2019

APPLICATION OF SENSOR FUSION FOR SI ENGINE DIAGNOSTICS AND COMBUSTION FEEDBACK

FNU Muralidhar Nischal
Michigan Technological University, nmuralid@mtu.edu

APPLICATION OF SENSOR FUSION FOR SI ENGINE DIAGNOSTICS AND COMBUSTION FEEDBACK

By

Muralidhar Nischal

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mechanical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2019

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Mechanical Engineering.

Department of Mechanical Engineering-Engineering Mechanics

Thesis Co-advisor: *Dr. Jeffrey D. Naber*

Thesis Co-advisor: *Dr. Jason R. Blough*

Committee Member: *Dr. Mahdi Shahbakhti*

Department Chair: *Dr. William W. Predebon*

Dedication

To my parents, family, teachers and friends

for having always helped me excel in my endeavors. I wouldn't be the person I am today without you'll.

Contents

List of Figures

List of Tables

Acknowledgments

I am grateful to Dr. Jeffrey D. Naber and Dr. Jason R. Blough for giving me an opportunity to be a part of their research group. I am also thankful to them for their guidance and supervision throughout this thesis work. I would also like to thank Dr. Mahdi Shahbakhti and Dr. Bo Chen for their suggestions along the way.

I would like to sincerely thank Ford Motor Company for sponsoring this research and my Masters degree. I am grateful to the Ford research team including Garlan Huberts, Chris Gugla, Chad Archer, Darren Nester, Qiuping Qu and Ken Rhodes for their constant support throughout the project and especially during my time at Ford. I'd also like to thank the Ford management, specifically Todd Rumpsa for the timely inputs and suggestions.

I would like to thank my dear friends Xin Wang, Amir Khameneian and Kaushik Prabhu for helping me conduct experiments in a timely fashion and also for their company throughout this experience. I would also like to acknowledge Paul Dice, Joel Duncan and the other staff members of the APS Labs for their support.

I would also like to thank my batchmates and friends Srihari, Vivek, Suman, Kruthika, Shiva, Sweta, Anupam and Sunit for their company and encouragement throughout my masters degree.

Last but not least I would like to thank my parents, family and buddies back home for all their love and encouragement.

- Muralidhar Nischal

Abstract

Shifting consumer mindsets and evolving government norms are forcing automotive manufacturers the world over to improve vehicle performance and also reduce greenhouse gas emissions. A critical aspect of achieving future fuel economy and emission targets is improved powertrain control and diagnostics.

This study focuses on using a sensor fusion based approach to improving control and diagnostics in a gasoline engine. A four cylinder turbocharged engine was instrumented with a suite of sensors including ion sensors, exhaust pressure sensors, crank position sensors and accelerometers. The diagnostic potential of these sensors was studied in detail. The ability of these sensors to detect knock, misfires and also correlate with pressure and combustion metrics was also evaluated.

Lastly a neural network based approach to combine individual sensor signal information was developed. The neural network was used to estimate mean effective pressure and location of fifty percent mass fraction fuel burn. Additionally, the influence of various neural network architectures was studied.

Results showed that under pseudo transient conditions a recursive neural network could use information from the low cost sensors to estimate mean effective pressure within an error of 0.1bar and combustion phasing within 2.5 crank-angle degrees.

Chapter 1

Introduction

The world today is striving towards using alternative forms of energy. The automotive sector is no exception to this migration. Global treaties, technological advancements and evolving consumer mindsets are driving auto manufacturers to adopt greener, more efficient strategies and produce vehicles with minimal emissions. Electric vehicles (EVs) present an alternative to fossil fuel operated vehicles however infrastructure constraints prevent electric vehicles from dominating the market.

A solution then is to improve the performance and reduce the emissions of fossil fuel operated vehicles until such time that the necessary infrastructure is built to make EV the primary vehicle architecture. Further, to enhance operation of conventional engines, one approach would be to achieve greater control of engine operation.

Engine control units have evolved by leaps and bounds since their advent, however much like any control system, the effectiveness of the controller is hugely dependent on the accuracy of information supplied by the sensors. Engines today use a plethora of sensors for control and diagnostic purposes, one such sensor is the in-cylinder pressure sensor (ICPS). In a research setup, it is quite common to use an ICPS, however in a production scenario, implementing ICPS is not viable due to cost and maintenance considerations. The challenge then is to develop sensors that could provide the same fidelity of information as an ICPS but at a fraction of the cost.

This study thus aims at studying the prospect of using low cost sensors, both existing and new, for their application in engine control and diagnostics. The fundamental idea is to fuse together the information supplied by multiple sensors to ascertain critical combustion metrics like indicated mean effective pressure (IMEP) and crank angle for 50 percent burn (CA50). Further, the diagnostic potential of the sensor suite is evaluated, specifically knock, misfire and partial/late burn detection.

The potential benefits of following such a sensor fusion approach includes :

- Improvement in fuel economy including enhanced knock and closed loop dilution control
- Reduced calibration effort and time
- Extended life of catalytic converter due to improved diagnostics

1.1 Goals and Objectives

The goals of this study are as listed below:

1. Identify signal artifacts in output of sensors and their correlation with combustion metrics
2. Study potential of using specified sensors to identify and quantify engine knock
3. Study potential for detecting abnormal combustion cycles i.e. late burn, partial burn and misfire
4. Perform transient testing, both vehicle level and using dynamometers
5. Assess feasibility of using artificial intelligence (AI) including advanced neural networks in tandem with multiple sensor inputs to estimate combustion metrics

1.2 Thesis overview

This thesis is divided into five chapters. In Chapter 1 and introduction to the study and the objectives of the current work is covered. A brief literature review is conducted in Chapter 2. In Chapter 3, details regarding the experimental setup including technical specification of the engine and various sensors used in this study are described. The various methods and algorithms developed as part of this study are discussed in Chapter 4. In Chapter 5, a detailed analysis of the data and results is

conducted. Finally, Chapter 6 concludes this study and offers recommendations for future work.

Chapter 2

Literature Review

The in-cylinder pressure sensor is a valuable control input. Significant effort has been devoted towards developing means and method to reconstruct the in-cylinder pressure signal. Studies by Jia et al. [3] showed that Frequency Response Functions(FRF) generated for a particular engine operating point can be altered to be applied over a range of operating points to recreate an in-cylinder pressure waveform. Particle swarm optimization was utilized to selectively alter low frequencies that were dominant in the FRF.Another approach to pressure waveform reconstruction using FRFs was carried out by Liu et al. [4] wherein the in-cylinder pressure close to the top dead center (TDC) was reconstructed with FRFs using crank-shaft velocity as an input. Researchers have also used structure borne signals [5] and fluctuations in engine speed [6] to reconstruct cylinder pressure waveforms.

Combustion diagnostics constitute a significant portion in evaluating component or system performance and consequently vehicle drive quality and operation. Work carried out by Bahri et al. [7] utilizes a neural network (NN) to identify misfires in a ethanol fueled HCCI engine. The study used a small subset of in-cylinder pressure readings to train a NN to predict misfires with tremendous accuracy. Another approach to detecting misfires in the absence of ICPS was showcased by Willimowski and Isermann [8] and also Prabhu [1] where the exhaust sensor was utilized as a means to successfully identify misfire events.The studies observed that exhaust signal artifacts specifically exhaust minima were a reliable indicator of engine misfire.

Giglio et al. [9] conducted a study to show the potential of a spark coil integrated ion sensor in detecting knock. The study used engine data of a number of steady state operating points and conducted an analysis to evaluate correlation between knock metrics and equivalent ion metrics developed as part of the study. Panousakis et al. [10] focused efforts on evaluating the ability of ion signals to detect misfire and ignition timing. The study also evaluated correlation between ion signal intensity and air-fuel ratio as well as compression ratio. Another study to explore the diagnostic capabilities of ion signals was conducted by Cavina et al. [11] wherein the ion integral was used to identify misfires and partial burns. Although ion integral could reliably identify misfires, identification of partial burns showed limited success.

Dev et al. [12] carried out a study to use ion signal as a means to determine combustion

phasing namely location of five percent mass fraction fuel burn (CA5) and location of fifty percent mass fraction fuel burn (CA50). A custom ion sensor integrated into a multi-electrode spark plug was used in the study to show that ion based combustion phasing estimates were comparable to combustion phasing information derived from pressure based measurements. Abhijit and Naber [13] investigated the ability of ion sensors to detect knock and found that when appropriately processed, frequency content dominant in ion signals were similar to that seen in the in-cylinder pressure signal. The study also showed correlation between pressure-based knock intensity (PI) and ion intensity (II).

Researchers have also studied the application of crankshaft speed for control and diagnostic purposes. A study by Yinhui et al. [14] demonstrated the use of crankshaft speed for misfire recognition where the filtered second derivative of the crank signal was used in tandem with a static threshold to identify misfiring cylinders at a given operating condition. Azzoni et al. [15] were able to identify misfires in a V12 engine using crankshaft speed and applying suitable signal processing techniques like Discrete Fourier Transforms. Other signal processing methods including Frequency Response Functions (FRFs) have also been used to process crank data to obtain estimates of in-cylinder pressure [16]. Brown and Neill [17] on the other hand employed pattern recognition of crank angular velocity for estimating in-cylinder pressure. The variation in angular velocity of a particular cycle was compared to a knowledge base of patterns based on the operating point to draw up an estimate of in-cylinder pressure.

Several studies have also used crankshaft speeds as input to neural networks to estimate combustion parameters and for signal reconstruction purposes. Taglialatela et al. [18] used crank speed and acceleration as inputs to a multilayer perceptron network in order to estimate peak cylinder pressure and location of peak pressure. The study was able to obtain network estimated pressure signals that were within 0.5bar-1bar (depending on operating point) of actual data and also proposed utilizing the network estimated in-cylinder pressure peak and location to identify abnormal combustion. Another study by Saraswati and Chand [19] made use of a recursive neural network with crankshaft speed and motorized pressure as inputs to develop a network capable of reconstructing in-cylinder pressure.

Thus through the brief literature survey it can be seen that several different studies have used various sensors and methodologies to develop alternatives for an ICPS, which is a valuable input for closed loop combustion control. Studies have also utilized these alternative sensors for diagnostic purposes with varying degrees of success and reliability. However none of the studies utilize the entire sensor suite as present in the current study. Previous studies in this area also primarily focused on evaluating sensor capabilities under steady state conditions. The current study however, includes analysis and results from not just steady state studies but also transient studies conducted both on an engine dynamometer as well as in-vehicle.

Chapter 3

Experimental setup and sensor specification

This chapter broadly describes the experimental setup used in this study and also presents the specification of instrumentation and sensors utilized as part of the study.

A brief schematic of the experimental setup used in this study is provided in Figure 3.1. As part of the study, the engine was instrumented with multiple sensors. Table 3.1 lists the various sensors instrumented onto the setup. A number of these sensors played a pivotal role in the current study. Specifications regarding the sensors utilized in this study are provided later in this chapter.

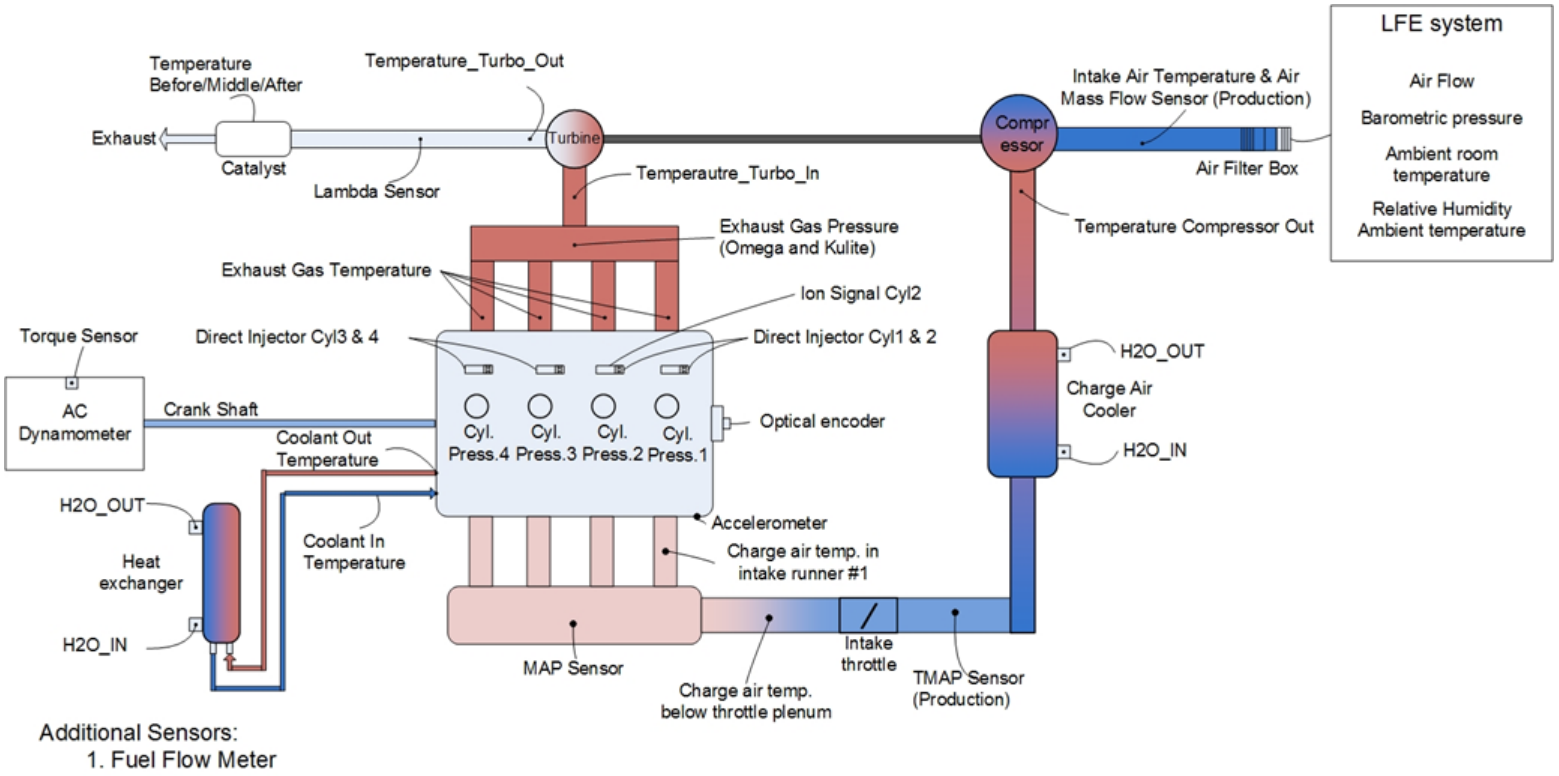


Figure 3.1: Schematic of engine setup used in study

Table 3.1
Sensors instrumented in setup

Sensor Name	**Primary Purpose**
Direct mounted exhaust pressure sensor	Combustion metric correlation and misfire diagnostics
Standoff mounted exhaust pressure sensor	Similar to high pressure sensor but mounted with a standoff
Standalone ion probe	Detection of flame front and combustion phasing information
Coil integrated ion probe	Combustion phasing detection
Crank position sensor	Measure engine speed variations
Accelerometer	Knock/misfire detection

3.1 Engine Specification

This study utilized an inline 4 cylinder 2.0L gasoline direct injection (GDI) engine as shown in Figure 3.2. The engine was provided to the Advanced Power Systems Research Center (APSRC) by Ford Motor Company as part of a larger collaborative research effort between Michigan Tech and Ford. Table 3.2 offers the specifications of the engine used.

Table 3.2
Engine specifications

Specification	Values	Units
Engine displacement	2.0	lit
Number of cylinders	4	N/A
Block/head material	Aluminum	N/A
Bore	87.5	mm
Stroke	83.1	mm
Connecting rod length	155.86	mm
Wrist pin offset	0.6	mm
Compression ratio	9.3:1	N/A
Firing order	1-3-4-2	N/A
Production crank sensor type	Hall effect	N/A
Crank sensor type used for data logging	Optical	N/A

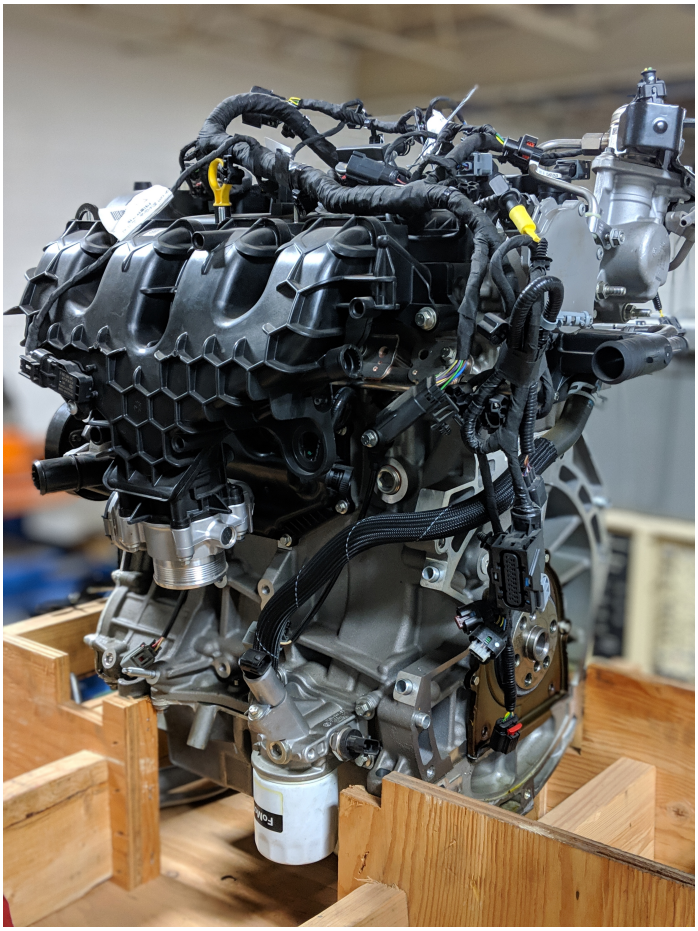


Figure 3.2: Representative image of engine used in study

As previously mentioned the engine is gasoline operated. AKI 87 fuel was used during testing. Fuel properties are further elaborated in Table 3.3 and are similar to a previous study [1] conducted using the same setup.

Table 3.3
Properties of fuel used

Properties	Values	Units
Carbon	83.06	Percentage weight
Hydrogen	13.48	Percentage weight
Oxygen	3.46	Percentage weight
Density	741.9	kg/m^3
Lower heating value	41.725	MJ/kg
Stoichiometric Air-Fuel Ratio (AFR)	14.06	N/A
Research Octane Number	91.7	N/A
Motor Octane Number	82.5	N/A

3.2 Exhaust Pressure Sensor

Exhaust pressure sensors function based on either piezo-resistivity or piezo-electricity . A piezo-resistive strain gauge is placed in a specific arrangement on a diaphragm. As the pressure changes, the diaphragm expands or contracts based on the pressure change. This in-turn causes the resistance of the strain gauge to change. The change in resistance leads to a voltage change which is consequently converted to a pressure reading. Piezo-electric based sensors on the other hand use a piezoelectric crystal which when subjected to a force, produce a charge which can then be suitably conditioned to obtain a voltage signal.

In this study two types of exhaust pressure sensors were used to obtain insights into engine performance and capture combustion dynamics. A high pressure piezoelectric exhaust sensor was mounted directly in the exhaust manifold, less than 10mm from the exhaust port of the engine as shown is Fig. 3.3. This directly mounted pressure sensor also referred to as the Kulite sensor (after the manufacturer name) was capable of operating at high temperatures and pressures. Table 3.4 lists the specifications of the Kulite sensor. The primary purpose of the Kulite sensor was to capture the exhaust gas dynamics. A second piezo-resistive exhaust pressure sensor, also called the Omega sensor(after the manufacturer) was placed at a standoff, about 18 inches from the exhaust port, as shown in Fig. 3.3. The Omega sensor had limited tolerance to high temperature and pressure in comparison to the Kulite sensor as the Omega sensor could only operate up to a temperature of 85 deg. C and 10.5 bar pressure while the Kulite could sustain temperatures of about 500 deg.C and 20 bar pressure. The tubing connecting to the Omega sensor acts as a low pass filter (cut-off frequency 325Hz [1]) thereby limiting the information captured in the signal. Specifications of of the Omega sensor are provided in Table 3.4. It is to be noted that the sensor details in Table 3.4 are the specifications of the sensors used in the test cell and not that of the sensors used in the vehicle tests.

Table 3.4
Specification of exhaust pressure sensors used in test cell

Property	**Unit**	**Kulite**	**Omega**
Rated Pressure	bar abs.	10	3.5
Maximum Pressure	bar abs	20	10.5
Excitation Voltage	VDC	12	28
Sensitivity	mV/bar	456.5	501.5
Operating Temperature Range	deg. C	-55 to 500	-20 to 85
Bandwidth	kHz	150	1
Output Range	VDC	0.5 to 5	0 to 5
Part number	N/A	ETL312M	PX309

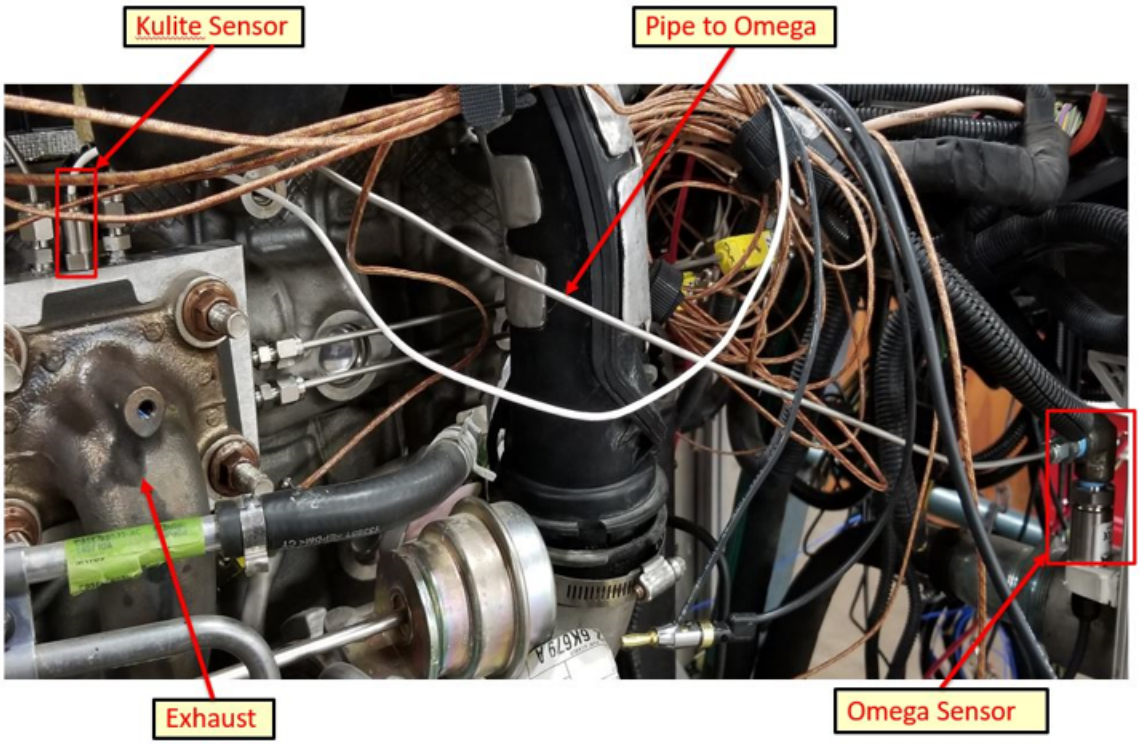


Figure 3.3: Positioning of exhaust pressure sensors on engine [1] . Reprinted with permission from original author. Refer Appendix B

The specifications of the sensors used in the test cell and in vehicle tests differ slightly due to lack of availability of sensors with the exact same specifications. Table 3.5 details the specifications of the sensors used in vehicle tests. Unlike the engine tests, the vehicle was mounted with 2 Kulite sensors instead of one. This was primarily to ensure testing would not be hindered due to sensor failures. Figure 3.4 shows the flange designed to mount the exhaust sensor close to the exhaust port of the engine in the vehicle, a Lincoln MKC. The flange design is similar to that used in the engine at the APS Labs.

Table 3.5
Specification of exhaust pressure sensors used in vehicle tests

Property	Unit	Kulite	Kulite	Omega
Rated Pressure	bar abs.	3.45	6.89	6.89
Maximum Pressure	bar abs	6.89	13.79	13.79
Excitation Voltage	VDC	12	12	28
Sensitivity	mV/bar	1296.85	659.30	724.5
Operating Temperature Range	deg. C	-55 to 538	-55 to 500	-20 to 85
Bandwidth	kHz	150	50	1
Output Range	VDC	0.5 to 5	0.5 to 5	0 to 5
Part Number	N/A	ETL312M	ETL190	PX359

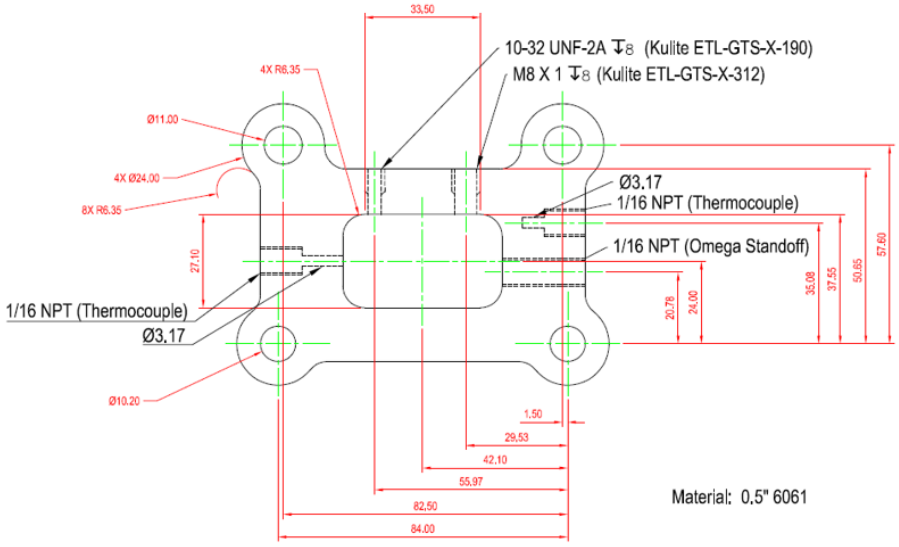


Figure 3.4: Mounting diagram for exhaust sensors used in vehicle testing

3.3 Ion Sensors

Ions probes function based on the principle that when a DC voltage is applied to two electrodes, charged particles in the vicinity of the electrode cause a current to flow which can be detected by the same sensor. The current is thought to be flowing due to ions in the combustion charge, present near the probe, and is hence termed as ionization current. The ion current is then electronically converted to a voltage signal which in turn is used for combustion sensing and diagnostics. Popular applications of ion sensors are in misfire and knock detection as well as estimating combustion phasing. This study utilizes ion sensors for similar purposes.

Ion sensors are predominantly of two types. Coil integrated ion probes and standalone ion probes. Coil integrated ion probes use the spark plug as an ion sensor. Once the spark discharge is complete the spark gap of the plug is utilized to sense ions. This form of ion sensors do not suffer from packing issues and minimize rework on engine head geometry, however the method suffers from ringing issues caused by the coil circuitry. Figure 3.5 offers a pictorial representation of the coil integrated ion probe used in this study along with the wiring diagram and pin configuration.

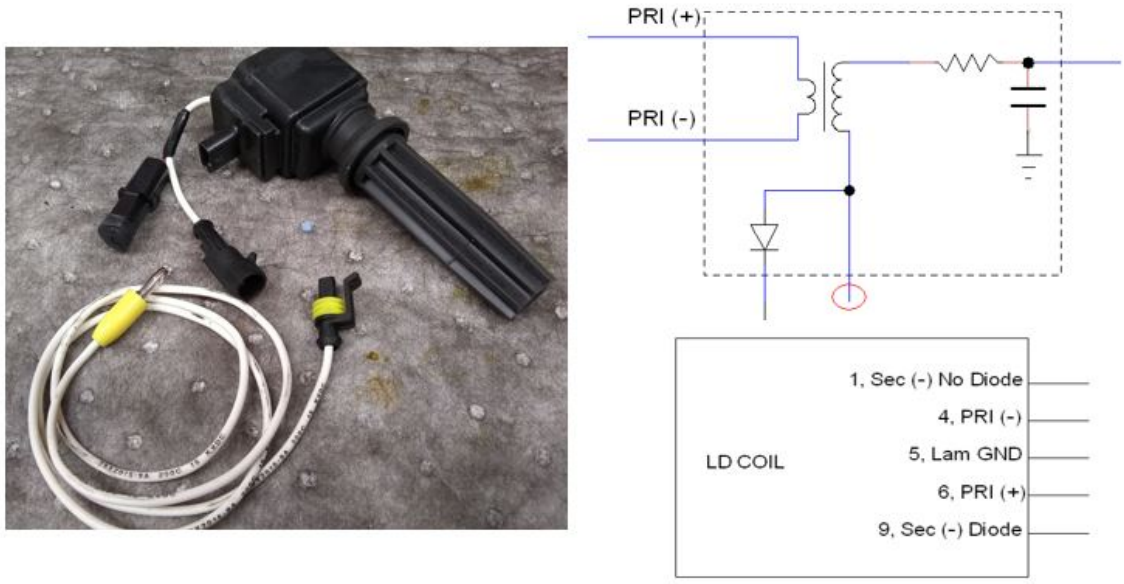


Figure 3.5: Ignition coil of coil integrated ion sensor

Another means of ion sensing, also used in this study, is to use a standalone ion probe as shown in Fig 3.6. The standalone ion probes used in this study were custom probes fabricated at MTU. Standalone ion probes do not suffer from the same ringing issues as in coil integrated probes but face issues in terms of packaging. It is to be noted that the standalone ion probe used in this study evolved over the course of the study, but for any given test only a single standalone ion probe was used. A list of specifications of the initial and final version of the custom ion probes are given in Tables 3.6 and 3.7 respectively. Additionally the ion sensors output was connected to a custom conditioning box provided to MTU by Ford. The conditioning box helped reduce the noise on sensor output thereby making it more suitably for data analysis. The rating of the boxes include 1MΩ, 300kΩ and 250kΩ.

Figure 3.6: Standalone ion sensor probes

Table 3.6
Specification of initial standalone ion probe

Electrode Properties	Values	Units
Material	steel	N/A
Tip protrusion	2.5	mm
Conditioning box rating	1	MΩ
Face insulation material	RTV	N/A
Location of sensor	Cylinder 2	N/A

Table 3.7
Specification of latest standalone ion probe

Electrode Properties	Values	Units
Material	Tungsten	N/A
Tip outer diameter	2.4	mm
Tip protrusion	4	mm
Conditioning box rating	300	kΩ
Location of sensor	Cylinder 2	N/A

3.4 Crank-angle Encoder

Encoders or more specifically rotary encoders are used as a means to monitor the position and speed of rotation of a shaft. Optical variants of an encoder use a light source and a disc with predefined transparent and opaque regions. As the shaft rotates, so does the disc. A photo-diode detects the light passing through the transparent portions of the disc. This in-turn generates a pulse train that is supplied to an external processor which consequently calculates the position and speed of the shaft.

Figure 3.7: Optical encoder

In engines, these sensors find applications in monitoring crankshaft rotation and speed. Production engines use crankshaft position sensors with lower resolution crank wheel, that work primarily based on the Hall effect as they are easy to package and sufficiently accurate for control applications. However in a research environment,

such as this study, greater resolution and higher accuracy in angular data is required. Figure 3.7 shows an image of the incremental optical encoder used in this study, made by BEI Sensors. Specifications of the same can be found in Table 3.8

Table 3.8
Specification of Optical encoder

Property	Value	Units
Shaft Material	Stainless Steel	N/A
Shaft diameter	3/8	inch
Maximum RPM	12000	RPM
Encoder type	Incremental	N/A
Input Voltage	5-28	V
Disc Resolution	0.5	CAD
Part Number	H25	N/A

3.5 Optical Engine

A portion of this study, pertaining to evaluating the influence of ion sensor location on correlation studies was conducted on an optical engine. The optical engine present at the APSRC was a modified 2.0L 4-cylinder Ford ecoboost engine developed by Mahle Powertrain [2]. The specifications of the engine are given in Table 3.9 and Figure 3.8 offers a visual representation of the same. Of the four cylinders, only cylinder two is active and optically accessible, the remaining three cylinders are deactivated by grounding off the cam lobes. In order to accommodate the piston extension, the cylinder head of the engine is separated from the engine block and placed at an elevation. The piston extension is in-turn threaded to a flat-top aluminum piston

with a quartz window insert.

Table 3.9
Specification of Optical engine

Property	Value	Units
Cylinder Displacement	0.6	L
Bore	87.5	mm
Stroke	100	mm
Compression Ratio	10.01:1	N/A
Fuel	Gasoline AKI 87	
Injector Pressure	4.5	MPa
Injector	Bosch 7-hole GDI injector	
Piston	Flat top piston with sapphire insert	
Intake	Ford 2L Ecoboost intake manifold	
Exhaust	Custom exhaust pipe	

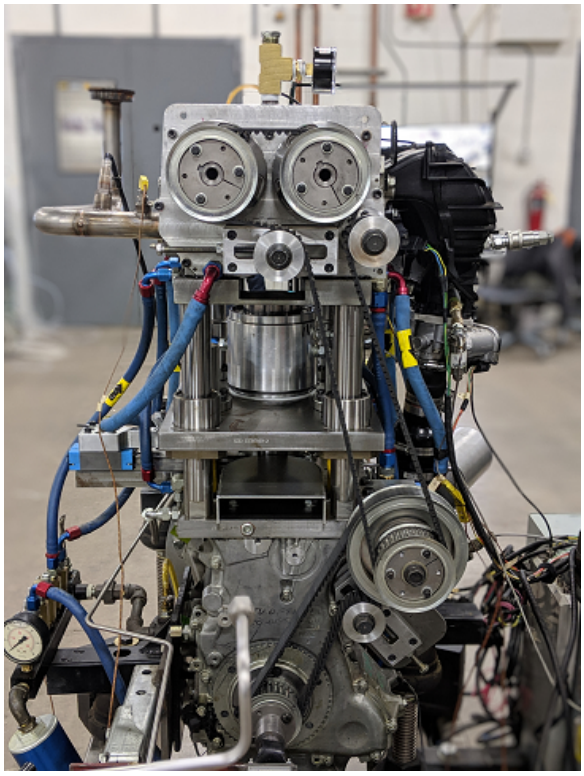


Figure 3.8: Optical Engine [2]

The optical engine was predominantly used for studying correlation of ions signal features with pressure metrics as it was instrumented with sensors at multiple positions unlike the metal engine. Figure 3.9 depicts the location of the four ion probes placed in the periphery of engine cylinder. The specifications of the ion probes are mentioned in Table 3.10.

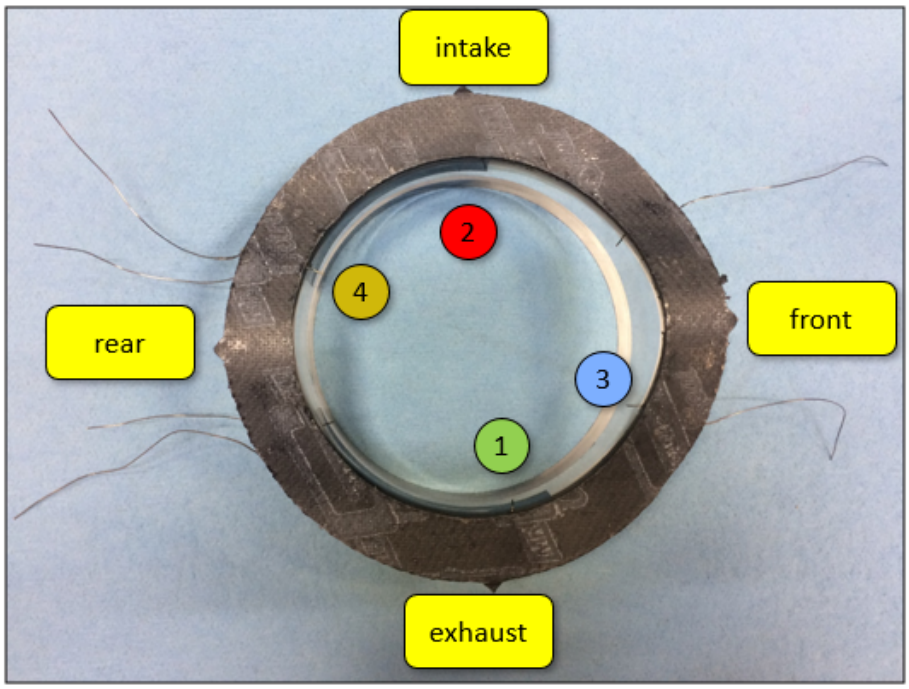


Figure 3.9: Ion probe location in optical engine [2]

Table 3.10
Specification of ion probe in optical engine

Electrode Properties	Values	Units
Material	304 Stainless Steel	N/A
Tip outer diameter	0.38	mm
Tip protrusion	5	mm
Conditioning box rating Probe 1	250	kΩ
Conditioning box rating Probe 2	250	kΩ
Conditioning box rating Probe 3	250	kΩ
Conditioning box rating Probe 4	1	MΩ

3.6 Accelerometer

The sensor suite installed onto the engine included accelerometers. Accelerometers are sensors capable of measuring acceleration and vibration. The engine was mounted with two triaxial accelerometers capable of simultaneous acceleration measurements along three orthogonal axes. The location of sensor mounting is shown in Figure 3.10 and brief specifications of the sensors are listed in Table 3.11. The accelerometer output was connected to a conditioning box and then onto the data acquisition system. Although the engine was instrumented with accelerometers, their use was not emphasized in this study. Studies related to these sensors were conducted in a

previous study by Prabhu [1].

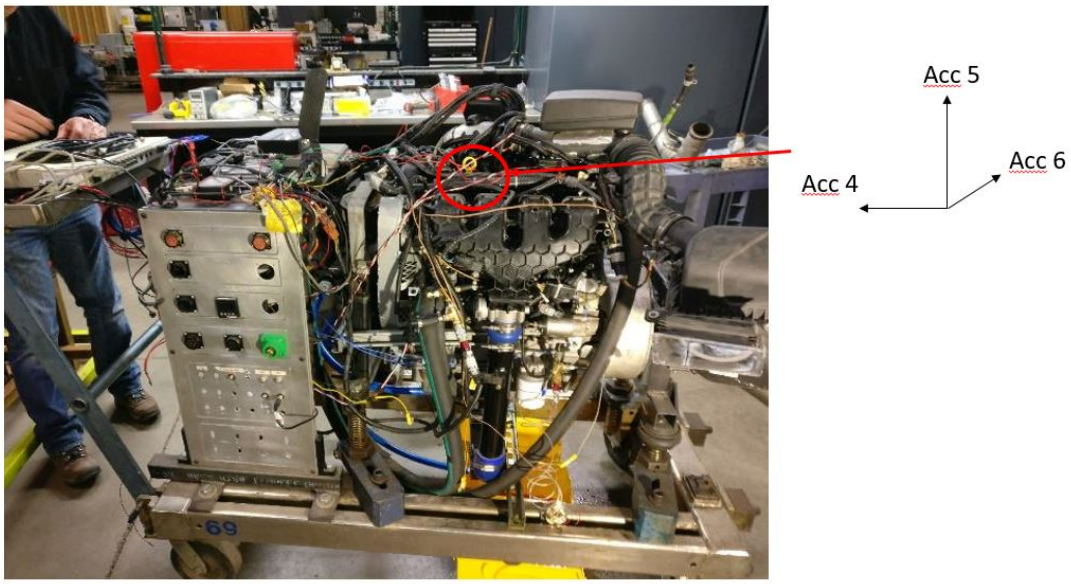


Figure 3.10: Final accelerometer mounting positions. Reprinted with permission from original author. Refer Appendix B

Table 3.11
PCB-356A03 Accelerometer specifications

Specification	Values (SI units)
Sensitivity (±20%)	1.02mV/(m/s^2)
Measurement range	±4905 m/s^2 pk
Frequency range (±5%) (y or z axis)	2 to 8000 Hz
Frequency range (±5%) (x axis)	2 to 5000 Hz
Resonant Frequency	≥50 kHz
Broadband resolution (1 to 10000 Hz)	0.03 m/s^2 rms
Non-Linearity	≤1%
Transverse sensitivity	≤5%
Temperature range (operating)	-54 to +121°C

3.7 Data Acquisition System

A data acquisition system (DAQ) is a device used to log (in digital format) the output signal of various sensors in a system. In this study, a Redline CAS system was used for data logging purposes. The DAQ was equipped with both real-time and analog-to-digital(A/D) modules. The specifications of the A/D modules are shown in Table 3.12. The DAQ was connected to the H25 crank encoder(3.4 to be used as a trigger to log data. Additionally a Redline SODEP encoder signal conditioner was used to

ensure high quality signal isolation, conditioning and transmission to the DAQ.

The DAQ chassis used was mounted with one 4344 real-time processor module and three 2840 analog modules thereby providing a total of 48 analog-digital channels. The 4344 module processes engine encoder information, stores digitized data and performs real-time calculations. The 2840 modules convert analog input to digital information which can then be used by the 4344 module for performing calculations and processing data. Further a combustion analysis software was used to post process data to obtain additional information. Depending on the settings established during data logging, results were obtained in time domain, angle domain or cyclic basis.

Table 3.12
Specification of analog input module

Paramter	**Value**
Number of channels	16
Max. sampling rate	1Msamples/sec
Full scale input (FS)	±10,±5,±2,±1
Bandwidth (3dB)	>1MHz@10V FS
Resolution	1 part in 4096 (12bits)
Accuracy	±0.05% of reading ± 1 LSB
Input impedence	1MΩ
Overvoltage protection	10VDC
Operating temperature	0-50degC

Chapter 4

Algorithm development

This study used multiple different sensors including ion sensor, exhaust pressure sensors and optical crank encoder. Each of these sensors offered varied information regarding engine combustion metrics including IMEP and CA50. A number of these sensors also offered diagnostic capabilities viz. knock detection, misfire detection etc. However in order to extract this information from the sensors, various different analysis techniques were required. This chapter presents details on the various techniques/algorithms developed and used for feature extraction, correlation studies and neural network development

4.1 Correlation Studies

Amongst the first sensors studied as part of this body of work, is the ion sensor (Section 3.3). A typical ion signal from the standalone ion probe is shown in Figure 4.1. As can be seen, the signal has multiple features that can be studied for possible correlation with combustion metrics. Table 4.1 lists the various features analyzed in this work. Once the features were extracted a Pearson correlation was used to determine which ion features best correlate with metrics including IMEP, in-cylinder peak pressure and peak pressure location.

