = datevec(TimeStr);% Generating Time Vec for all timing events
    DateVecMesh = meshgrid(DateVec, 1:size(TimeStr,1));
    TimeVec(:,1:3) = DateVecMesh;% Appending the correct year-month-day to the Time↙
Vectors
    TimeNum = datenum(TimeVec); % Generating Time Numbers for all timing events
    for i=1:size(TimeStr,1)% Calculating the minutes from scheduler start for all timing↙
events (Elapse Time)
        TimeEla(i,1) = (60*TimeVec(i,4)+TimeVec(i,5)+TimeVec(i,6)/60)-(60*TimeVec(4,4)↙
+TimeVec(4,5)+TimeVec(4,6)/60);
    end

    % Determining the number of modes from the time file
    ModeNum = size(TimeNum,1)-2;

    % Generating vector of mode bound times
    % datenum format for axes with time of day
    TickTime=TimeNum(2:end,1)';
    % elapse time elapse time axes needed for overlaying multiple runs
    ElapseTickTime = TimeEla(2:end,1)';
    % Defining Quality control Names
    QCName = {'Pre-IB ' 'Pre-TB ' 'Engine On ' 'Post-TB ' 'Post-IB '};

    %% Retrieving EEPS data
    % Changing to the directory containing the EEPS data
    cd(EOutputD);

    % list files in current directory
    DirCon=ls;
    % Checking for an EEPS File match
    EE=strmatch(Filename{1,1}, DirCon,'exact');

    %% If there is an EEPS File, retreieve
    if EE>0
        % Read EEPS Data
        %-------------------------------------------------------------
        [EEPS.data, EEPS.txt] = xlsread(Filename{1,1}{:},'AllData'); % Read in Data
```



```
        EENumR = size(EEPS.data,1); % Set number of rows to a variable
        EEStart = zeros(1,5); % Initialize a variable to save start rows
        EEEnd = zeros(1,5); % Initialize a variable to save end row

        for k = 1:32
            EEPS.txt{1,14+k} = [char(039), EEPS.txt{1,14+k}];% Adding apostrophe before↙
bin numbers so they are saved as text
        end

        for h = 1:5
            for i = 1:EENumR
                if EEPS.data(i,14) < h
                    EEStart(1,h) = EEStart(1,h)+1;
                end
                if EEPS.data(i,14) > h
                    EEEnd(1,h) = EEEnd(1,h)+1;
                end
            end
        end
        EEStart = EEStart + 1;
        EEEnd = EENumR - EEEnd;

        h = 1;
        for i = 1:EENumR
            if EEPS.data(i,14) ~= 3
                EEBlankData(h,:) = EEPS.data(i,:);
                h = h+1;
            end
        end

        for h = 1:5
            % Calculating bin by bin averages for each QC_ID
            EEBinAve(h,:) = mean(EEPS.data(EEStart(1,h):EEEnd(1,h),15:47));
            % Calculating bin by bin averages for each QC_ID
            EEBinStDev(h,:) = std(EEPS.data(EEStart(1,h):EEEnd(1,h),15:47));
        end

        % Calculating detection limit with Pre-tunnel Blank only (Average + 3xStDev)
        EEDetLim = EEBinAve(2,:)+ 3.*EEBinStDev(2,:);

        % Correcting EEPS engine on data
        CorrEEPS = EEPS.data(EEStart(1,3):EEEnd(1,3),:);
        for h = 1:size(CorrEEPS,1)
            CorrEEPS(h,15:47) = CorrEEPS(h,15:47)-EEDetLim(1,:);
            for i = 1:33
                if CorrEEPS(h,14+i) < EEDetLim(1,i)
                    CorrEEPS(h,14+i) = EEDetLim(1,i);
                end
            end
        end
```

```
%% Plots for EEPS
% Setting page orientation and margins of figure to be saved
set(0,'defaultfigurepaperunits','inches');
set(0,'defaultfigurepaperorientation','portrait');
set(0,'defaultfigurepapersize',[8.5 11]);
set(0,'defaultfigurepaperposition',[.25 .25 [8.5 11]-0.5]);

figure ('Name', 'EEPS_Bin_BoxPlots','NumberTitle','off')
q=[1 2 4 5];
for i = 1:4
    subplot(3,2,i)
    boxplot(EEPS.data(EEStart(1,q(1,i)):EEEnd(1,q(1,i)),15:46))
    xlabel ('EEPS Bins')
    ylabel ('dN/dlogDp [#/cm³]')
    title (strcat(QCName(1,q(1,i)}, ' Box Plots (RAW)'))
    set(gca,'XTicklabel',' ')
    hold on
    plot(EEBinAve(q(1,i),1:32),'ko','MarkerFaceColor', 'k','MarkerSize',4)
    if i ==1
        legend ('Mean')
    end
    hold off
end
subplot(3,2,5)
boxplot(EEBlankData(:,47), EEBlankData(:,14),'labels',{QCName{1,[1 2 4 5]}})
ylabel ('dN [#/cm³]')
title ('Total Concentration Box Plots')
hold on
plot([1:4], EEBinAve([1 2 4 5],33),'ko','MarkerFaceColor', 'k','MarkerSize',4)
hold off

% Save figure as a PDF
saveas(gcf, strcat(RunID, ' EEPSBlanks.pdf'),'pdf');


% Outputing Blank Corrected EEPS Data
EEDataName=strcat(RunID,'_Blank_Corrected_EEPS.xls');

%****Using the standard xlswrite in Matlab***
xlswrite(EEDataName,EEPS.txt,'Sheet1','A1');
xlswrite(EEDataName,CorrEEPS,'Sheet1','A2');

end

%% Retrieving FTIR data
% Changing to the directory containing the FTIR data
cd(FOutputD);

% list files in current directory
DirCon=ls;
% Checking for an FTIR File match
```



```
FT=strmatch(Filename{1,2}, DirCon,'exact');

%% If there is an FTIR File, retrieve
if FT>0
    % Read FTIR Data
    %------------------------------------------------------------
    [FTIR.data, FTIR.txt] = xlsread(Filename{1,2}{:},'AllData');
    FTNumR = size(FTIR.data,1); % Set number of rows to a variable
    FTStart = zeros(1,5); % Initialize a variable to save pre-tunnel blank start row
    FTEnd = zeros(1,5); % Initialize a variable to save pre-tunnel blank end row
    for h = 1:5
        for i = 1:FTNumR
            if FTIR.data(i,14) < h
                FTStart(1,h) = FTStart(1,h)+1;
            end
            if FTIR.data(i,14) > h
                FTEnd(1,h) = FTEnd(1,h)+1;
            end
        end
    end
    FTStart = FTStart + 1;
    FTEnd = FTNumR - FTEnd;

    h = 1;
    for i = 1:FTNumR
        if FTIR.data(i,14) ~= 3
            FTBlankData(h,:) = FTIR.data(i,:);
            h = h+1;
        end
    end

    for h = 1:5
        % Calculating constituent averages for each QC_ID
        FTConAve(h,:) = nanmean(FTIR.data(FTStart(1,h):FTEnd(1,h),15:57));
        % Calculating constituent averages for each QC_ID
        FTConStDev(h,:) = nanstd(FTIR.data(FTStart(1,h):FTEnd(1,h),15:57));
    end


    % Calculating detection limit with pre and post tunnel blanks (Average + 3xStDev)
    %         FTDetLim = (nanmean([FTIR.data(FTStart(1,2):FTEnd(1,2),15:45) ; FTIR.↙
data(FTStart(1,4):FTEnd(1,4),15:45)]))+ 3.*(nanstd([FTIR.data(FTStart(1,2):FTEnd(1,2),15:↙
45) ; FTIR.data(FTStart(1,4):FTEnd(1,4),15:45)]));

    FTDetLimPre = (nanmean(FTIR.data(FTStart(1,2):FTEnd(1,2),15:45)))+ 3.*(nanstd↙
(FTIR.data(FTStart(1,2):FTEnd(1,2),15:45)));
    FTDetLimPost = (nanmean(FTIR.data(FTStart(1,4):FTEnd(1,4),15:45)))+ 3.*(nanstd↙
(FTIR.data(FTStart(1,4):FTEnd(1,4),15:45)));

    % Correcting FTIR engine on data
    CorrFTIR = FTIR.data(FTStart(1,3):FTEnd(1,3),:);
```