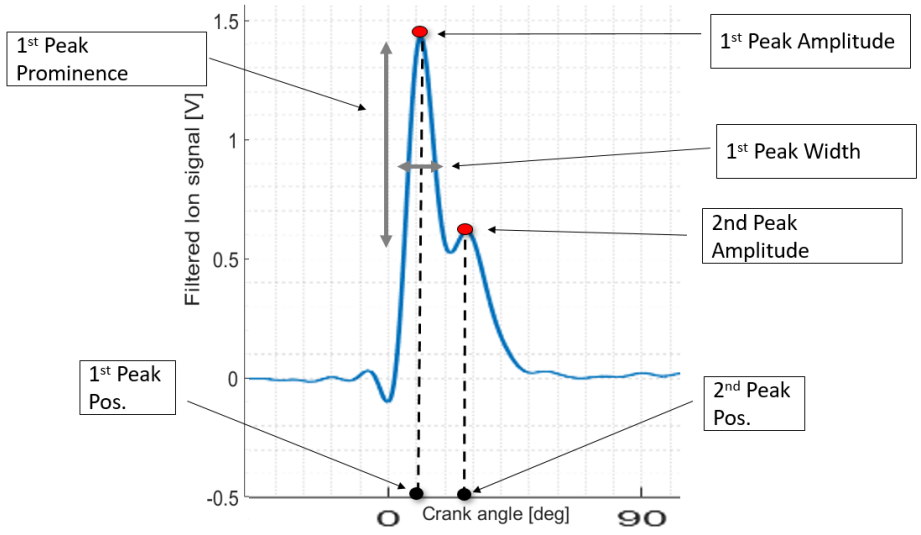


Figure 4.1: Representative standalone ion sensor signal at load of 750kPa and speed of 1500RPM

A Pearson correlation coefficient (Eqn: 4.1), developed by Karl Pearson, is a means

to find the linear correlation between two random variables X and Y. The value for the coefficient ranges between -1 and 1, where a value of -1 indicates total linear negative correlation. A value of 1 indicates total positive correlation and zero indicates no correlation.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{4.1}$$

where cov(X,Y) is the covariance, σ_X and σ_Y are the standard deviation in variable X and variable Y

Table 4.1
List of Ion features analyzed in correlation studies

	Feature
Ion metrics	Area under ion curve
First peak metrics	Amplitude of first peak
	Position of first peak
	Half width of first peak
	Prominence of first peak
Second peak metrics	Amplitude of second peak
	Position of second peak
	Half width of second peak
	Prominence of second peak

The same method as mentioned for ion signal was used at multiple points in this study including the analysis of correlation between features in the exhaust pressure signal and IMEP as well as CA50. Figure4.2 shows a representative image of in-cylinder pressure and the corresponding exhaust pressure for the same cycle. The prominent features studied are also highlighted including Pressure maxima/minima, area beneath the curve and pressure at exhaust valve opening.

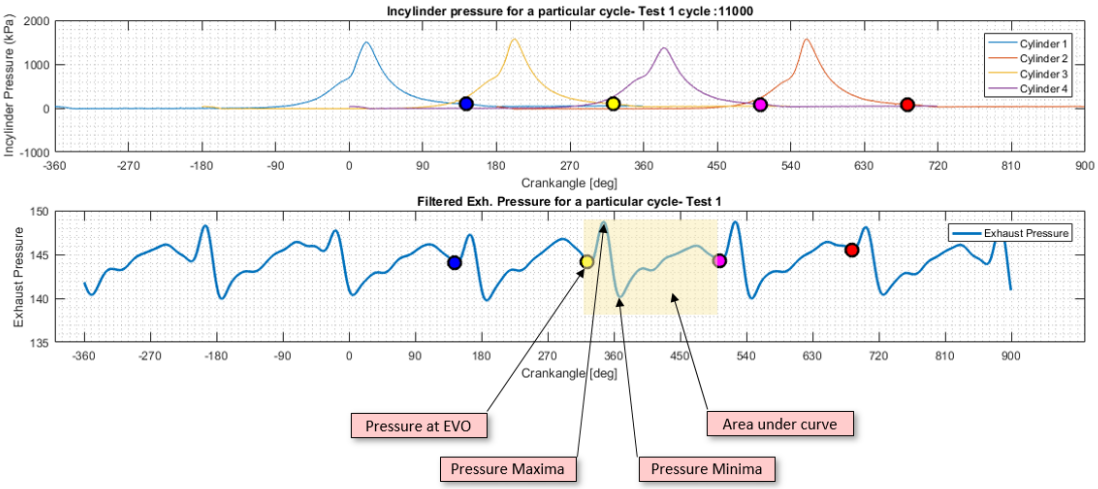


Figure 4.2: Representative Exhaust pressure signal

4.2 Order Tracking

A prominent technique used for feature extraction throughout this study is order tracking. Order tracking is a technique in signal processing that is used to analyze signals with time varying frequency, specifically when the frequency is proportional to the speed of rotation of a primary shaft or machine. This technique finds prominence in analysis of rotary equipment including motors, engines, bearings etc.

The generic representation for a time vary frequency signal is given by Equation 4.2 [20]

$$X(t) = A(k,t)sin(2\pi(\frac{k}{p})t + \phi_k) \tag{4.2}$$

where;

k is the order being tracked

A(k,t) is amplitude of order k as a function of time

t is time

p is the period of primary order in seconds

In this study the crankshaft is the primary rotary shaft of reference. Also since most signals logged by the combustion analysis software are relative to crankshaft position, the signals are logged in angle domain. Thus the order analysis was conducted in the angle or order domain and not time domain as given in Equation 4.2. Thus by conducting a Fourier transform on the data it is possible to extract the orders and corresponding phase and amplitude independent of engine speed. A list of steps involved in order extraction is given below:

- If signal is in time domain convert it to angle domain using an RPM (Tachometer) signal
- Filter the signal to remove any high frequency noise if need be

- Create signal blocks of appropriate length to obtain desired order resolution
- Apply window and conduct FFT on the signal
- Visually verify the computation using order colormap or waterfall plot
- Extract amplitude and phase information of desired orders

Figures 4.3 and 4.4 show a representative images of an ordermap for crank data and a waterfall plot for orders in exhaust data respectively. Here crank data refers to the speed signal derived from the output of the crank position sensor. The plots are for transient tests conducted as part of the study. Further, event index refers to a firing event or cycle count. The graphs are also color coded by the amplitude of the orders as shown in the scales beside the plots.

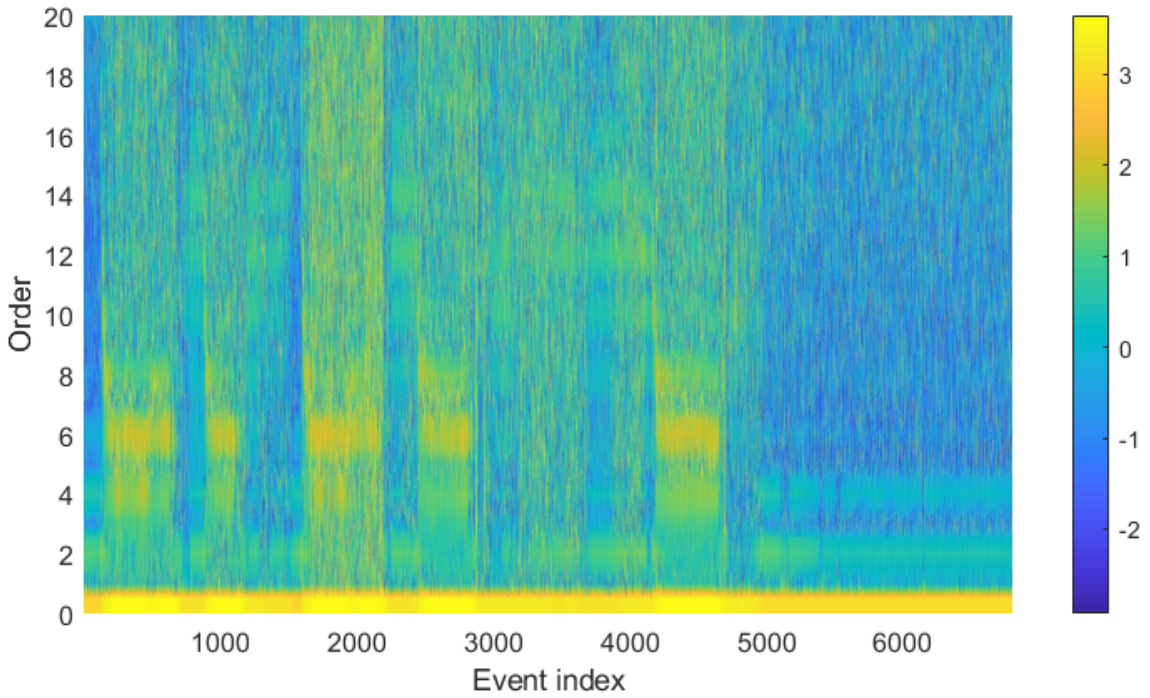


Figure 4.3: Representative order-map of crank signal

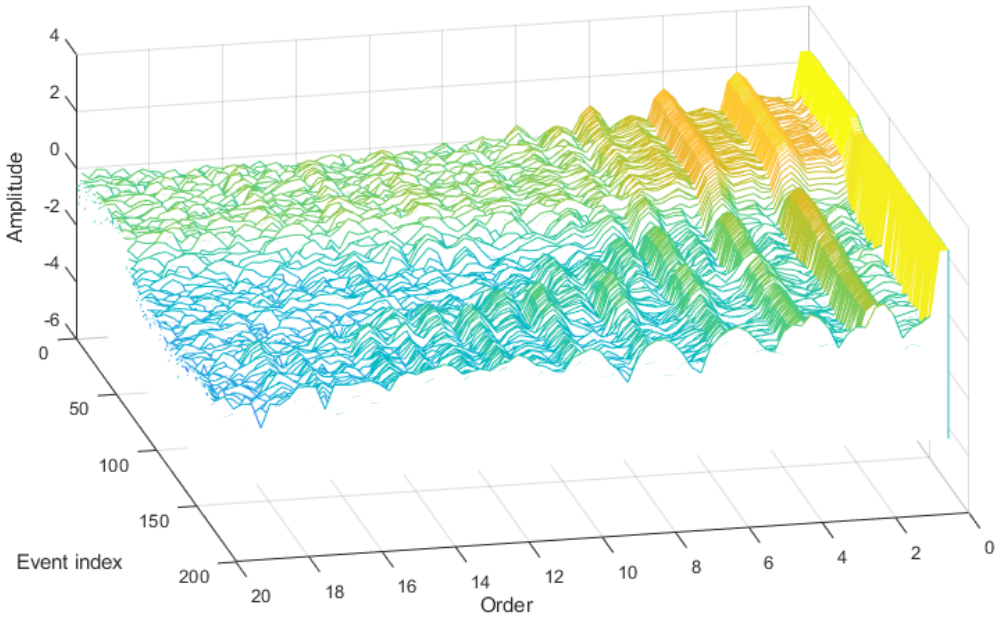


Figure 4.4: Representative waterfall plot of kulite exhaust pressure signal

4.3 Knock Integral Calculator

Knocking is an engine phenomenon, wherein the unburnt gaseous mixture outside/beyond the normal flame front undergoes rapid combustion causing high frequency pressure oscillations to be setup in the engine cylinder [21]. Knocking is a form of abnormal combustion that can be caused due to a number of reasons, one of which is improper spark timing. For certain operating conditions, the likelihood of knock increases with advancing the spark as this leads to combustion being initiated sooner than optimal timing which inturn causes pressure and temperature conditions to be

conducive for mixture beyond the normal flame front to be ignited and thus cause engine knock.

From a measurement standpoint it is essential to detect and quantify knock as high engine knock could cause component damage or even engine failure in severe cases. This study thus looks at means of quantifying engine knock not just using the ICPS but also using the ion signal. Knock detection was also evaluated using accelerometer by Prabhu [1]. Figure 4.5 shows a representative image of an engine cycle in which knock was occurring.

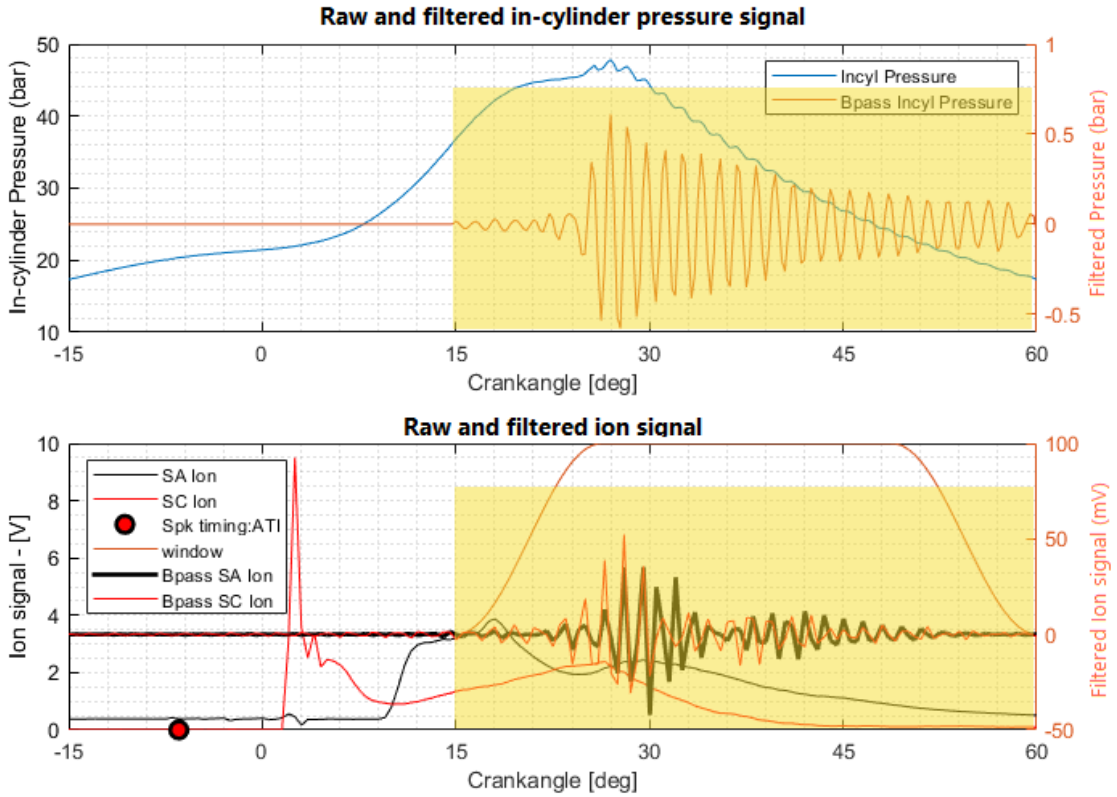


Figure 4.5: Representative figure of knock integral calculation (Speed : 1500RPM, IMEP : 11bar)

Note in the Figure 4.5 'Bpass' is an acronym for Bandpassed; also SC refers to spark coil ion signal and SA refers to standalone ion signal. The yellow highlighted region signifies the knock window (15 dATDC to 60 dATDC) considered for the specific test. The figure shows both in-cylinder pressure signal as well as the ion signal from both the coil integrated ion (SC) and the standalone ion probe (SA). The method involved in calculating the knock integral are similar to that presented by Naber et al. [22] . The specified signal of interest was bandpass filtered before calculating the knock integral as given in equation 4.3

$$S(N) = \frac{1}{t_2 - t_1} \int_{t_1=t(\theta_1)}^{t_2=t(\theta_2)} |s_f(t)|dt = \frac{1}{n} \sum |s_f(i)| \tag{4.3}$$

where

s is the signal of interest (Pressure or Ion in this case)

n is the number of sample or datapoints

θ_1 is the start of the knock window in crankangle degrees

θ_2 is the end of the knock window in crankangle degrees

In Equation4.3 's' could be any signal. The equation is developed assuming the signal is sampled in time domain. However it is applicable to angle domain sampled signals as well with minimal modifications.

4.4 Misfire Generator

A portion of this study involved in-vehicle transient testing which was conducted at Ford's high speed test track facility in Dearborn, MI. The primary focus of the vehicle testing was to evaluate the performance of the exhaust sensor in transient conditions. A portion of the study also involved studying misfires during transient driving conditions. In order to generate these misfires a custom Ford Misfire Generator software was used. Figure 4.6 shows a portion of the user interface as observed in ATI Vision. ATI Vision is an integrated calibration and data acquisition tool used to collects signals from the ECU and other sources.

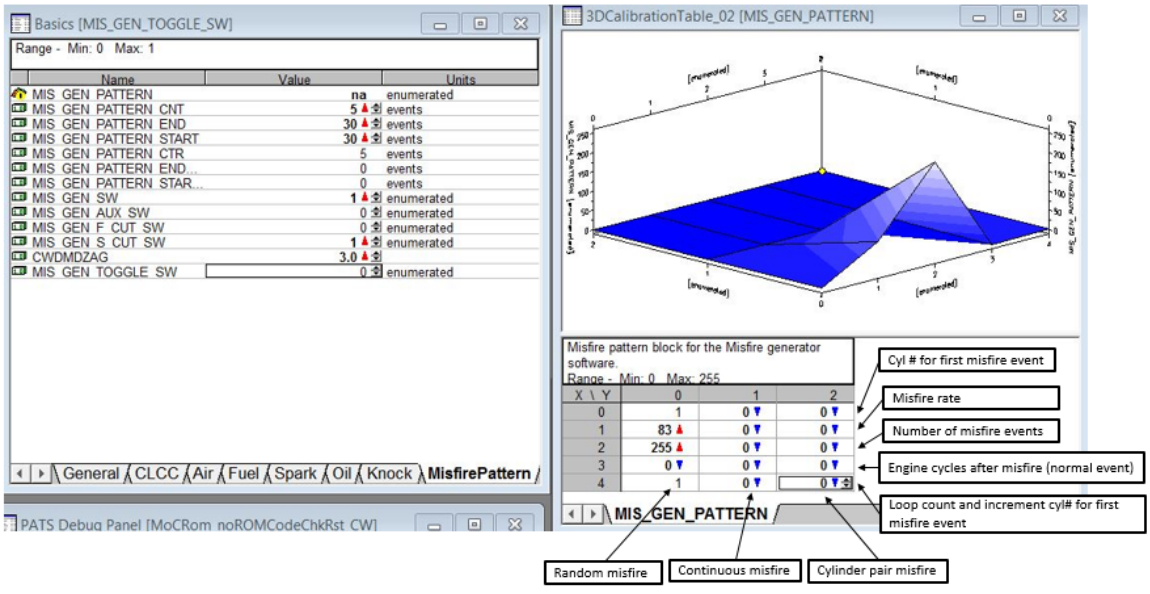


Figure 4.6: User-interface of Ford's Misfire Generator Software [1] . Reprinted with permission from original author. Refer Appendix B

The same software was also used for misfire generation in the steady state dynamometer tests conducted at the Michigan Tech. As shown in Figure 4.6 the software has multiple misfire patterns that can be generated. For example, depending on the parameter setting, it can create misfires at predefined intervals in a particular cylinder or create a walking pattern where a misfire is generated after a predefined number of firing events i.e. independent of cylinder. The software can create a random misfire pattern but this feature is not utilized for ease of analysis. The software is activated by using the toggle switch parameter at any point while conducting the test. It is however recommend to not have multiple misfire events occur continuously as this could damage the catalytic converter.

4.5 Neural Networks

Artificial Neural Networks (ANN) are a method of machine learning that was developed to mimic the neurons in the brain [23] . Typical uses of neural networks include linear and non-linear regression as well as logistical regression. A simple neural network like that shown in Figure 4.7 consists of three segments or layers, an input layer, an output layer and a hidden layer. The hidden layer is called so because the values in this layer are typically not seen or displayed.

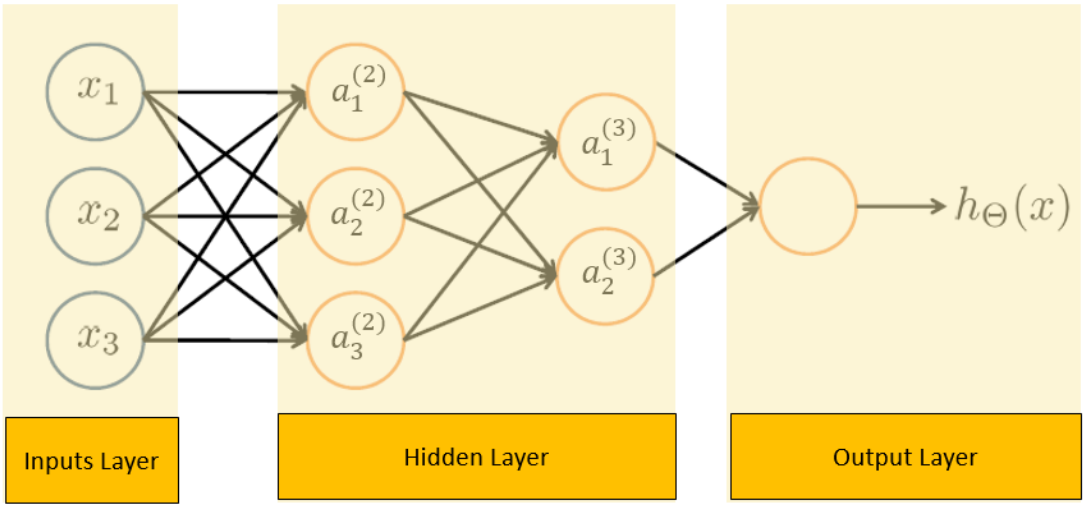


Figure 4.7: A simple neural network

Any neural network is composed of neurons or units also called activation units. The activation function of these units is typically a sigmoid function (shown in equation 4.4. Wherein 'z' is an input and the output g(z) ranges between -1 and 1.

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.4}$$

The system of equations (Eqn 4.5) , adapted from the work of Andrew [23], offer a mathematical representation of the simple network shown in Figure 4.7. Further, $a_i^{(j)}$ is the activation of unit i in layer j and $\Theta^{(j)}$ is the matrix of weights controlling the function mapping layer j to layer j+1.

$$
\begin{aligned}
a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3), \\
a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3), \\
a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3), \\
a_1^{(3)} &= g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}), \\
a_2^{(3)} &= g(\Theta_{20}^{(2)} a_0^{(2)} + \Theta_{21}^{(2)} a_1^{(2)} + \Theta_{22}^{(2)} a_2^{(2)} + \Theta_{23}^{(2)} a_3^{(2)}), \\
h_\theta(x) &= a_1^{(4)}, \\
a_1^{(4)} &= g(\Theta_{10}^{(3)} a_0^{(3)} + \Theta_{11}^{(3)} a_1^{(3)} + \Theta_{12}^{(3)} a_2^{(3)}).
\end{aligned}
\tag{4.5}
$$

There are many different types of neural networks such as the feed forward network, radial basis network, support vector machine, recursive network etc. The networks are typically distinguished based on the flow of information from one unit (or layer) to another. This study primarily utilized a feed forward network and a recursive network. Figures 4.8 and 4.9 show the architecture of such a network. The primary difference between the two networks is that the recursive network uses previous output/estimates (i-1) to generate an estimate for a given cycle (i). Further, 'W' and 'b' refers to the weight matrix and bias vector. Each layer has a set of weights assigned to the nodes as well as a bias i.e a vector of ones applied for computational purposes. Chapter 5 delves into greater details on the performance obtained using the different networks.

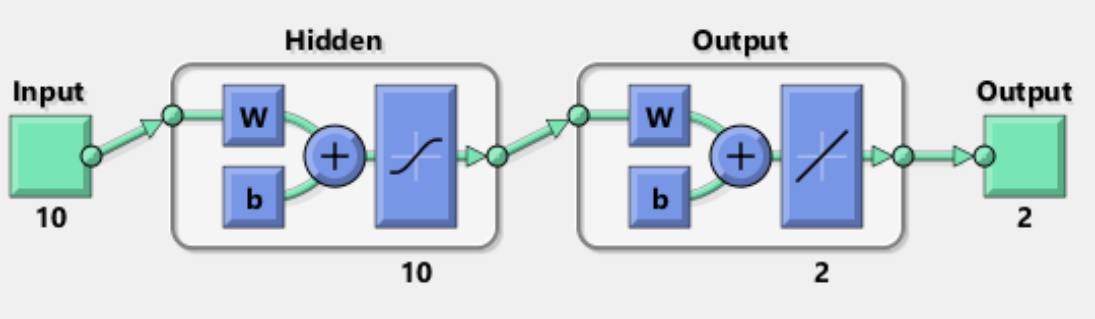


Figure 4.8: Feed forward network

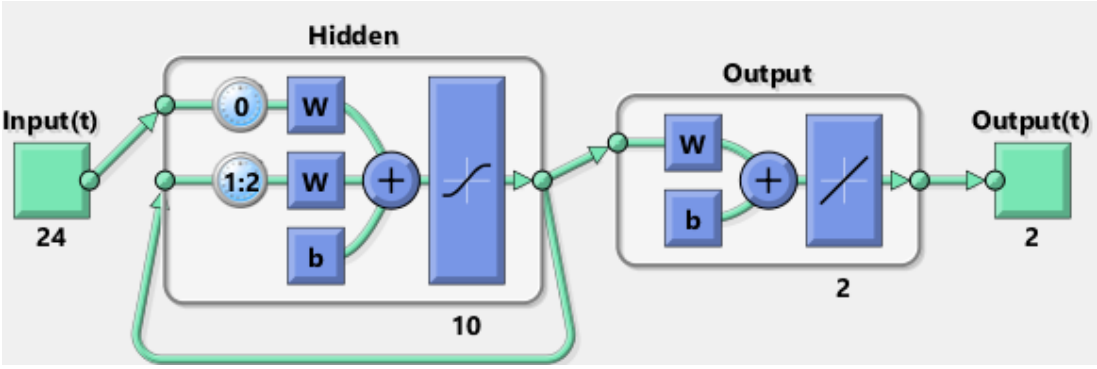


Figure 4.9: Recursive neural network

4.5.1 Feature Scaling

A neural network often uses multiple inputs; as such the inputs can each have varied ranges. For example, in a transient engine study, if a parameter like peak cylinder pressure is estimated using a neural network whose inputs are engine speed, spark timing and manifold air pressure (MAP). The range or scale of each of the three inputs is different i.e. engine speed could vary from 0-5000 rpm, spark could vary

from -30 dATDC to 10 dATDC and MAP from 0-1.5 bar. This being the case, it is often beneficial to alter the inputs (also called as features) such that they are all of a similar scale. This process of scaling the inputs is called feature scaling and is analogous to normalization. Any input X_i can be scaled using equation 4.6

$$X_i = \frac{X_i - \mu_i}{\sigma_i} \tag{4.6}$$

where μ_i and σ_i is the mean and variance of the feature

4.5.2 Random Initialization

Previously it was mentioned that $\Theta^{(j)}$ is a matrix of weights that controls the mapping of a function from one layer to the next. When training a network these weights are progressively altered to find an optimal weighting factor, this is done using various algorithms like gradient descent and advanced optimization algorithms.

In order for these optimization algorithms to function, an initial value is to be assigned to these weights. If the same value is assigned to all the wights then all the units would compute the same feature or function of the inputs i.e. the function for $a_1^{(2)}$ would be the same as that for $a_2^{(2)}$ and so on. This would make the system redundant or more technically, lead to the problem of symmetric weights.

Random initialization helps eliminate the problem of symmetric weights and carries out symmetry breaking by assigning the weights of the parameters in a random fashion, but within a specified bound (ϵ) as shown in Equation 4.7.

$$-\epsilon \leq \Theta_{ij}^{(l)} \leq \epsilon \tag{4.7}$$

4.5.3 Principal Component Analysis

Neural networks quite often utilize multiple inputs, making for a high dimensionality input data space. However with data sets that have high dimensionality there is always a possibility of cross correlation between the various inputs, thereby making them redundant. Principal component analysis (PCA) is method to transform a dataset with multiple correlated variables into a set of linearly uncorrelated variables known as principal components [3]. The principal components are arranged in descending magnitude of variance. Thus using PCA helps reduces the demensionality of the data while also conserving the most relevant information from the original dataset. This study as well, utilized PCA initially to reduce the demensionality of the network used for combustion metric estimation.

Implementation of PCA is a two step process i.e

- Compute the covariance matrix Σ

- Compute the eigen-vectors of the covariance matrix using singular value decomposition

Considering an input matrix or data space with 'n' features and 'm' samples, the covariance matrix can be calculated as given in eqn. 4.8

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T \tag{4.8}$$

where $x^{(i)}$ is a n$\times$ 1 vector

Further any vector A can be represented as

$$A_{(m\times n)} = U_{(m\times r)}S_{(r\times r)}(V_{(n\times r)})^T \tag{4.9}$$

The matrices U, S and V in Equation 4.9 are the left singular vector, singular values and right singular vector respectively. These matrices can also be found using singular value decomposition of the covariance matrix mentioned in Equation 4.8. Lastly to reduce a data matrix with 'n' features or dimensions to 'k' features the first 'k' columns of the left singular matrix is convolved with the input matrix (Eqn. 4.10)

$$z_{(m\times k)} = x_{(m\times n)}U_{(n\times k)} \tag{4.10}$$

Chapter 5

Results and Discussion

The algorithms and methodologies used for processing the data of the various sensors were discussed in the previous chapter. This chapter presents an analysis of the results and discusses the performance of the sensor suite across the various tests conducted C. This chapter is divided into four sections. Section 5.1 talks about the analysis conducted using the ion sensor. Results obtained both on the regular and optical engine are discussed. Results pertaining to the exhaust sensor are discussed in Section 5.2. This section primarily talks about sensor performance during transient (i.e in-vehicle) test conditions. Section 5.3 presents select results pertaining to the use of the crank angle encoder. Lastly, Section 5.4 discusses the process of using the various sensor outputs to develop a neural network for predicting combustion metrics including IMEP and CA50.

5.1 Ion sensor

This study used two variants of ion sensors, the standalone ion sensor and the spark coil integrated ion sensor discussed in chapter 3. The standalone ion sensors were custom built prototypes and had undergone several design iterations over the course of the study. Suitable mentions of the variants are made throughout this section. Additionally, during the initial phase of the study only the standalone ion senor was used, the coil ion probe was a later addition.

5.1.1 Correlation of ion features with pressure metrics

The first task in utilizing a particular sensor for control and diagnostic purposes is to understand if there are signal artifacts within the signal that correlate with the parameter needed to be sensed or controlled. Thus a range of steady state tests were conducted to evaluate the correlation of various ion signal artifacts with in-cylinder pressure metrics including the pressure peak location and amplitude. A list of the acronyms for the features used in ion studies is listed in Table 5.1 and shown in Figure 5.1

Table 5.1
Ion correlation studies : Feature acronyms

Acronym	**Feature**
CA I Pk1	Location of first ion peak
CA P Pk	Location of pressure peak
P Pk Amp	Amplitude of pressure peak
I Pk2 Amp	Amplitude of second ion peak
I Area	Area under ion curve
I Pk1 width	Half width of first ion peak

5.1.1.1 Ion Correlation studies - Metal engine

The various ion artifacts investigated were discussed in Chapter 4 Section 4.1. Figure 5.1 offers a depiction of the same. Table 5.2 lists the steady state tests conducted to evaluate correlation of ion signal artifacts with pressure metrics. The alumina based standalone ion probe was used during this phase of the study. The raw ion signal was connected to a 1MΩ conditioning box developed by Ford Motor Co and mentioned in Section 3.3 Table 3.6. The signal was sampled at intervals of 0.5CAD. Additionally, during post processing, a fourth order (one-way) Butterworth filter with cutoff frequency at 25 orders was used. Two way filtering was done to avoid phase delay.

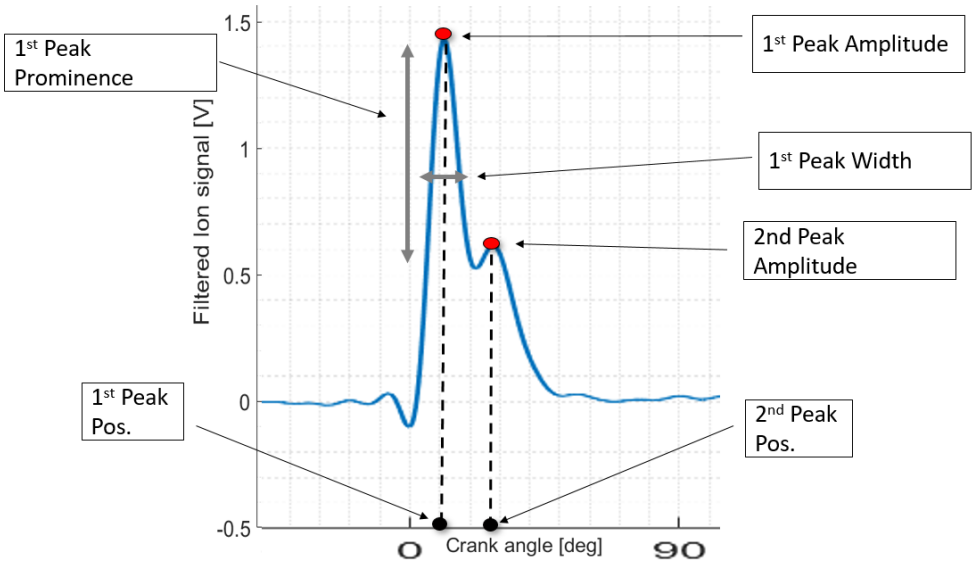


Figure 5.1: Ion sensor signal artifacts studied

Table 5.2
Test matrix for ion sensor evaluation - Cylinder 2

Test	**Speed**	**IMEP**	**Intake advance**	**Exhaust retard**	**CA50**	**COV**
	RPM	KPa	deg	deg	deg	%
Test 1	1500	250	0	0	8	0.74
Test 2	1500	750	0	0	8	0.59
Test 3	3500	250	0	0	8	0.68
Test 4	3500	750	0	0	8	0.42

Figure 5.2 shows the in-cylinder pressure and standalone ion probe signal for a typical

combustion cycle. The spark timing is also marked on the ion signal. The slight oscillation in ion signal after the spark but before the TDC is an artifact introduced due to the filtering process.

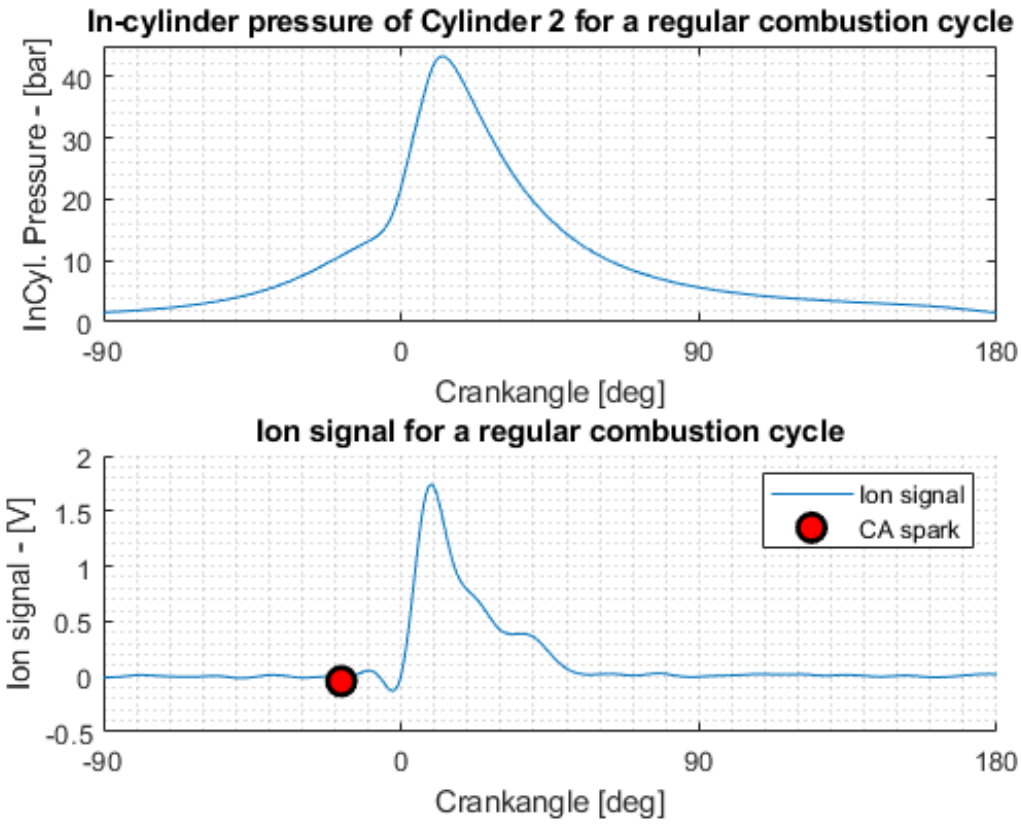


Figure 5.2: Filtered ion signal for a normal combustion cycle (Speed : 1500RPM , IMEP : 750kPa)

Upon extracting the features of interest, a correlation study was conducted between the various features extracted and in-cylinder pressure metrics. Figure 5.3 shows the correlation (Pearson Correlation Coefficient - refer Section 4.1) of first ion peak

artifacts with pressure metrics for Test 1. The dataset was composed of 300 cycles of data with the engine operating at a speed of 1500 RPM and IMEP of 250kPa. The ion peak was extracted for the ion signal corresponding to each cycle and evaluated for correlation with the pressure metrics listed. Figure 5.3 lists ion metrics on the x-axis and pressure metrics on the y-axis. Table 5.1 lists the expansion for the acronyms used in the figure. It was seen that for Test 1 (Figure 5.3), the correlation of location of first ion peak (CA I Pk1) with pressure peak amplitude (P Pk Amp) and locations (CA P Pk) was greater than 0.6. However, the other ion features including area under ion curve (I Area), ion first peak amplitude (I Pk1 Amp) etc. showed poor correlation with pressure metrics including IMEP, pressure peak amplitude and location.

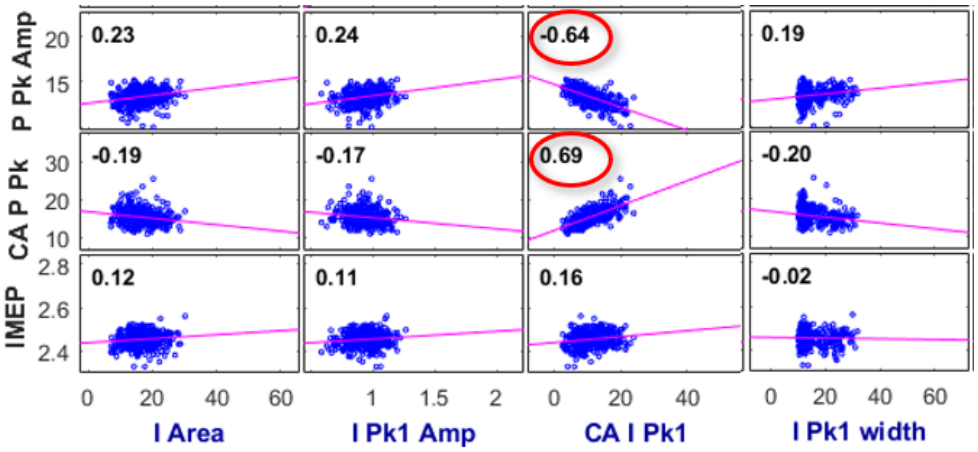


Figure 5.3: Correlation of Ion sensor signal artifacts with pressure metrics - Test 1 : 1500RPM, 250kPa

Conducting the same analysis on the dataset of Test 2 (Figure 5.4), it was seen that the location of first ion peak showed highest correlation with pressure peak amplitude

and location. However the magnitude of the correlation coefficients was lower than at low load conditions.

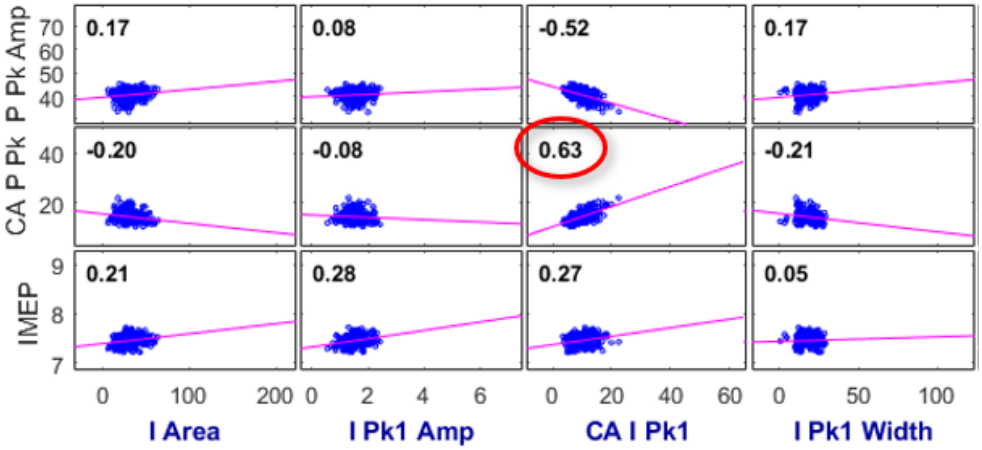


Figure 5.4: Correlation of Ion sensor signal artifacts with pressure metrics - Test 2 : 1500RPM, 750kPa

Similar analysis was conducted across the various datasets mentioned in Table 5.2 and the results are tabulated in Table 5.3. A correlation coefficient of above 0.6 was considered to be satisfactory to use a particular artifact for pressure metric estimation and other diagnostic applications based on prior work by Naber et al. [24].

Thus from Table 5.3 it can be seen that the ion signal offers a good estimate of peak pressure location. This is evidenced by the fact that the correlation of ion peak location with pressure peak location is greater than the previously mentioned threshold of 0.6. However, the first ion signal peak does not offer consistently good correlation ($R \not> 0.6$) with amplitude of pressure peak.

Table 5.3
Results of correlation studies of ion sensor

Correlation		**Test 1**	**Test 2**	**Test 3**	**Test 4**
First peak	CA I Pk1 - P Pk Amp	-0.64	-0.52	-0.60	-0.52
& Pressure	CA I Pk1 - CA P Pk	0.69	0.63	0.64	0.62
Between Ion	I Pk2 Amp - I Area	0.71	0.65	0.59	0.63
Metrics	I Pk2 Amp - I Pk1 width	0.73	NA	NA	0.70
	No. of 2^{nd} peaks	210	127	233	158
First peak &	CA I Pk1 - P Pk Amp	-0.64	-0.66	-0.60	-0.58
Pressure *	CA I Pk1 - CA P Pk	0.69	0.74	0.65	0.66

* Correlation in cycles with 2 peaks N/A - Not calculated

Further, in cycles with two ion peaks, the first ion peak showed a correlation of greater than 0.65 with location of peak pressure. The amplitude of second ion peak also correlated well (R >0.6) with area under ion curve.

5.1.1.2 Optical engine

The ion studies on the metal engine showed that there are correlations between the ion signal characteristics and pressure metrics. The metal engine however has certain restrictions on the spatial position of the ion sensor, thereby making it hard to study the influence of sensor location on correlation studies. The optical engine however,

was instrumented with four ion sensors in Figure 3.9. Thus the optical engine could help offer a visualization of the combustion process and also showcase the influence of sensor position on correlation studies. Table 5.4 lists a subset of tests conducted on the optical engine to understand the influence and signal quality of ion sensors placed on the cylinder periphery. It is to be noted that all the tests are conducted at low speed-load conditions to prevent damage to the optical engine.

Table 5.4
Test matrix for ion sensor evaluation in optical engine

Test	**Speed**	**IMEP**	**CA50**	**Spark Adv.**	**Start of inj.**	**Tumble**
	RPM	KPa	dATDC	deg	dBTDC	
Test 1	1000	236	4.2	18	360	OFF
Test 2	1000	257	12.3	12	360	OFF
Test 3	1000	253	8.1	12	360	ON

Figure 5.5 shows the output of the ion sensors for a normal combustion cycle. It can be observed that based on the location of the sensor, the waveform intensity and shape changes. The intensity of each signal rises at a different crank-angle, indicative of the fact that the flame speed and in-cylinder combustion dynamics play a role in determining the signal amplitude. Figure 5.6 shows the ion signal for a misfire cycle in the same dataset.

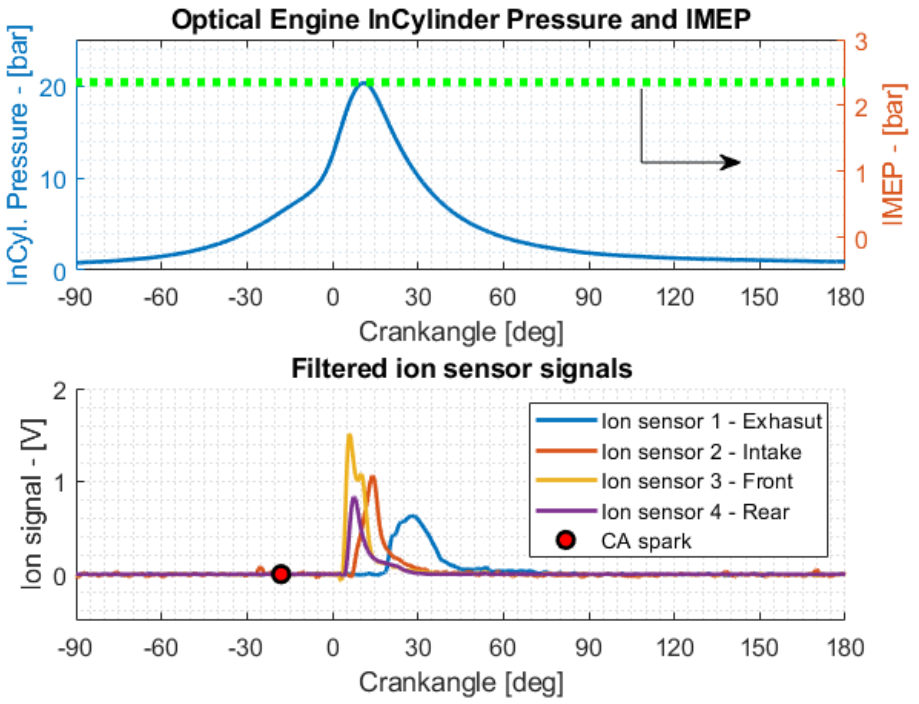


Figure 5.5: Output of ion probes on optical engine - normal combustion cycle - Test 1

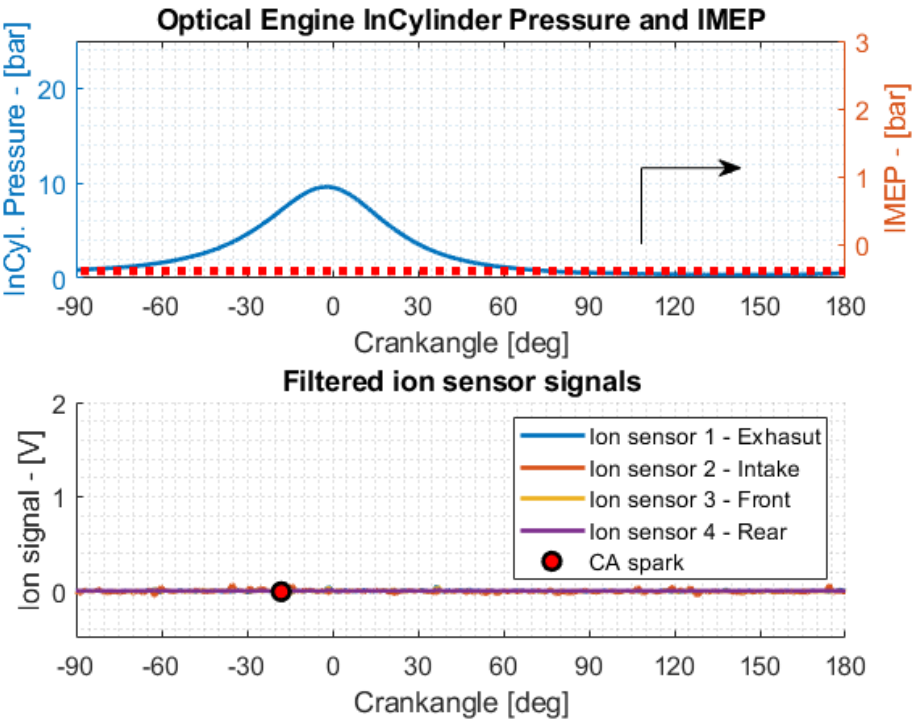
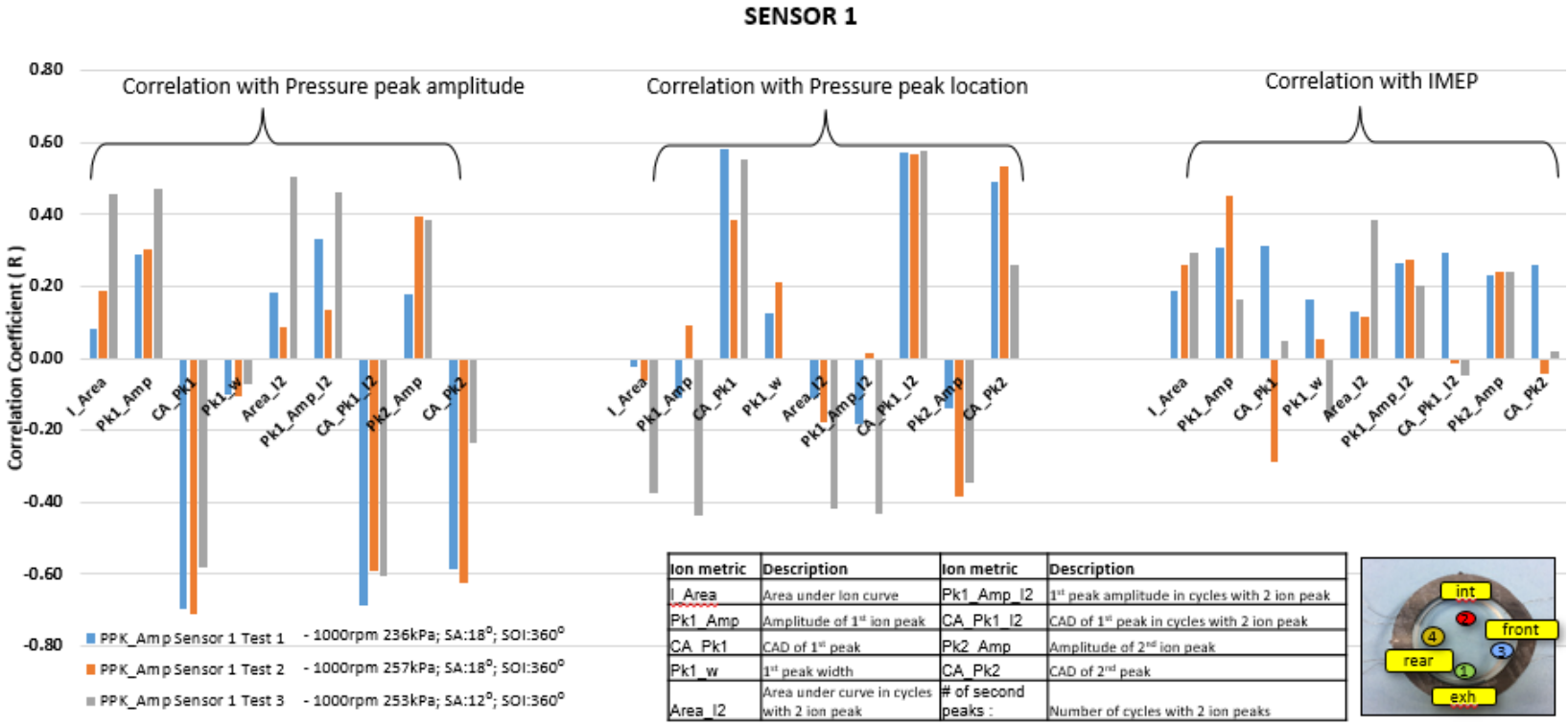


Figure 5.6: Output of ion probes on optical engine - misfire cycle - Test 1

In Figure 5.6 can be seen that the absence of combustion inhibits ion formation, thereby causing the ion signal to be negligibly small. The lack of combustion also results in low/negative IMEP.

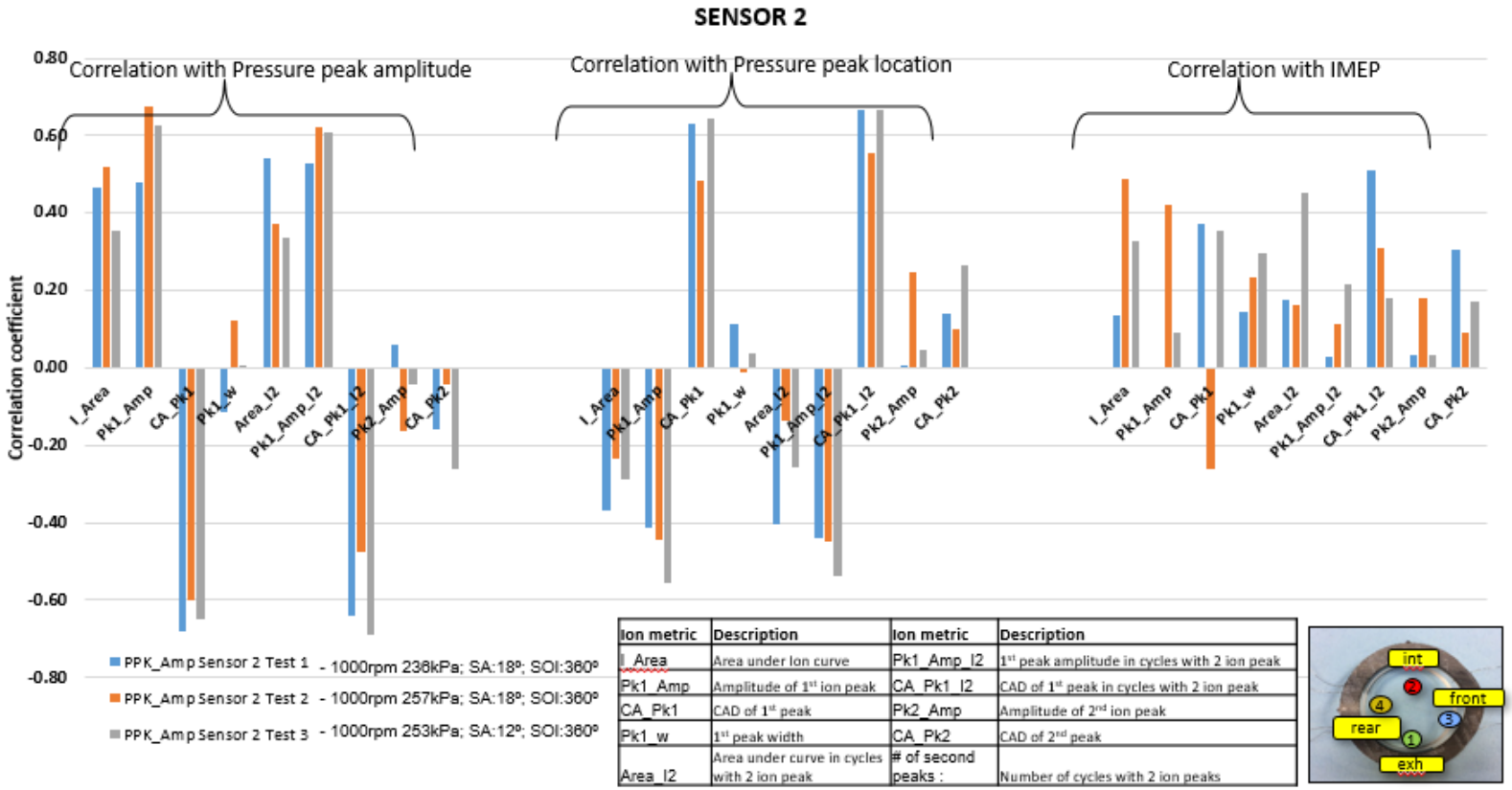
Going beyond merely just visualizing the ion signals, correlation studies conducted using the output of the four ion sensors, offered some interesting results. Figure 5.7 shows the correlation of the various ion sensor artifacts with the pressure metrics. The figure showcases the correlation study results across the various tests conducted as well. For sensor 1 it was seen that none of the features showed consistently good correlation (i.e $R \geq 0.6$) for all the tests conducted. Crankangle location of the first peak of the ion signal showed the strongest correlation with both amplitude and location of the pressure peaks. However low correlation was observed with IMEP. Figures 5.8, 5.9 and 5.10 offer a similar depiction but for ion sensors 2, 3 and 4 respectively. Observing the plots it can be seen that the trends are largely the same but the magnitudes are different based on the sensor location. This goes to show that the location of the ion sensor plays a vital role in combustion sensing.

Thus if the ion sensor is to be used for control and diagnostic applications, location of the sensor would have to be a key consideration. Further, for the test conditions studies, ion sensors 3 and 4 showed the strongest correlation with pressure peak location and amplitude as seen from Figures 5.7 - 5.10. The results were also consistent across the datasets. The results are also similar to that obtained on the metal engine.



Ion metric	Description	Ion metric	Description
I_Area	Area under Ion curve	Pk1_Amp_I2	1st peak amplitude in cycles with 2 ion peak
Pk1_Amp	Amplitude of 1st ion peak	CA_Pk1_I2	CAD of 1st peak in cycles with 2 ion peak
CA_Pk1	CAD of 1st peak	Pk2_Amp	Amplitude of 2nd ion peak
Pk1_w	1st peak width	CA_Pk2	CAD of 2nd peak
Area_I2	Area under curve in cycles with 2 ion peak	# of second peaks :	Number of cycles with 2 ion peaks

Figure 5.7: Correlation of ion sensor 1 with pressure metrics, across all tests



Ion metric	Description	Ion metric	Description
I_Area	Area under Ion curve	Pk1_Amp_I2	1st peak amplitude in cycles with 2 ion peak
Pk1_Amp	Amplitude of 1st ion peak	CA_Pk1_I2	CAD of 1st peak in cycles with 2 ion peak
CA_Pk1	CAD of 1st peak	Pk2_Amp	Amplitude of 2nd ion peak
Pk1_w	1st peak width	CA_Pk2	CAD of 2nd peak
Area_I2	Area under curve in cycles with 2 ion peak	# of second peaks :	Number of cycles with 2 ion peaks

Figure 5.8: Correlation of ion sensor 2 with pressure metrics, across all tests

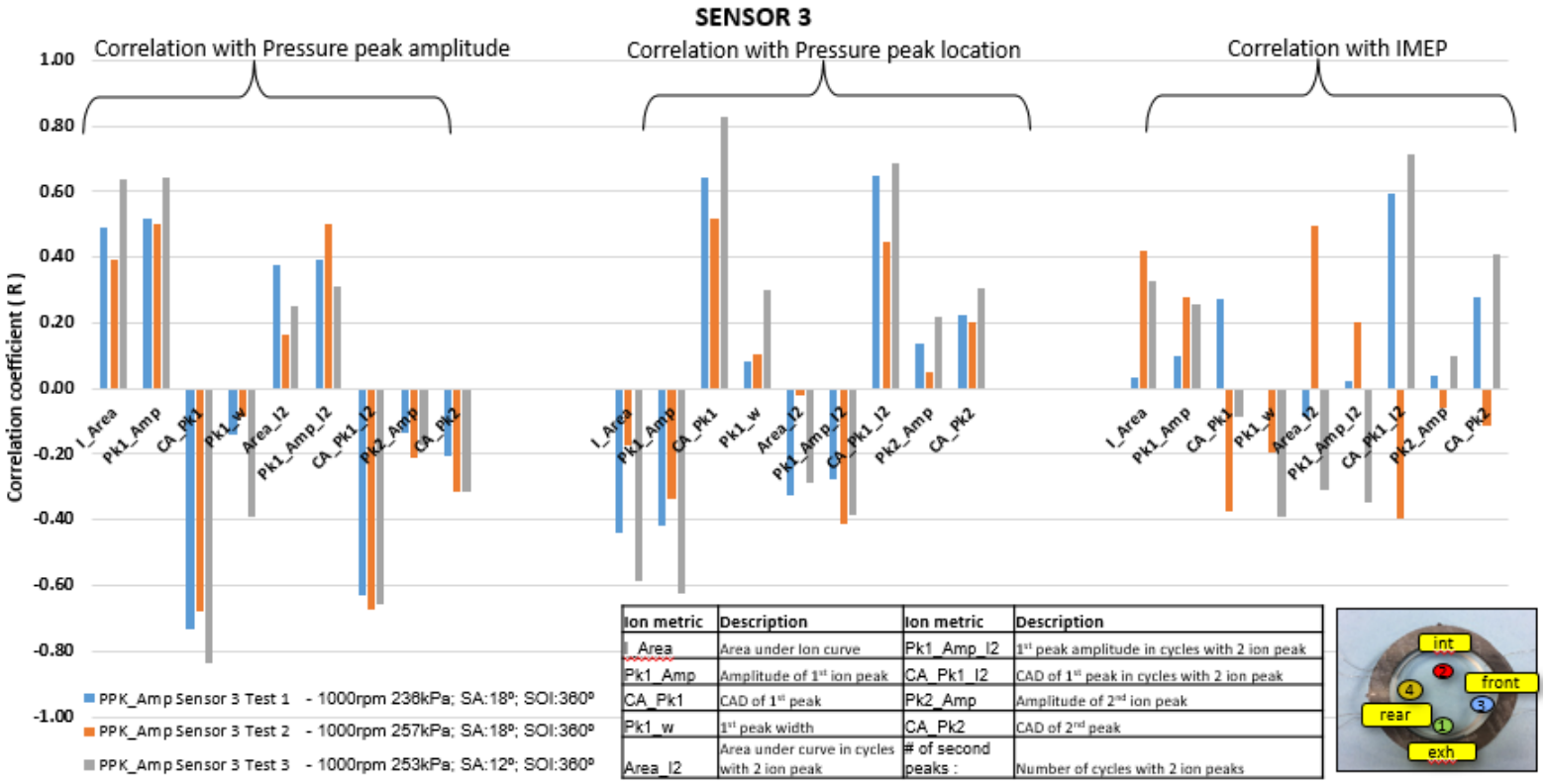
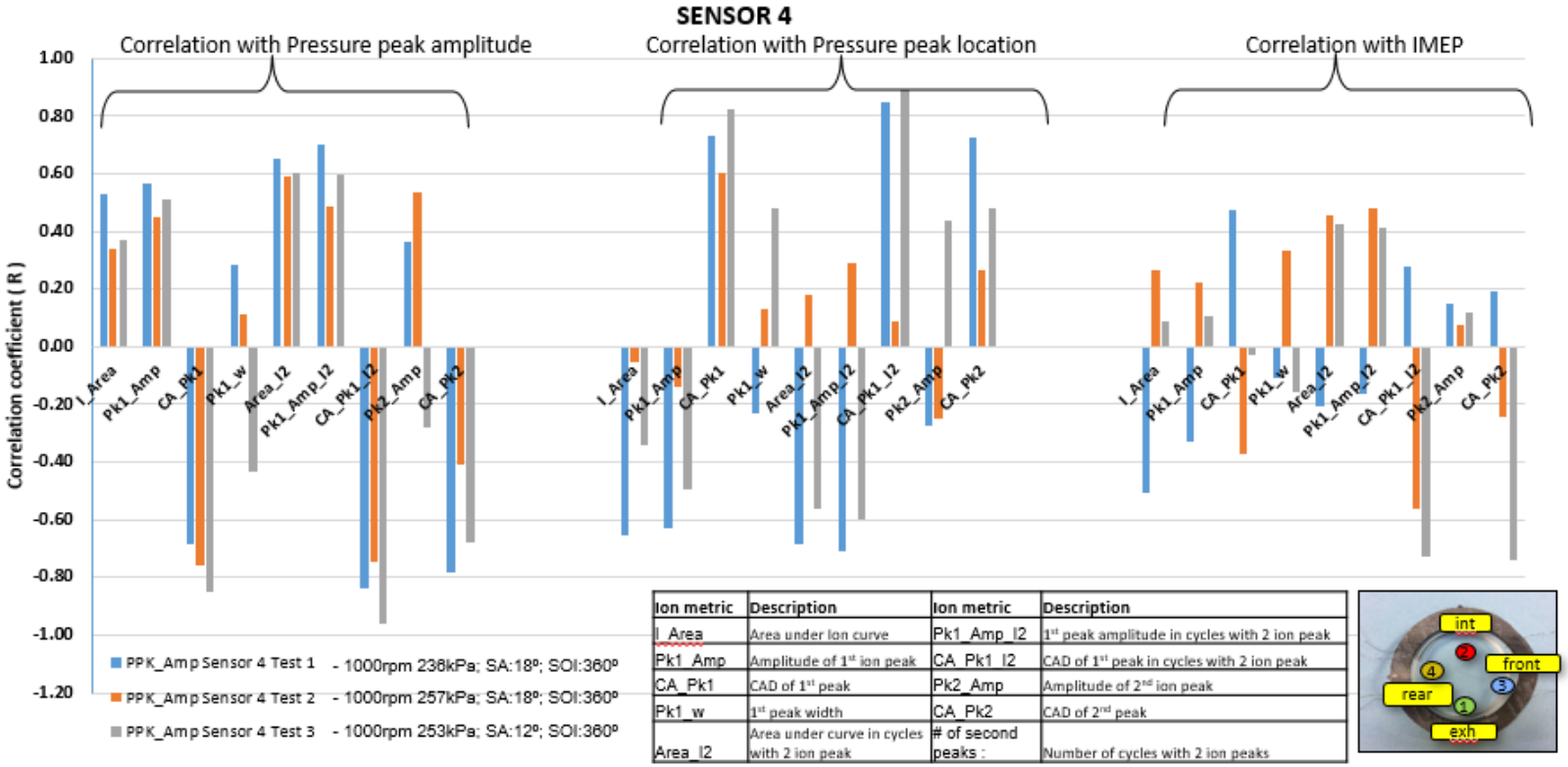


Figure 5.9: Correlation of ion sensor 3 with pressure metrics, across all tests



Ion metric	Description	Ion metric	Description
I_Area	Area under Ion curve	Pk1_Amp_I2	1st peak amplitude in cycles with 2 ion peak
Pk1_Amp	Amplitude of 1st ion peak	CA_Pk1_I2	CAD of 1st peak in cycles with 2 ion peak
CA_Pk1	CAD of 1st peak	Pk2_Amp	Amplitude of 2nd ion peak
Pk1_w	1st peak width	CA_Pk2	CAD of 2nd peak
Area_I2	Area under curve in cycles with 2 ion peak	# of second peaks :	Number of cycles with 2 ion peaks

Figure 5.10: Correlation of ion sensor 4 with pressure metrics, across all tests

5.1.2 Knock detection using Ion probe

Knock detection is a critical aspect of engine control as knock amplitudes over a certain limit could be detrimental if not fatal, to the engine. Accurate information regarding the occurrence of in-cylinder knock and its amplitude is vital in ensuring the powertrain control module (PCM) takes corrective measure to prevent damage to the engine. The in-cylinder ion sensors were thus studied to evaluate if they could offer accurate information regarding knock, thereby proving to be an alternative for conventional knock sensors.

Table 5.5 offers a list of operating conditions under which the ion sensor's knock detection capabilities were evaluated. In accordance with an in-house procedure to conduct knock tests, first the spark is altered on all cylinders until knock is observed in cylinder 2. Once knocking occurs in the desired cylinder i.e. cylinder 2, then the spark on all cylinder except cylinder 2 (which houses the ion sensors) were retarded to prevent knocking in them. Care was also taken to ensure that the magnitude of knock in cylinder 2 was below a nominal value of 2 bar. This was done by ensuring the spark timing for cylinder 2 is not advanced more than about 30degBTDC at 2500RPM, as high knock could damage the sensors and the engine. The threshold on spark timing is also dependent on throttle and intake air temperature amongst other factors. In the knock test, the valve timings and wastegate setting were set to

'auto' and thus were not manually controlled. All tests were also conducted under stoichiometric conditions. Further, the latest version of the standalone ion sensor was installed during these knock tests and was connected to the 300kΩ conditioning box. The coil ion probe was connected to the 250kΩ conditioning box.

Table 5.5
Test matrix for knock detection using ion sensors

Test	**Speed** (RPM)	**IMEP** (kPa)	**Knock Amp*** (bar)	**Spark** (dATDC)	**CA50** (dATDC)
Test 1	1500	1113	1.1	-6.5	20
Test 2	1500	1044	1.62	-2.0	24.7
Test 3	1500	921	0.19	3.5	33.9
Test 4	2500	680	0.57	-25.3	4.0
Test 5	2500	673	0.83	-26.3	1.5
Test 6	2500	662	1.05	-29.3	-1.0
Test 7	2500	724	0.69	-20.5	6.0
Test 8	2500	713	1.24	-23.5	3.0
Test 9	2500	694	1.98	-27.5	2.1

* 95^{th} percentile of knock amplitude

In the knock test previously listed (Table 5.5), the sampling rate for the in-cylinder pressure and ion signals were not uniform throughout. A knock window was defined,

in the test conducted, that spanned -30dATDC to 70dATDC. Within the knock window the sampling rate was 0.25CAD, outside the knock window it was 0.5CAD. This was done so as to capture the knock events. Figure 5.11 offers a visual representation of the knock window. A higher resolution was not achievable due to issues with the acquisition system used at the time of testing.

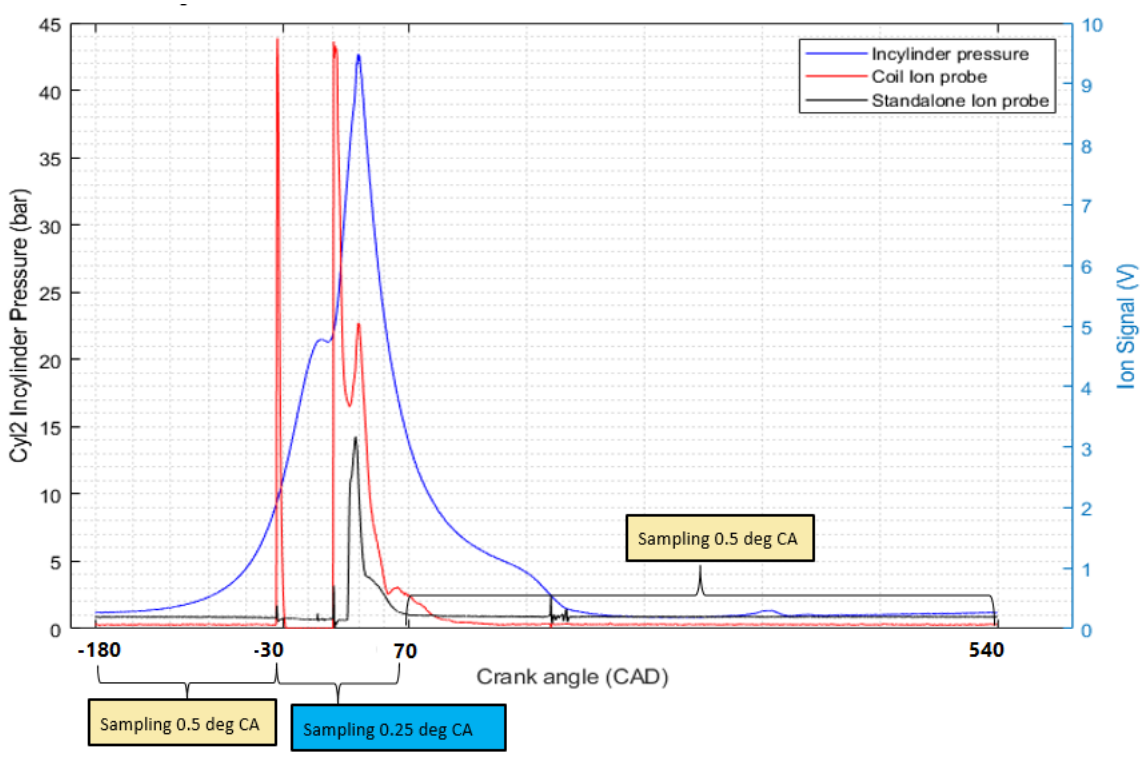


Figure 5.11: Knock window for knock tests

Considering the dual sampling rates of the pressure and ion signals, the signals had to be processed to obtain a uniform sampling rate. Figure 5.12 offers a flow chart of steps involved in obtaining the uniform sampling rate for the pressure and ion signals. First the pressure and ion signals outside of the knock window (i.e before -30dATDC

and after 70dATDC) were upsampled by a factor of two so that $\Delta\theta$ is 0.25 CAD throughout. Then the signal is low pass filtered to remove aliasing using an elliptical filter (default Matlab lowpass filter). For ease of understanding, filtering is conducted in frequency domain. The pass band of the lowpass (Fpass) was 45% of the sampling frequency(Hz) of the low resolution portion (i.e 0.45*(360/0.5)*RPM/60) and the sampling rate(Fs) was equal to the sampling rate(Hz) of the high resolution portion (i.e. (360/0.25)*RPM/60).

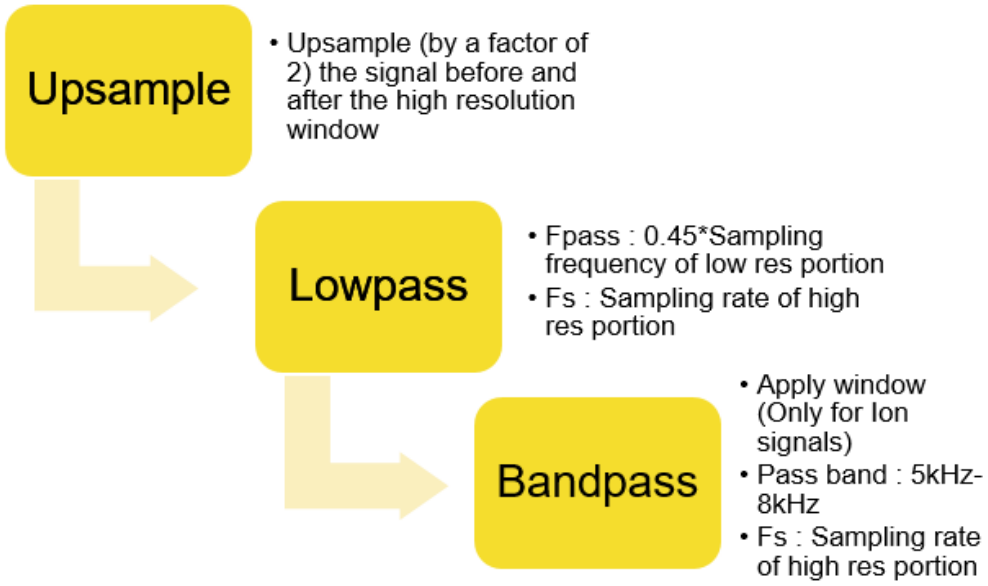


Figure 5.12: Filtering method to obtain uniform sampling rate

The pressure signal was then bandpass filtered using elliptical filters with a pass band of 5kHz to 8kHz and sampling rate equal to the sampling rate(Hz) of the high resolution portion (i.e. (360/0.25)*RPM/60) to obtain the filtered pressure signal

(termed as knock pressure in Figure 5.13). Additionally, the same bandpass filters were applied to the ion signals but before filtering the signal a window was applied to the ion signals. The type of window applied varied based on the algorithm used and included static Tukey windows and adaptive windows. Section 5.1.2.1 has more details of the methods developed to create the various windows used. Figure 5.13 and 5.14 show the bandpassed signal for a knocking and non knocking cycle of a particular dataset. It is also to be noted that in the graphs henceforth, the standalone ion probe would be referred to as SA and the coil ion probe as SC. Also 'BPass' refers to bandpassed signal.

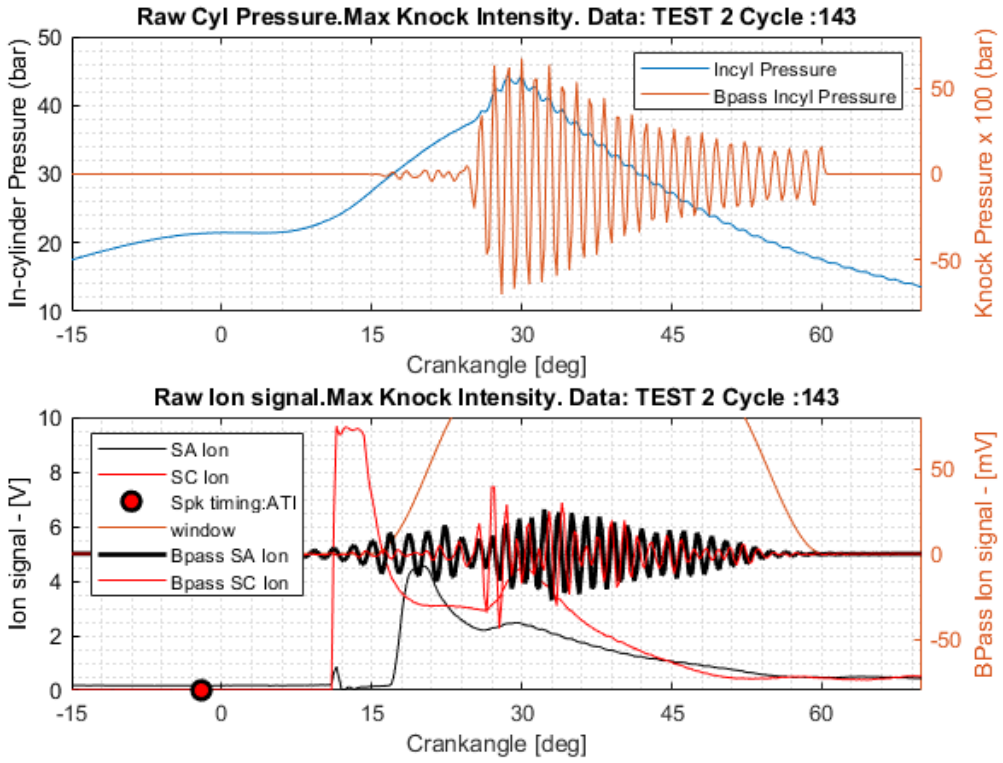


Figure 5.13: In-cylinder Pressure and ion signal for cycle with highest knock - elliptical bandpass(5-8kHz) filter

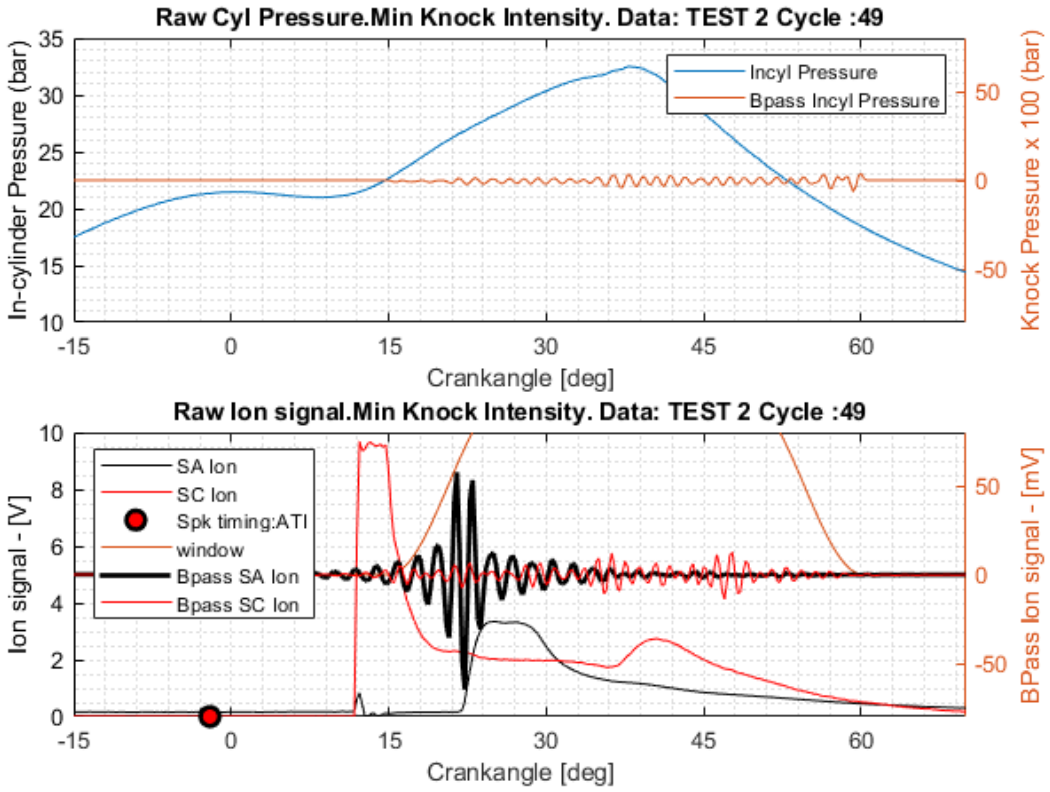


Figure 5.14: In-cylinder Pressure and ion signal for cycle with least knock-elliptical bandpass(5-8kHz) filter

Once the signals were bandpass filtered the pressure intensity (PI) and ion intensity(II) were calculated. The procedure to calculate PI and II are similar to the procedure described in Section 4.3. The primary intent of this step was to know if the ion intensity follows a log-normal distribution. The pressure intensity or knock integral is known to follow a log-normal distribution under knock conditions, based on previous work conducted by Naber et al. [22]. Thus if under the same knocking conditions, the ion integral as well correlates with pressure intensity and followed a log-normal distribution ion signal can potentially be indicative of in-cylinder knock.

Figures 5.15-5.17 show the distribution of the pressure and ion intensity respectively. Each figure also shows the probability and cumulative distribution function.