```
        for h = 1:size(CorrFTIR,1)
            CorrFTIR(h,15:45) = CorrFTIR(h,15:45)-FTDetLimPre(1,:);
            for i = 1:31
                %               if CorrFTIR(h,14+i) < FTDetLim(1,i)
                %                   CorrFTIR(h,14+i) = FTDetLim(1,i);
                %               end
                if CorrFTIR(h,14+i) < 0
                    CorrFTIR(h,14+i) = 0;
                end
            end
        end

        %% FTIR Plots

        g = 1;
        for i = 1:6
            figure ('Name', strcat('FTIR_Con_BoxPlots_', num2str(i)),'NumberTitle','off')
            for j = i*6-5: i*6
                if j < 34
                    subplot(3,2,g)
                    boxplot(FTBlankData(:,j+14), FTBlankData(:,14),'labels',{QCName{1,[1↙
2 4 5]}})
                    title (FTIR.txt{1,j+14})
                    hold on
                    plot([1:4], FTConAve([1 2 4 5],j),'ko','MarkerFaceColor',↙
'k','MarkerSize',4)
                    hold off
                end
                g = g+1;
            end
            % Save figure as a PDF
            saveas(gcf, strcat(RunID, ' FTIRBlanks_',num2str(i),'.pdf'),'pdf');
            g = 1;
        end


        % Outputing Blank Corrected FTIR  Data
        FTDataName=strcat(RunID,'_Blank_Corrected_FTIR.xls');

        %****Using the standard xlswrite in Matlab***
        xlswrite(FTDataName,FTIR.txt,'Sheet1','A1');
        xlswrite(FTDataName,CorrFTIR,'Sheet1','A2');

    end

    close all
    clearvars -except indCur runType DateStr DateStrAll FuelType FuelTypeAll RunNum↙
RunNumAll FAstr FAstrAll

end
```

CODE_3_Time_Alignment_12JUN2014.m


```
6/16/15 9:29 PM  H:\...\CODE_3_Time_Alignment_12JUN2014.m   1 of 22

%Tyler Feralio
%CM12 Engine Dyno Data compiler for 2013/2014 transient / steady state test sequence
%Retrieves files generated by codes 1 and 2, time aligns all data, and
%outputs a single file containing all data sharing a single timestamp

%Armfield
%Scantool (VCDS)
%Labview
%EEPS
%FTIR

%Last Updated 08JUN2014 by KS

close all; clear all; clc;

%% ------------------------------------------
%Manual input

% Input needed to build run ID
%Date data was collected
DateStr='06DEC2013';
%Fuel used for data collection
FuelType='B000';
%Run of the day
RunNum='1';
%Feedstock association -- 'WVO' or 'SOY'
FAstr = 'SOY';
%% ------------------------------------------

%------------------------------------------
%Hz at which Armfield is logging
AHz = 2;
%------------------------------------------


%------------------------------------------
% Begin processing
% Defining a Nominal Value to Feedstock Association
if strcmp(FAstr,'WVO') == 1
    FAnom = 1;
end

if strcmp(FAstr,'SOY') == 1
    FAnom = 2;
end

% Percentage of Biodiesel in fuel
BioPer = str2num(strrep(FuelType,'B',''));

%Reference Date for second of the year calcualtion
```

```
6/16/15 9:29 PM  H:\...\CODE_3_Time_Alignment_12JUN2014.m   2 of 22

RefDateVec = [2013 1 1 0 0 0];
% RefDateStr = datestr(RefDateVec,'mmmm dd, yyyy HH:MM:SS.FFF');
RefDateNum = datenum(RefDateVec);

%% Begin program
%--------------------------------
% Generating datevec and datenum using input data
DateVec=datevec(DateStr);
%Retaining only the YYYY MM DD data
DateVec=DateVec(:,1:3);

%Using user input data to generate file names
RunID=strcat(RunNum,'_',DateStr,'_',FuelType);

%% ------------------------------------------
%Construction File Names to look for
% Timing
TimeFile = strcat(RunID,'_Timing.xls');

%Emissions Data

    %FTIR
        FTfilename = strcat(RunID,'_Blank_Corrected_FTIR.xls');

    %EEPS
        EEfilename = strcat(RunID,'_Blank_Corrected_EEPS.xls');

%Operational
    AFfilename = strcat(RunID,'_AFOut.xls');
    LVfilename = strcat(RunID,'_LVOut.xls');
    STfilename = strcat(RunID,'_STOut.xls');

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

%------------------------------------------
%Defining directories names to retrieve data from
%Setting Directory that matlab files are in as the Parent Directory
PD=cd;
%Generating month portion of file path needed to retrieve data
month=strcat(strrep(sprintf('%0.1f',(DateVec(1,2))/10),'.',''),'_',DateStr(1,3:5));
%Defining directories:
%Main Data Directory path
Dir=strcat('H:\SP2_ArmfieldEngine\CM12_data\', num2str(DateVec(:,1)),'\', month,'\',↙
DateStr);
%Checking if multiple runs were made in the same day by looking for
%subfolders
mr=exist(strcat(Dir,'\',RunID));
%deciding which directory to pull data from
if mr>0
    %Defining file paths if more then one run was collected in the same day
```

```
    TDataD=(strcat(Dir,'\',RunID,'\Instrument_Data'));
    EPlotD=(strcat(Dir,'\',RunID,'\Data_Plots\EEPS'));
    OPlotD=(strcat(Dir,'\',RunID,'\Data_Plots\Operational'));
    EOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\EEPS\Matlab_Output'));
    AOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Armfield\Matlab_Output'));
    SOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Scantool\Matlab_Output'));
    LOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\Labview\Matlab_Output'));
    FOutputD=(strcat(Dir,'\',RunID,'\Instrument_Data\FTIR\Matlab_Output'));
else
    %Defining file paths if only one run was collected in the same day
    TDataD=(strcat(Dir,'\Instrument_Data'));
    EPlotD=(strcat(Dir,'\Data_Plots\EEPS'));
    OPlotD=(strcat(Dir,'\Data_Plots\Operational'));
    EOutputD=(strcat(Dir,'\Instrument_Data\EEPS\Matlab_Output'));
    AOutputD=(strcat(Dir,'\Instrument_Data\Armfield\Matlab_Output'));
    SOutputD=(strcat(Dir,'\Instrument_Data\Scantool\Matlab_Output'));
    LOutputD=(strcat(Dir,'\Instrument_Data\Labview\Matlab_Output'));
    FOutputD=(strcat(Dir,'\Instrument_Data\FTIR\Matlab_Output'));
end

% Checking to see if folders already exist to deposit Matlab output in.  If
% they don't, Matlab will generate them.
InstLet={'E' 'A' 'S' 'L' 'F'};
FileLet = {'EE' 'AF' 'ST' 'LV' 'FT'};
for i=1:size(InstLet,2)
    var1=strcat(InstLet{i},'Out');
    var2=strcat(InstLet{i},'OutputD');
    %checking to see if 'Matlab_Output' folder exists
    eval(strcat(var1,'=exist(',var2,');'));
    %if it does not exist, create it
    if eval(strcat(var1,'==0;'));
        eval(strcat('mkdir(',var2,');'));
    else
    end
end


%% Changing Directory to retrieve Timing Data
cd(TDataD);
%Reading Timing Data for the Run
Timing = importdata(TimeFile,'\t');
TimeStr = Timing.Sheet1(2:end,2);
TimeVec = datevec(TimeStr);% Generating Time Vec for all timing events
DateVecMesh = meshgrid(DateVec, 1:size(TimeStr,1));
TimeVec(:,1:3) = DateVecMesh;% Appending the correct year-month-day to the Time Vectors
TimeNum = datenum(TimeVec); % Generating Time Numbers for all timing events
for i=1:size(TimeStr,1)% Calculating the minutes from scheduler start for all timing↙
events (Elapse Time)
    TimeEla(i,1) = (3600*TimeVec(i,4)+60*TimeVec(i,5)+TimeVec(i,6))-(3600*TimeVec(4,4)↙
+60*TimeVec(4,5)+TimeVec(4,6));
end
```



```
%Counting the rows to each scan start
ModeNum = size(TimeNum,1)-2;

% Generating vector of mode bound times
%datenum format for axes with time of day
TickTime=TimeNum(2:end,1)';
%elapse time elapse time axes needed for overlaying multiple runs
ElapseTickTime = TimeEla(2:end,1)';