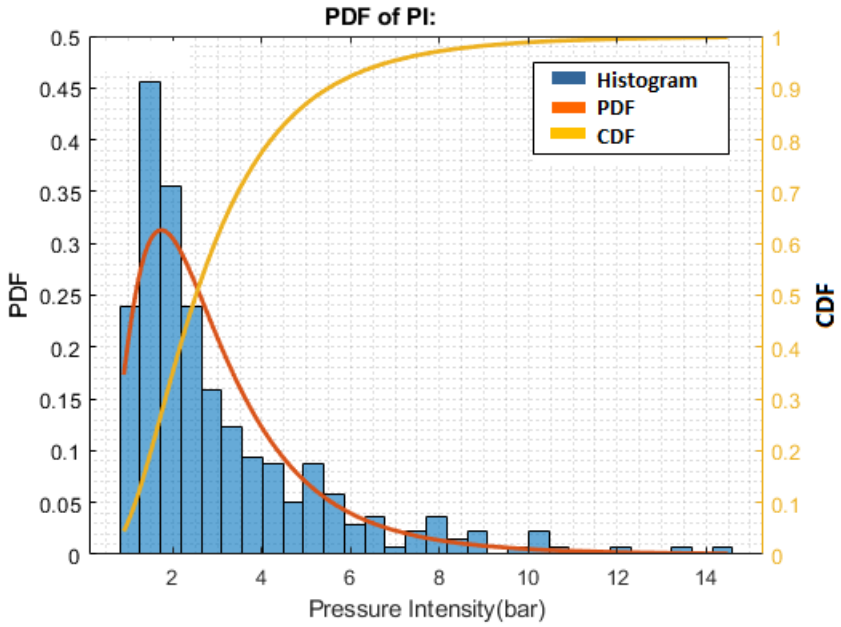


Figure 5.15: Distribution of pressure intensity - Test 1

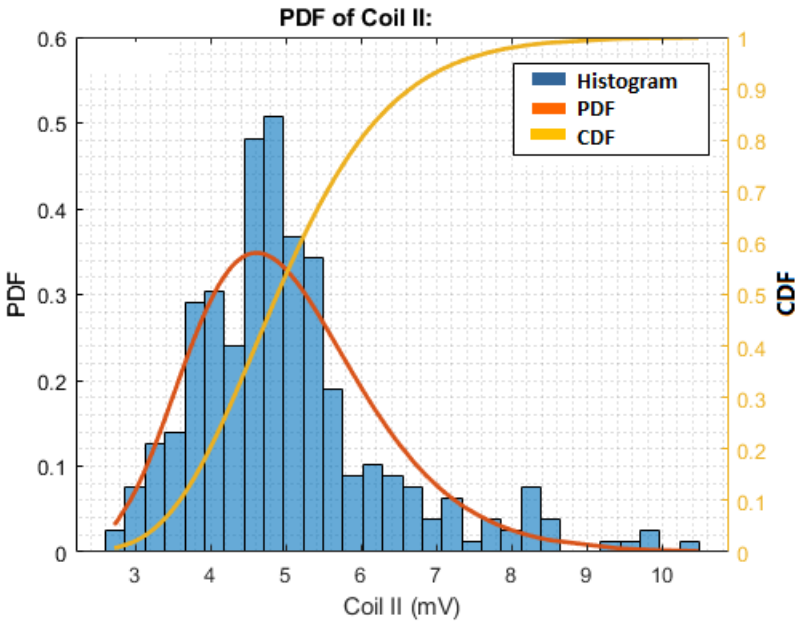


Figure 5.16: Distribution of ion intensity for coil ion probe - Test 1

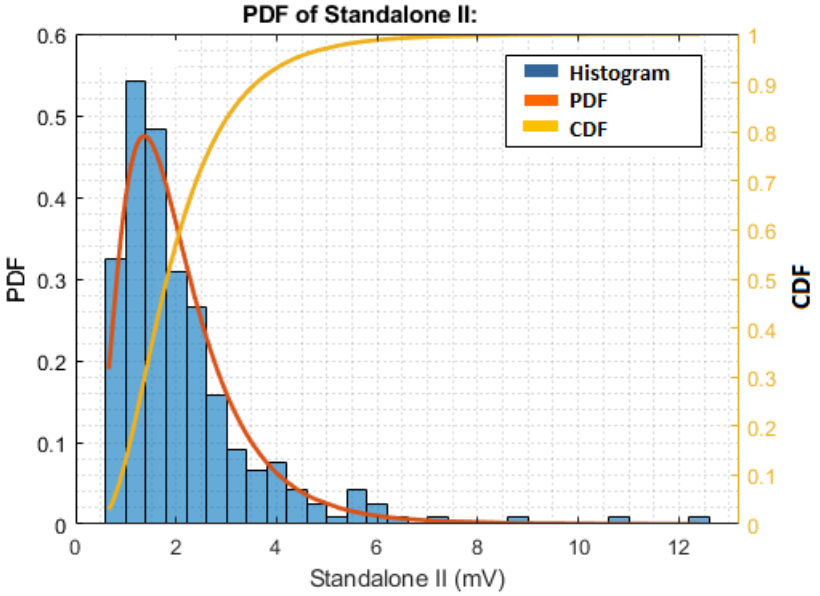


Figure 5.17: Distribution of ion intensity for standalone ion probe - Test 1

Through visual verification it was seen that a log normal distribution is a good fit

for pressure intensity and ion intensity (both probes). Both ion intensities showed similar skewness but the standalone ion intensity showed higher kurtosis.

In the knock study, initially, a static Tukey window(between 15dATDC to 60dATDC) was applied to the ion signals, both coil and standalone. The use of this stationary window causes an issue in cycles where the flame front occurs within the window. For example in Figure 5.14, the flame front as detected by the standalone ion probe, is within the knock window of -30dATDC to 70dATDC. The steep rise in ion concentration (at 21dATDC) causes the large oscillations seen on the bandpassed standalone ion signal. This in turn leads to an improper estimate of ion intensity and could lead to false classification of the cycle as one with high knock. To counteract this problem various methods were developed that changed the start of the windows applied based on a set of criteria specified. Subsequent sections describe the approaches developed.

5.1.2.1 Adaptive window

One method developed was to use an adaptive window instead of a simple static window. In this method, the location of the flame front as observed by the standalone ion sensor is first detected. The start of the window applied is then offset by a specified number of CAD w.r.t the location of the flame front detected. Figure 5.18 shows a visual representation of the algorithm developed, wherein the first blue triangle indicates the location of flame front w.r.t. standalone ion probe and the second triangle

indicates the start of the window applied. In this particular dataset a threshold of 1V was used to identify flame front and an offset of 4 CAD was used. It is to be noted that in this method the same window is applied to the coil and standalone ion signals.

Figure 5.19 shows the correlation of the ion intensity with the pressure intensity for the coil and standalone ion signals after implementing the adaptive window. The results show that with the application of adaptive windowing technique, the performance of the standalone ion probe was good (R=0.7>0.6) but the performance of the coil ion probe was not satisfactory(R=0.5<0.6).

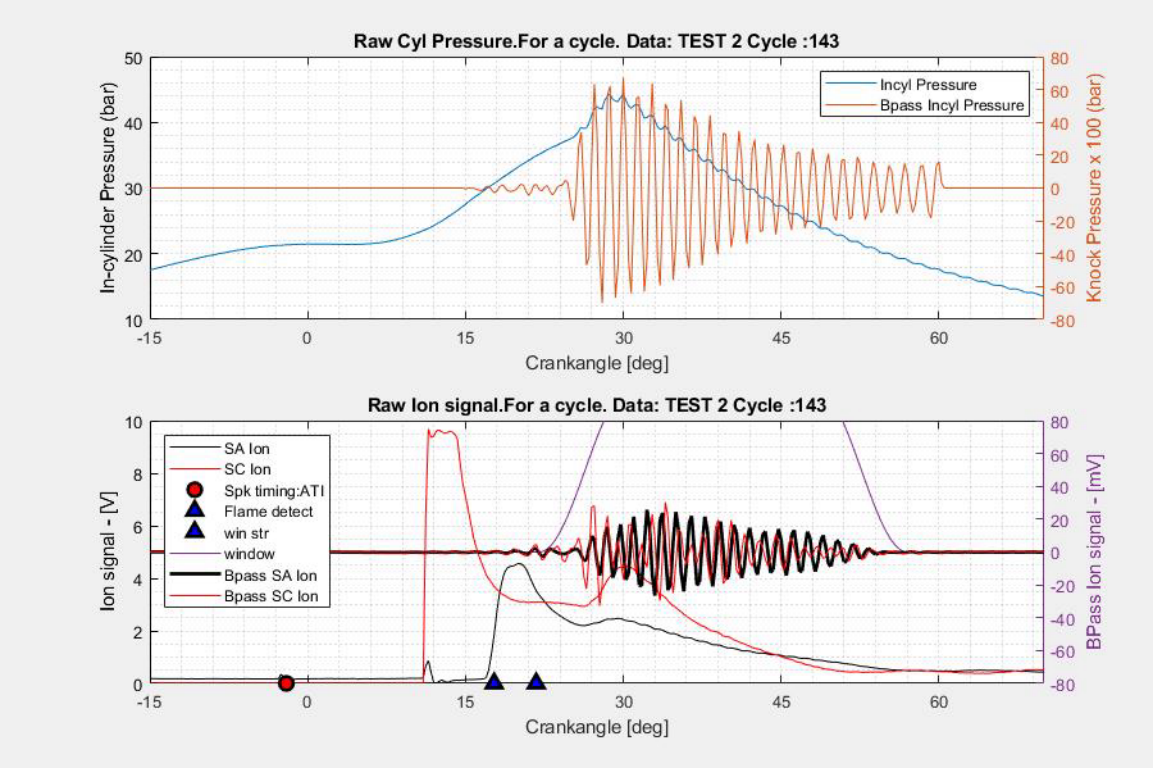


Figure 5.18: Implementation of Adaptive windowing -Test 2

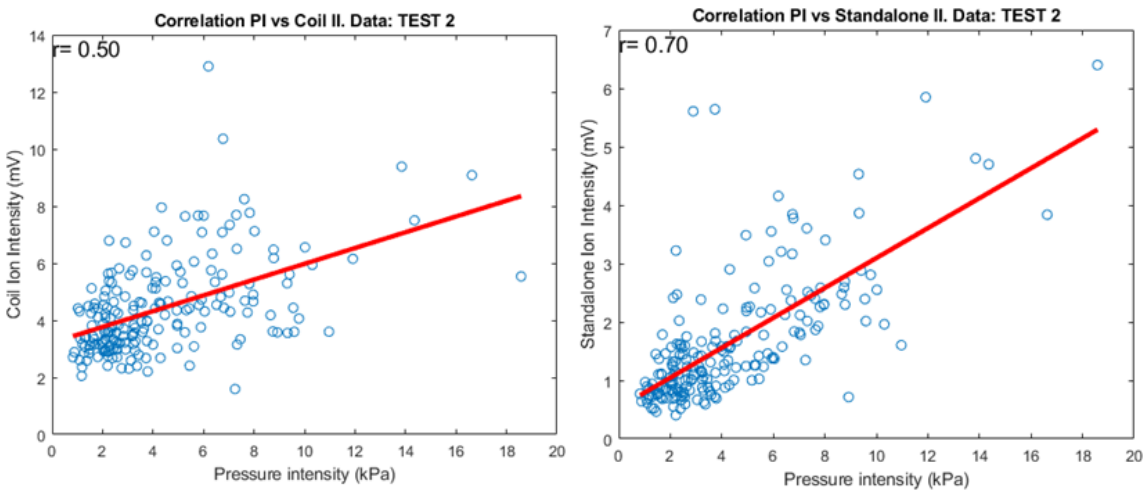


Figure 5.19: Correlation of ion intensities with adaptive windowing -Test 2

5.1.2.2 Adaptive and Static window

Another approach tried was to use separate windows for each of the ion sensors. The coil ion probe used a static window and the standalone ion probe used the adaptive window, mentioned previously. Figure 5.20 offers a visual representation of the algorithm. The window applied to the coil ion signal starts at 15dATDC and is 45CAD long, while the window applied to the standalone ion signal is 35CAD long and is offset from the flame front by 4CAD. The blue triangles indicate the flame front location and the start of the adaptive window. The lengths of the window and the offset was decided by trial and error.

Figure 5.21 shows the correlation of the ion intensities with pressure intensity for

the two ion sensors using the adaptive and static windowing technique. The window applied to the standalone ion signal in this technique is similar to that of the previous technique and hence there isn't any improvement in the correlation achieved for the standalone ion probe. However, applying the static window, to the coil ion signal did not show a noticeable improvement in correlation with pressure intensity. Further, in certain cases, when the ringing portion of the coil ion signal was within the static window zone, it lead to large oscillations being observed in the bandpassed coil ion signal. This in turn lead to an improper estimate of the coil ion intensity. To avoid this scenario, a modified dual adaptive windowing technique was developed.

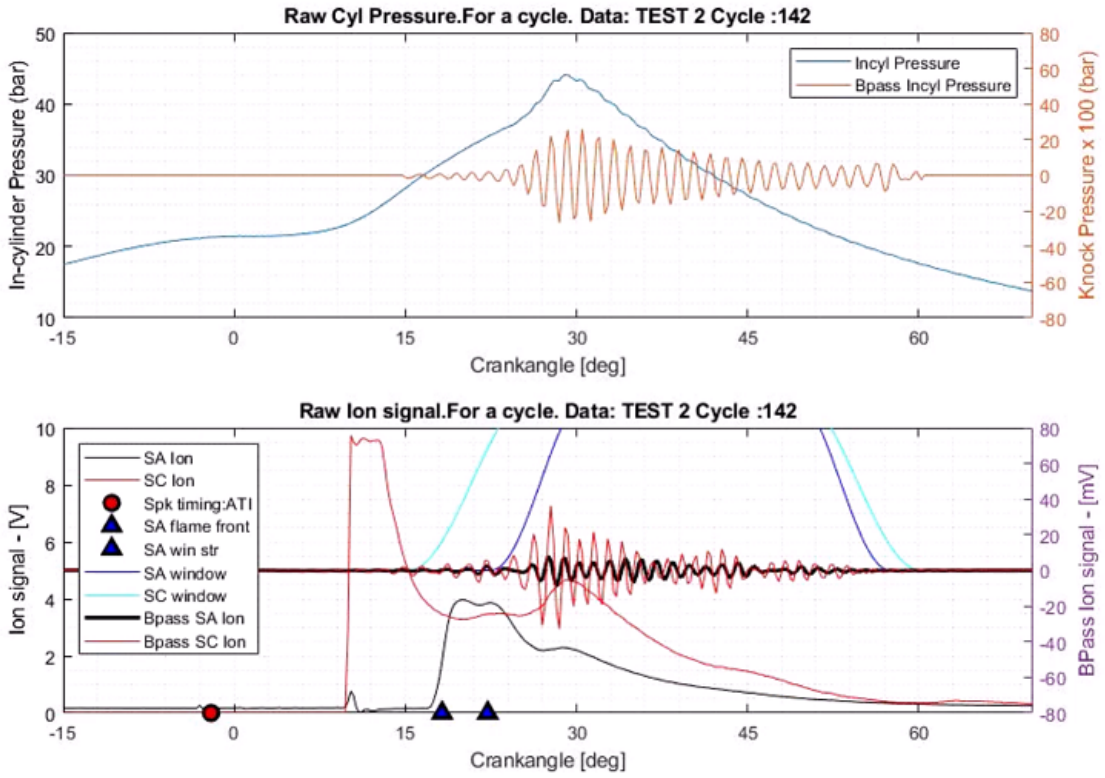


Figure 5.20: Implementation of Adaptive static windowing- Test 2

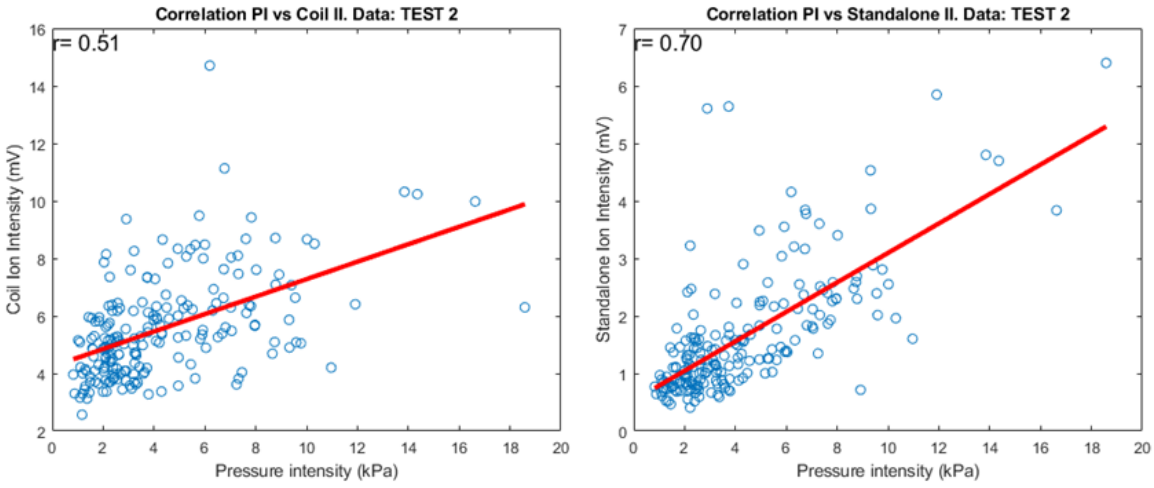


Figure 5.21: Correlation of ion intensities with Adaptive static windowing- Test 2

5.1.2.3 Modified dual adaptive window

The final method developed was the modified dual window technique, wherein a separate window was used by each of the ion sensors and both of them were adaptive in nature. The window used by the coil ion probe, 35CAD long, was designed to avoid the ringing phase of the coil ion signal and the window applied to the standalone ion probe, also 35CAD long, was designed to be applied at an offset with respect to the flame front detected. Figure 5.22 offers a visual representation of the algorithm. The blue triangles indicates the point of flame detection and the window start for the standalone probe ion signal(6CAD offset used here). The red triangles indicate the location of the start of ringing and the start of the window for the coil ion probe signal(8CAD offset used here). It was seen that using the modified adaptive windowing technique, correlation of both ion intensities with pressure intensity was improved. Thus the technique implemented reduced the influence of the flame front and ringing on the ion intensity estimates.

Using this method offered a good correlation ($R > 0.6$) with pressure intensity for both ion probes as can be seen from Figure 5.23. This was true both at low and high speed regimes, making the method the more reliable amongst the techniques developed. Further a slight decrease in R value of the standalone ion intensity is observed due to a change of dataset used for analysis.

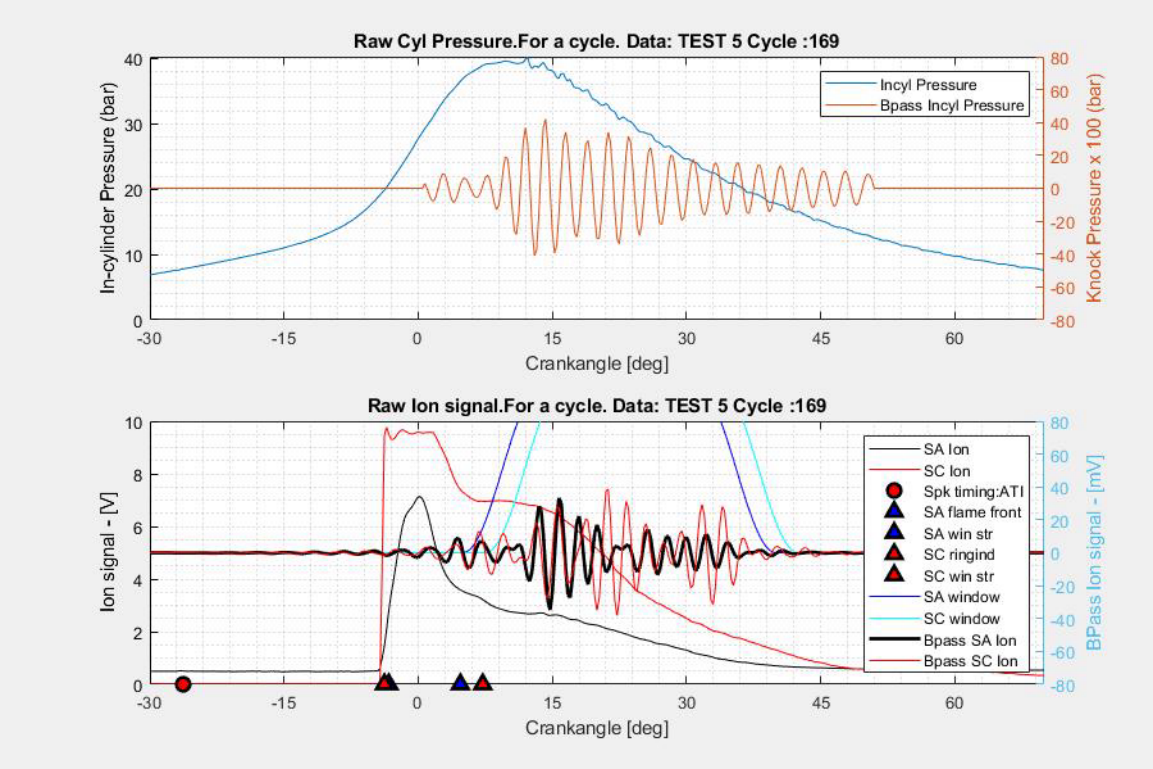


Figure 5.22: Implementation of Modified adaptive windowing- Test 5.

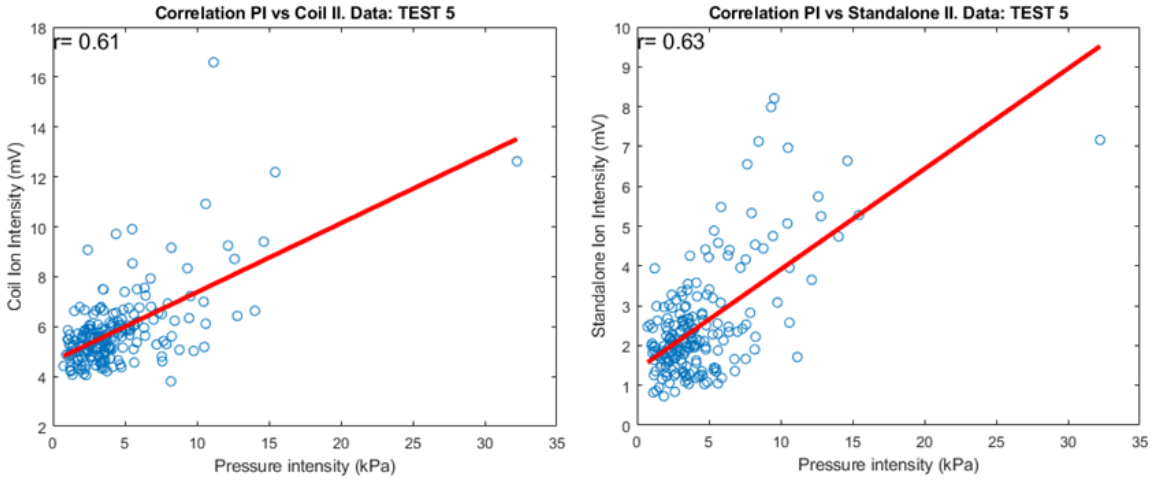


Figure 5.23: Correlation of ion intensities with modified adaptive windowing - Test 5

5.1.2.4 Effect of using the adaptive windowing

In Section 5.1.2.1, the effect of using the various techniques in knock detection were discussed. It was seen that the modified adaptive windowing technique showed improved performance in terms of offering a correlation between the ion intensity and pressure intensity for knock tests. Figure 5.24 and 5.25 show a comparison of the linear spectra (in a log scale) of pressure and ion signals. The plots shows the linear spectrum for the cycle with the highest and lowest knock amplitudes within the dataset (i.e Test 5).

Figure 5.24 shows the results when a simple static window is applied to the three signals viz in-cylinder pressure, standalone ion and coil ion signal. It can be seen that for the cycle with highest knock, the spectrum of the in-cylinder pressure shows a peak around 6kHz-7kHz, which corresponds to the frequency of knock for the engine used in this study. However, the standalone and coil ion signals for the same high knock cycle does not show a prominent peak when a static window is used. It is to be noted that though a minor peak is observed on the standalone, it is indistinguishable from peaks at other frequencies. This indicated that using a static window for processing ion signal is not beneficial in obtaining information regarding knock.

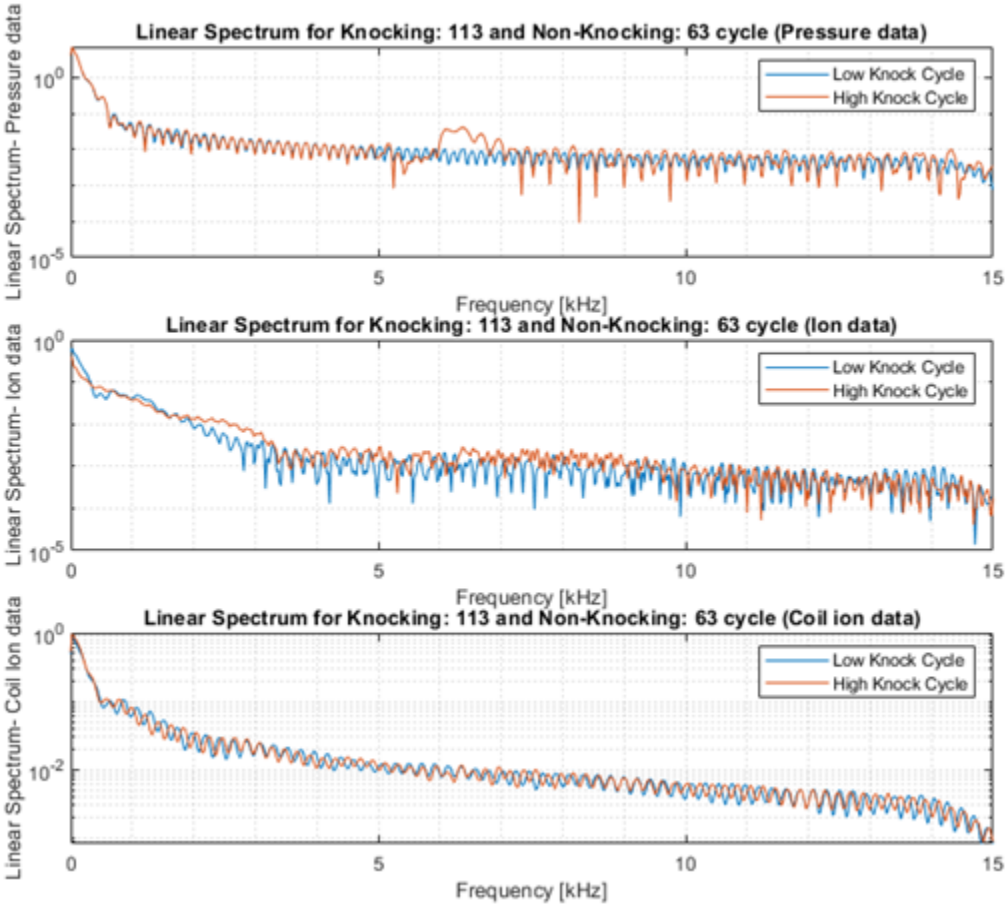


Figure 5.24: Linear spectrum visualization without using the custom windowing algorithm - Test 5

Figure 5.25 on the other hand shows the liner spectrum of the in-cylinder pressure and ion signals when the ion signals are processed using the modified adaptive windowing technique discussed earlier. The results are shown for the same dataset and cycles in Figure 5.24.

Further in Figure 5.25 it can be seen that the linear spectrum of the standalone ion signal shows a peak around the knock frequency of 6kHz-7kHz, for the cycle with high

knock, when the modified windowing technique is used. However the linear spectrum of the coil ion probe do not show as a prominent peak around the knock frequency. One possible cause for this could be that the position of the coil ion probe causes an issue in offering reliable information regarding knock. None the less the results show that for the conditions tested, the standalone ion probe is capable of knock detection.

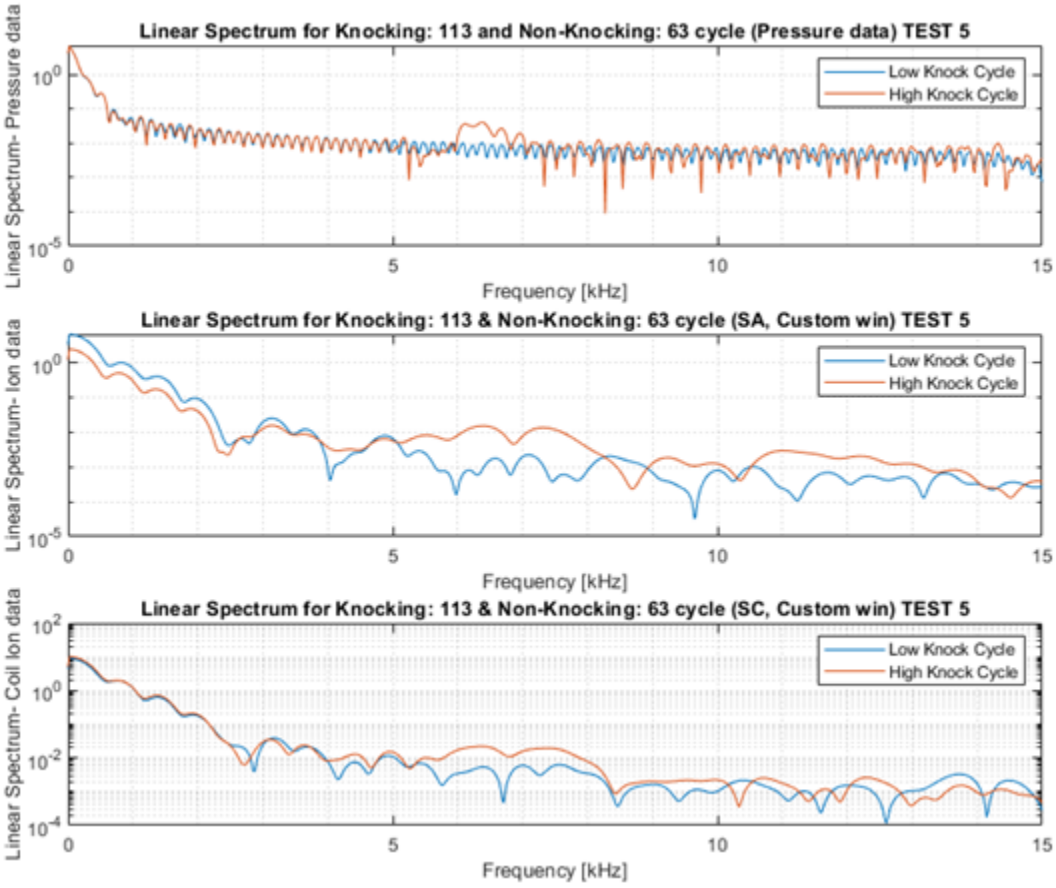


Figure 5.25: Linear spectrum visualization with using the custom windowing algorithm - Test 5

5.1.2.5 Conclusions on ion sensor studies

Through the ion sensor studies conducted on the metal engine, it was found that location of ion signal peak could be used to correlate with pressure metrics. Further, the knock studies conducted also showed that the ion signals required additional processing to be able to detect knock. The use of the modified adaptive windowing technique showed better results. Despite the knock amplitude being relatively low (<2bar) the standalone ion sensor was able to detect knock. Through the optical engine studies it was found that location of the sensor also plays a critical role in obtaining good correlations with pressure and combustion metrics. Detailed conclusions of the ion sensor studies are listed in Chapter 6.

5.2 Exhaust pressure sensor

The study of exhaust pressure, as showcased in this body of work, was conducted across two facilities. The steady state tests were conducted in the APS labs at Michigan Tech. while the in-vehicle transient tests were conducted at the Ford test facility. As mentioned previously, the engine was instrumented with two exhaust sensors. A Kulite sensor, placed close to the exhaust port (10mm) and an Omega sensor, located at a standoff(18in). The bulk of this study utilizes the Kulite sensor as it was able to capture the exhaust dynamics better than the Omega. A comparative analysis of the Omega and the Kulite is also discussed towards the end of this section.

5.2.1 Feature extraction and correlation with combustion metrics in steady state

Table 5.6 lists a subset of various operating conditions at which steady state data was acquired. The collected data was analyzed to identify features in the exhaust signal that correlate with pressure and combustion metrics. Figure 5.26 shows the exhaust signal under normal combustion conditions. The various sections of the exhaust signal are color coded based on the corresponding cylinder pressure. The "star" markers signify in-cylinder peak pressure and exhaust maxima; circles indicate EVO.

Table 5.6
Test matrix for exhaust sensor evaluation

Test	**Speed**	**IMEP**	**Intake advance**	**Exhaust retard**	**CA50**	**COV**
	RPM	KPa	deg	deg	deg	%
Test 1	1500	250	0	0	8	0.74
Test 2	1500	750	0	0	8	0.59
Test 3	3500	250	0	0	8	0.68
Test 4	3500	750	0	0	8	0.42
Test 5*	1500	250	-35	35	8	9.89

* Lambda=0.9 - Misfire testing

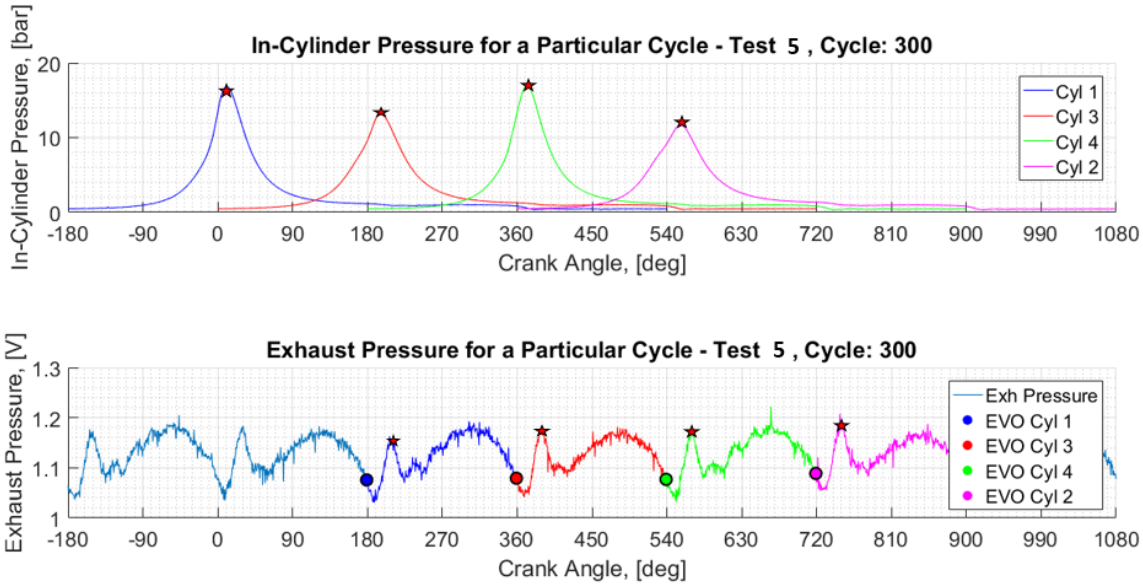


Figure 5.26: Exhaust signal during healthy combustion in all cylinders - Test 5 . "star" indicates maxima , "circle" indicates EVO

Previous studies by Prabhu [1] showed that the exhaust pressure could be used to identify misfire events. Figure 5.27 shows the visual representation of in-cylinder and exhaust pressure of a cycle with misfire in cylinder 2. The engine used in this study has four cylinders and follows the firing order 1-3-4-2. Thus in Figure 5.27 it can be seen from the in-cylinder pressure and IMEP mentioned, that cylinder 2 misfired. Concretely, the exhaust pressure of cylinder 2 showed a deep trough, indicative of the misfire in the cylinder. Thus by using the amplitude of the exhaust pressure minima, misfire events can be easily distinguished from normal combustion cycles. However, as shown in Figure 5.28 and 5.29, the exhaust pressure offers additional information as well.

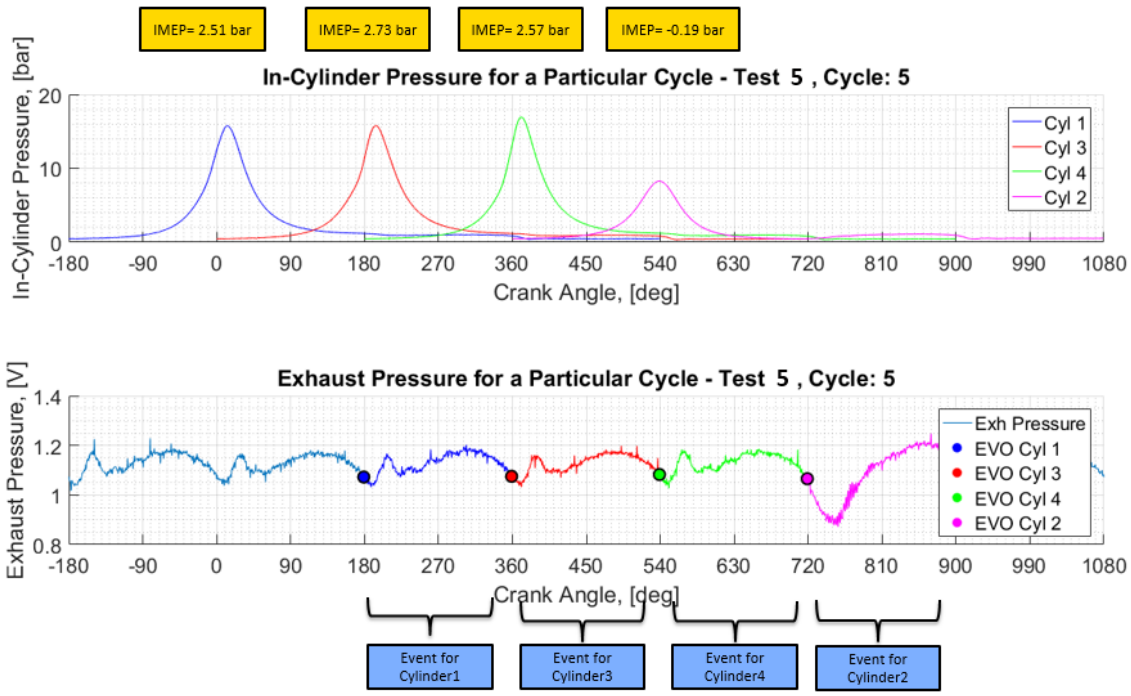


Figure 5.27: Exhaust signal for a cycle with misfire - Test 5

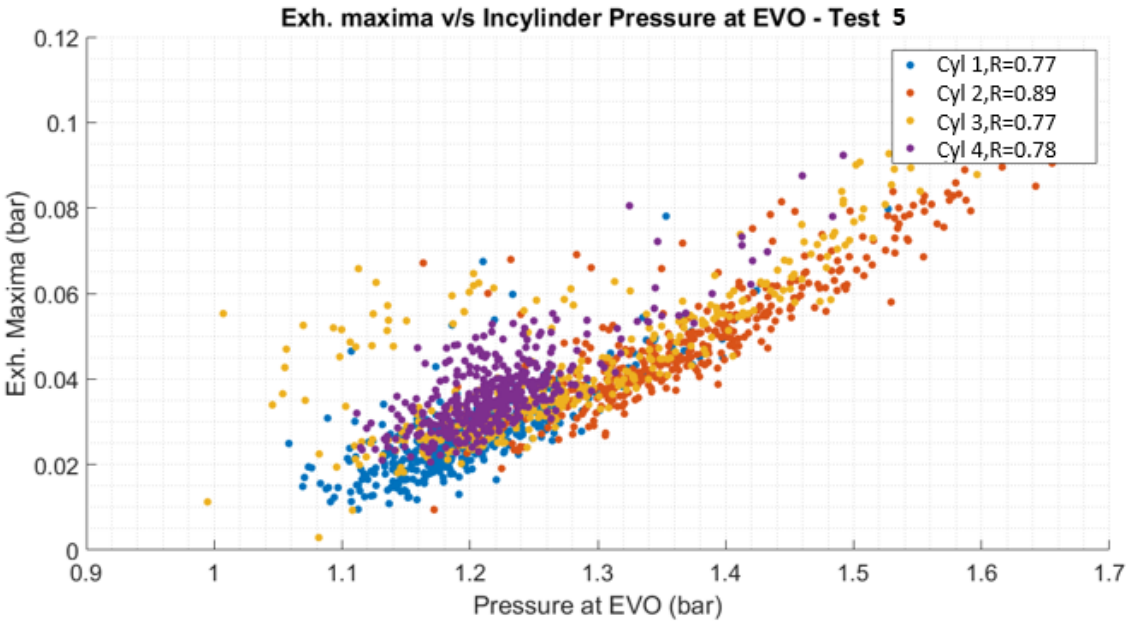


Figure 5.28: Correlation of Exhaust pressure with in-cylinder pressure at EVO - Test 5

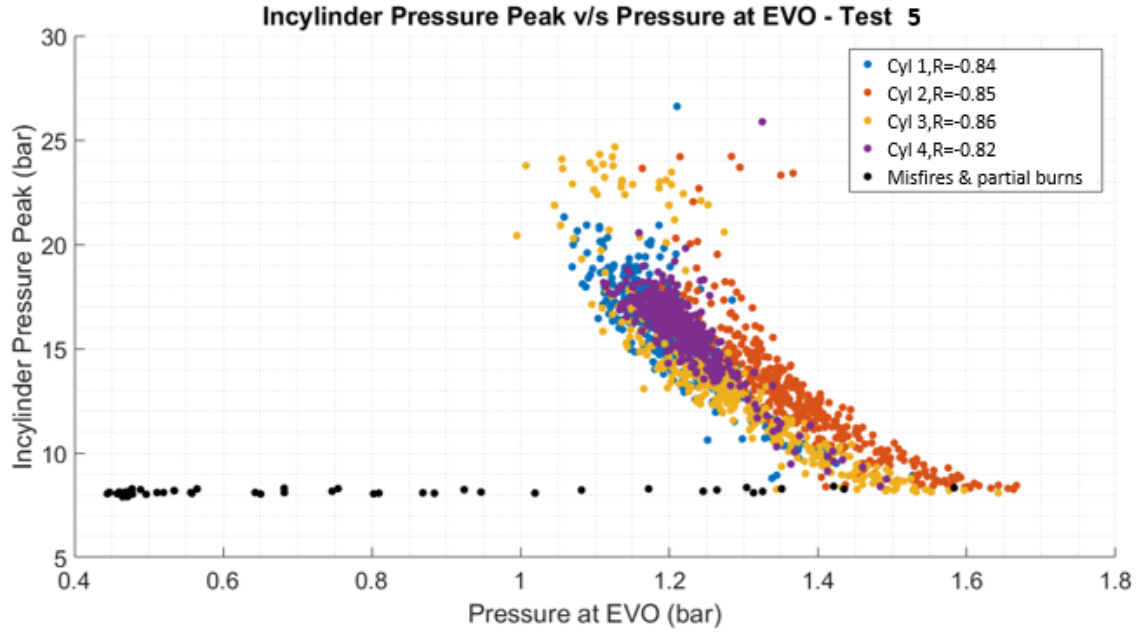


Figure 5.29: Correlation of incylinder pressure peak with pressure at EVO - Test 5

When evaluating correlation of exhaust pressure with incylinder pressure metrics, it was observed that a direct correlation between the two parameters was weak. However, both exhaust pressure and incylinder peak pressure correlate well (i.e R >0.6) with in-cylinder pressure at EVO as shown in Figure 5.28 and 5.29. The figures show the correlation for all four cylinders at a particular operating condition i.e Test 5.

5.2.2 Factors affecting exhaust signatures

The steady state studies showed that the exhaust pressure contains valuable information that could be used for diagnostic and control applications. However, as the operating conditions change, the exhaust waveform shape change, as seen in Figure 5.30.

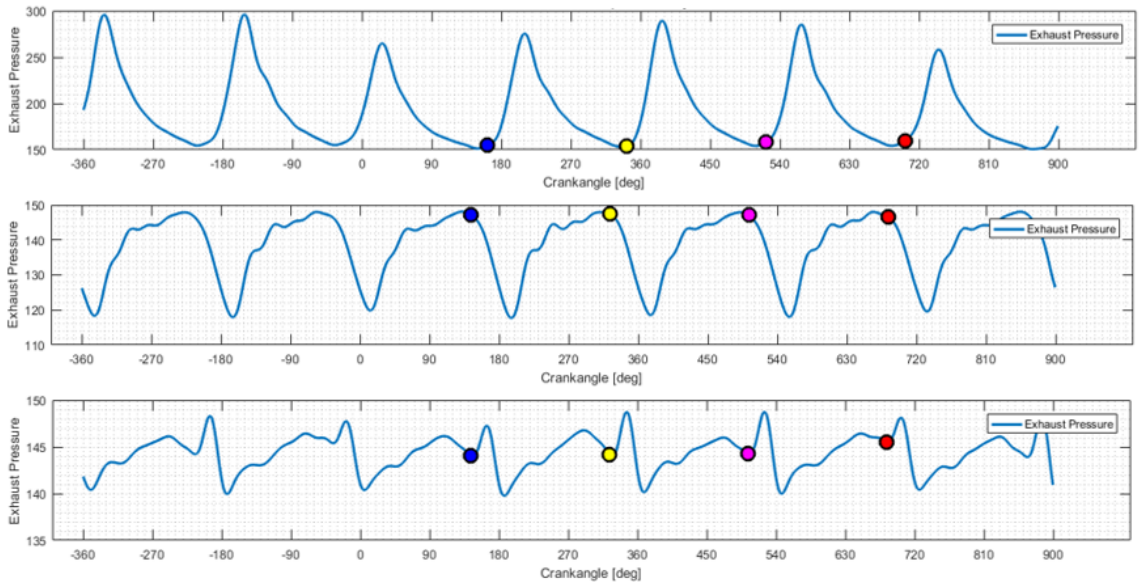


Figure 5.30: Exhaust pressure waveform at various operating conditions

In Figure 5.30, each exhaust waveform corresponds to a particular operating condition, and the ∘ markers indicate the EVO of each cylinder. Further, considering how drastically the waveform changes, it would be incorrect to use a single signal artifact (ex. signal maxima) as a feature to evaluate exhaust correlation with combustion metrics. Thus it was critical to understand exhaust pressure signatures and factors that affect waveform signatures. This exercise helped identify signal artifacts that could be used to correlate exhaust pressure with combustion metrics.

In an effort to make the identification of exhaust signatures and in-turn the correlation studies more comprehensive, tests were conducted in a vehicle setup under real-world testing conditions. Figure 5.31 shows a section of the drive profile for a particular road test conducted. The effect of various engine parameters including load, engine speed, spark timing etc. on exhaust signatures was studied.

Initially it was found that engine load could be used to classify waveforms into two categories. A normalized load index of 0.27 was used as a threshold to segregate waveforms as Type I (ones with prominent peaks) and Type II (ones with troughs). The threshold was found by trial and error. In type I waveforms, the exhaust peak showed good correlation with combustion metrics as seen in Figures 5.32 - 5.35.

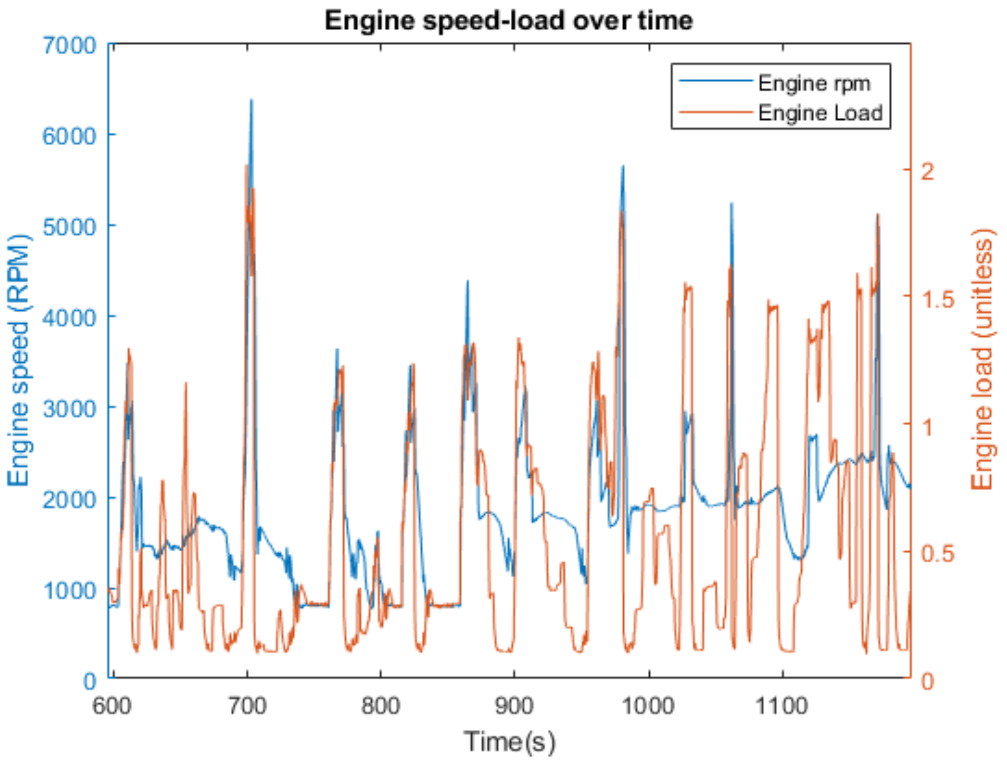


Figure 5.31: Section of transient drivecycle

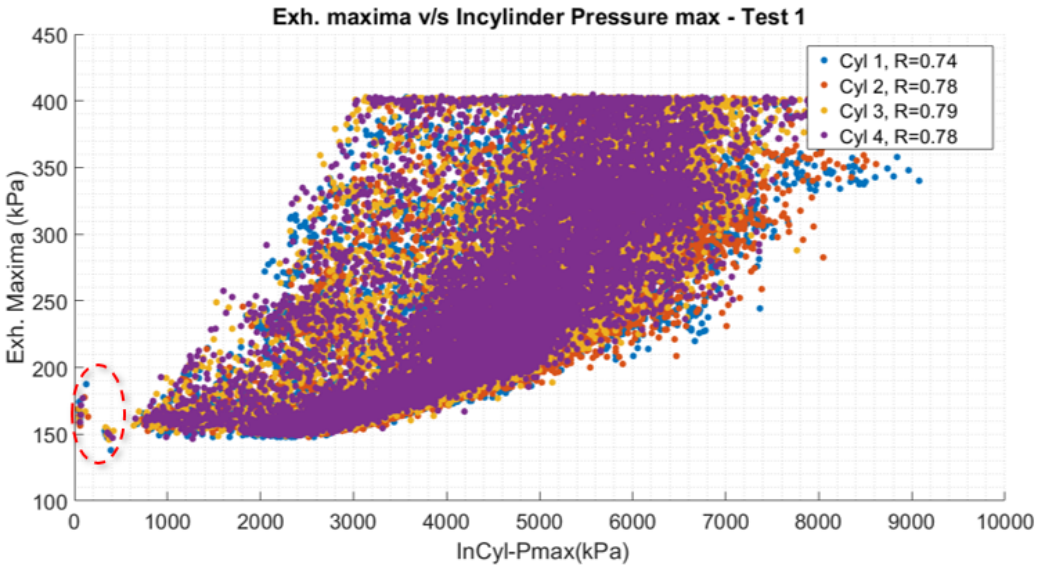


Figure 5.32: Correlation of Type I waveform exhaust peaks with in-cylinder pressure

Figure 5.32 shows the correlation of exhaust peak pressure with in-cylinder pressure for all cylinders to be greater than 0.74. Additionally the exhaust maxima was seen to saturate at 400kPa; This was due to the rating of the pressure sensor used for this particular test. In subsequent tests the sensor was suitably altered to avoid saturation. Also the cycles encircled in the bottom left represent cycles with deceleration fuel shut off (DFSO) and cycles with a synchronization issue. The exhaust peaks however did not show as good a correlation with location of in-cylinder peak pressure. Figure 5.33 shows the correlation. It can be seen that all the cylinders showed a correlation lower than 0.6. Further the data-points encapsulated in the rectangle indicate cycles with DFSO, sync issues and spark retard (i.e the compression peak was detected).

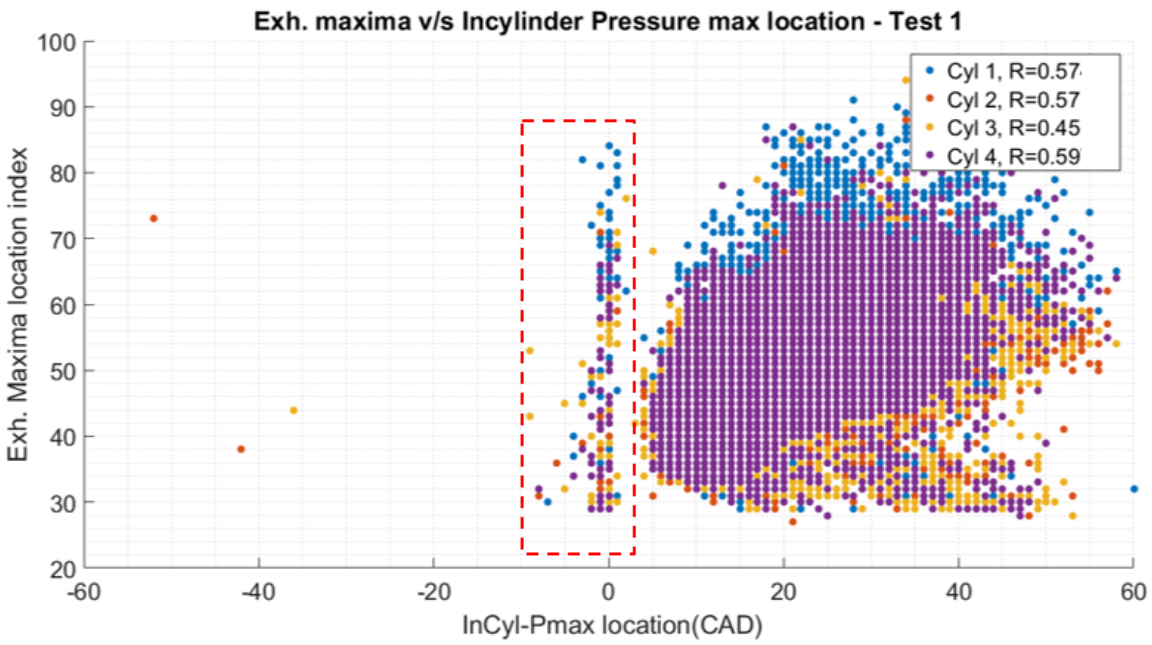


Figure 5.33: Correlation of Type I waveform exhaust peak location with in-cylinder pressure location

The exhaust maxima for Type I cycles also showed good correlation with combustion metrics including IMEP and CA50. Figure 5.34 shows the correlation of the exhaust maxima with IMEP for Type I cycles. It can be seen that irrespective of the cylinder, the exhaust maxima shows a correlation greater than 0.97 with IMEP. The cycles with near zero and negative IMEP were cycles with DFSO, sync issues or spark retard.

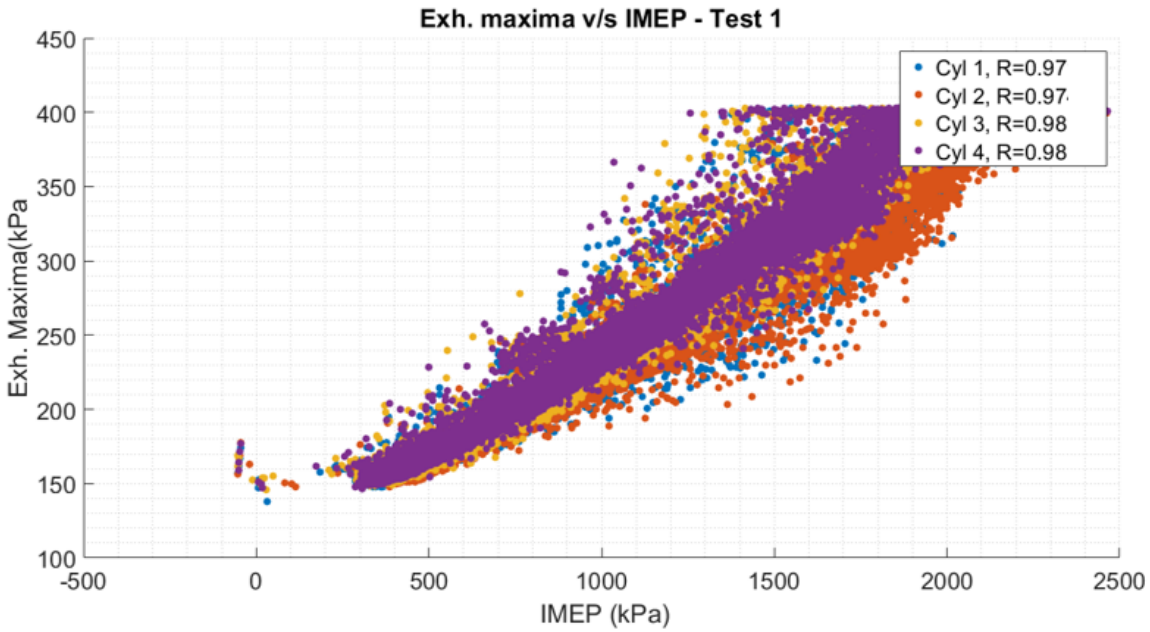


Figure 5.34: Correlation of Type I waveform exhaust peak with IMEP

Similarly, the exhaust peak also showed a correlation greater than 0.69 with location of 50% MFB or also called CA50. This correlation was not as high as that seen for IMEP, thereby indicating that exhaust pressure can be used for IMEP estimation with greater accuracy than for CA50 estimation.

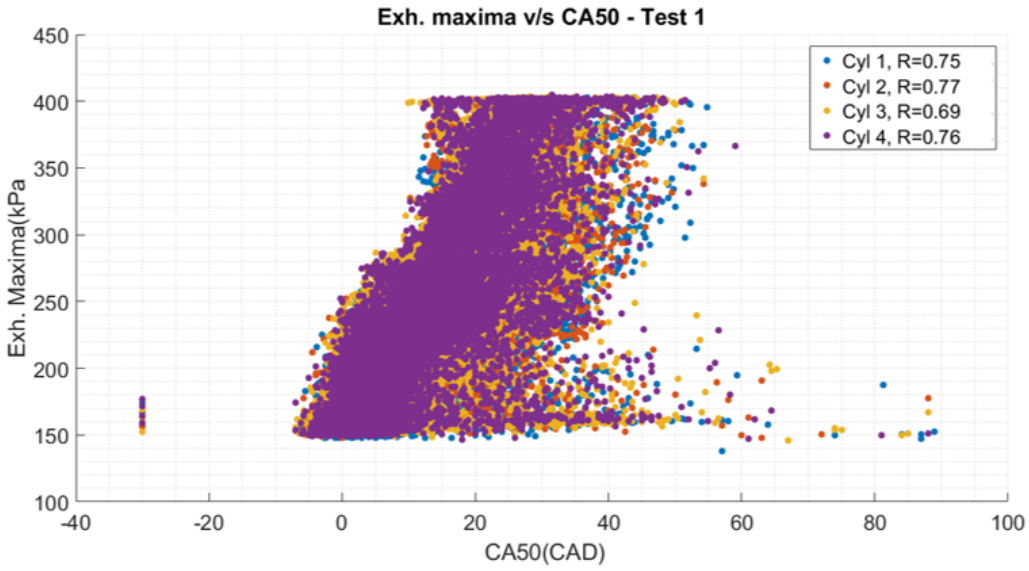


Figure 5.35: Correlation of Type I waveform exhaust peak with CA50

Moving on to the low load cycles or the Type II waveforms; The exhaust minima was used for correlation studies as the waveforms for a number of cycles did not have a prominent peak. Figure 5.36 shows the results of the correlation studies.

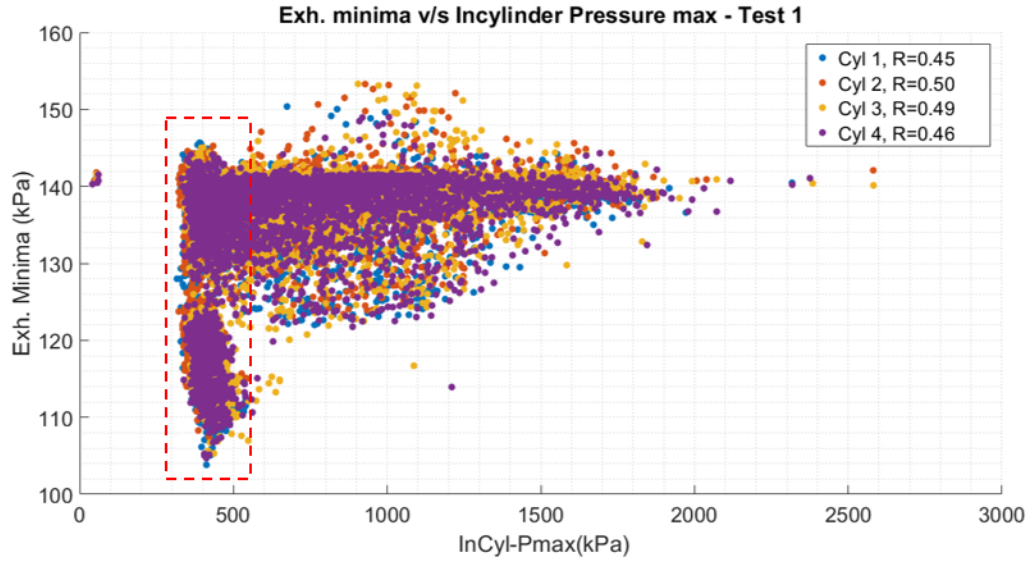


Figure 5.36: Correlation of Type II waveform exhaust trough with in-cylinder pressure peak

As seen in Figure 5.36, apart from the encapsulated datapoints that signify DFSO, the amplitude of exhaust minima does not vary by more than about 25kPa with changes in in-cylinder peak pressure. This lead to a correlation of less than 0.5 across all the four cylinders.

In order to study exhaust waveforms under low load conditions in greater detail, an algorithm was developed that used any particular engine parameter (Engine speed in this case) and color coded exhaust waveforms based on their intensity of the chosen parameter. This helped identify parameters that could be used to distinguish the various waveform signatures. Additional test data was also acquired to conduct this phase of the study. The drivecycle of the transient test is shown in Figure 5.37.

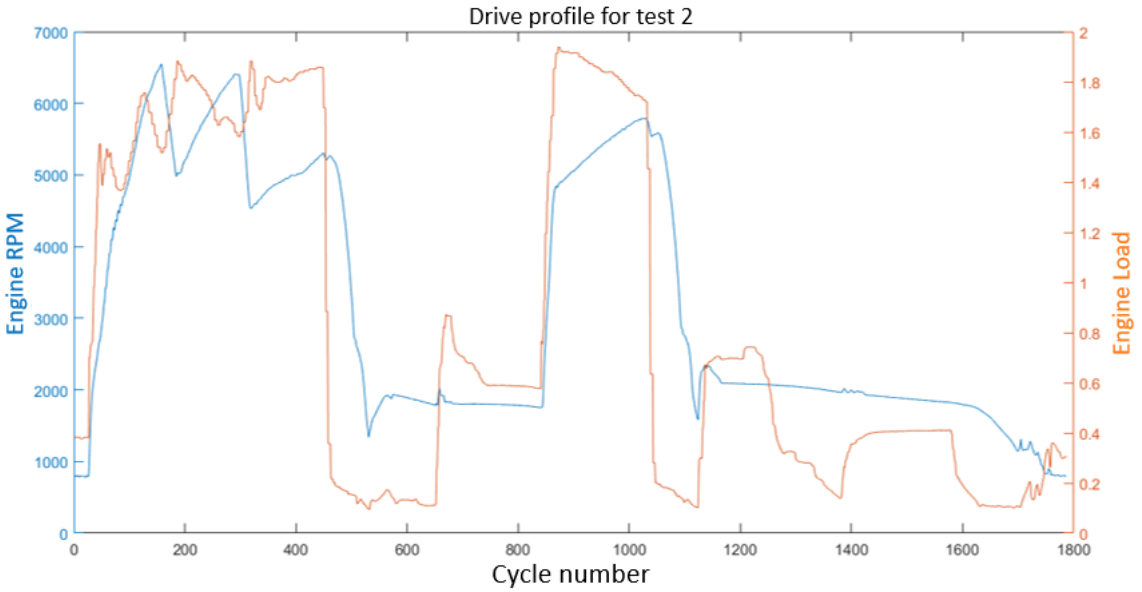


Figure 5.37: Transient drivecycle for evaluating exhaust signatures of low load cycles

When analyzing the Type II exhaust waveforms using the tool developed it was found that engine speed played an important role in defining waveform signatures as well. Figure 5.38 shows a series a images where engine speed was used to segregate exhaust waveforms pertaining to a particular cylinder.

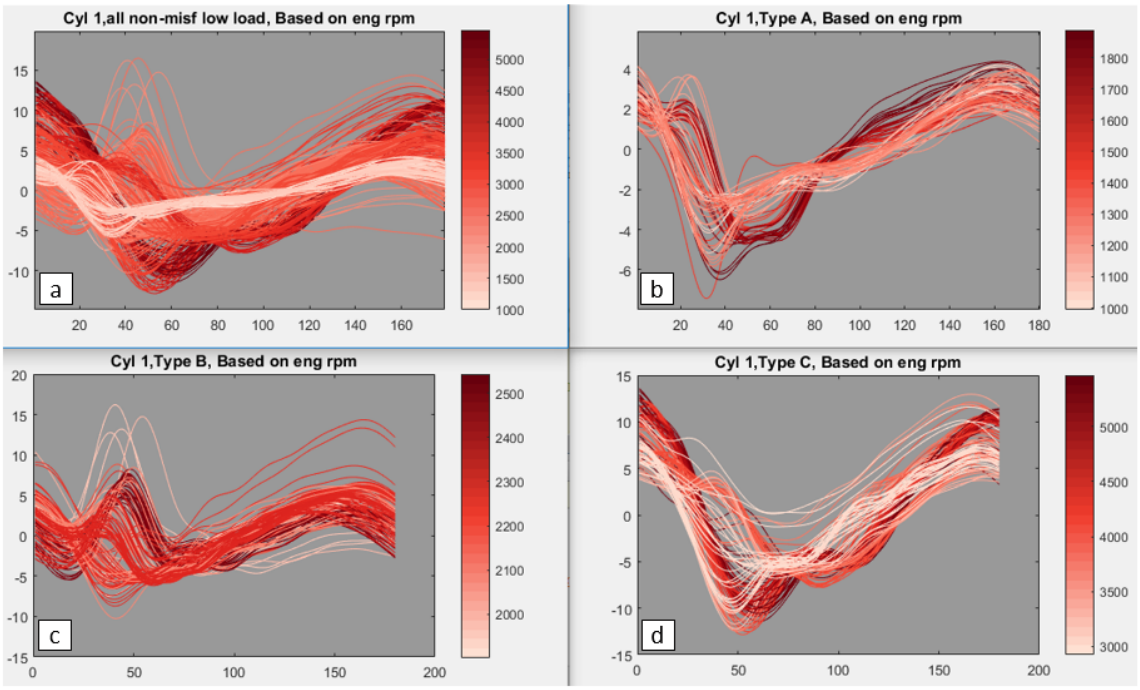


Figure 5.38: Segregation of Type II waveforms using engine speed

Figure 5.38a shows all the Type II waveforms color coded by RPM. In doing so, three RPM bands were observed, each having a peculiar signature. Each of these bands were segregated and shown in figures 5.38 b, 5.38 c and 5.38 d.

The three RPM bands were found to be as listed below

- Type A - Engine speed less than 1900 RPM
- Type B - Engine speed between 1900 RPM and 2800 RPM
- Type C - Engine speed greater than 2800 RPM

However, within the RPM bands a split distribution was observed. Thus the secondary classification of waveforms was further refined with a tertiary classification viz. Type A was further segregated using brake torque, Type B using spark timing and Type C using load. This lead to the classification diagram shown in Figure 5.39 that helped identify the various exhaust signatures.

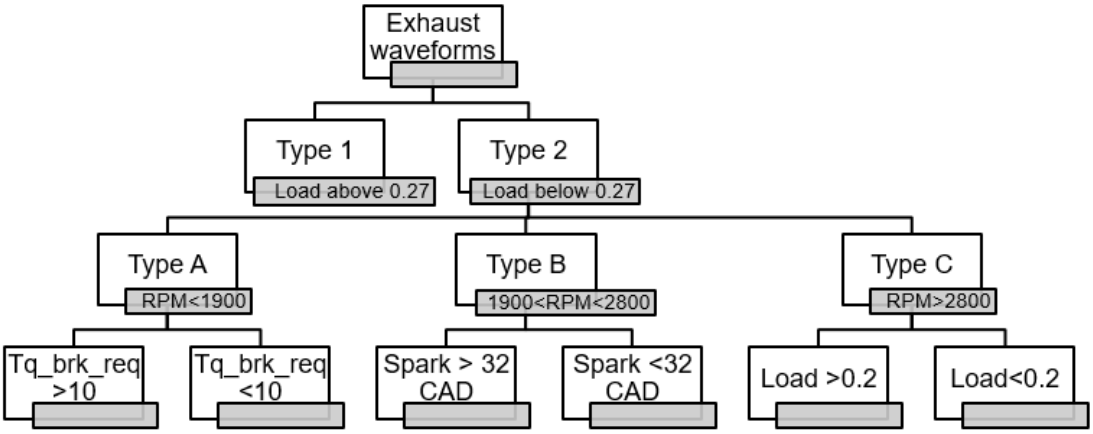


Figure 5.39: Segregation of Type II waveforms using engine speed

* $TqBrkreq$ signifies brake torque

Upon classifying the low load cycles, a correlation study was conducted to identify features in the respective exhaust waveforms to find correlations with combustion

metrics. Table 5.7 lists the various artifacts studied and color codes the results. Green indicates that a correlation of greater than 0.6 was obtained for the correlation of the parameter with the respective combustion metric and red indicates a poor correlation ($R<0.6$). In a few cases there were not enough cycles with a particular type of waveform to conduct a correlation study. A notable feature is that the Type C cycles showed a poor correlation with in-cylinder pressure despite a number of different artifacts being evaluated. In part this might be due to the fact that the Type C cycles mostly occurred during DFSO/Tip-out conditions as seen from Figure 5.40. The location of the other low load cycles are also mentioned in the same figure. The lack of combustion during the tip-out/DFSO portions of the cycle, results in minimal exhaust pressure rise during the blowdown portion of the cycle thereby making it hard to find signal artifacts that correlate with combustion metrics as there was a lack of combustion itself.

Table 5.7
Correlations studied for low load cycles

Level 1	Level 2	Level3	Parameter	Incyl Pmax. Amp	Incyl Pmax. CAD.
Type 2	Type A	High torque	Exh maxima Amp		
			Exh maxima Loc		
		Low torque	N/A		
	Type B	Advs spark	Exh maxima Amp		
			Exh maxima Loc		
		Rtd. spark	N/A		
	Type C		Exh minima Amp		
			Exh minima Loc		
			Area under curve		
			Exh P at EVO		
			First exh peak		
			First exh peak loc.		
			Dist. from EVO to exh. minima		

*N/A Not evaluated due to low number of cycles

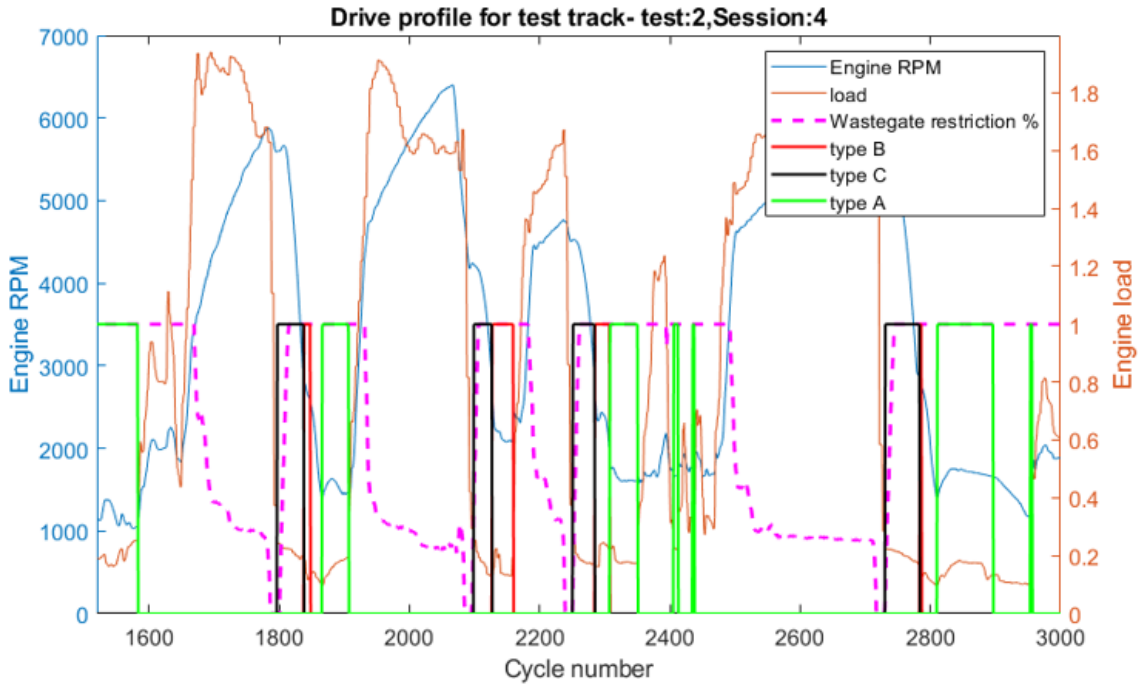


Figure 5.40: Location of various Type II waveforms over a drivecycle

The exhaust classification and correlation studies showed that the exhaust pressure was rich in information. However, using a signature/pattern recognition approach to extract features that correlate with combustion metrics was found to be a cumbersome process. Order tracking/extraction of the exhaust pressure provides the same information with lesser computational effort. Thus subsequently, this study used order tracking to extract information from the exhaust sensor.

5.2.3 Misfire detection under transient conditions

Previous studies by Prabhu [1] have shown that exhaust pressure signals can be used for misfire detection. However, a number of the tests in these studies were conducted under steady state conditions. This study conducted tests to identify misfire under transient on-road conditions. The Ford misfire generator software (refer Section 4.4) was used to generate misfires in a predefined sequence/pattern. The pattern used is listed in Table 5.8. An input of 83 causes the misfires to occur in a walking fashion.

Table 5.8
Pattern input to misfire software

1	83	255	0	1
0	X	X	X	X
0	X	X	X	X

The data collected over the transient test was processed to analyze misfire events on an individual cylinder basis. Figure 5.41 shows the exhaust waveforms corresponding to cylinder 1 color coded by load. The misfire and DFSO events/cycles are the waveforms with a significant trough. Although misfire and DFSO both result in a lack of combustion in the cylinder chamber, they are quite different in nature. DFSO is intentionally induced as part of the PCM strategy while misfire can occur due to a

number of causes including insufficient charge, improper spark or valve timings etc. Misfire detection is essential to ensure there is no power loss in the drivetrain and also to minimize hydrocarbon emissions.

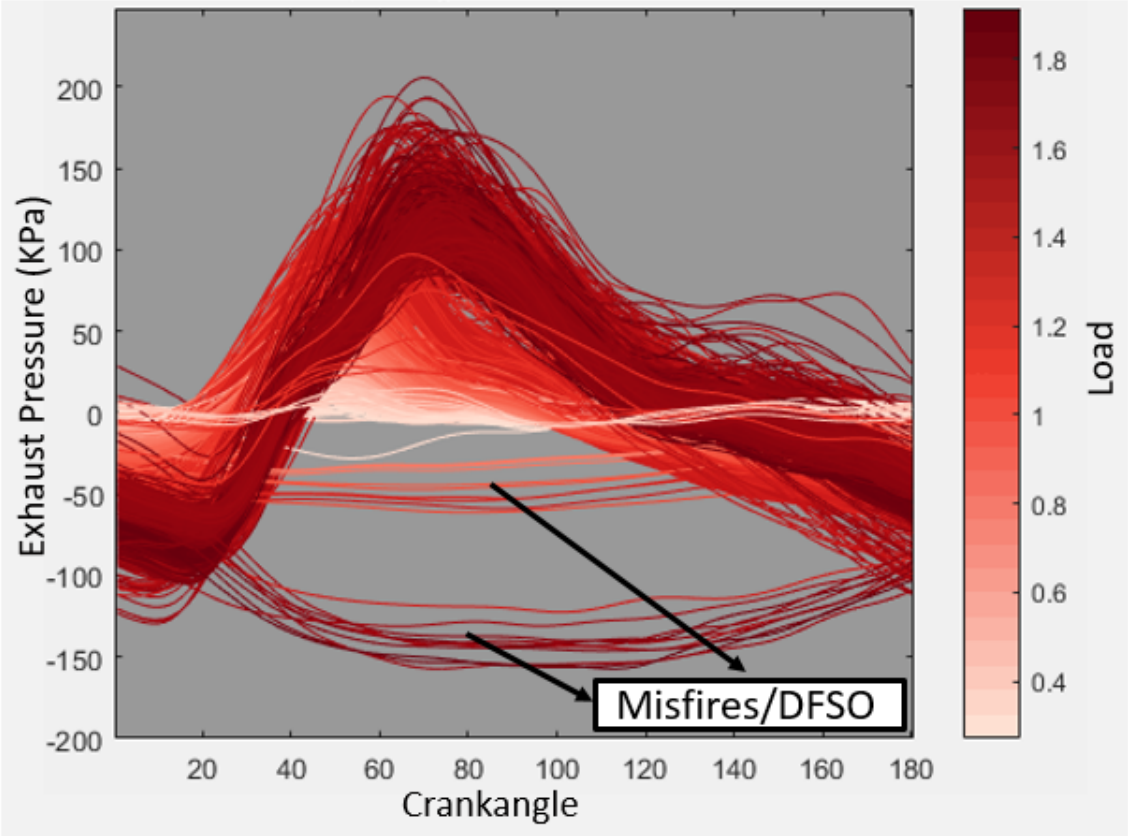


Figure 5.41: Misfire and DFSO detection

Using the amplitude of the exhaust troughs the misfire events could be distinguished from the normal combustion cycles. An algorithm was developed to extract the required signal features and detect the misfire events. Figure 5.42 shows the performance of the algorithm with respect to a particular cylinder. For the purpose of the analysis, cycles with IMEP lesser than zero were classified as misfires and cycles with

an IMEP lower than 46% of the mean IMEP were classified as partial burns. The thresholds used were based on work done by Cesario et al. [25]. Further, the DFSO events were differentiated from the misfire events using PCM parameters/commands.

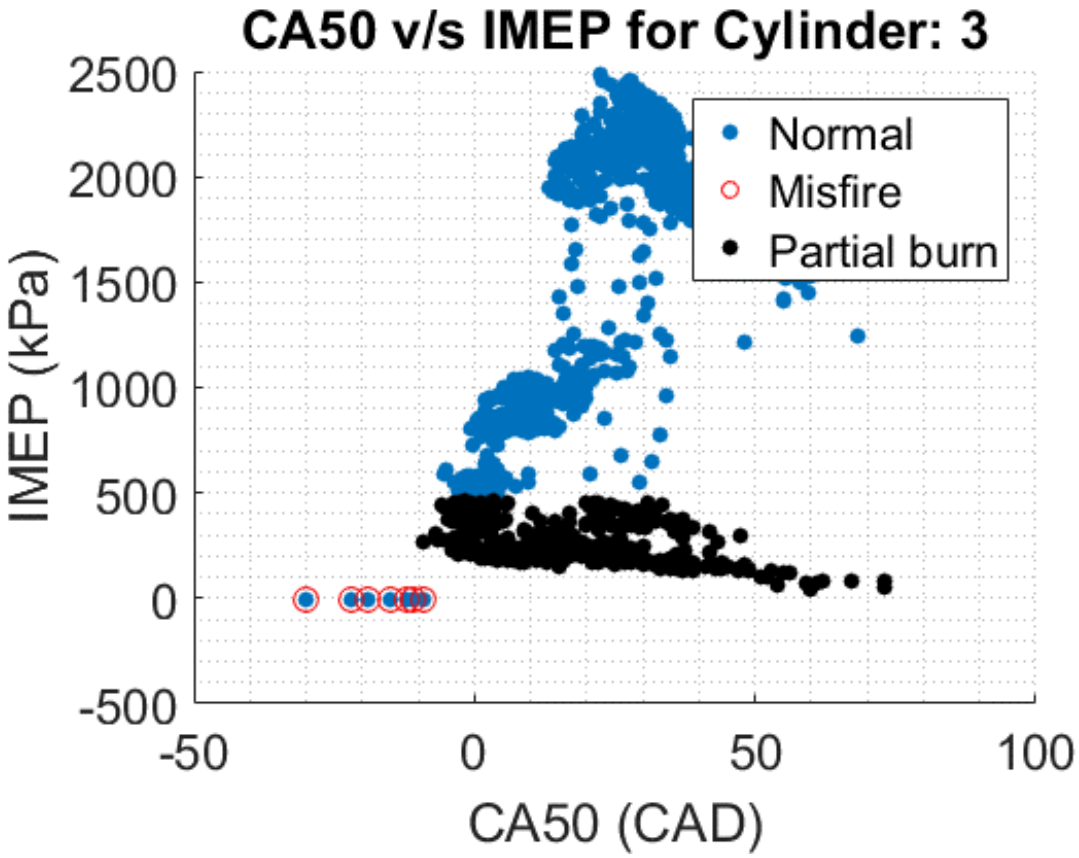


Figure 5.42: Misfire events as detected by algorithm

To verify if the algorithm developed was functioning as expected, two indices were developed called as Detectability index(DI) and False flag index (FFI). DI is a measure of how well the algorithm is able to identify misfires/DFSO. FFI indicates the number of cycles that were detected as a misfire/DFSO but were not actually a misfire/DFSO event. Mathematically the indices could be defined as:

$$DI = \frac{X \cap Y}{\text{number of actual misfires}}$$

$$FI = \frac{X - Y}{\text{number of detected misfires}}$$

where X represents a set of cycles detected as misfires by the algorithm and Y represents as set of cycles with actual misfire. Table 5.9 shows the performance of the algorithm in terms of the indices discussed. It can be seen that the algorithm was able to detect most of the misfire events. However there were a few cases when the false flag index was greater than a nominal value of 5%; possible causes for this are listed below

1. Thresholds used to classify or identify misfire/DFSO events would need to be fine tuned
2. On the top level, waveforms are classified based on load index. Since data is logged in sync mode, certain type I waveforms get classified as type II because of the time ATI takes to re-measure/refresh the load index
3. Exhaust waveform had bias (a drift in sensor output). This causes waveform classification to become harder

Table 5.9
Detection algorithm performance

DI	Value(%)	FFI	Value (%)
Cyl. 1	95.73	Cyl. 1	4.85
Cyl. 2	95.02	Cyl. 2	7.19
Cyl. 3	100.00	Cyl. 3	0.58
Cyl. 4	100	Cyl. 4	0.00

5.2.4 Order tracking

Previously in Section 5.2.2 it was shown that the use of exhaust signatures and feature extraction resulted in good correlation with combustion metrics for a number of operating conditions but the technique involved mapping a sizable dataspace and could be computationally intensive. Another technique utilized to extract exhasut pressure information was order tracking. The procedure to implement order tracking is mentioned in Section 4.2. The window applied to the exhaust signal is shown in Figure 5.43. Order analysis was conducted such that each block had 2880 data points (2-cycles, sampled at 0.5CAD); The Tukey window used was of the same length as well. The window was then shifted by 360 data points with respect to the previous block to cover the next 2880 points. This centers the window with the exhaust peak of a particular cylinder and every consecutive shift centers the window over the exhaust

signal of the next firing event. In Figure 5.43 for example the window (solid line) first aligns with exhaust peak of Cylinder 1 in the first block and exhaust peak of Cylinder 3 in the second block, where the window is showed with the dashed line. Thus by conducting the order analysis in the manner mentioned, order information on a cyclic and cylinder basis was obtained over the drivecycle. Further, an order resolution of 0.25 was obtained for the analysis, with a block size of 2880 points, sampled at an angle resolution of 0.5CAD.

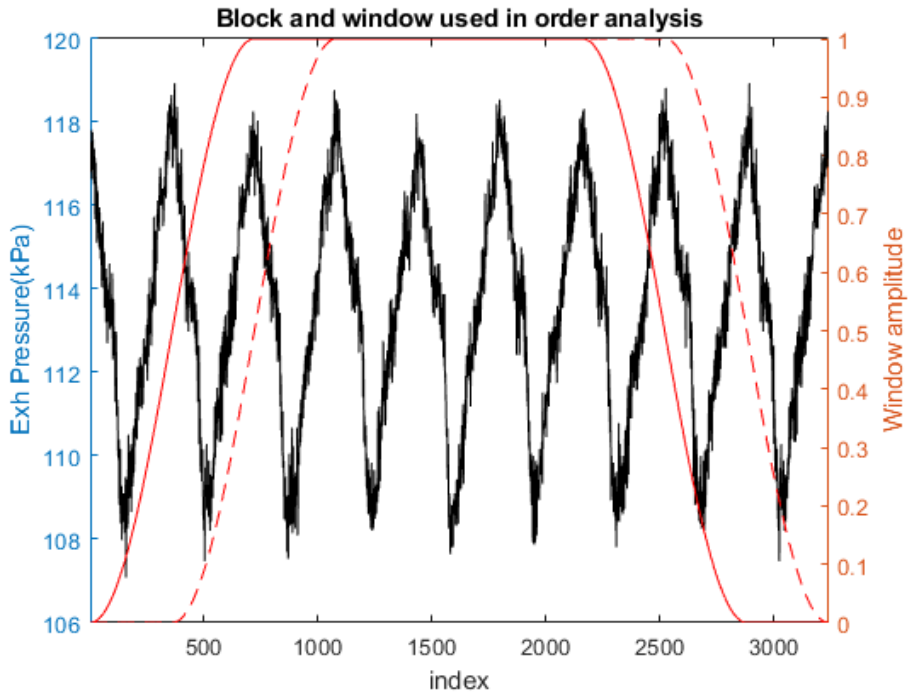


Figure 5.43: Window applied to signal for order analysis

Results on implementing the order tracking technique to a section of the data collected over the transient vehicle tests (Figure 5.44) is showcased in Figure 5.45. Figure 5.44 shows the engine speed, load, wastegate position and gear command on a cyclic basis,

for a section of the drive cycle from tip-in (cycle 2176) to tip-out and onto the low load section(cycle 2351). The location of the low load Type II cycles (i.e A, B C) are also highlighted.