%% Calculating Fuel Volume
%Density (lbs/gal)
B100Den = 7.314;
B000Den = 6.779;
Ratio = str2num(strrep(FuelType,'B',''))/100;
Den = Ratio*B100Den + (1-Ratio)*B000Den;

q = strcmp(FuelType, 'BurnOut');
if q == 1
    Den = B000Den;
end

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


%  It appears as though different parameters from the same instrument have
%  slightly different time offsets complicating instrument to instrument
%  time adjustment.  These time offsets are the result of three different
%  sources.
%
%     1. Faulty synchronization of instrument clocks
%     2. Sensor time constant
%        - How long it takes a sensor to 'see' a change in what it is
%          measuring.
%     3. Real physical delay
%        - If comparing two different parameters like Injection QTY and
%          Torque, it is feasible that there is a real physical time delay
%          between when the fuel is injected and when the torque output
%          changes.
%
%  With this in mind, redundant parameters (same parameter measured on both
%  instruments) will primarily be used for time alignment.  In instances
%  where there are no redundant parameters, parameters will be carefully
%  selected taking into account both sensor type and physical response.
%
%     For Scantool - Armfield adjustment, Scantool data will be
%       interpolated from its current ~3Hz form to 2Hz so it matches the
%       Armfield data.  Then the Armfield Data will be shifted in 0.5
%       second intervals.  Correlations between these shifted data sets and
%       the Scantool data will then be generated.  The time shift
%       associated with the highest correlation value will be the 'winner'.
```

```
%       Here, Throttle Position and RPM were both logged with both
%       instruments, however, the RPM sensor used for the Armfield data
%       acquisition measures camshaft speed.  A calculation is then made of
%       RPM.  A large lag has been found between Armfield RPM and Scantool
%       RPM relative to other parameters, therefore, it has been decided
%       that Armfield data set will be aligned to the scantool data via
%       Throttle position.  Armfield RPM will then be aligned to scantool
%       RPM separately so that RPM engine on and engine off RPM can be used
%       for EEPS to RPM alignment.
%

    % Begin


    %--------------------------------------------------------------------
    % Pulling in all instrument data
    for i=1:size(InstLet,2)
        if i==1 || i==5;
            j = 'Sheet1';
        else
            j = 'AllData';
        end
        %-------------------------------------
        % Changine to the directory containing the data
        eval(strcat('cd(',InstLet{1,i},'OutputD);'));

        %list files in current directory
        DirCon=ls;
        %Checking for an Filename match
        eval(strcat( InstLet{1,i}, ' = strmatch(', FileLet{1,i}, 'filename, DirCon,',↙
char(039),'exact', char(039),');'));

        %If the filename does exist, retrieve
        if eval(strcat(InstLet{1,i},' > 0'))
            %Read Data
            %-----------------------------------------------------
            eval(strcat('[',FileLet{1,i},'.data, ',FileLet{1,i},'.txt] = xlsread(',↙
FileLet{1,i},'filename,', char(039),j,char(039),');')); % Read in Data
            eval(strcat(FileLet{1,i},'NumR = size(',FileLet{1,i},'.data,1);' ));% Set↙
number of rows to a variable
        end
        %-------------------------------------
    end

    %--------------------------------------------------------------------
    % Determining Maximum Start Time/ row and minimum stop time/ row from all
    % instruments in seconds to avoid rounding error
    for i=1:size(InstLet,2)
        eval(strcat('InstBounds(1,i) = ', FileLet{1,i},'.data(1,8);'));
        eval(strcat('InstBounds(2,i) = ', FileLet{1,i},'.data(end,8);'));
    end
```



```
    %Rounding to the closest second
    InstBounds(1,:) = ceil(InstBounds(1,:));
    InstBounds(2,:) = floor(InstBounds(2,:));

    TimeBounds(1,1) = max(InstBounds(1,:));
    TimeBounds(2,1) = min(InstBounds(2,:));

    %Finding bounding rows for each of the original data sets [EE AF ST LV FT]
    BoundingRow = zeros(2,size(FileLet,2));
    for h = 1 : size(FileLet,2)
        for  i = 1: eval(strcat(FileLet{1,h},'NumR'))
            if eval(strcat(FileLet{1,h},'.data(i,8) < TimeBounds(1,1)'))
                BoundingRow(1,h) = BoundingRow(1,h)+1;
            end
            if eval(strcat(FileLet{1,h},'.data(i,8) <= TimeBounds(2,1)'))
                BoundingRow(2,h) = BoundingRow(2,h) + 1;
            end
        end
    end
    BoundingRow(1,:) = BoundingRow(1,:) + 1;
    %--------------------------------------------------------------------


    %% ------------------------------------------------------------------
    % Interpolating Scantool data to 2Hz and aligning with Armfield data via throttle↙
position
    % Hz to interpolate data to
    Hz = 2;
    % Defining elapse time stamps to interpolate Scantool data to - Time
    % stamps span entire span of collected scantool data
    STInterpTime = (InstBounds(1,3):(1/Hz):InstBounds(2,3))';

    % Interpolating Scantool data to timestamps defined above
    STInterp = interp1(ST.data(:,8),ST.data(:,[9, 14:21]), STInterpTime);
    STInterpHeader = {ST.txt(:,[9, 14:21])};
    % Plot to overlay raw data and interpolated data
    figure (1)
    plot(STInterpTime, STInterp(:,4), ST.data(:,8), ST.data(:,16))
    legend('InterpData ThrotPos', 'RawData ThrotPos')
    ylabel ('Throttle Position (%)')
    xlabel ('Elapsed Time (Sec)')

    %Finding bounding rows for the interpolated scantool data
    STInterpBound = zeros(2,1);

    for  i = 1: size(STInterpTime,1)
        if STInterpTime(i,1) < TimeBounds(1,1)
            STInterpBound(1,1) = STInterpBound(1,1)+1;
        end
        if STInterpTime(i,1) <= TimeBounds(2,1)
```

```
            STInterpBound(2,1) = STInterpBound(2,1) + 1;
        end
    end

    STInterpBound(1,:) = STInterpBound(1,:) + 1;

    % Generating a matrix with .5 sec time lags
    % Number of seconds (+/-) to check (input 3 for -3 to +3 seconds of data
    CorrWinSec = 4;
    % +/- Rows
    CorrWinRow = CorrWinSec*Hz;
    % Total Window size in terms of Rows
    CorrWinRowTot = CorrWinRow*2+1;

    %Initializing Matrix of time Offset data for Pearsons
    STCorr = zeros(STInterpBound(2,1)-STInterpBound(1,1)+1-2*CorrWinRow, CorrWinRowTot);
    % Filling in Matrix
    for i = 1:CorrWinRowTot
        STCorr(:,i) = [STInterp(STInterpBound(1,1)+i-1: STInterpBound(2,1)-2↙
*CorrWinRow+i-1,4)];
    end

    %Initializing a matrix to store r values if
    Corr = zeros(CorrWinRowTot,1);
    %Filling in correlation matrix
    for i = 1:CorrWinRowTot
        z = corrcoef(AF.data(BoundingRow(1,2)+CorrWinRow:BoundingRow(2,2)-CorrWinRow,↙
14), STCorr(:,i));
        Corr(i,1) = z(1,2);
    end

    %Determining the max r values and saving the positions
    [R,Lag] = max(Corr);
    STLagRow = (Lag -(CorrWinRow+1));
    STLagSec = STLagRow/Hz

    if Lag == 0 || Lag == CorrWinRowTot || isnan(Corr(Lag,1))
        error('Increase time window size')
    end

    STTime = STInterpTime - STLagSec;

    % Overlay Interpolated Scantool data and Interpolated/ Time adjusted
    % Scantool data with Armfield Data
    figure (2)
    plot(AF.data(BoundingRow(1,2)+CorrWinRow:BoundingRow(2,2)-CorrWinRow, 8),...
        AF.data(BoundingRow(1,2)+CorrWinRow:BoundingRow(2,2)-CorrWinRow, 14),...
        STInterpTime(STInterpBound(1,1):STInterpBound(2,1),1),...
        STInterp(STInterpBound(1,1):STInterpBound(2,1),4),...
        STTime(STInterpBound(1,1):STInterpBound(2,1),1),...
        STInterp(STInterpBound(1,1):STInterpBound(2,1),4));
```



```
    legend('Armfield', 'Interpolated Scantool', 'Interpolated and Time Aligned Scantool')
    ylabel('Throttle Position (%)')
    xlabel('Elapsed Time (Sec)')