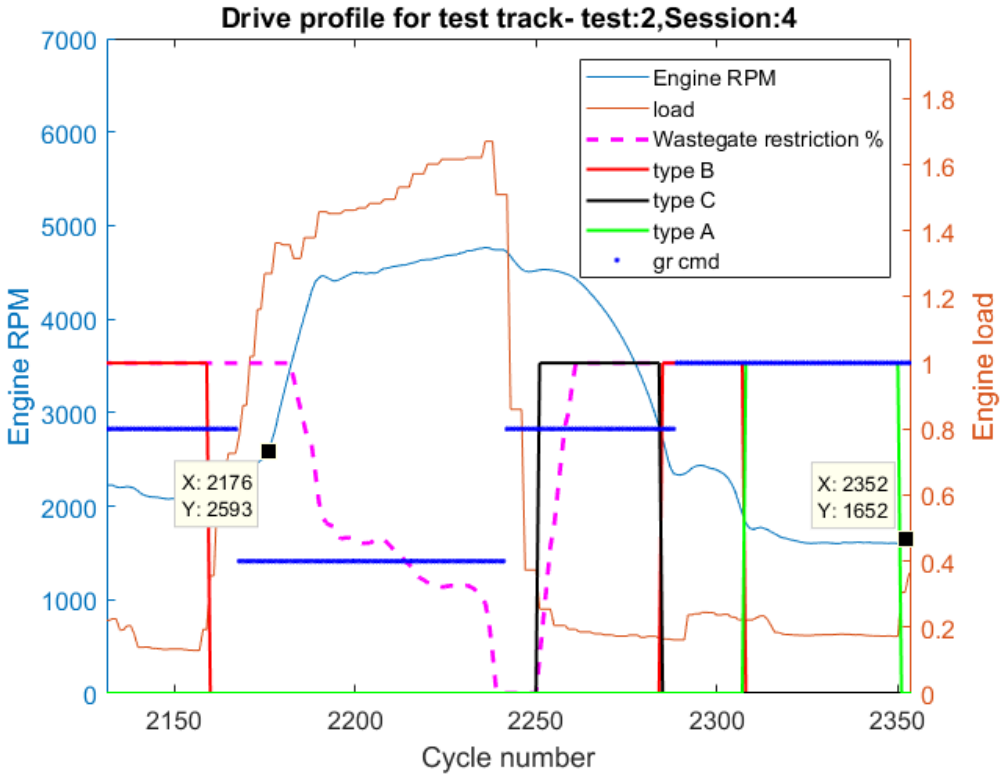


Figure 5.44: Cycles for which order analysis was conducted

Observing the waterfall plot in Figure 5.45 it can be seen that the orders present vary based on operating point. It is to be noted that in the waterfall plot the FFTs are stacked in firing order for the specified cycles. It is for this reason that though from tip-in to low load there are only 175 cycles, the event index in Figure 5.45 spans about 700 events i.e 175 cycles x 4 cylinders.

Further, the following inferences can be made from Figure 5.45 :

- Second order is consistent as it corresponds to firing order
- The harmonics of the second order viz. 4,6,8 etc are also seen to be present
- Order 7 and 9 seen to be excited as waste gate closes
- Odd orders, i.e Orders 1, 3 and 5 seen to have low amplitude during low load cycles
- Gear shift events coincide with drops in amplitude of orders. The two arrows in the waterfall plot indicate the 3-5 upshift and 5-6 upshift gear events.

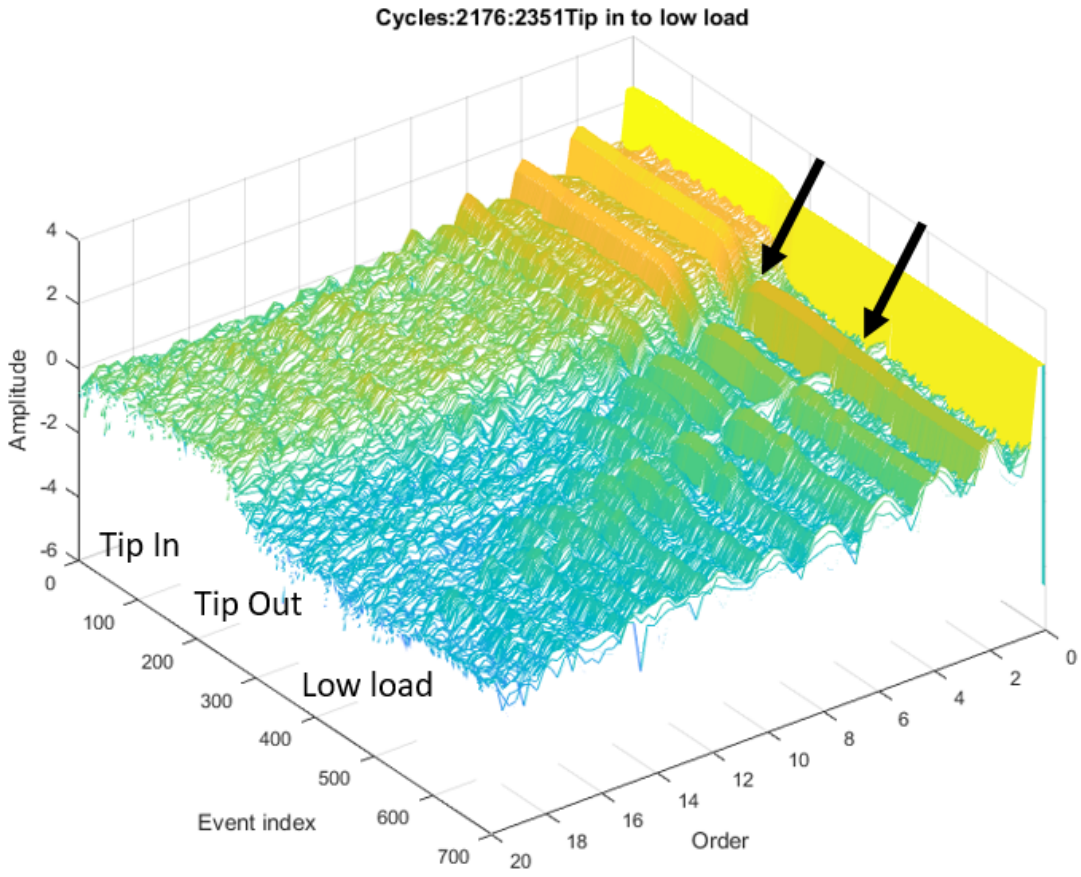


Figure 5.45: Waterfall plot of exhaust order analysis for cycles shown

Through the waterfall plot it was seen that the exhaust signal during transient vehicle tests, could be affected by numerous factors, the prominent noise element being gear shift events. Thus in an effort to simplify the analysis and neural network studies described later, transient tests were conducted on the engine dynamometer. The test cycles would mimic engine operation over a transient drivecycle, so the exhaust signal would be similar to that seen in a transient vehicle test but devoid of disturbances including gear shift events.

Figure 5.46 shows the order colormap for the exhaust signal over a transient cycle conducted in the engine test cell. A colormap is similar to a waterfall plot but is two dimensional. The color indicates amplitude; yellow being high and blue indicates low amplitude. The methodology to extract orders was the same as mentioned before. Further, the drivecycle of the test is also shown in the figure. The engine speed and load had to be scaled to ensure safe operation in laboratory conditions, i.e. engine speed was limited to 4500RPM and the normalized engine load was restricted to a range of 0.12 to 1.12.

The order colormap shows orders 2, 3 and 4 being excited during high speed-load conditions. Orders 4-6-8 etc are harmonics of the primary firing order 2 and are seen to be excited as well. For the neural network studies, orders 2, 3 and 4 were extracted and their amplitude and phase information was used in the NN studies for combustion metric estimation. Figure 5.47 shows an order cut of exhaust orders 2,

3 and 4. The figure shows the amplitude of the exhaust orders on a cycle-by-cycle basis. The figure is plotted on a logarithmic scale to accentuate signal features for visualization. Similar to the colormap, it can be seen that the amplitudes of the orders extracted increase during the high speed-load regime. The amplitude and phase of the exhaust orders extracted was later used as an input to the neural network for IMEP and CA50 estimation.

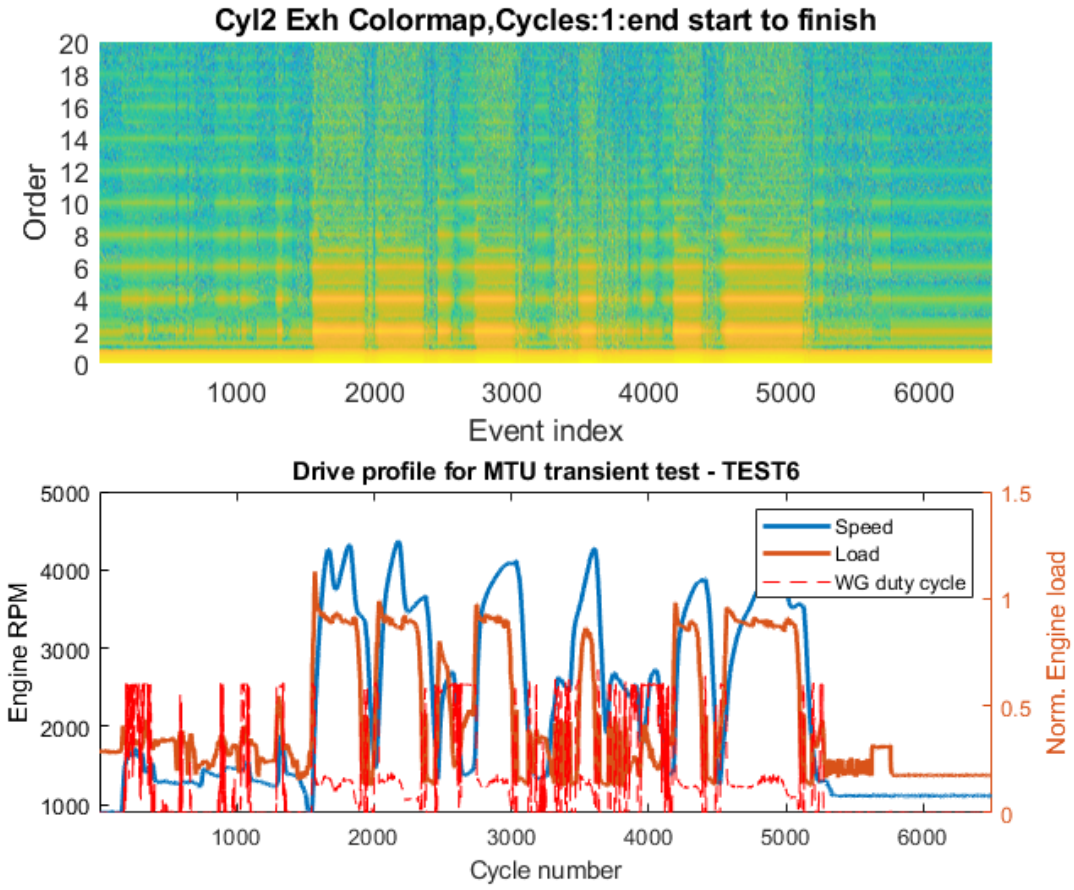


Figure 5.46: Order map of exhaust signal for transient engine testing

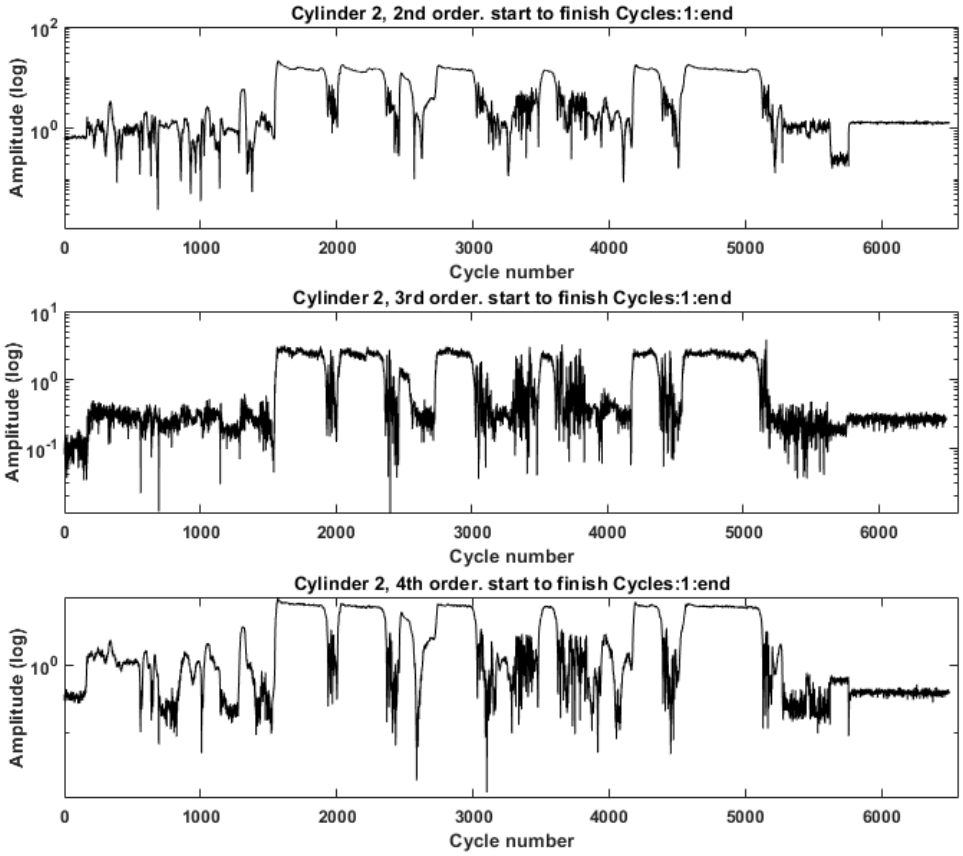


Figure 5.47: Order cut of exhaust signal for transient engine testing

5.2.5 Comparison of Omega and Kulite sensor

This study predominantly used the Kulite exhaust sensor, however the Omega exhaust sensor which was placed at a standoff could also be used for the same purpose as shown in this section. Table 5.10 lists the set of tests conducted to show that the Omega sensor contains the necessary order content to be used for estimation of combustion

metrics. The tests were conducted across various speed-load regimes.

Table 5.10
Test matrix for exhaust sensor evaluation

Test	**Speed**	**IMEP**	**Intake advance**	**Exhaust retard**	**CA50**
	RPM	kPa	deg	deg	deg
Test 1	1500	250	0	0	8
Test 2	1500	750	0	0	8
Test 3	3500	250	-35	35	8
Test 4	3500	750	-35	35	8

In all the tests mentioned in Table 5.10 the exhaust signals were sampled at an angle resolution of 0.5CAD. To generate order colormaps, the exhaust signals were divided into overlapping blocks of 2880 points and a Tukey window was applied to each block before conducting an FFT. Figure 5.48 to 5.53 shows the results of the FTT as colormaps to compare the Kulite and the Omega sensor output. It is to be noted that all the colormaps are color coded by the amplitude of the orders for a given cycle (event index). Figure 5.48 and 5.49 showcase the order colormap of the exhaust sensors for Test 1 viz low speed-load conditions. It can be seen that under low speed conditions the Omega sensor has significant content till about 12^{th} order in comparison to the Kulite sensor which shows content upto about 18^{th} order.

Conversely when the engine operates under high speed-load conditions similar to that of Test 4 it was seen that the Kulite sensor (Figure 5.50) showed that a number of odd orders were also excited which were not observed under low speed-load conditions. Figure 5.51 shows the orders present in the Omega signal for the same test and it can be seen that the sensor showed significant order content up till about 8^{th} order. The neural network studies use only orders 2, 3 and 4 as inputs to the neural network for IMEP and CA50 estimation. Thus through the results it was concluded that the Omega sensor had the necessary order content to replace the Kulite sensor.

Figure 5.52 shows a comparison of the linear spectra of the two exhaust sensor for low speed-load conditions. The graph is plotted on a logarithmic scale. Figure 5.53 shows a similar comparison but for high speed-load conditions. The figures also show the frequency corresponding to orders 2, 4 and 6. It can be seen that under low speed-load conditions the amplitude of orders are largely the same on both sensors up to about 325Hz but when the engine operates under high speed-load conditions, the amplitude of the orders on the Omega sensor (after about 150Hz) are seen to be significantly lower than that observed on the Kulite sensor.

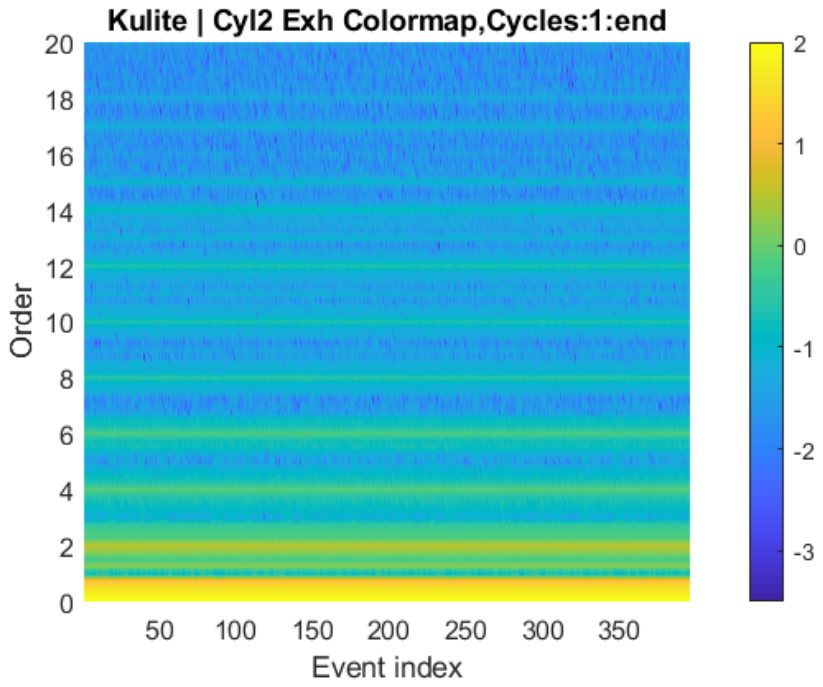


Figure 5.48: Order Colormap of Kulite sensor for Test 1

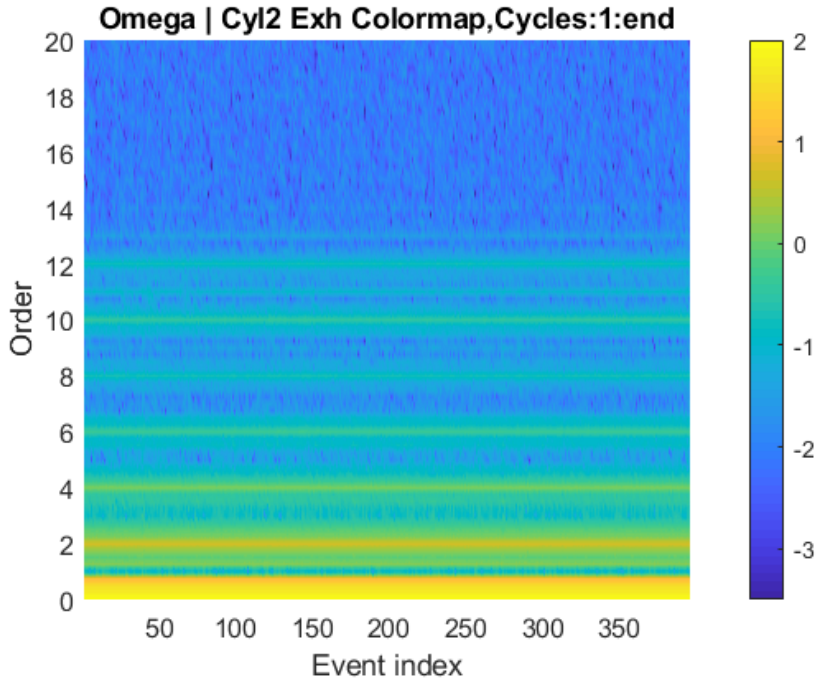


Figure 5.49: Order Colormap of Omega sensor for Test 1

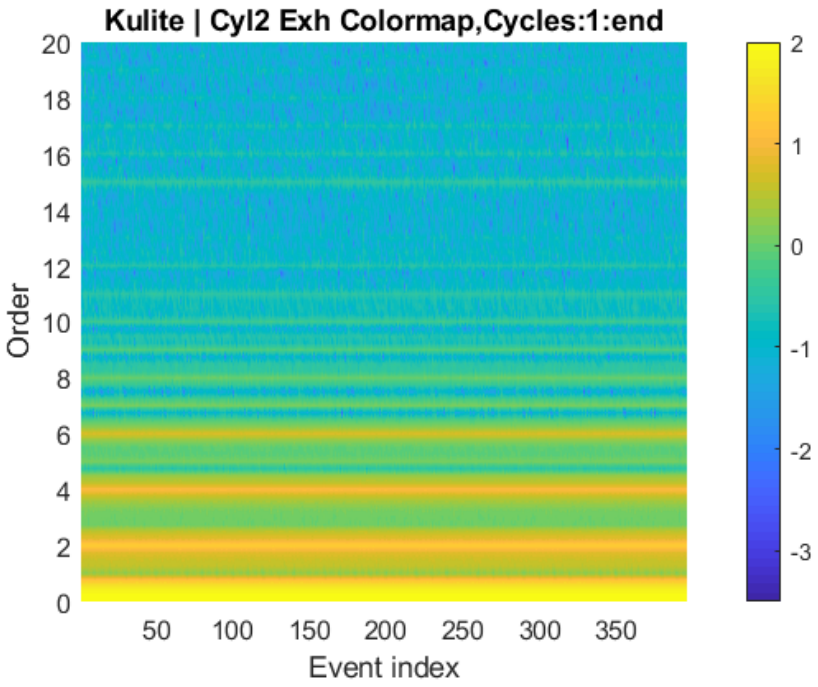


Figure 5.50: Order Colormap of Kulite sensor for Test 4

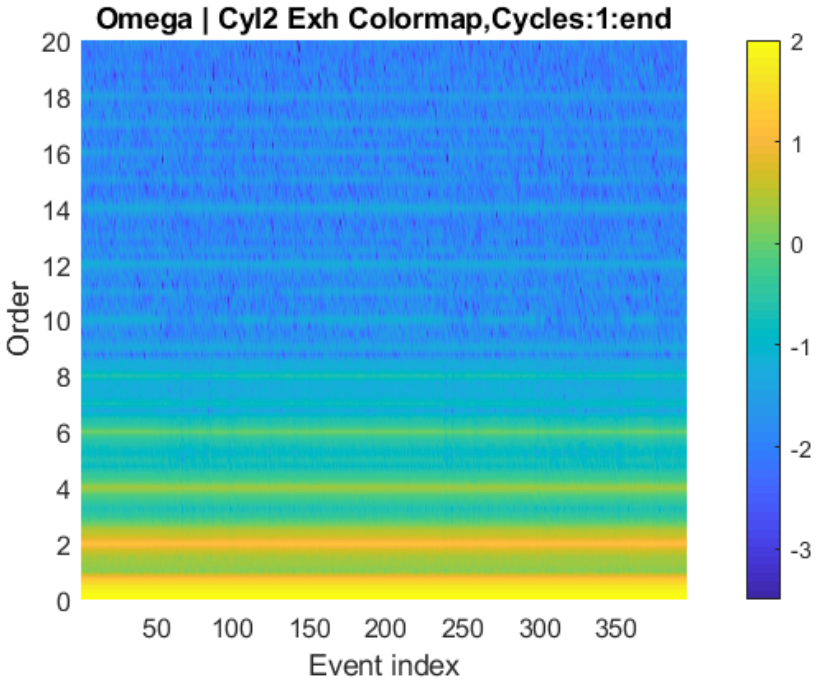


Figure 5.51: Order Colormap of Omega sensor for Test 4

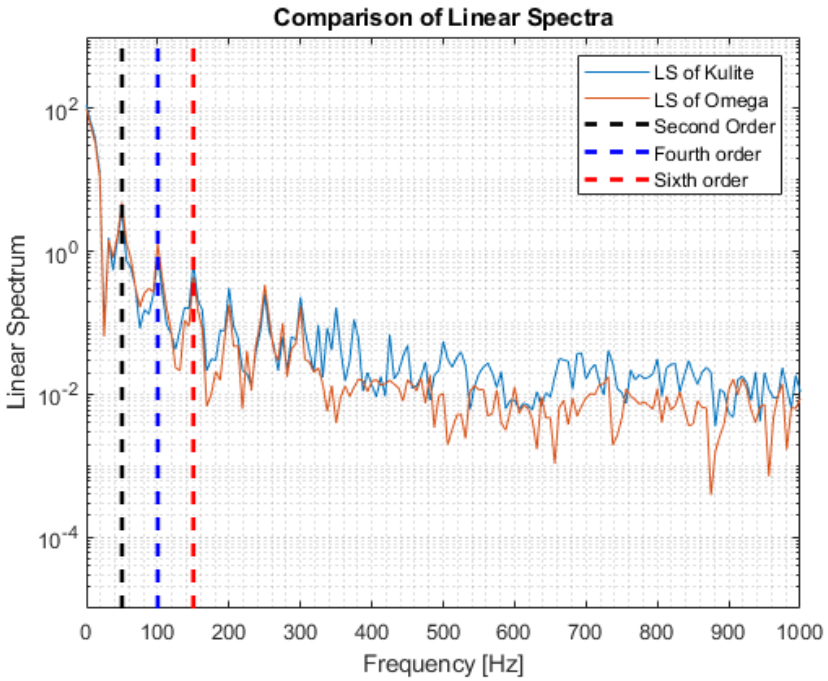


Figure 5.52: Comparison of Linear spectra for Test 1

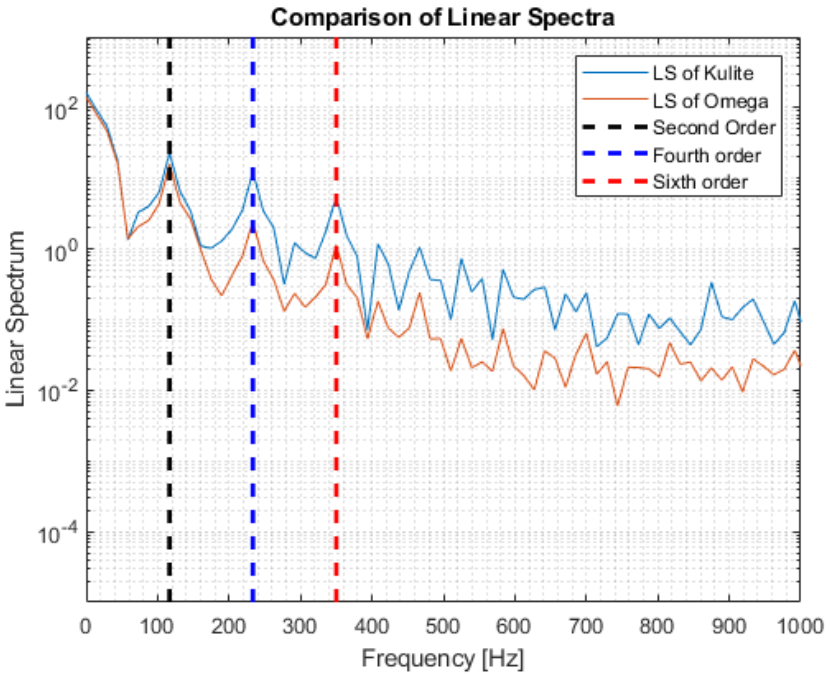


Figure 5.53: Comparison of Linear spectra for Test 4

The results show that the Omega sensor could be used as a alternative for the Kulite sensor however it is to be noted that since the Omega sensor is placed at a standoff, there would be time delay involved between the exhaust event and pressure measurement. Work conducted by Willimowski and Isermann [8] offers a methodology to calculate the time delay involved with the measurement. The method accounts for transport delay occurring due to the standoff and group delays introduced due to the anti-alias filter and pressure transmitter.

5.2.6 Conclusion on exhaust sensor studies

Under steady state conditions it was found that the exhaust minima could be used for misfire detection. Analyzing the exhaust pressure signal under transient conditions it was found that a number of engine parameters including load, speed etc could be used to identify exhaust signatures. Knowledge of the signatures helped extract features that could be used to obtain correlations with combustion metrics. However, the order tracking technique was later used to extract information from the exhaust sensor. Lastly, the standoff exhaust pressure sensor was seen to have the necessary order content to subsititute the sensor placed closer to the exhaust port.

5.3 Crank angle encoder

The output of the crank encoder installed in the engine is processed to identify the position of the crank/piston with respect to time. Thus the output signal is a time-stamp based on the desired angular resolution. This signal can be differentiated to obtain a crank speed signal. Fluctuations in the crank speed could offer helpful information for engine control and diagnostics.

5.3.1 Order extraction

Similar to the analysis conducted using the exhaust pressure signal; order analysis was conducted on the crank speed signal under transient conditions. A block size of 1440 points (i.e 1 cycle) sampled at 0.5 CAD resolution was used for the analysis. A simple Hanning window, the same size as the block, was applied to each signal block to avoid leakage when performing the Fourier transform. Further the internal timer resolution of the data acquisition system was set to about 5 microseconds. This was done so that there is sufficient resolution in the timestamp during high speed conditions.

Thus by processing the data as previously mentioned, an order map was generated as shown shown in Figure 5.54. The drivecycle followed is shown in the figure as

well. It can be seen that when the engine is operating in a high speed-load regime, orders 4 and 6 are excited. Here as well, second order refers to the firing order and orders 4, 6 etc. are harmonics of the firing order. The sixth order showed the highest amplitude when engine was operating in high speed load conditions. Similar to the exhaust pressure analysis orders 2, 3, 4 and 6 were extracted to be used as inputs to the neural network. Figure 5.55 shows the amplitude of the various orders previously mentioned on a cycle-by-cycle basis. The orders are plotted on a logarithmic scale.

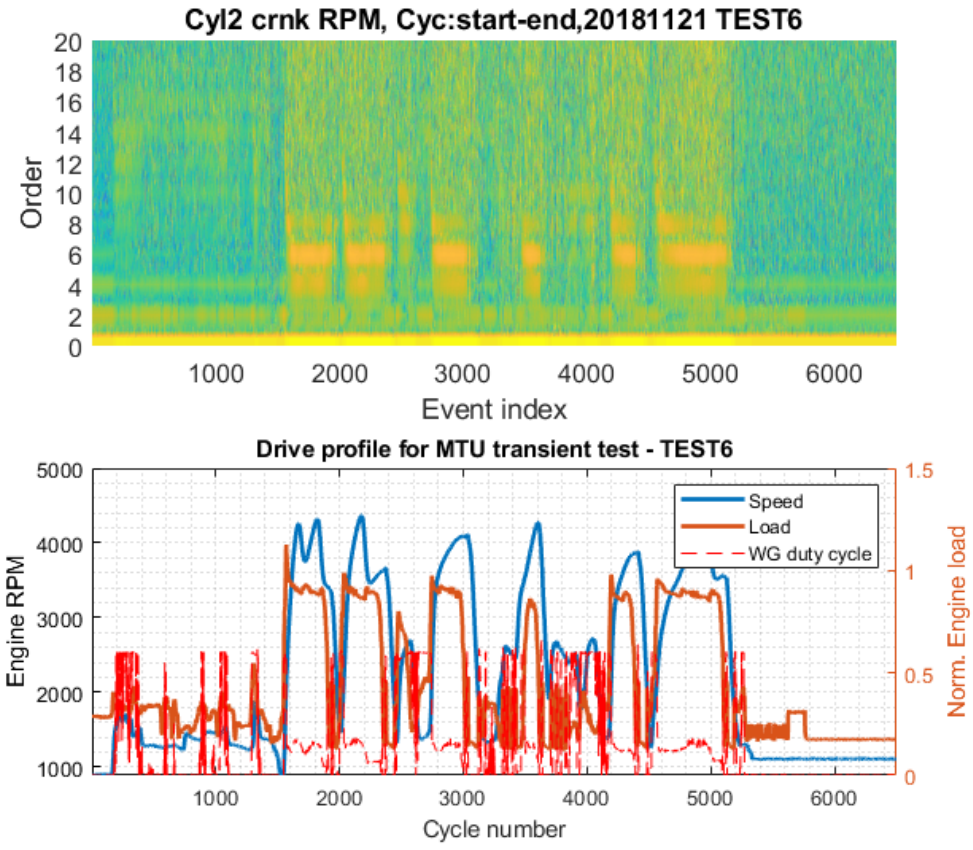


Figure 5.54: Order map of crank signal for transient engine testing

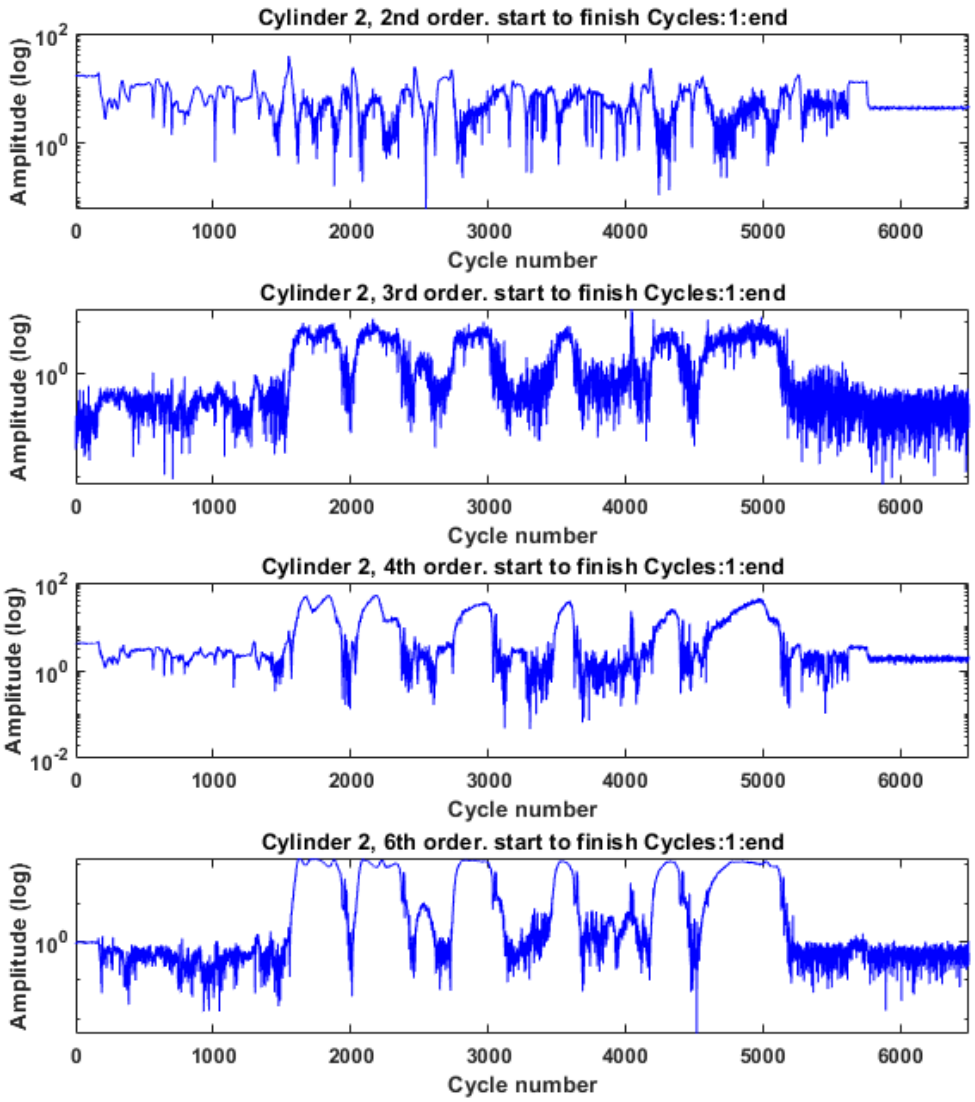


Figure 5.55: Order cut of crank signal for transient engine testing

5.4 Neural network

The previous sections showed how each of the sensors studied offered insights and correlations with combustion metrics. This section talks about how the information provided by the various sensors were consolidated to obtain estimates of combustion metrics including CA50 and IMEP.

In this study a neural network based approach was used to combiine the various sensor signals to obtain estimates of combustion metrics on a cyclical basis. Initially a matlab based simple feed-forward neural net was used. Later on, more advanced neural network architectures were also explored. Figure 5.56 shows the drivecycle over which the engine to obtain data for the neural network studies. Table 5.11 lists the initial set of inputs given to the neural network. Apart from the order amplitude and phase information from the exhaust and crank sensors, the flame locations and ion peak information from the ion sensors were also supplied. Besides the sensor inputs, engine operating parameters including engine speed, MAP, cam timing and spark were also provided as inputs to increase accuracy of network estimates. Initially the same network was used to estimate both the CA50 and IMEP.

Figure 5.57 offers a graphical representation of the network used and Table 5.12 provides insights about the settings of the neural network used in this phase of the study. Bayesian regularization was used as the training algorithm as it can result in

good generalization for difficult, small or noisy datasets even though it requires more processing time.

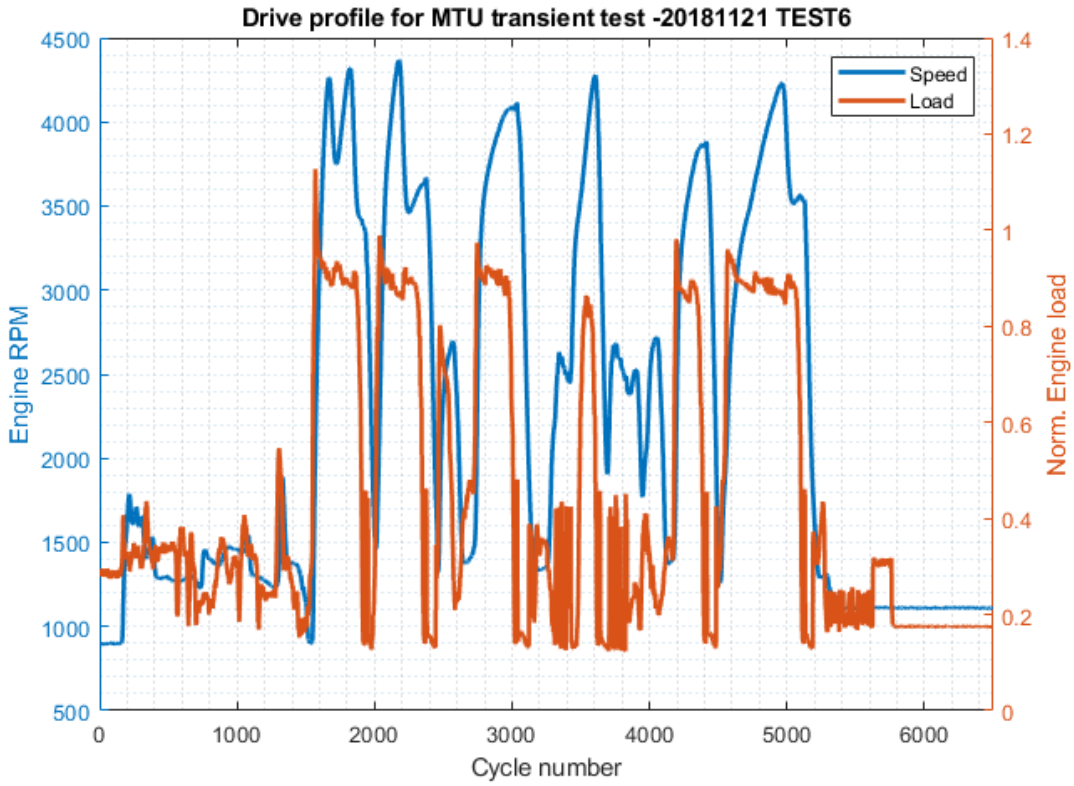


Figure 5.56: Transient cycle used in neural network studies

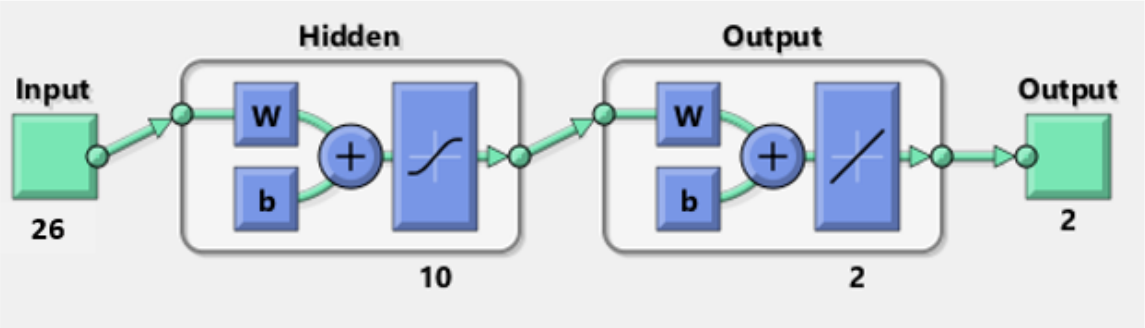


Figure 5.57: Feed forward neural network used for initial studies

Table 5.11
Inputs to feed forward network

ANN Inputs
Engine Speed
MAP
Spark Advance
Intake and Exhaust Cam phasing
Location of waste gate
Amplitude of 2nd, 3rd and 4th order of exhaust pressure
Phases of 2nd, 3rd and 4th order of exhaust pressure
Amplitude of 2nd, 3rd, 4th and 6th order of crank data
Phases of 2nd, 3rd, 4th and 6th order of crank data
Amplitude of first peak of standalone ion signal
Location of first peak of standalone ion signal
Amplitude of first peak of coil ion signal
Location of first peak of coil ion signal
Flamefront location detected by standalone ion probe

Table 5.12
Setting of feed forward neural network

Parameter	**Value**
Number of samples	6500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	26
Number of Outputs	2
Number of Hidden Layers	10
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

Figure 5.58 and 5.59 show the performance of the neural network in estimating the IMEP and CA50. Figure 5.58 shows the results on a cycle by cycle basis. The results show that the network estimates of IMEP follow the general trend of the actual IMEP values but show some deviation during the tip-in and tip-out zones. Similarly for the CA50 estimates the network estimates show significant deviation during deceleration and tip-out regions.

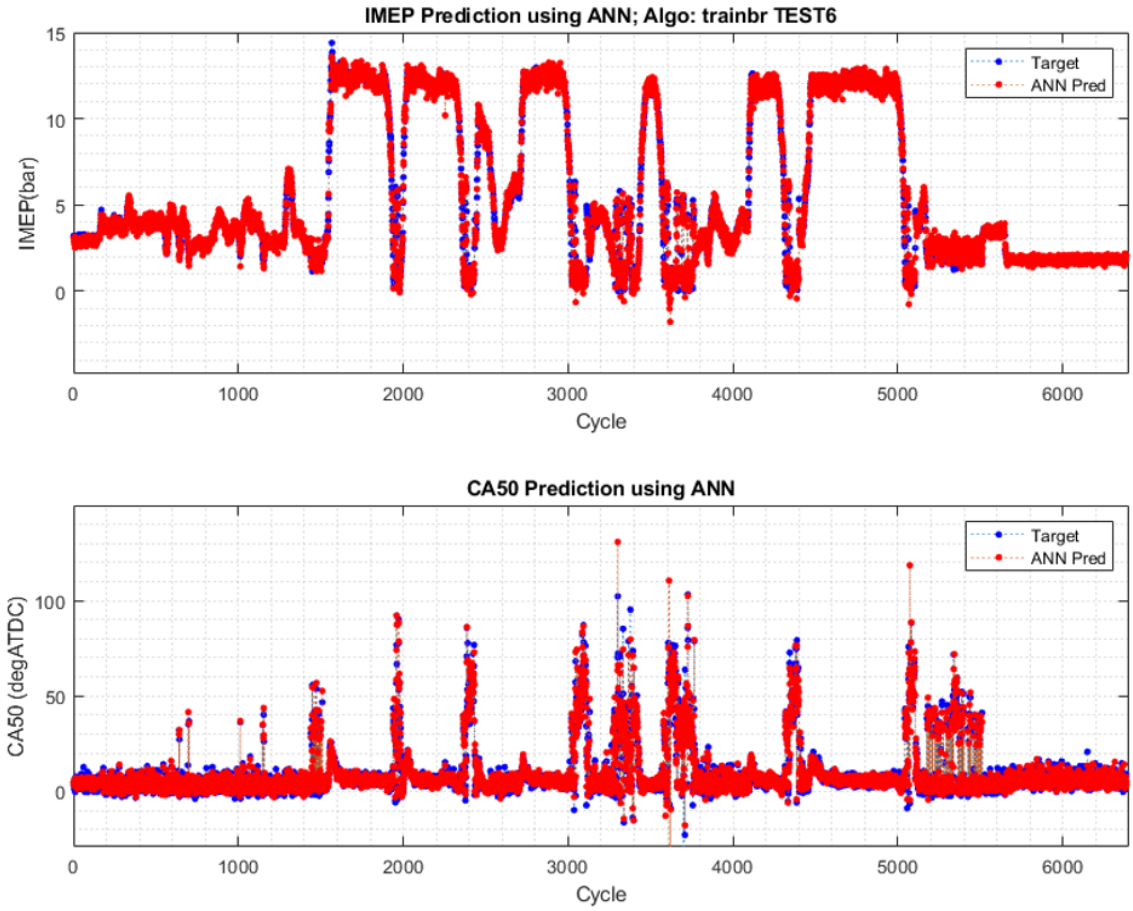


Figure 5.58: Actual and estimated combustion metrics for feed forward neural network

Figure 5.59 shows the same results but in a different format of actual value of combustion metric against network estimated value of combustion metric. The data is also color coded by engine speed to know if there was significant deviation at any particular engine speed or operation regime. However, the results showed that this was not the case.

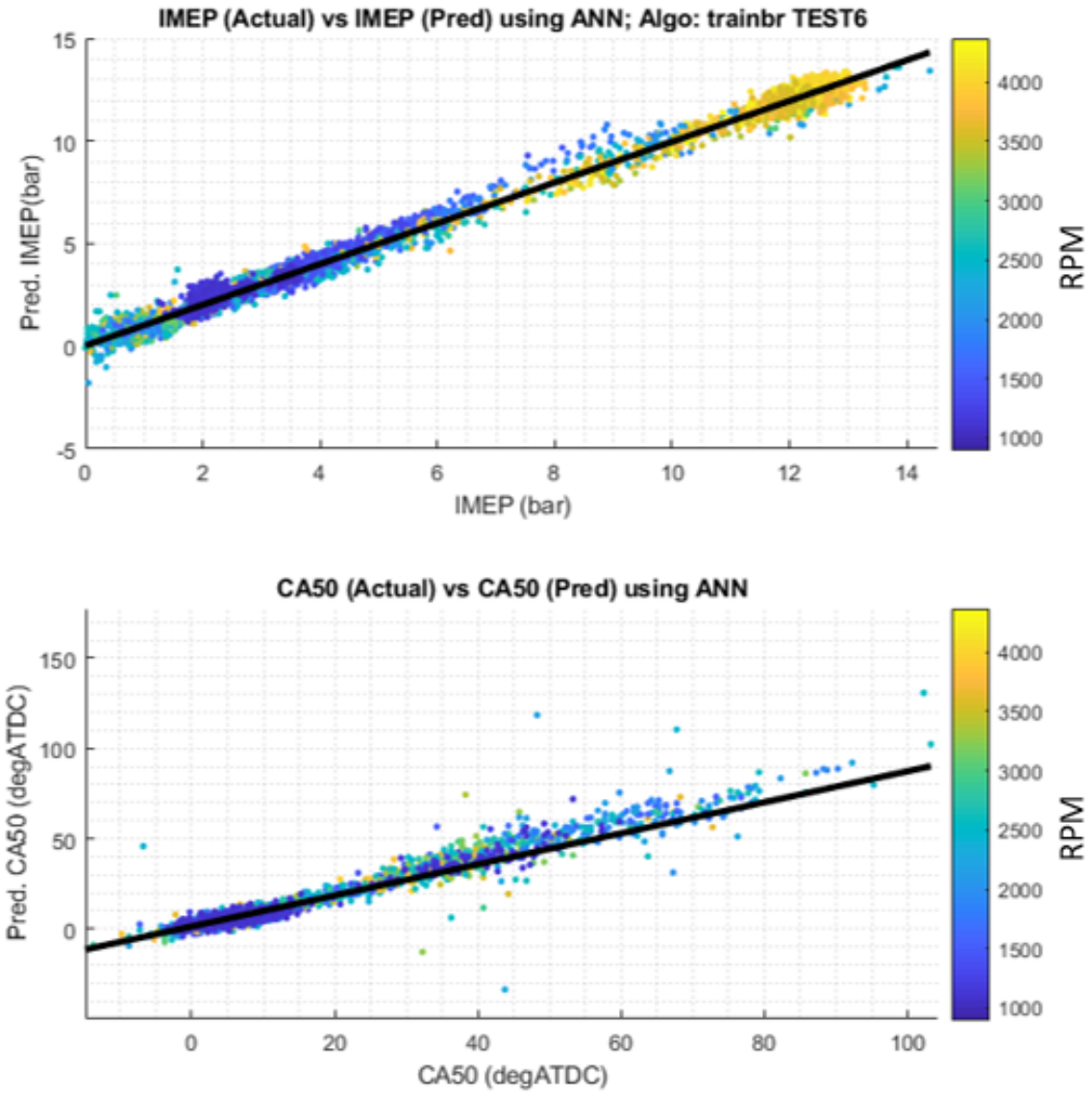


Figure 5.59: Actual against estimated combustion metrics for feed forward neural network

The observation in Figure 5.58 are corroborated by Figure 5.60 that shows the error in estimation. It can be seen that the error in IMEP estimates was highest during tip-in and tip-outs and the error in CA50 estimates was large during the tip-out portions

of the drivecycle.

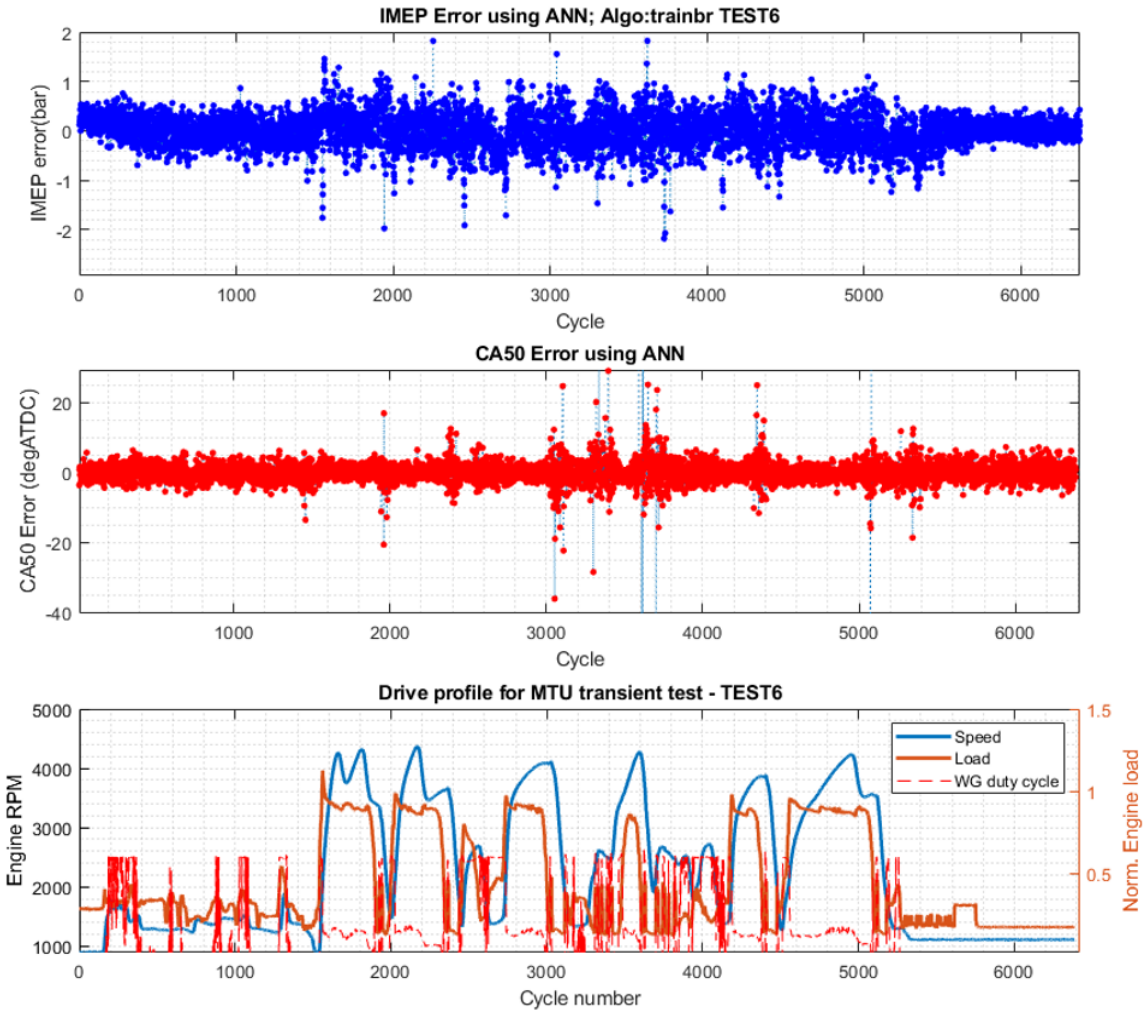


Figure 5.60: Error analysis of simple feed forward network

Figure 5.61 shows the distribution of errors in estimation of CA50 and IMEP. Using the simple feed forward neural network, it was observed that a majority of the cycles had IMEP estimates that were within 0.4 bar of the actual value and CA50 estimates that were within 3 CAD.

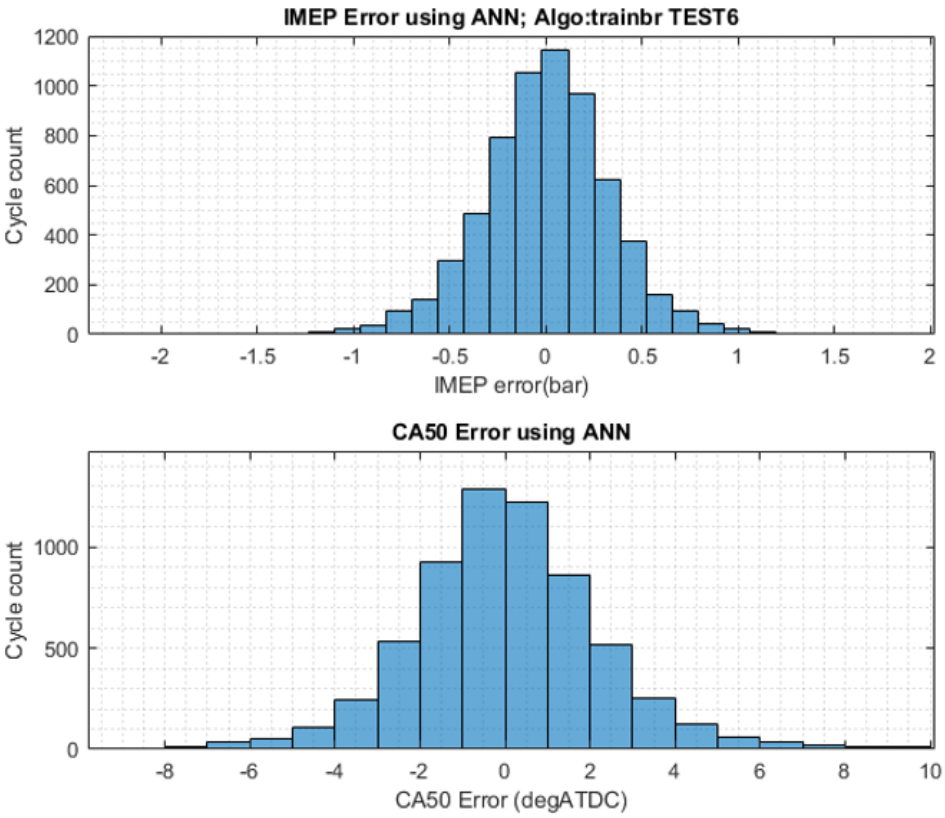


Figure 5.61: Distribution of error of simple feed forward network

The initial results proved promising but had to be improved. The initial choice was to retain the same network architecture but to improve the accuracy of estimates. To achieve this, feature normalization and principal component analysis was conducted.

Further in terms of the inputs used it was found that the coil integrated ion signal was offering erroneous results in certain cases. The algorithm used to process coil ion signals was detecting the false peak as first ion peak in certain cases. Despite trying various approaches to identify the corect peak in all cases, there were shortcomings. Thus it was decided to drop the coil ion signal from the list of inputs. Additionally, to

offer the network with inputs regarding fuel concentration, the mass of fuel injected into the cylinder on a cyclic basis was also used as an input.

5.4.1 Feature scaling and PCA

Feature scaling or featuring normalization is the process of altering or scaling the neural network inputs (i.e features) such that they all have a similar range . This process helps make the network equally sensitive to all inputs. The method to conduct feature scaling is mentioned in Section 4.5.1.

Using the simple feed forward network, it was observed that the network used 26 different inputs. Considering the training set had over 4800 cycles (samples), the dimensionality of the input matrix was quite large. Thus there was a need to make the analysis less computationally expensive without compromising the accuracy of estimation. One method to achieve this was thorough PCA. Thus in an effort to improve accuracy of estimation, feature scaling and PCA were implemented. Table 5.13 shows the inputs used in the modified feed forward neural network. There were 24 inputs but using PCA the number of inputs were reduced to 10. Figure 5.62 shows the contribution of each of the principal components and the marker indicates the number of components needed to retain 90% of the variance present in the data, which came out to be 10 components in the study conducted.

Table 5.13
Inputs to modified feed forward network

ANN Inputs
Engine Speed
MAP
Spark Advance
Mass of fuel injected
Intake and Exhaust Cam phasing
Location of waste gate
Amplitude of 2nd, 3rd and 4th order of exhaust pressure
Phases of 2nd, 3rd and 4th order of exhaust pressure
Amplitude of 2nd, 3rd, 4th and 6th order of crank data
Phases of 2nd, 3rd, 4th and 6th order of crank data
Amplitude of first peak of standalone ion signal
Location of first peak of standalone ion signal
Flamefront location detected by standalone ion probe

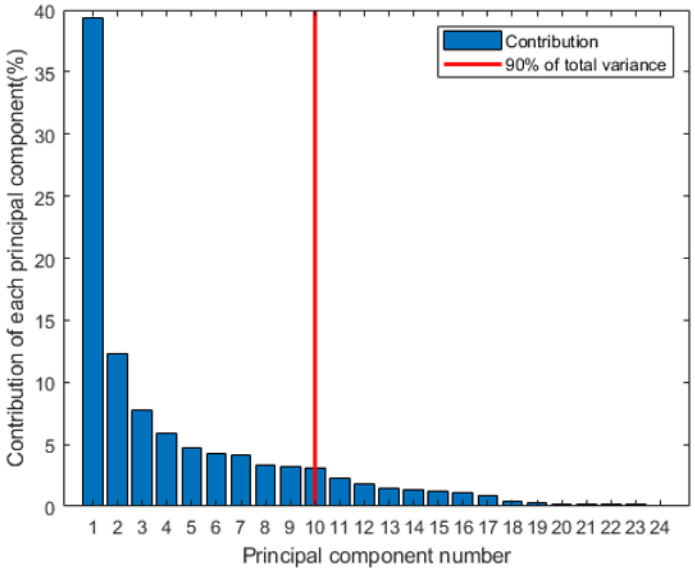


Figure 5.62: Contribution of each principal component

Figure 5.63 shows the network used to conduct the analysis and specifications regarding the network are mentioned in Table 5.14.

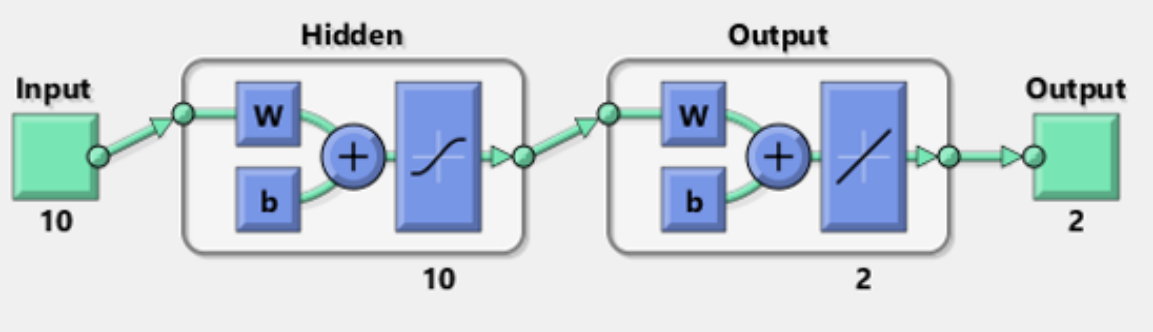


Figure 5.63: Feed forward neural network with feature scaling and PCA

Table 5.14
Setting of modified feed forward neural network

Parameter	**Value**
Number of samples	6500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs (after PCA)	10
Number of Outputs	2
Number of Hidden Layers	10
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

The results of implementing feature scaling and PCA are shown in Figure 5.64 to 5.67. Figure 5.64 shows that the estimates of modified feed forward network followed the same general trend as the previous network. However, the network was still unable to offer accurate estimates during highly transient operation. Further it can be seen

from Figure 5.65 that the deviation between the actual and estimated CA50 and IMEP was higher than the previous network. the deviation was spread across all load and speed conditions. It is to be noted that the data in Figure 5.65 is color coded by engine speed.

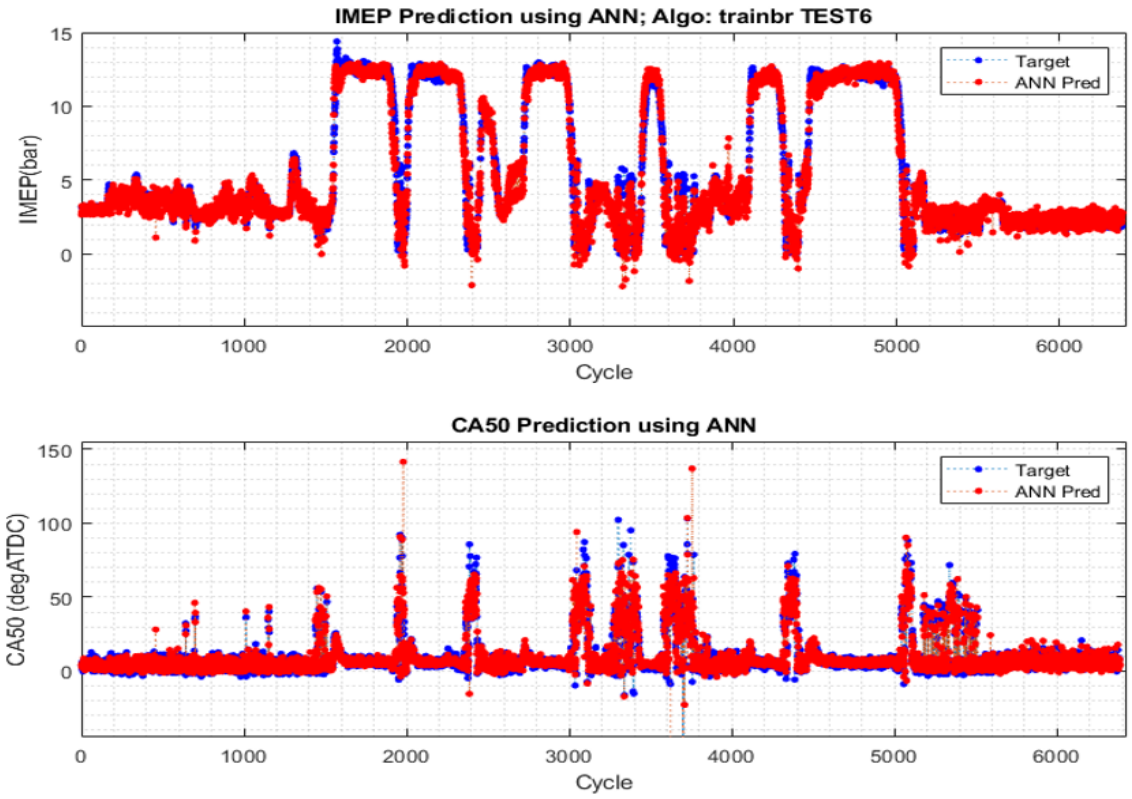


Figure 5.64: Actual and estimated combustion metrics for modified feed forward neural network

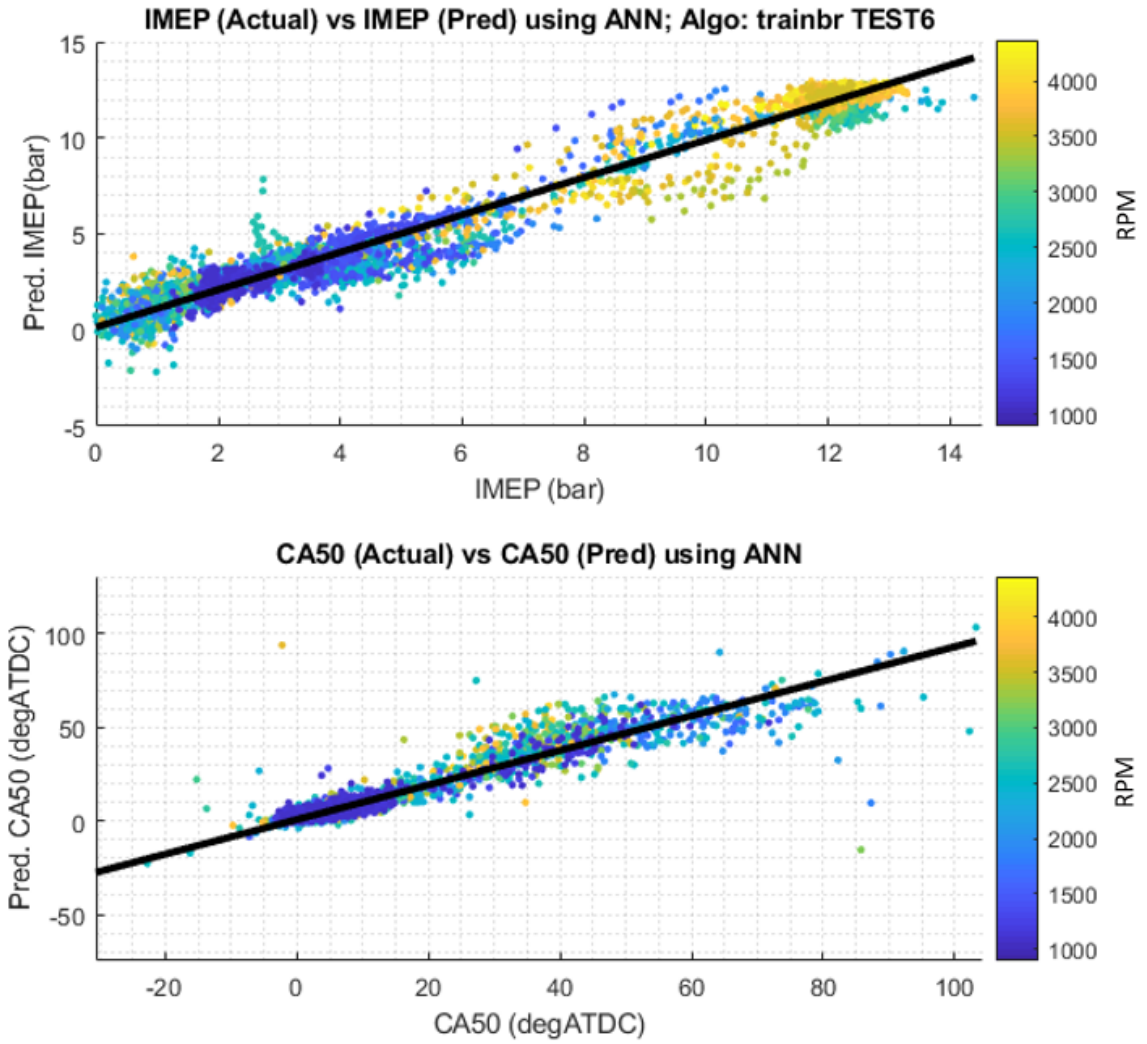


Figure 5.65: Actual against estimated combustion metrics for modified feed forward neural network

Figure 5.66 shows the error in estimation of the combustion metrics on a cycle by cycle basis. it can be seen that despite the implementation of the feature scaling, the network was unable to improve accuracy of estimates, especially under heavy transience conditions.

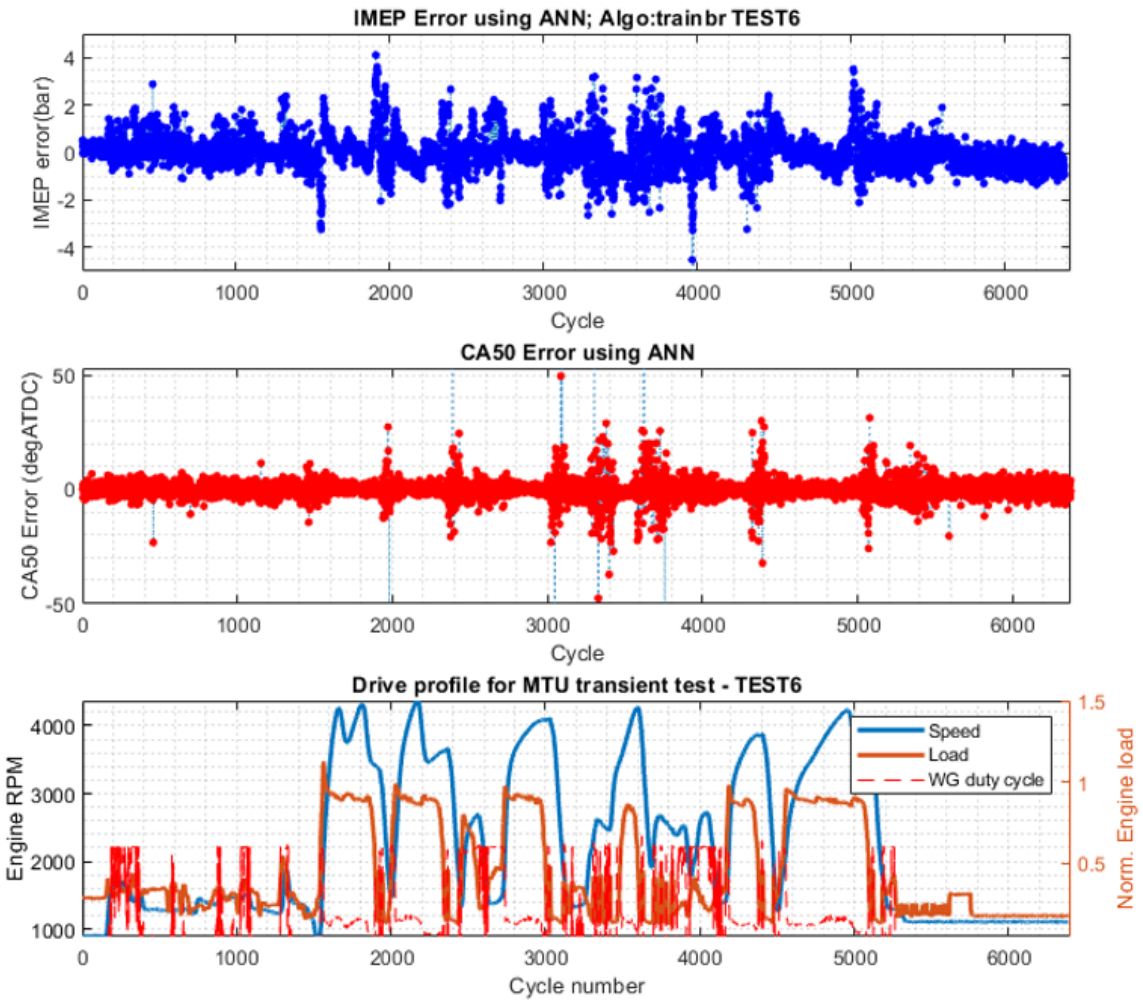


Figure 5.66: Error analysis of modified feed forward network

Figure 5.67 shows the distribution of errors in estimation of CA50 and IMEP. Using the modified feed forward neural network the standard deviation in IMEP was seen to be about 0.6bar and the standard deviation in CA50 was about 5CAD. Thus though implementation of PCA reduced input dimensionality it caused a significant deterioration of network accuracy.

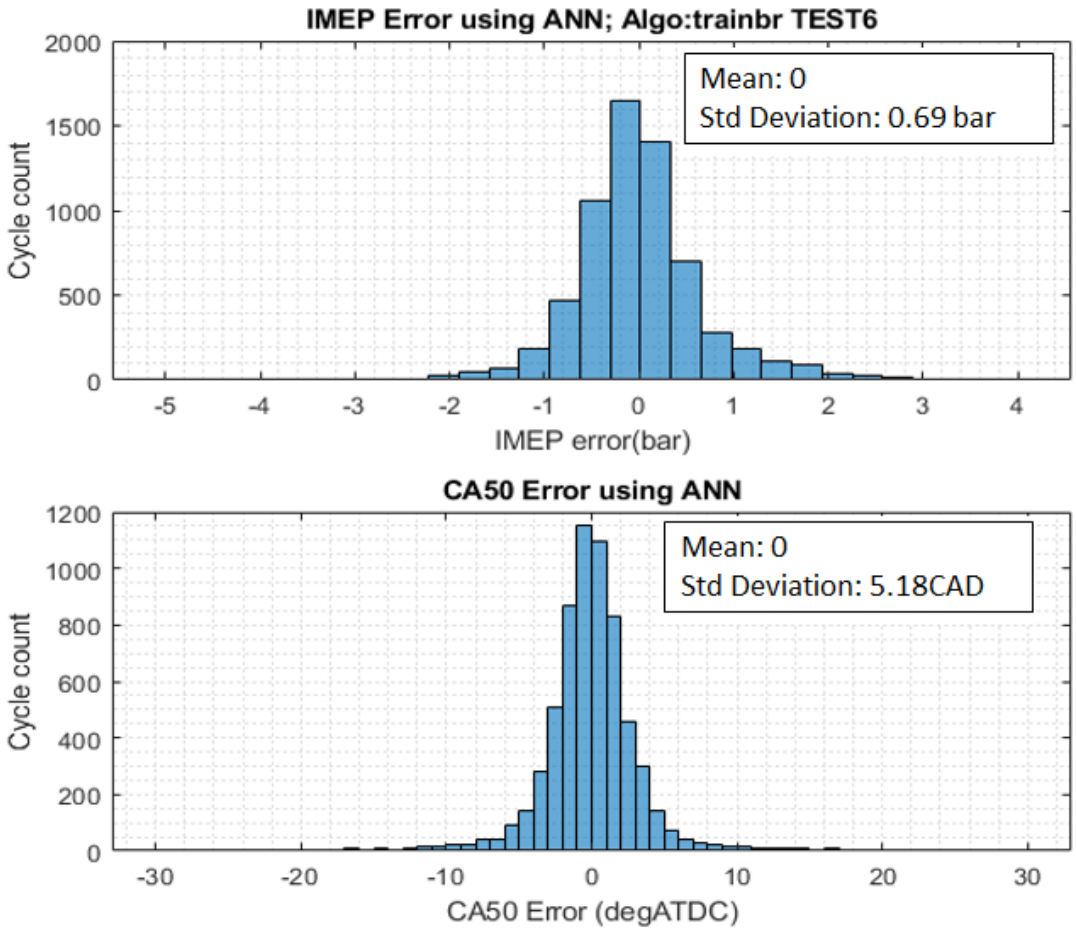


Figure 5.67: Distribution of error of modified feed forward network

The modified feed forward network was unable to improve the performance obtained. Further, the network never used any information of the prior cycle to estimate combustion metrics of a given cycle i.e it was not recursive in nature. In reality, the sort of combustion occurring in a given cycle is significantly affected by the performance of the previous cycle. Thus it was decided to change the architecture of neural networks used to include a recursive element.

5.4.2 Recursive neural net

As mentioned previously, in an effort to enhance the performance of the neural network, a recursive approach was needed. Two neural net architectures were evaluated for this application viz. nonlinear auto-regressive with external input (NARX) network and a recursive neural network (RNN). Figure 5.68 and 5.69 show a visualization of the two architectures. The same inputs as mentioned in Table 5.13 were used in the recursive studies too.

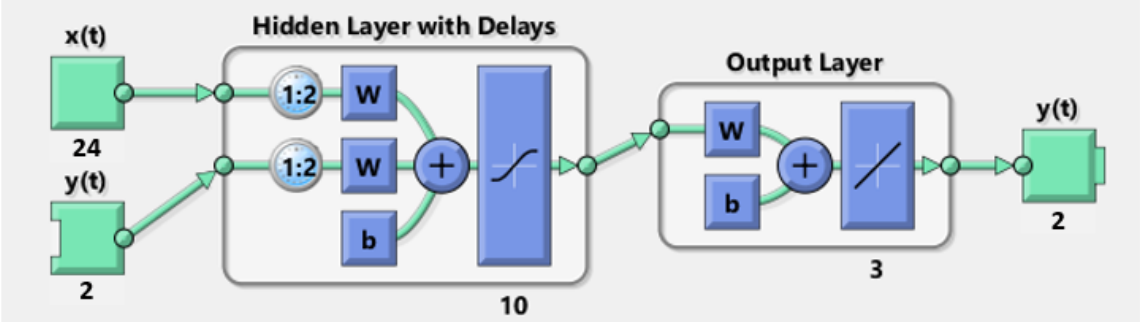


Figure 5.68: Nonlinear auto-regressive with external input (NARX) network

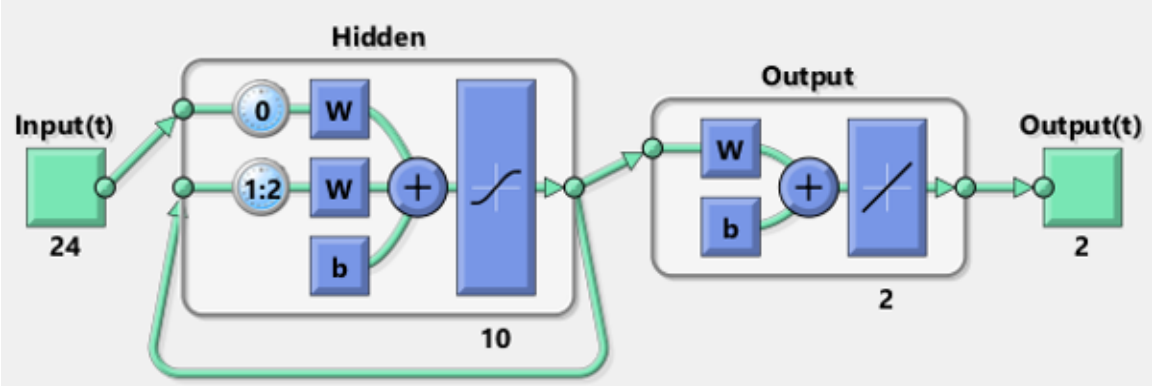


Figure 5.69: Recursive neural network (RNN)

The NARX network showed large errors in estimation. This might be due to the fact that a NARX network uses previous cycle data of all inputs, including inputs whose historical data might not affect a given cycle's estimates. Thus in the interest of maintaining brevity, results pertaining to NARX are not discussed in this study.

The recursive neural net however used IMEP and CA50 estimates of cycle 'i-1' and inputs of cycle 'i' to estimate the CA50 and IMEP of cycle 'i'. This architecture was more representative of capturing engine combustion dynamics. Table 5.15 presents the specifications of the network used.

Table 5.15
Settings of recursive neural network

Parameter	**Value**
Number of samples	6500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	24
Number of Outputs	2
Number of Hidden Layers	10
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

Figure 5.70 and 5.71 show the results of the RNN for estimation of CA50 and IMEP. From the figures it can be seen that the network was able to follow the general trend however, in terms of the accuracy of estimation, the network had to be improved especially in the estimation of CA50.

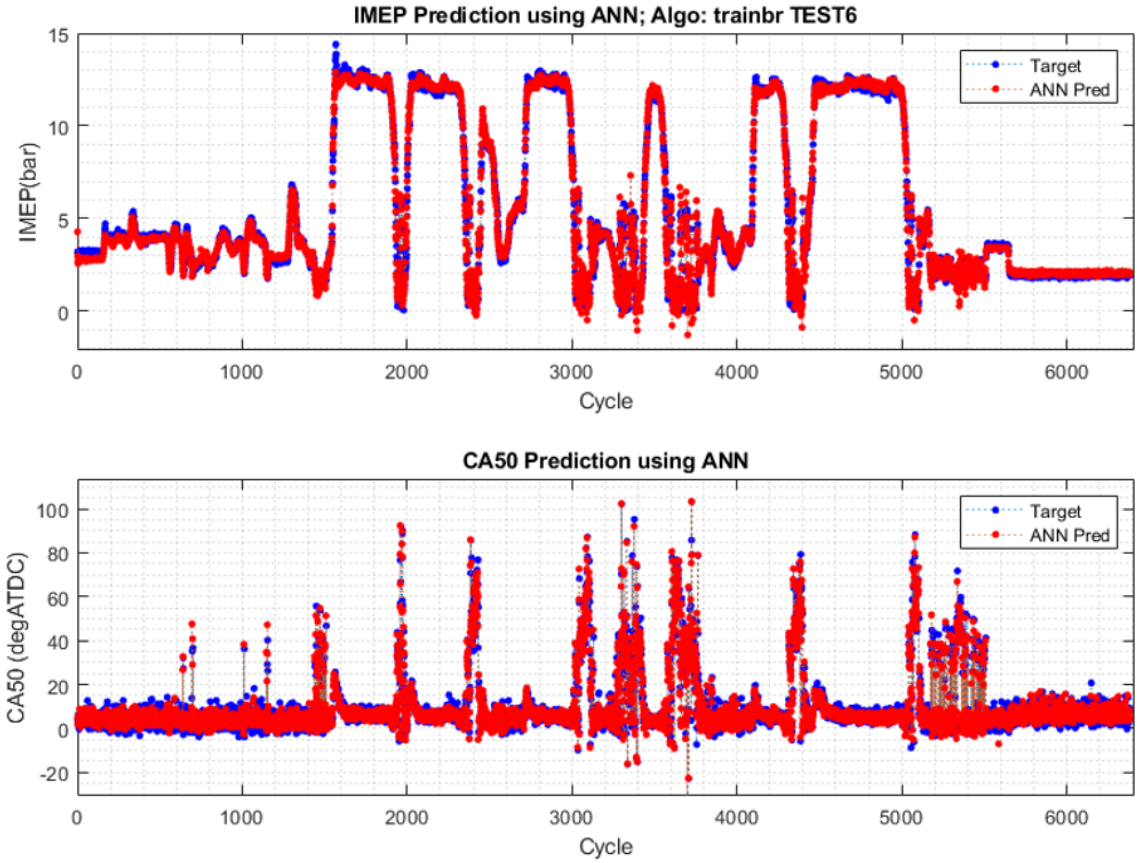


Figure 5.70: Actual and estimated combustion metrics for recursive neural network

From Figure 5.71 it can be seen that there was significant deviation in network CA50 estimates, especially at low speed conditions (marked by the dark blue points). The

IMEP estimates however showed much better results in comparison to the CA50 estimates.