    %Buidling Matrix of 2Hz time aligned Scantool Data
    ST2Hz = zeros(size(STInterp,1),size(STInterp,2)+1);
    ST2Hz(:,1) = STTime;
    ST2Hz(:,2:end) = STInterp;
    ST2HzHeader = {'Elapse Time',  STInterpHeader{:}};
    %--------------------------------------------------------------------

    %--------------
    % Time aligning Armfield RPM to Scantool RPM adjusting in 0.5 sec
    % intervals
    % Generating a matrix with .5 sec time lags
    % Number of seconds (+/-) to check (input 3 for -3 to +3 seconds of data
    CorrWinSec = 4;
    % +/- Rows
    CorrWinRow = CorrWinSec*Hz;
    % Total Window size in terms of Rows
    CorrWinRowTot = CorrWinRow*2+1;

    %Initializing Matrix of time Offset data for Pearsons
    AFRPMCorr = zeros(BoundingRow(2,2)-BoundingRow(1,2)+1-2*CorrWinRow, CorrWinRowTot);
    % Filling in Matrix
    for i = 1:CorrWinRowTot
        AFRPMCorr(:,i) = [AF.data(BoundingRow(1,2)+i-1: BoundingRow(2,2)-2*CorrWinRow+i-↙
1,22)];
    end

    %Initializing a matrix to store r values if
    Corr = zeros(CorrWinRowTot,1);
    %Filling in correlation matrix
    for i = 1:CorrWinRowTot
        z = corrcoef(ST2Hz(STInterpBound(1,1)+CorrWinRow:STInterpBound(2,1)-CorrWinRow,↙
3), AFRPMCorr(:,i));
        Corr(i,1) = z(1,2);
    end

    %Determining the max r values and saving the positions
    [R,Lag] = max(Corr);
    AFRPMLagRow = (Lag -(CorrWinRow+1));
    AFRPMLagSec = AFRPMLagRow/Hz

    %Adjusting AF RPM to Scantool RPM
    %Appending a 'Corrected RPM' column to the Armfield Data
    AF.data(:,end+1)=0;

    if AFRPMLagRow < 0
        for i = 1-AFRPMLagRow:AFNumR
            AF.data(i,end) = AF.data(i+AFRPMLagRow,22);
```

```
    end
end
if AFRPMLagRow >= 0
    for i = 1:AFNumR-AFRPMLagRow
        AF.data(i,end) = AF.data(i+AFRPMLagRow,22);
    end
end
AF.txt{1,end+1} = 'Corrected Engine Speed [RPM]';
%--------------
 %Overlaying raw AF RPM, Scantool RPM, and corrected AF RPM
figure (3)
plot(AF.data(BoundingRow(1,2):BoundingRow(2,2),8),...
    AF.data(BoundingRow(1,2):BoundingRow(2,2),22),...
    ST2Hz(STInterpBound(1,1):STInterpBound(2,1),1),...
    ST2Hz(STInterpBound(1,1):STInterpBound(2,1),3),...
    AF.data(BoundingRow(1,2):BoundingRow(2,2),8),...
    AF.data(BoundingRow(1,2):BoundingRow(2,2),26))
legend('AF RPM', 'ST RPM','Corr AF RPM')
ylabel('RPM')
xlabel('Elapsed Time (Sec)')

%--------------------------------------------------------------------------
%Interpolating Armfield and Scantool Data to 1Hz to match the remaining
%instruments
Hz = 1;
% Defineing elapse time stamps to interpolate Scantool and Armfield data to 1Hz
STInterpTime2 = (STInterpTime(1,1):(1/Hz):STInterpTime(end,1))';
AFInterpTime = [ceil(AF.data(1,8)):(1/Hz):floor(AF.data(end,8))]';

%Interpolating Scantool to 1Hz
ST1HzHeader = ST2HzHeader;
STInterp2 = interp1(ST2Hz(:,1),ST2Hz(:,2:end), STInterpTime2);
ST1Hz = zeros(size(STInterp2,1),size(STInterp2,2)+1);
ST1Hz(:,1) = STInterpTime2;
ST1Hz(:,2:end) = STInterp2;

%Interpolating Armfield to 1Hz
AF1HzHeader = {'ElapseTime' AF.txt{1,[9, 14:end]}};
AFInterp = interp1(AF.data(:,8),AF.data(:,[9, 14:end]), AFInterpTime);
AF1Hz = zeros(size(AFInterp,1),size(AFInterp,2)+1);
AF1Hz(:,1) = AFInterpTime;
AF1Hz(:,2:end) = AFInterp;
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%Finding Time associated with first elapse time stamp of interpolated
%data
for i = 1 : size(AF.data,1)
    if AF.data(i,8) == AF1Hz(1,1)
        AFStartNum = AF.data(i,7);
    end
```



```
    end
    AFStartVec = datevec(AFStartNum);

    % Starting to generate output file
    TATimeNum = [datenum(TimeVec(4,:) - [0 0 0 0 25 0]):datenum([0 0 0 0 0 1]):datenum↙
(TimeVec(4,:) + [0 0 0 0 115 0])]';
    TANumR = size(TATimeNum,1);
    for i = 1:TANumR
        TATimeVec(i,:) = datevec(TATimeNum(i,1));
        if TATimeVec(i,4) == AFStartVec(1,4) && TATimeVec(i,5) == AFStartVec(1,5) &&↙
TATimeVec(i,6) == AFStartVec(1,6)
            AFStartRow = i;
        end
    end

    TAHeader = {'Yr' 'Mon' 'Day' 'Hr' 'Min' 'Sec' 'DateNum' 'ElapseTime' 'SOY_Master'↙
'RunOfDay' 'FeedstockAssociation' 'BIO%' 'Phase'};
    TAData = nan(TANumR, 13);% Initializing Output matrix
    TAData(:,1:6) = TATimeVec(:,1:6);% Assigning DateVec time Stamp
    TAData(:,7) = TATimeNum(:,1);% Assigning DateNum time stamp
    TAData(:,8) = AF1Hz(1,1)-AFStartRow+1:1:AF1Hz+(size(TATimeNum,1)-AFStartRow)';%↙
Assigning elapse time
    TAData(:,9) = (TAData(:,7)-RefDateNum)*86400; % Assigning 'SOY'
    TAData(:,10) = str2num(RunNum); % Assigning Run number
    TAData(:,11) = FAnom; % Assigning Biodiesel association
    TAData(:,12) = BioPer; % Assigning Biodiesel percentage

    % Filling Phase Column with zeros
    TAData(:,13) = 0;

    % Defining Mode Numbers - 0 = engine off; 1 = initial idel; 2 = warm up; 3 =
    % transient; 4 = SS1 stabilization; 5 = SS1 Stabilized; 6 = SS2
    % stabilization; 7 = SS2 Stabilized; 8 = SS3 Stabilization; 9 = SS3
    % Stabilized; 10 = Cool down Idle
    PhaseNum = [1:10];

    for k=1:ModeNum+1
        n=0;
        for r=1:TANumR
            if k < ModeNum+1
                if TAData(r,7)>=TickTime(:,k) && TAData(r,7)<TickTime(:,k+1)
                    TAData(r,13) = PhaseNum(1,k);
                end
            end
        end
    end

    % Finding bounding rows for correlation using 1 Hz data
    for i = 1:TANumR
        if TAData(i,8) == TimeBounds(1,1)
            TARowBound(1,1) = i;
```

```
        end
        if TAData(i,8) == TimeBounds(2,1)
            TARowBound(2,1) = i;
        end
    end

    CorrNumR = TARowBound(2,1)-TARowBound(1,1);

    %Adding Armfield Data
    TANumC = size(TAData,2);
    TAData = [TAData, nan(TANumR, size(AF1Hz,2)-1)]; % Adding NAN positions to matrix for↙
AF Data
    TAData(AFStartRow:AFStartRow+size(AF1Hz,1)-1,TANumC+1:end) = AF1Hz(:, 2:end); %↙
Addign data
    TAHeader = {TAHeader{:}, AF1HzHeader{1,2:end}};

    %Adding Scantool Data
    TANumC = size(TAData,2);
    for i=1:TANumR
        if TAData(i,8) == ST1Hz(1,1)
            STStartRow = i;
        end
    end
    TAData = [TAData, nan(TANumR, size(ST1Hz,2)-1)]; % Adding NAN positions to matrix for↙
ST Data
    TAData(STStartRow:STStartRow+size(ST1Hz,1)-1,TANumC+1:end) = ST1Hz(:, 2:end); %↙
Addign data
    TAHeader = {TAHeader{:}, ST1HzHeader{1,2:end}};
    %--------------------------------------------------------------------

    %Overlaying 1sec resolution raw AF RPM, Scantool RPM, and corrected AF RPM
    figure (4)
    plot(TAData(:,8), TAData(:,23),...
        TAData(:,8), TAData(:,29),...
        TAData(:,8), TAData(:,27))
    legend('AF RPM', 'ST RPM','Corr AF RPM')
    ylabel('RPM')
    xlabel('Elapsed Time (Sec)')



    %--------------------------------------------------------------------
    % Correlating Labview MAF to Armfield MAP
    % Since Labview data is logged at 1 Hz (all time stamps exactly 1
    % second apart) interpolation will not be used, just correlation

    %Building a new matrix of LV data just incase there's a missing time
    %stamp
    LVTime = (LV.data(1,7):datenum([0 0 0 0 0 1]):LV.data(end,7))'; %Generating a new↙
time array for LV Data
    LVData = nan(size(LVTime,1),size(LV.data,2));% Initiallizing new LV Data matrix
```



```
    LVData(:,7) = LVTime; % Assigning the datenum data to a row in the matrix
    k = 1;
    for i = 1:size(LVTime,1)
        LVData(i,1:6) = datevec(LVTime(i,1));% Filling in datevec data
        LVData(i,8) = (3600*LVData(i,4)+60*LVData(i,5)+LVData(i,6))-(3600*TimeVec(4,4)↙
+60*TimeVec(4,5)+TimeVec(4,6));% calculating and filling in alapse time
        if k < (LVNumR - 5)
            l = k+5;
        else
            l = LVNumR;
        end
        for j = k:l
           if LV.data(j,4) == LVData(i,4) && LV.data(j,5) == LVData(i,5) && LV.data(j,6)↙
== LVData(i,6)% Tranfering data from original matrix to new one
               LVData(i,9:end) = LV.data(j,9:end);
               k = k+1;
           end
        end
    end

    % Finding the bounding rows associated with bounding times for correlation
    LVBoundingRow(1:2,1) = 0;
    for i = 1:size(LVData,1)
        if LVData(i,8)< TimeBounds(1,1)
            LVBoundingRow(1,1) = LVBoundingRow(1,1)+1;
        end
        if LVData(i,8)< TimeBounds(2,1)
            LVBoundingRow(2,1) = LVBoundingRow(2,1)+1;
        end
    end

    % Generating a matrix with 1 sec time lags
    % Number of seconds (+/-) to check (input 3 for -3 to +3 seconds of data
    CorrWinSec = 6;
    % +/- Rows
    CorrWinRow = CorrWinSec*Hz;
    % Total Window size in terms of Rows
    CorrWinRowTot = CorrWinRow*2+1;

    %Initializing Matrix of time Offset data for Pearsons
    LVCorr = zeros(CorrNumR+1-2*CorrWinRow, CorrWinRowTot);
    % Filling in Matrix
    for i = 1:CorrWinRowTot
        LVCorr(:,i) = [LVData(LVBoundingRow(1,1)+i-1: LVBoundingRow(2,1)-2*CorrWinRow+i-↙
1,26)];
    end

    %Initializing a matrix to store r values if
    Corr = zeros(CorrWinRowTot,1);
    %Filling in correlation matrix
    for i = 1:CorrWinRowTot
```

```
        z = corrcoef(TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 18),↙
LVCorr(:,i),'rows','pairwise');
        Corr(i,1) = z(1,2);
    end

    %Determining the max r values and saving the positions
    [R,Lag] = max(Corr);
    LVLagRow = (Lag -(CorrWinRow+1));
    LVLagSec = LVLagRow/Hz

    if Lag == 0 || Lag == CorrWinRowTot || isnan(Corr(Lag,1))
        error('Increase time window size')% Throwing an error if the maximum R value is↙
at either end of the time window
    end

    %Adding Labview Data
    TANumC = size(TAData,2);
    LVStartRow = 0;
    for i=1:size(LVData,1)
        if TAData(1,8) > LVData(i,8)
            LVStartRow = LVStartRow+1;
        end
    end
    LVStartRow = LVStartRow + LVLagRow;% applying time lag found
    TAData = [TAData, nan(TANumR, size(LVData,2)-12)]; % Adding NAN positions to matrix↙
for LV Data

    if LVData(end,7) > TAData(end,7)
        TAData(:,TANumC+1:end) = LVData(LVStartRow:LVStartRow+TANumR-1, [9 14:end]); %↙
Assign data
    else
        TAData(1:size(LVData,1)-LVStartRow+1,TANumC+1:end) = LVData(LVStartRow:end, [9↙
14:end]); % Assign data
    end

    TAHeader = {TAHeader{:}, LV.txt{1,[9 14:end]}};% appending to header


    % Normalizing data for plotting
    MAPBounds = [min( TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow,↙
18));...
        max(TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 18))];
    MAFBounds = [min( TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow,↙
50));...
        max(TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 50))];
    NormMAP =  (TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 18) -↙
MAPBounds(1,1))/(MAPBounds(2,1)-MAPBounds(1,1));
    NormMAF =  (TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 50) -↙
MAFBounds(1,1))/(MAFBounds(2,1)-MAFBounds(1,1));

    %Plotting normalized data for visual vrification of time alignment
```



```
    figure (5)
    plot(TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 8),...
        NormMAP,...
        LVData(LVBoundingRow(1,1)+CorrWinRow:LVBoundingRow(2,1)-CorrWinRow, 8),...
        NormMAF,...
        TAData(TARowBound(1,1)+CorrWinRow:TARowBound(2,1)-CorrWinRow, 8),...
        NormMAF);
    legend('Armfield MAP', 'LabView MAF', 'Time Aligned LabView MAF')
    %--------------------------------------------------------------------

     %Adding a column for a moving average of MAG DR
    TANumC = size(TAData,2);
    TAHeader{1,TANumC+1} = 'MovAve MAG DR';
    TAData(:,TANumC+1) = 0;
    WinSize = 60;
    for i = WinSize/2+1:TANumR - WinSize/2
        TAData(i,TANumC+1)= mean(TAData(i-WinSize/2:i+WinSize/2,54));
    end

    %Plotting DR Data
    figure (6)
    plot(TAData(:, 8), TAData(:, 53), TAData(:, 8), TAData(:, 54), TAData(WinSize/2+1:↙
TANumR-WinSize/2, 8), TAData(WinSize/2+1:TANumR-WinSize/2, 56))
    ylabel('DR')
    xlabel('Time(sec)')
    legend('OMEGA DR','MAG DR','MovAve MAG DR')

% % %     %--------------------------------------------------------------------
% % %
% % %     %Writing Time Aligned Data to File
% % %
% % %     cd(TDataD);
% % %     %Outputing Armfield Data
% % %     TADataName=strcat(RunID,'_TAOperationalOut.xls');
% % %
% % %     %****Using the standard xlswrite in Matlab***
% % %     xlswrite(TADataName,TAHeader,'Sheet1','A1');
% % %     xlswrite(TADataName,TAData,'Sheet1','A2');
% % %     %--------------------------------------------------------------------



%% EEPS Data
% ------------------------------------------------------------------------
% Correlating EEPS Total PN to Scantool RPM
% Since EEPS data is logged at 1 Hz (all time stamps exactly 1
% second apart) interpolation will not be used, just correlation

% Prompt user on source for EEPS lag data
EELagSource = input('Which source would you like to use for the EEPS lag (type M for↙
manual adjustment or P for previously calculated)? ','s')
```

```
%Building a new matrix of EE data just in case there's a missing time
%stamp
EETime = (EE.data(1,7):datenum([0 0 0 0 0 1]):EE.data(end,7))'; %Generating a new time↙
array for LV Data
if size(EETime,1) ~= EENumR

    EEData = nan(size(EETime,1),size(EE.data,2));% Initializing new EE Data matrix
    EEData(:,7) = EETime; % Assigning the datenum data to a row in the matrix
    k = 1;
    for i = 1:size(EETime,1)
        EEData(i,1:6) = datevec(EETime(i,1));% Filling in datevec data
        EEData(i,8) = (3600*EEData(i,4)+60*EEData(i,5)+EEData(i,6))-(3600*TimeVec(4,4)↙
+60*TimeVec(4,5)+TimeVec(4,6));% calculating and filling in alapse time
        if k < (EENumR - 5)
            l = k+5;
        else
            l = EENumR;
        end
        for j = k:l
            if EE.data(j,4) == EEData(i,4) && EE.data(j,5) == EEData(i,5) && EE.data(j,6)↙
== EEData(i,6)% Tranfering data from original matrix to new one
                EEData(i,9:end) = EE.data(j,9:end);
                k = k+1;
            end
        end
    end
else
    EEData = EE.data;
end

% only prompt the user for graphical input if manual method for lag
% chosen above
if strmatch('M',EELagSource) == 1;
    EERowBound = [0;0];
    for i = 1 : size(EETime,1)
        if EEData(i,8)<ElapseTickTime(1,end) %KMS changed to end to deal with runs that↙
were not complete
            EERowBound(1,1) = EERowBound(1,1) + 1;
        end
        if EEData(i,8)<ElapseTickTime(1,end)
            EERowBound(2,1) = EERowBound(2,1) + 1;
        end
    end

    %Plotting Corrected AF RPM over EEPS TPN at engine off for G-input time alignment of↙
EEPS Data
    figure (7)
    [haxes,hline1,hline2] = plotyy(TAData(:, 8),TAData(:, 27), EEData(:, 8), EEData(:,↙
47));
    legend('Corr AF RPM', 'EEPS TPN')
```



```
    %Setting X axis limits for 1st plot
    set(haxes(1), 'XLim', [ElapseTickTime(1,end)-30, ElapseTickTime(1,end)+30])
    %Setting X axis limits for 2nd plot
    set(haxes(2), 'XLim', [ElapseTickTime(1,end)-30, ElapseTickTime(1,end)+30])
    %Setting Y axis limits for 1st plot
    set(haxes(1), 'YLim', [500, 3500])
    %Setting Y axis limits for 2ndt plot
    set(haxes(2), 'YLim', ...
        [min(EEData(EERowBound(1,1)-30:EERowBound(2,1)+30,47))-0.1*((max(EEData↙
(EERowBound(1,1)-30:EERowBound(2,1)+30,47))-min(EEData(EERowBound(1,1)-30:EERowBound(2,1)↙
+30,47)))),...
        max(EEData(EERowBound(1,1)-30:EERowBound(2,1)+30,47))+0.1*((max(EEData(EERowBound↙
(1,1)-30:EERowBound(2,1)+30,47))-min(EEData(EERowBound(1,1)-30:EERowBound(2,1)+30,↙
47)))))]

    [x1 y1] = ginput(2);

    EERowBound = [0;0];
    for i = 1 : size(EETime,1)
        if EEData(i,8)<ElapseTickTime(1,1)-30
            EERowBound(1,1) = EERowBound(1,1) + 1;
        end
        if EEData(i,8)<ElapseTickTime(1,3)
            EERowBound(2,1) = EERowBound(2,1) + 1;
        end
    end

    %Plotting Corrected AF RPM over EEPS TPN at engine on for G-input time alignment of↙
EEPS Data
    figure (8)
    [haxes,hline1,hline2] = plotyy(TAData(:, 8),TAData(:, 27), EEData(:, 8), EEData(:,↙
47));
    legend('Corr AF RPM', 'EEPS TPN')
    %Setting X axis limits for 1st plot
    set(haxes(1), 'XLim', [ElapseTickTime(1,1)-30, ElapseTickTime(1,1)+30])
    %Setting X axis limits for 2nd plot
    set(haxes(2), 'XLim', [ElapseTickTime(1,1)-30, ElapseTickTime(1,1)+30])
    %Setting Y axis limits for 1st plot
    set(haxes(1), 'YLim', [-500, 1500])
    %Setting Y axis limits for 2ndt plot
    set(haxes(2), 'YLim', ...
        [min(EEData(EERowBound(1,1):EERowBound(2,1),47))-.25*max(EEData(EERowBound(1,1):↙
EERowBound(2,1),47))-min(EEData(EERowBound(1,1):EERowBound(2,1),47)),...
        max(EEData(EERowBound(1,1):EERowBound(2,1),47))]);

    [x2 y2] = ginput(2);

    EngineOnLag = (x2(2,1)-x2(1,1));
    EngineOffLag = (x1(2,1)-x1(1,1));
    EELagSec = round((EngineOnLag + EngineOffLag)/2);
```

```
    if EngineOnLag == 0
        EELagSec = round(EngineOffLag);
    end

    if EngineOffLag == 0
        EELagSec = round(EngineOnLag);
    end

elseif strmatch('P',EELagSource) == 1;
    % only lag adjustments for WVO
    FilenameWVOSoyAll = {'1_18JUN2013_B000';
        '1_25JUN2013_B000';
        '2_05AUG2013_B000';
        '1_06AUG2013_B000';
        '1_29AUG2013_B010';
        '1_30AUG2013_B010';
        '1_31AUG2013_B010';
        '1_04SEP2013_B020';
        '1_05SEP2013_B020';
        '1_06SEP2013_B020';
        '1_09SEP2013_B050';
        '1_10SEP2013_B050';
        '1_11SEP2013_B050';
        '1_19SEP2013_B100';
        '1_20SEP2013_B100';
        '2_20SEP2013_B100';
        '1_22OCT2013_B000';
        '1_24OCT2013_B000';
        '1_25OCT2013_B000';
        '1_01MAY2014_B000';
        '1_02MAY2014_B000';
        '1_05MAY2014_B010';
        '1_06MAY2014_B010';
        '2_06MAY2014_B010';
        '1_07MAY2014_B020';
        '2_07MAY2014_B020';
        '1_12MAY2014_B020';
        '2_12MAY2014_B020';
        '1_13MAY2014_B020';
        '2_13MAY2014_B020';
        '1_14MAY2014_B020';
        '1_15MAY2014_B020';
        '1_20MAY2014_B050';
        '2_20MAY2014_B050';
        '1_22MAY2014_B050';
        '1_23MAY2014_B100';
        '2_23MAY2014_B100';
        '1_26MAY2014_B100';
        '1_27MAY2014_B000'};
```



```
    EELagSecAll = [6;5;6;5;6;5;6;5;5;5;5;6;6;7;6;6;8;6;6;5;6;6;6;5;8;6;NaN;6;6;6;7;6;6;7;↙
6;6;6;6;6];

    EELagSec = EELagSecAll(strmatch(RunID,FilenameWVOSoyAll));
else
    error('A method for calculating the EEPS lag is needed.')
end

EELagSec

EELagRow = EELagSec*Hz;


% Finding the bounding rows associated with bounding times for correlation
EEBoundingRow(1:2,1) = 0;
for i = 1:size(EEData,1)
    if EEData(i,8)< TimeBounds(1,1)
        EEBoundingRow(1,1) = EEBoundingRow(1,1)+1;
    end
    if EEData(i,8)< TimeBounds(2,1)
        EEBoundingRow(2,1) = EEBoundingRow(2,1)+1;
    end
end


%Adding EEPS Data
TANumC = size(TAData,2);
for i=1:TANumR
    if TAData(i,4) == EEData(1,4) && TAData(i,5) == EEData(1,5) && TAData(i,6) == EEData↙
(1,6)
        EEStartRow = i;
    end
end
% KMS added condition that if the start row was not defined that the start
% row was the first row of the TA data set and then find the corresponding
% EEPS data and limit the EEPS data going forward to the start of the TA
% file
if exist('EEStartRow','var') == 0;
   EEStartRow = 20;
   EETAStartInd = find(EEData(:,4) == TAData(1,4) & EEData(:,5) == TAData(1,5) & EEData↙
(:,6) == TAData(1,6));
EEData = EEData(EETAStartInd:end,:);
end

TAData = [TAData, nan(TANumR, size(EEData,2)-13)]; % Adding NAN positions to matrix for↙
ST Data
for i = EEStartRow-EELagRow:EEStartRow+size(EEData,1)-1-EELagRow
    TAData(i,TANumC+1) = EEData(i+1-(EEStartRow-EELagRow), 9); % Adding EEPS SOY

    %******Correcting EEPS data for DR*******
    TAData(i,TANumC+2:end) = [EEData(i+1-(EEStartRow-EELagRow), (15:47))*TAData(i,56),↙
```

```
EEData(i+1-(EEStartRow-EELagRow), (48:49))]; % Adding Bin Data
end
% KMS to add check, if EEPS data too long, will write zeros to end of
% TAData making the matrix larger than it should be
if size(TAData,1) > TANumR;
    TAData = TAData(1:TANumR,:);
end

% Adding Header
for k = 15:46
    EE.txt{1,k} = [char(039), EE.txt{1,k}];% Adding apostrophe before bin numbers so they
are saved as text
end

TAHeader = {TAHeader{:}, EE.txt{1,[9 15:end]}};

% Calculating additional Particle info
TANumC = size(TAData,2);
TAHeader = {TAHeader{:}, 'Calc TPN [#/cm³]','Calc TUFP [#/cm³]'};
TAData(:,TANumC+1) = sum(TAData(:,58:89),2)/16;
TAData(:,TANumC+2) = sum(TAData(:,58:77),2)/16;

% Normalizing data for plotting
AFRPMBounds = [min(TAData( :, 27)); max(TAData(:, 27))];
STRPMBounds = AFRPMBounds;
TPNBounds = [min(TAData( :, 90)); max(TAData(:, 90))];
EETPNBounds = [min(EEData( :, 47)); max(EEData(:, 47))];

NormSTRPM =  (TAData(:, 29) - STRPMBounds(1,1))/(STRPMBounds(2,1)-STRPMBounds(1,1));
NormAFRPM =  (TAData(:, 27) - AFRPMBounds(1,1))/(AFRPMBounds(2,1)-AFRPMBounds(1,1));
NormTPN =  (TAData(:, 90) - TPNBounds(1,1))/(TPNBounds(2,1)-TPNBounds(1,1));

NormEETPN = ((EEData(:, 47)-EETPNBounds(1,1))/(EETPNBounds(2,1)-EETPNBounds(1,1)));

%Plotting normalized data for visual verification of time alignment
figure (9)
plot(TAData(:, 8), NormSTRPM, TAData(:, 8), NormAFRPM, EEData(:, 8), NormEETPN, TAData(:,
8), NormTPN);
legend('ST RPM','Corr AF RPM', 'TPN', 'Time Aligned & DR Corrected TPN')
ylabel('Normalized RPM, Blank corrected TPN, and Blank & DR Corrected TPN')
xlabel('Time (sec)')
ylim([0 1])

%% FTIR Data
% ------------------------------------------------------------------------
% Correlating FTIR CO2 to Scantool RPM
% Since FTIR data is logged at about 1 Hz, interpolation is necessary

% Find data without background data records -- necessary because
% reprocessed data has background data at the front end from much earlier
% in the day
```



```
FTindDiff60 = [ ];
for FTind = 2:size(FT.data,2);
    FTdiffTime = (FT.data(FTind,strmatch('Elapse Time',FT.txt)) - FT.data(FTind-1,
strmatch('Elapse Time',FT.txt))) > 60;
    if FTdiffTime == 1;
        FTindDiff60 = [FTindDiff60;FTind];
    end
end
% check if there is background data at the top of the file, if not, just
% assign the data to a new variable name
if isempty(FTindDiff60) == 1;
    FT.data1 = FT.data;
elseif isempty(FTindDiff60) == 0;
    FT.data1 = FT.data(FTindDiff60(1,1):end,:);
end

% fix the instrument bounds to match the bounds of file 3 if data was
% reprocessed and background data included in the file
InstBounds(1,strmatch('F',InstLet)) = ceil(FT.data1(1,strmatch('Elapse Time',FT.txt)));

% Create matrix of elapsed time at 1 Hz increments for interpolation of
% data to the nearest one-second
FTNewElapsed = [InstBounds(1,strmatch('F',InstLet)):1:InstBounds(2,strmatch('F',
InstLet))]';

% Create matrix of interpolated FTIR data using the new elapsed time
FTInterp = interp1(FT.data1(:,strmatch('Elapse Time',FT.txt)),[FT.data1(:,strmatch
('SOY_FTIR',FT.txt)) FT.data1(:,strmatch('1,2,4-Trimethylbenzene 191c',FT.txt):strmatch
('Pressure (Atm)',FT.txt))],FTNewElapsed);
FTInterpHeaders = FT.txt(1,[strmatch('SOY_FTIR',FT.txt) strmatch('1,2,4-Trimethylbenzene
191c',FT.txt):strmatch('Pressure (Atm)',FT.txt)]);


TANumC = size(TAData,2);

FTBoundLow = find(TAData(:,strmatch('ElapseTime',TAHeader)) == InstBounds(1,strmatch('F',
InstLet)));
FTBoundHigh = find(TAData(:,strmatch('ElapseTime',TAHeader)) == InstBounds(end,strmatch
('F',InstLet)));
FTBounds = zeros(size(TAData,1),1);
% if the bounding rows from FTIR are not in TAData, then assign all of the
% FTIR data and limit the size to TAData
if sum(FTBoundHigh) == 0 && sum(FTBoundLow) == 0;
    FTBounds(1:end) = 1;
    FTLims(1:size(TAData,1)) = 1;
    FTBoundLow = 1;
    FTBoundHigh = size(TAData,1);
    % if just the high bounding row is defined
elseif sum(FTBoundLow) == 0;
    FTBounds(1:FTBoundHigh) = 1;
    FTLims(1:FTBoundHigh) = 1;
```

```
    FTBoundLow = 1;
    % if just the low bounding row is defined
elseif sum(FTBoundHigh) == 0;
    FTBounds(FTBoundLow:end) = 1;
    FTLims(1:size(FTInterp,1)) = 1;
    FTBoundHigh = 1;
    % if both bounding rows are defined
else
    FTBounds(FTBoundLow:FTBoundHigh) = 1;
    FTLims(1:size(FTInterp,1)) = 1;
end
FTBounds = logical(FTBounds);
FTLims = logical(FTLims);

% Read in FTCo2 data to FTBounds
FTCo2 = FTInterp(FTLims,strmatch('CO2% 191C',FTInterpHeaders));

% Align CO2 and RPM
% Correlation check
AFRpm = TAData(FTBounds,strmatch('Engine Speed  [rpm]',TAHeader));
STRpm = TAData(FTBounds,strmatch('Engine Speed (RPM)',TAHeader));

FTshift = [-200:200]';

[FTrhoTA,FTpValTA] = corr(STRpm(:,1),FTCo2(:,1),'rows','pairwise');
FTco2match = FTCo2;
FTco2shift = NaN(size(FTco2match,1),size(FTshift,1));
for indShift = 1:size(FTshift,1)-1;
    if FTshift(indShift) < 0;
        FTco2shift(1:end-abs(FTshift(indShift)),indShift) = FTco2match(abs(FTshift↙
(indShift))+1:end);
    elseif FTshift(indShift) >= 0;
        FTco2shift(FTshift(indShift)+1:end,indShift) = FTco2match(1:end-FTshift↙
(indShift));
    end
end
for indShift = 1:size(FTshift,1)-1;
    [FTrhoSA(indShift,1),FTpValSA(indShift,1)] = corr(STRpm,FTco2shift(:,↙
indShift),'rows','pairwise');
end
[FTmaxRho,FTindMax] = max(FTrhoSA);
FTcorrResults = [FTrhoSA(FTindMax) FTrhoSA(FTindMax) FTshift(FTindMax)];
FTLagRow = FTshift(FTindMax);
FTLagRow

TAData = [TAData, NaN(size(TAData,1), size(FTInterp,2))]; % Adding NAN positions to↙
matrix for ST Data
% KMS Add condition if negative lag and low bounding row
if FTBoundLow+FTLagRow < 0
    for i = FTBoundLow:FTBoundLow+size(FTInterp(FTLims,:),1)-1+FTLagRow
        TAData(i,TANumC+1:TANumC+size(FTInterp,2)) = FTInterp((i-FTLagRow),:); % Adding↙
```



```
All FTIR data
    end
else
    for i = FTBoundLow+FTLagRow:FTBoundLow+size(FTInterp,1)-1+FTLagRow
        TAData(i,TANumC+1:TANumC+size(FTInterp,2)) = FTInterp(i+1-↙
(FTBoundLow+FTLagRow),:); % Adding All FTIR data
    end
end
TAHeader = {TAHeader{:},FTInterpHeaders{1,:}};

% Visual confirmation of lag alignment
figure
[ax,l1,l2] = plotyy(TAData(:,strmatch('ElapseTime',TAHeader)),TAData(:,strmatch('CO2%↙
191C',TAHeader)),TAData(:,strmatch('ElapseTime',TAHeader)),TAData(:,strmatch('Engine↙
Speed (RPM)',TAHeader)));
ylabel(ax(1),'CO_2 (%)')
ylabel(ax(2),'Scantool RPM')
xlabel('Elapsed Time (sec)')


%% Calculate Averages

% %     AveOmegaDR = mean(TAData(:, 53))
% %     StDevOmegaDR = std(TAData(:, 53))
indices = TAData(:,13) ~= 0;
    AveMAGDR = nanmean(TAData(indices, 54))
% %     StDevMAGDR = std(TAData(:, 54))
% %     AveMovAveMAGDR = mean(TAData(WinSize/2+1:TANumR-WinSize/2, 94))
% %     StDevMovAveMAGDR = std(TAData(WinSize/2+1:TANumR-WinSize/2, 94))

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

    %% Writing Time Aligned Data to File
    cd(TDataD);
    %Outputing Armfield Data
    TADataName = strcat(RunID,'_ASLEFOut_',upper(datestr(now, 'ddmmmyyyy_HHMM')),'.xls');

    LagNames = {'Date computed'; 'Master time = Armfield'; 'Scantool Lag (sec)';'AF RPM↙
Lag (sec)'; 'Labview Lag (sec)'; 'EEPS Lag (sec)'; 'FTIR Lag (sec)'; 'Ave Engine On DR'};
    LagTimes = [STLagSec;AFRPMLagSec; LVLagSec; EELagSec; FTLagRow; AveMAGDR];
    CompDate = {upper(datestr(now, 'ddmmmyyyy_HH:MM'))};
    %****Using the standard xlswrite in Matlab***
    xlswrite(TADataName,LagNames,'Sheet1','A1');
    xlswrite(TADataName, CompDate,'Sheet1','B1');
    xlswrite(TADataName, LagTimes,'Sheet1','B3');
    xlswrite(TADataName,TAHeader,'Sheet1','A9');
    xlswrite(TADataName,TAData,'Sheet1','A10');
    %--------------------------------------------------------------------
```

CODE_4_26JUN2014.m


```
%Tyler Feralio
%CM12 Engine Dyno Data compiler for 2013 transient / steady state test sequence
%Retrieves files generated by codes 3, concatenates them, and outputs one
%large .txt file

%Armfield
%Scantool (VCDS)
%Labview
%EEPS


% Record of Revisions:
%   Date        Programmer        Description of Change
%   ====        ==========        =====================
%   3/6/14      Tyler Feralio
%   6/26/14     Karen Sentoff
%   2/27/15     Karen Sentoff     Add in DEC B000 runs; change File.data
%                                   calls for structure in case Sheet1 is
%                                   not created in data files

close all; clear all; clc;

%% Manual input for run IDs

% Input needed to build run ID
%Run ID's of the data to be concatenated
% 1_12MAY2014_B010 and 2_05AUG2014_B000 not included; should be 37 runs
RunIDs = {'1_18JUN2013_B000', '1_25JUN2013_B000', '1_06AUG2013_B000',...
    '1_29AUG2013_B010', '1_30AUG2013_B010', '1_31AUG2013_B010',...
    '1_04SEP2013_B020', '1_05SEP2013_B020', '1_06SEP2013_B020',...
    '1_09SEP2013_B050', '1_10SEP2013_B050', '1_11SEP2013_B050',...
    '1_19SEP2013_B100', '1_20SEP2013_B100', '2_20SEP2013_B100',...
    '1_22OCT2013_B000', '1_24OCT2013_B000', '1_25OCT2013_B000',...
    '1_04DEC2013_B000', '1_05DEC2013_B000', '1_06DEC2013_B000',...
    '1_01MAY2014_B000', '1_02MAY2014_B000', '1_05MAY2014_B010',...
    '1_06MAY2014_B010', '2_06MAY2014_B010', '1_07MAY2014_B020',...
    '2_07MAY2014_B020', '2_12MAY2014_B020', '1_13MAY2014_B020',...
    '2_13MAY2014_B020', '1_14MAY2014_B020', '1_15MAY2014_B020',...
    '1_20MAY2014_B050', '2_20MAY2014_B050', '1_22MAY2014_B050',...
    '1_23MAY2014_B100', '2_23MAY2014_B100', '1_26MAY2014_B100',...
    '1_27MAY2014_B000'};
NumRuns = size(RunIDs,2);

RunQA = [1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;0;1;1;1;1;0;1;1;0;1;1;1;1;1;1;1;1;1;↙
1];

%% Pull data to write to file

%Defining directories names to retrieve data from
%Setting Directory that matlab files are in as the Parent Directory
PD=cd;
```



```
for i = 20:21 %1:NumRuns % check with 20:25
    Date = datevec(RunIDs{1,i}(3:11));
    Yr = Date(1,1);
    Month = strcat(strrep(sprintf('%0.1f',(Date(1,2))/10),'.',''),'_',RunIDs{1,i}(5:7));
    %Defining directories:
    %Main Data Directory path
    Dir=strcat('H:\SP2_ArmfieldEngine\CM12_data\', num2str(Yr),'\', Month,'\',RunIDs{1,i}↙
(3:11));
    %Checking if multipule runs were made in the same day by looking for
    %subfolders
    mr=exist(strcat(Dir,'\',RunIDs{1,i}));
    %deciding which directory to pull data from
    if mr>0
        %Defining file paths if more then one run was collected in the same day
        TDataD=(strcat(Dir,'\',RunIDs{1,i},'\Instrument_Data'));

    else
        %Defining file paths if only one run was collected in the same day
        TDataD=(strcat(Dir,'\Instrument_Data'));
    end
    cd(TDataD);
    Folder = dir;
    k=0;
    for j = 1:size(Folder,1)
        % KMS changed number of characters to 26 from 31
        d = strncmp(strcat(RunIDs{1,i},'_ASLEFOut_'), Folder(j,1).name,26);
        if d ==1
            FilePos = j;
        end
    end
    FileName = Folder(FilePos,1).name;
    File = importdata(FileName,'\t');
    % KMS added lines to check the structure of the File.data to see if a
    % Sheet name is necessary to call or not
    tfStruct = isstruct(File.data);
    if tfStruct == 1
        NumR(i,1) = size(File.data.Sheet1(8:end,:),1);
        NumC(i,1) = size(File.data.Sheet1,2);
        FileData = File.data.Sheet1;
        FileText = File.textdata.Sheet1;
    elseif tfStruct == 0;
        NumR(i,1) = size(File.data(8:end,:),1);
        NumC(i,1) = size(File.data,2);
        FileData = File.data;
        FileText = File.textdata;
    end
    %
    if i == 20%1 %check with 20
        % KMS changed hardcoded data limits to limits of actual data
        Output = zeros(NumR(i,1), NumC(i,1)+2);
        Output(:,1) = i;
```

```
        % KMS added QA 1=valid and 0=not valid
        Output(:,2) = RunQA(i);
        Output(:,3:end) = FileData(8:end,:);
        % KMS added for check of data length (in case FTIR data was
        % reprocessed and did not match)
        OutputHeader = {'Run','QA',FileText{9,1:end}};
    else
        clear Output1
        Output1 = zeros(NumR(i,1), NumC(i,1)+2);
        Output1(:,1) = i;
        % KMS added QA 1=valid and 0=not valid
        Output1(:,2) = RunQA(i);
        Output1(:,3:end) = FileData(8:end,:);
        % KMS added for check of data length
        Output1Header = {'Run','QA',FileText{9,1:end}};
        % KMS added check to make sure number of columns matches, if not check
        % headers and create dummy columns where needed (won't be necessary
        % in the future, but necessary if different number of FTIR
        % columns)
        if size(Output,2) ~= size(Output1,2);
            OutputDiffInd = strncmp(OutputHeader,Output1Header(1,1:size(OutputHeader,2)),↙
size(OutputHeader,2));
            OutputDiffFix = OutputDiffInd == 0;
            Output = [Output(:,OutputDiffInd) NaN(size(Output,1),1) Output(:,↙
OutputDiffFix)];
        end

        % write new data to existing matrix
        Output = [Output; Output1];
    end
end
NumC = size(Output,1);
% KMS added QA data header
OutputHeader = {'Run','QA',FileText{9,1:end}};

%% Write file
cd('H:\SP2_ArmfieldEngine\CM12_data\2014')
fileID = fopen(strcat('Biodiesel_DataSet_',upper(datestr(now, 'ddmmmyyyy')),'.txt'),'w');
fprintf(fileID,[repmat(' %s\t',1,size(Output,2)-1),' %s\r\n'],OutputHeader{:});
fprintf(fileID,[repmat(' %f\t',1,size(Output,2)-1), ' %f\r\n'],Output');
fclose(fileID);
```