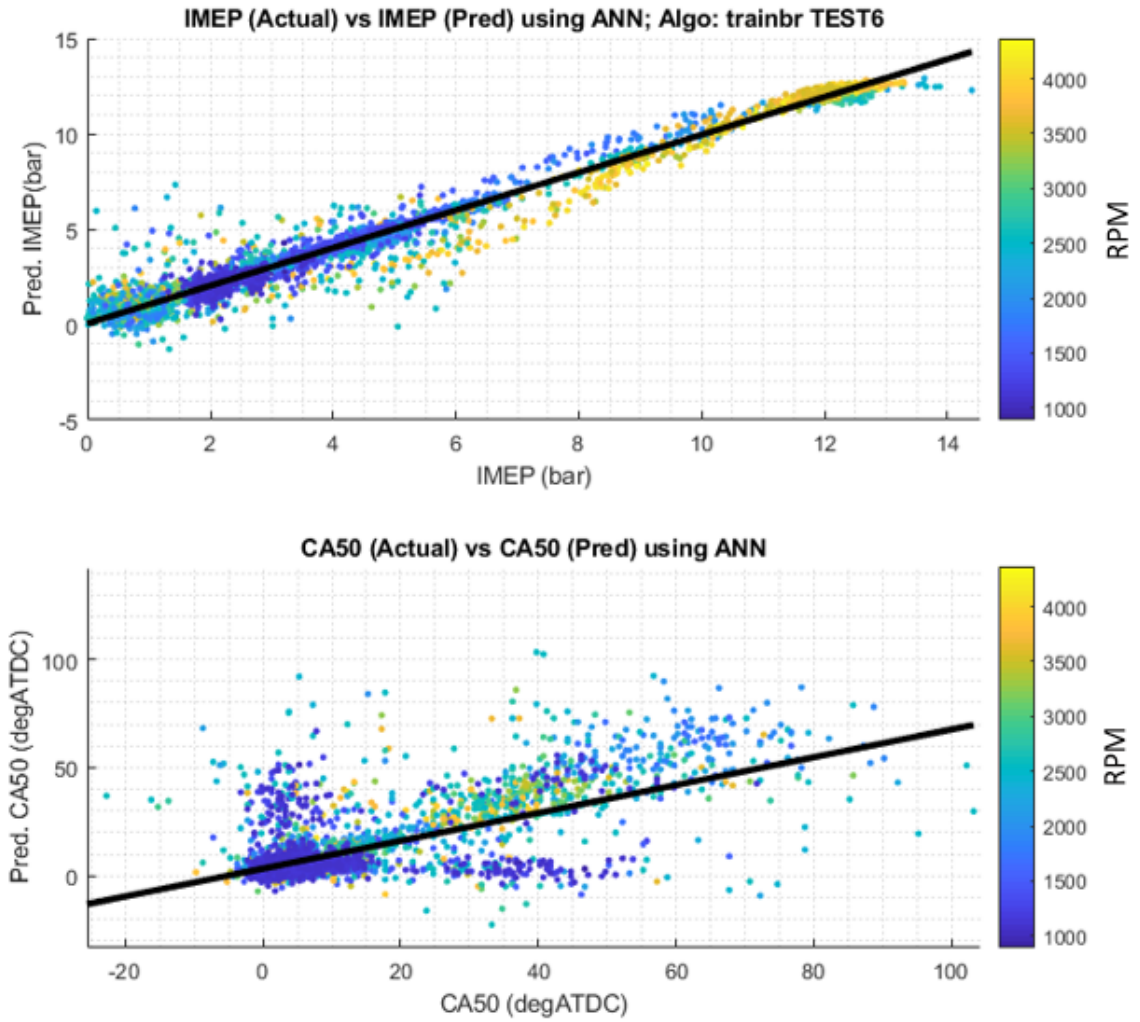


Figure 5.71: Actual against estimated combustion metrics for recursive neural network

Figures 5.72 and 5.73 shows the location and distribution of the errors. The highest errors in estimation were mostly during the heavy transient phases. The errors in CA50 estimates were seen to be larger than previously observed. Figure 5.73 shows

the distribution of errors and it can be seen that the standard deviation of error in CA50 estimates was about 12CAD. The standard deviation of error observed in IMEP estimation was about 0.5 bar.

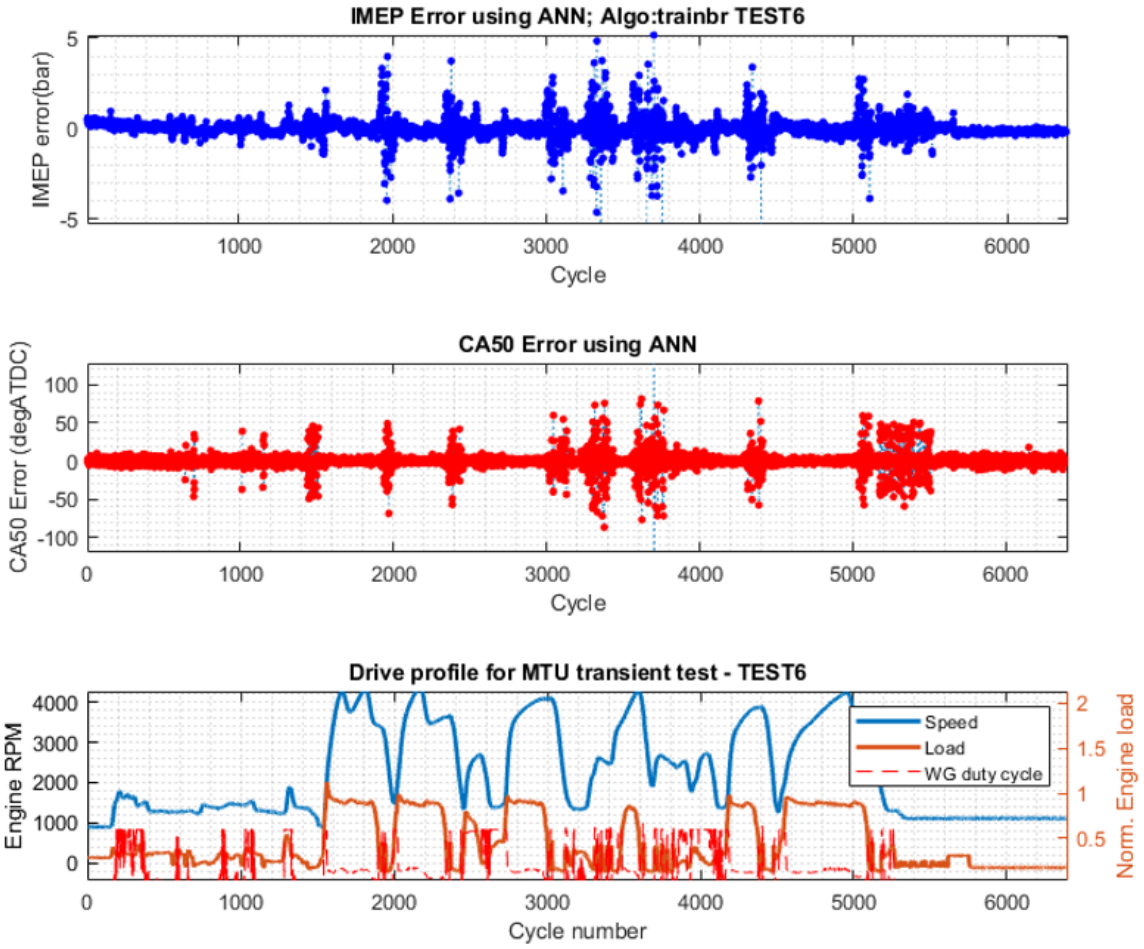


Figure 5.72: Error analysis of recursive network

Thus it was seen through the initial studies of using RNN that the network had to be improved by a large margin; especially CA50 estimation. When conducting a root cause analysis of the problem, it was found that the use of inputs that were

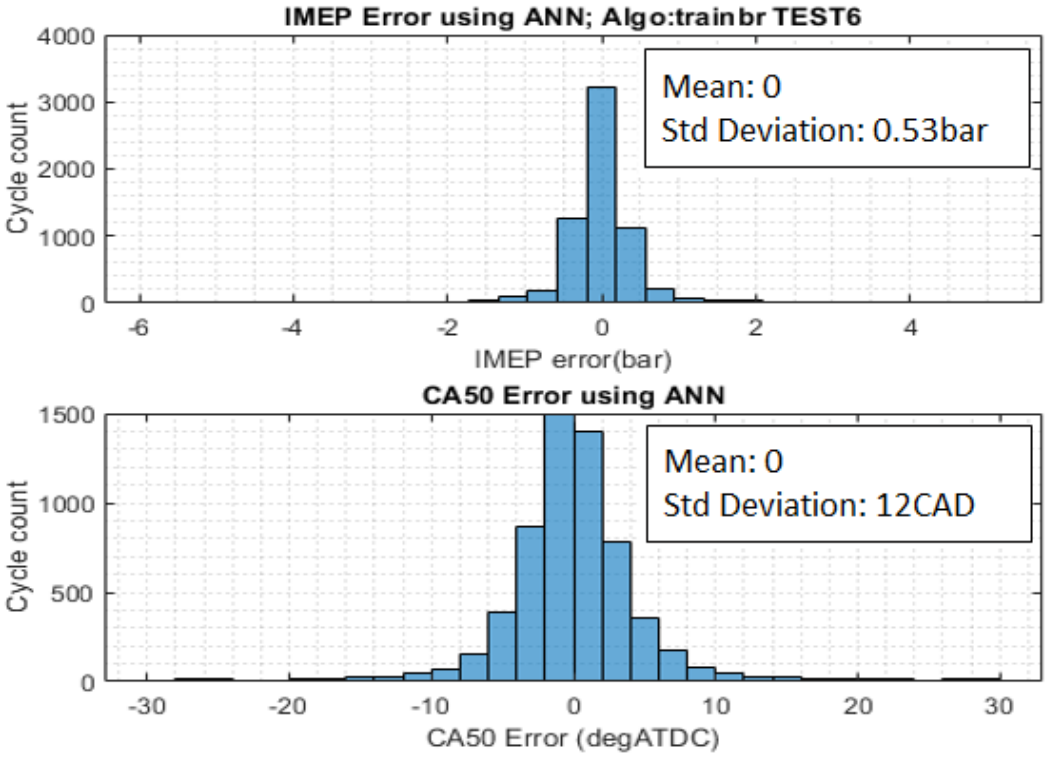


Figure 5.73: Distribution of error of recursive network

not strongly correlated with the parameter being estimated could cause issues with accuracy of estimation. Further, considering the large number of inputs being used, several inputs could be correlated with other inputs thereby making them redundant. To resolve this issue, the correlation of the combustion metrics with the various inputs being used was evaluated. The cross-correlation of the various inputs being ustilized was also evaluated. Lastly, using the same network to estimate both CA50 and IMEP could be problematic as inputs that strongly correlate with one combustion parameter need not strongly correlate with the other combustion parameter.

Table 5.16 lists the various input parameters evaluated in the correlation studies and their acronyms.

Table 5.16
Parameters evaluated and acronyms

Parameter	**Acronym**
Engine Speed	RPM
Manifold Abs. Pressure	MAP
Intake cam phasing	IVT
Exhaust cam phasing	EVT
Location of waste gate	WG_{pos}
Fuel mass injected	MF
Amplitude of 2nd exhaust order	Exh2A
Amplitude of 3rd exhaust order	Exh3A
Amplitude of 4th exhaust order	Exh4A
Phase of 2nd exhaust order	Exh2P
Phase of 3rd exhaust order	Exh3P
Phase of 4th exhaust order	Exh4P
Amplitude of 2nd crank order	Cnk2A
Amplitude of 3rd crank order	Cnk3A
Amplitude of 4th crank order	Cnk4A
Amplitude of 6th crank order	Cnk6A
Phase of 2nd crank order	Cnk2P
Phase of 3rd crank order	Cnk3P
Phase of 4th crank order	Cnk4P
Phase of 6th crank order	Cnk6P
Amplitude of standalone ion peak	SI_{pk}
Location of standalone ion peak	SI_{pl}
Location of flame front wrt standalone	SI_f
Spark Advance	SA

Figure 5.74 offers a visual representation of the cross-correlation between various inputs given to the neural network. The matrix is color coded by the magnitude of the correlation coefficient. It can be seen that the crank data (i.e amplitude and phase of various crank orders) has a strong correlation (R >0.6) with the exhaust data (i.e amplitude and phase of various exhaust orders). Further, the amplitude of

crank orders were also observed to have strong positive correlation with speed and load as well as a strong negative correlation with valve timing. Thus it can be inferred that the crank data was not offering any new information and could thus be omitted for the neural network studies.

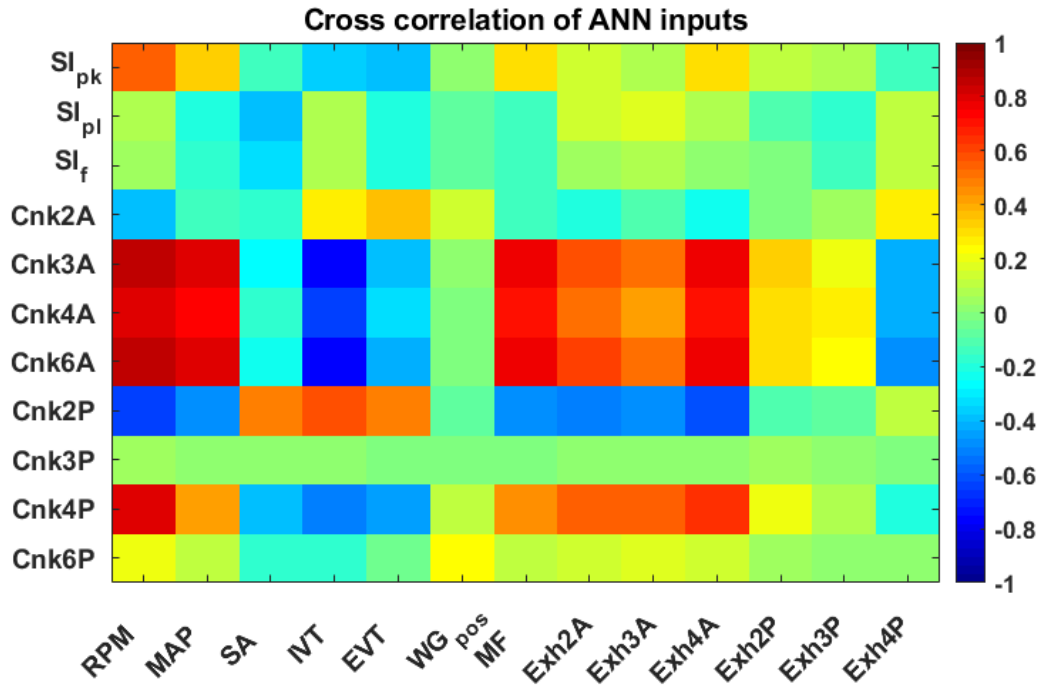


Figure 5.74: Cross-correlation between inputs

The correlation of the various parameters with IMEP and CA50 were also evaluated. The results of the correlation study are listed in Table 5.17. With regard to IMEP, it can be seen that engine speed, load, valve timing, fuel mass, amplitude of exhaust and crank orders have the highest correlation (indicated by green cells). The ion signals shows poor correlation with IMEP. However, for CA50 estimation, the location of the

ion peak and the location of the flame front showed the highest correlation. None of the other inputs used showed a significant correlation with CA50.

Table 5.17
Correlations of various inputs with combustion metrics

	IMEP		IMEP		CA50		CA50
'RPM'	0.79	'Exh4P'	-0.39	'RPM'	0.05	'Exh4P'	0.16
'MAP'	0.99	'SI_pk'	0.4	'MAP'	-0.25	'SI_pk'	-0.07
'SA'	-0.27	'SI_pl'	-0.21	'SA'	-0.42	'SI_pl'	0.78
'IVT'	-0.92	'SI_f'	-0.18	'IVT'	0.11	'SI_f'	0.71
'EVT'	-0.26	'Cnk2A'	-0.25	'EVT'	-0.22	'Cnk2A'	0.08
'WG_pos'	0.13	'Cnk3A'	0.85	'WG_pos'	-0.07	'Cnk3A'	-0.1
'MF'	0.94	'Cnk4A'	0.78	'MF'	-0.17	'Cnk4A'	-0.17
'Exh2A'	0.95	'Cnk6A'	0.86	'Exh2A'	-0.07	'Cnk6A'	-0.15
'Exh3A'	0.9	'Cnk2P'	-0.5	'Exh3A'	-0.03	'Cnk2P'	-0.33
'Exh4A'	0.98	'Cnk3P'	0.01	'Exh4A'	-0.12	'Cnk3P'	-0.02
'Exh2P'	-0.41	'Cnk4P'	0.49	'Exh2P'	0.19	'Cnk4P'	0.27
'Exh3P'	-0.35	'Cnk6P'	0.12	'Exh3P'	-0.28	'Cnk6P'	0.18

Thus it can be seen that the inputs needed for IMEP estimation were different from those needed for CA50 estimation. Thus it was decided to use two separate networks for estimation of IMEP and CA50 respectively. Another observation from Table 5.17 is that the spark advance did not show significant correlation with CA50 as would be expected in actuality. A possible cause for this issue could be a synchronization issue between CAS and ATI.

Based of the results found in Table 5.17 a network was developed whose inputs were only parameters that correlated well with IMEP. Table 5.18 shows the list of inputs used. The network developed and the network specifications are mentioned in Figure

5.75 and Table 5.19.

Table 5.18
Inputs to recursive network to estimate IMEP

ANN Inputs
Engine Speed
MAP
Mass of fuel injected
Intake Cam phasing
Amplitude of 2nd, 3rd and 4th order of exhaust pressure

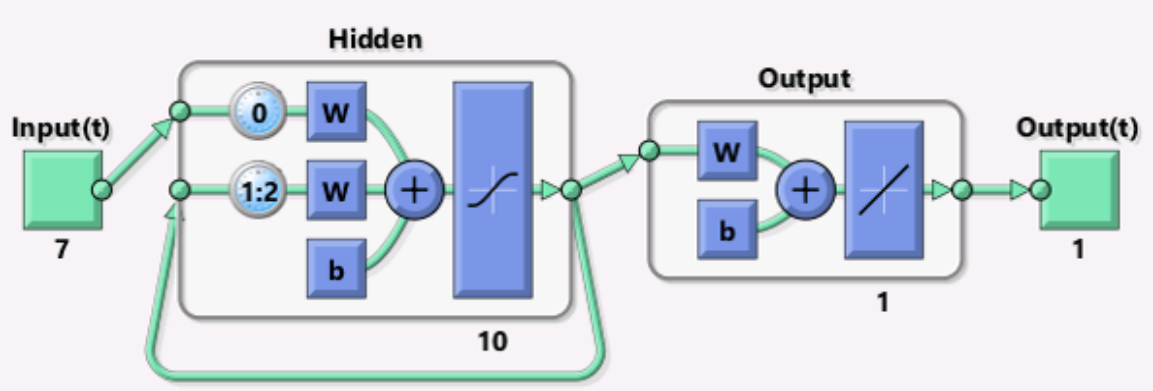


Figure 5.75: Recursive neural network for IMEP estimation

Table 5.19
Specifications of recursive network to estimate IMEP

Parameter	Value
Number of samples	6500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	7
Number of Outputs	1
Number of Hidden Layers	10
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

Results of using the network specifically for IMEP estimation are shown in Figure 5.76. It can be seen from the results that despite the reduced number of inputs the network was able to offer reasonably good estimates that had an error lesser than 0.5bar. The largest errors were still under heavy transience conditions.

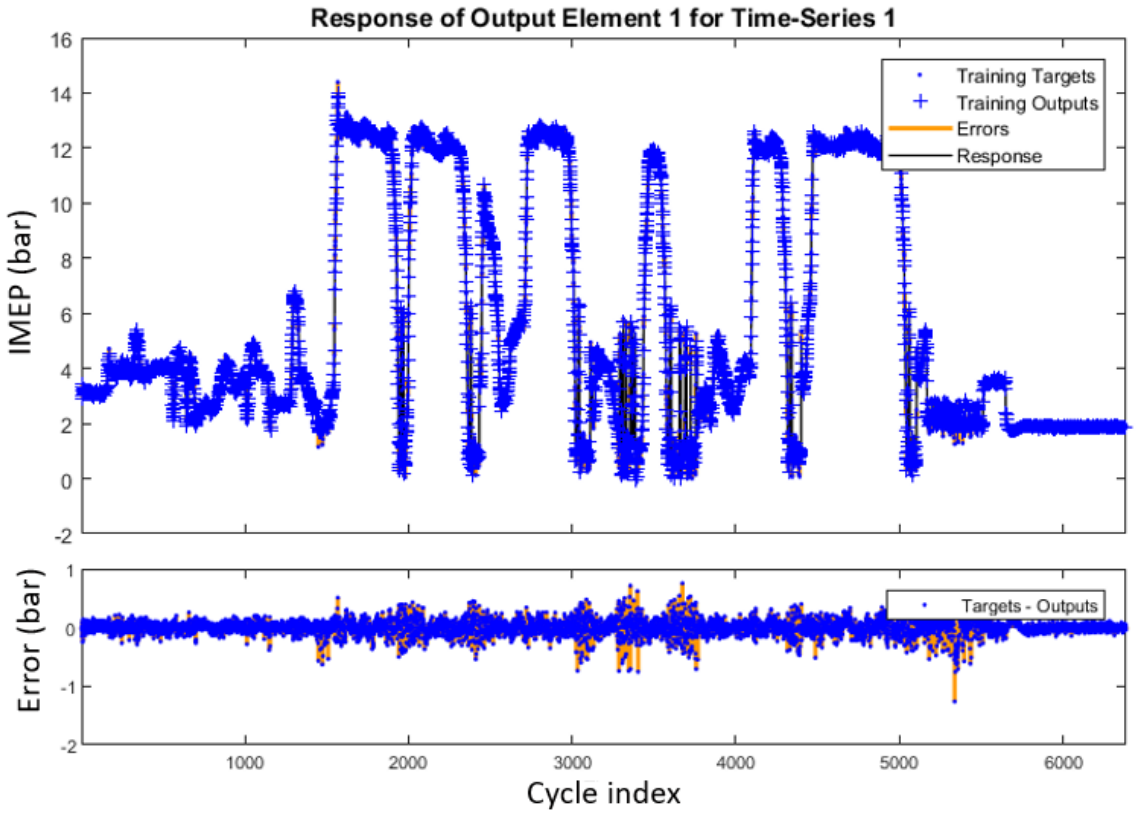


Figure 5.76: Network prediction and error

Figure 5.77 shows the estimated IMEP plotted against the actual IMEP for a particular cycle. The data points are color coded by engine speed. It can be seen that the there was significant deviation observed between 2-6bar IMEP. The deviation is

much lesser at higher loads. This is due to the fact that the points showing large deviation correspond to cycles under heavy transience.

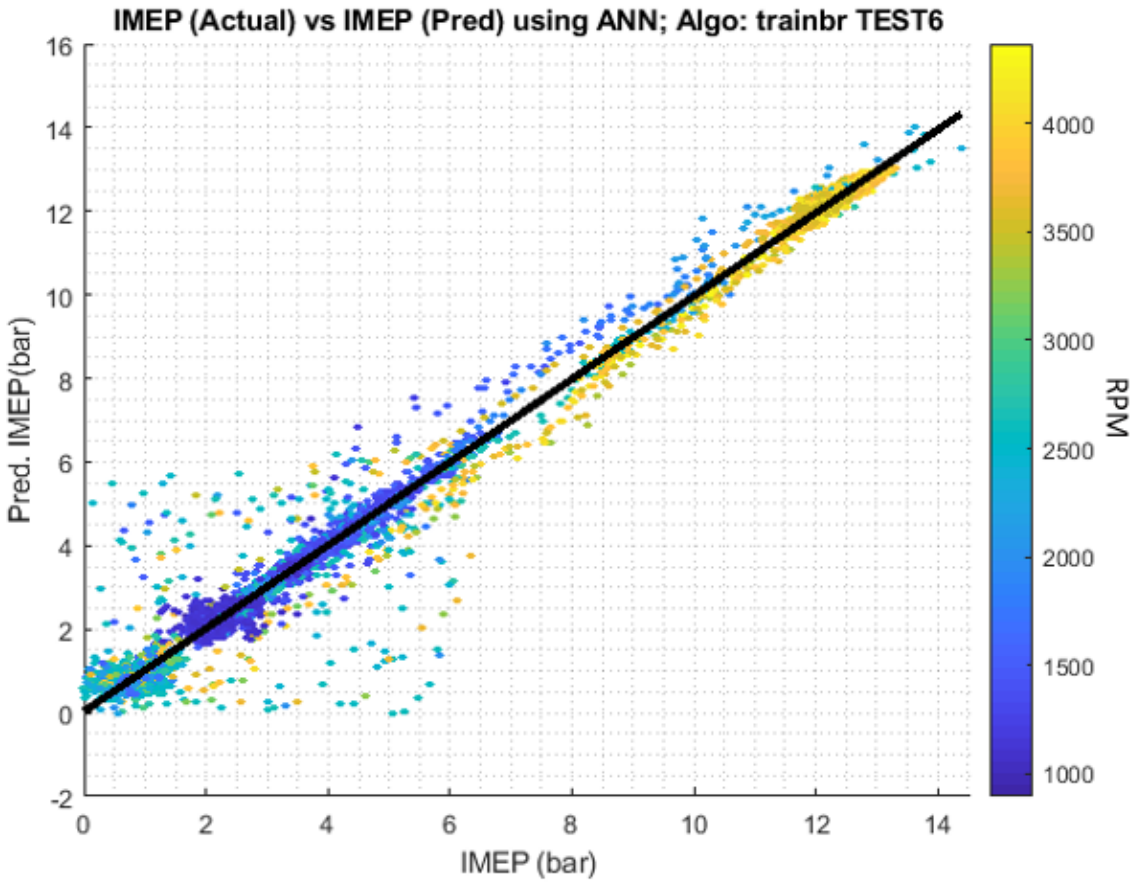


Figure 5.77: Error analysis of recursive network for IMEP estimation

Figure 5.78 shows the distribution of errors and it can be seen that the standard deviation of error in IMEP estimates was about 0.47 bar. This error can be assumed to lower if the estimates for high transience is ignored. Thus it was demonstrated that by using a dedicated recursive network for IMEP estimation could offer good estimates on a cyclic basis.

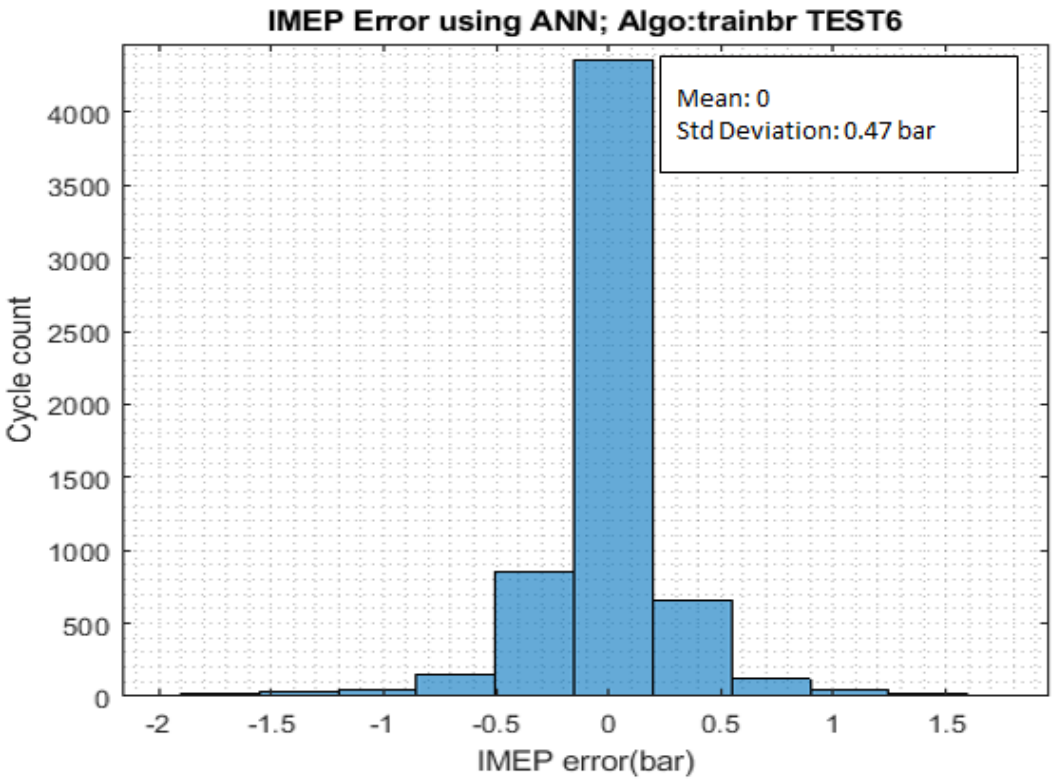


Figure 5.78: Distribution of error of recursive network

Similar to the network developed for estimating IMEP, a network for estimating CA50 as well was developed. Similar to the IMEP estimation network, the CA50 estimation network only used inputs that highly correlated with CA50. Table 5.20 shows the list of inputs used; none of the engine parameters were used as they had shown poor correlation. The network developed and the network specifications are mentioned in Figure 5.79 and Table 5.21.

Table 5.20
Inputs to recursive network to estimate CA50

ANN Inputs
Location of standalone ion peak
Location of flame front wrt standalone ion probe

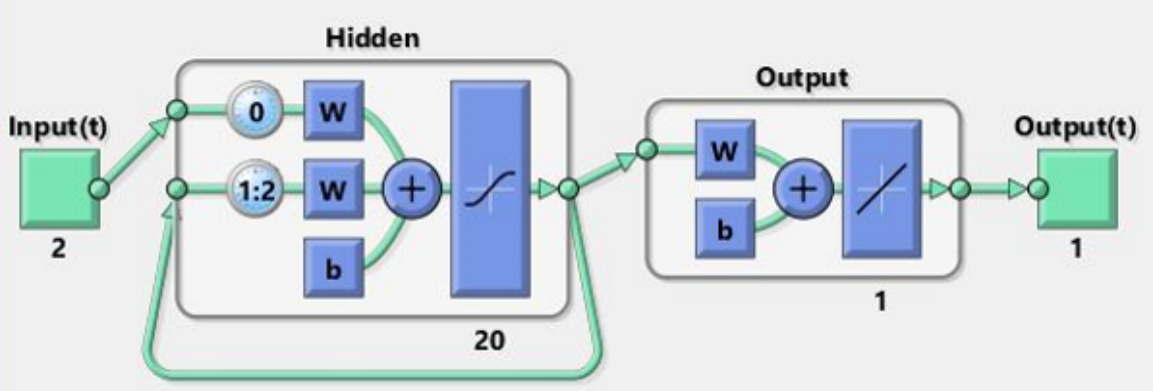


Figure 5.79: Recursive neural network for CA50 estimation

Table 5.21
Specifications of recursive network to estimate CA50

Parameter	**Value**
Number of samples	6500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	2
Number of Outputs	1
Number of Hidden Layers	10
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

The results of using a dedicated network for CA50 estimation are shown in Figure 5.80 and 5.81. It can be seen in Figure 5.80 that the errors in CA50 estimation were considerable high especially under heavy transience. Figure 5.81 shows the estimated CA50 plotted against the actual CA50 of the cycle. The datapoints are color coded by the location of the ion peak. It can be seen that there is significant deviation between the estimated and the actual CA50.

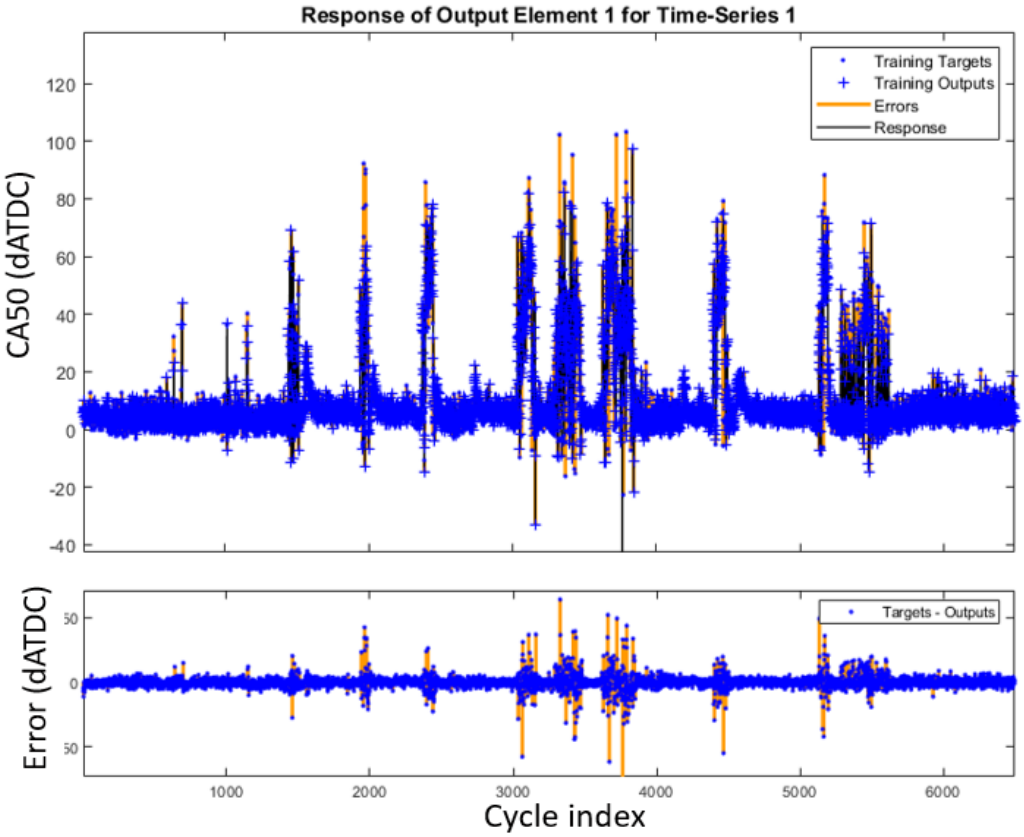


Figure 5.80: Network prediction and error for CA50 estimation

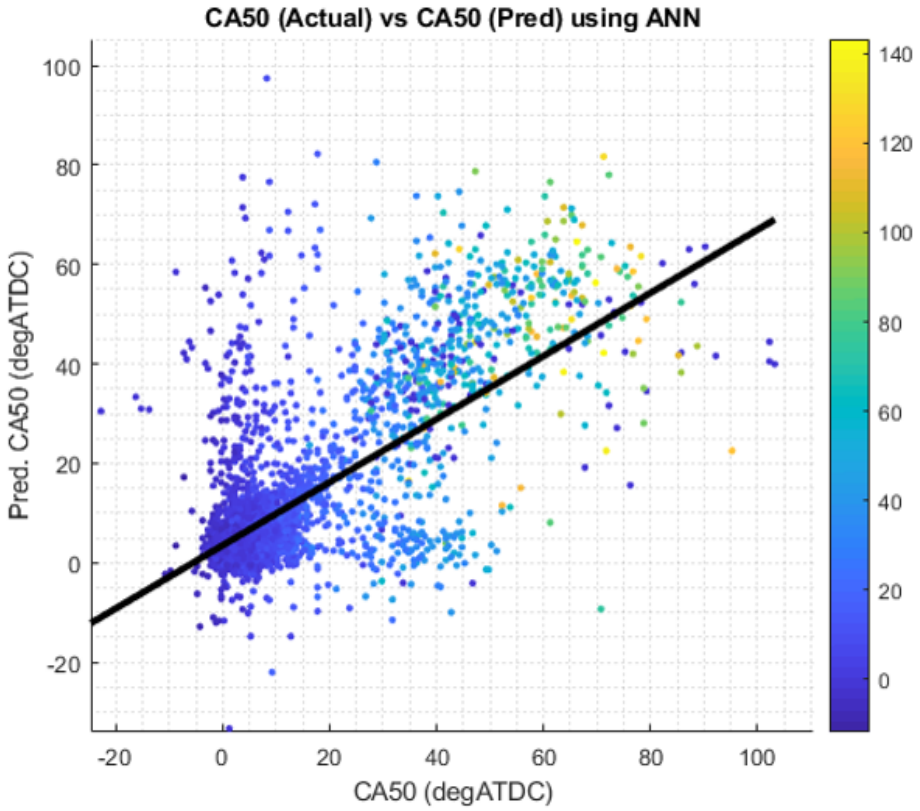


Figure 5.81: Error analysis of recursive network for CA50 estimation

5.4.2.1 Pseudo-steady state tests for combustion metric estimation

Through the various iterations of neural networks developed it was observed that under heavy transience, the estimates of the network showed large deviation. In order to evaluate network performance under conditions with relatively lower transience, a pseudo-steady state test was developed wherein the engine was subjected to step changes in operating conditions. This test helps highlight the performance of the network under conditions with low transience.

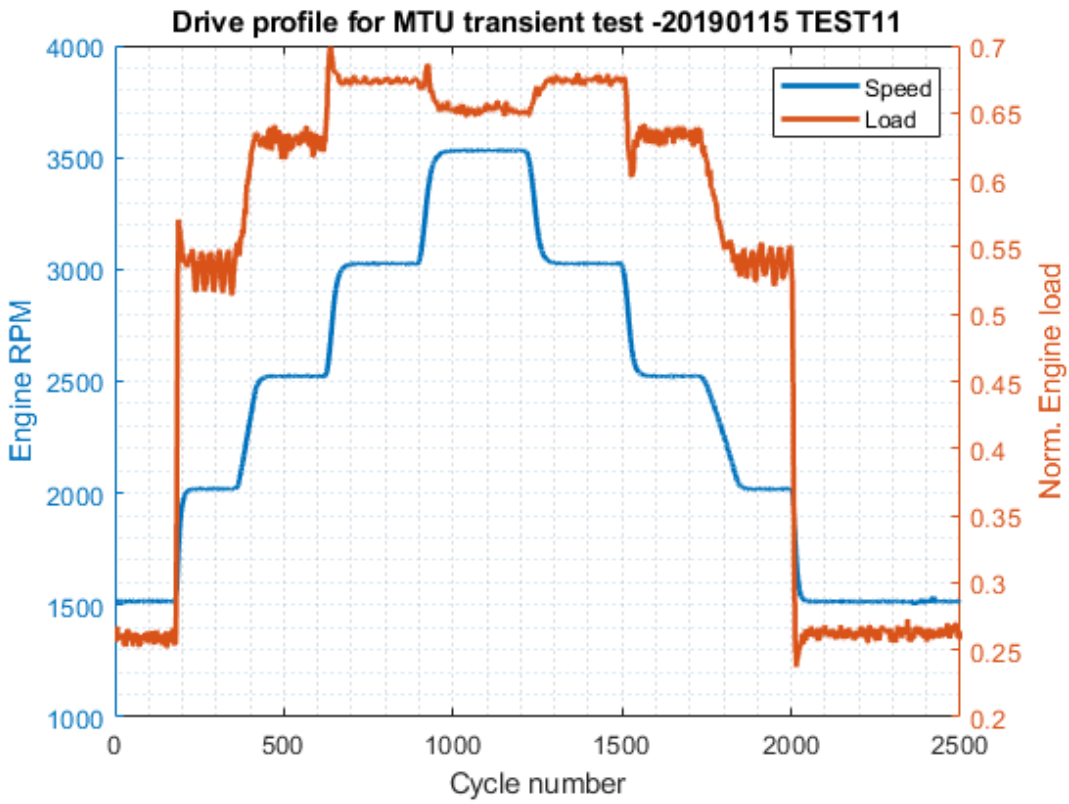


Figure 5.82: Drive cycle for pseudo steady state test

Figure 5.82 shows the drivecycle used for the pseudo steady state studies. As can be seen, the drivecycle consisted of a sequence of step changes in engine operating points. The engine speed varied from 1500rpm-3500rpm and the normalized engine load varied from about 0.25 to 0.7.

IMEP estimation for pseudo-steady state test

Similar to the previous analysis a seperate network was used for IMEP estimation and CA50 estimation. The list of inputs used for the current network are listed in Table 5.22. The correlation of the various inputs with IMEP was re-evaluated for the

current dataset and it was found that in addition to the parameters shown in Table 5.17, the exhaust cam timing and phase of exhaust pressure orders also showed a strong correlation. They were thus included as inputs to the network. Additionally the number of hidden layers were increased to help obtain better results.

Table 5.22
Inputs to RNN to estimate IMEP in pseudo-steady state test

ANN Inputs
Engine Speed
MAP
Mass of fuel injected
Intake and exhaust Cam phasing
Amplitude of 2nd, 3rd and 4th order of exhaust pressure
Phase of 3rd and 4th order of exhaust pressure

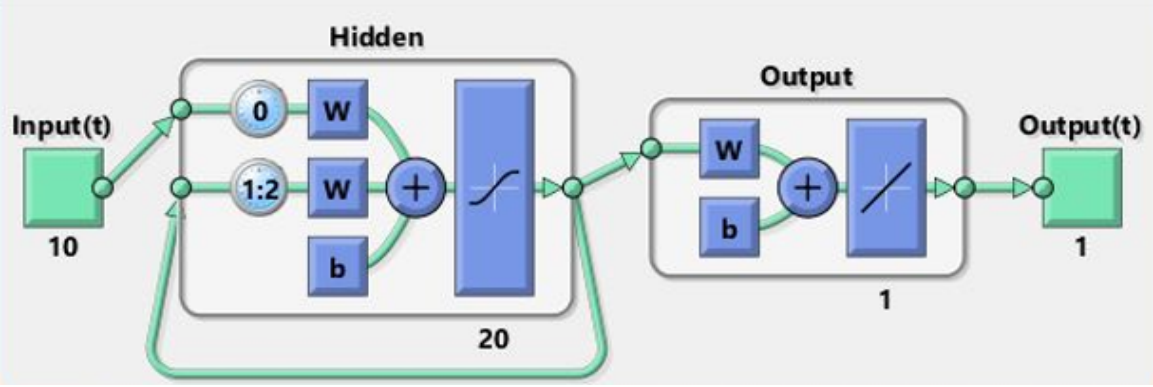


Figure 5.83: RNN to estimate IMEP in pseudo-steady state test

Table 5.23
Specifications of RNN to estimate IMEP in pseudo-steady state tests

Parameter	**Value**
Number of samples	2500
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	10
Number of Outputs	1
Number of Hidden Layers	20
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

The performance of the neural network in estimating IMEP over the pesudo-steady state test is shown in Figures 5.84, 5.85 and 5.86. It can be seen that the network was able to estimate the IMEP with minimal errors. Figure 5.85 shows that across the various operating conditions the network did not show large deviations with respect to the actual IMEP value. Further, the distribution of errors shown in Figure 5.86, shows that the IMEP estimates of about 95% of the data points of the dataset were within about 0.28bar of the actual value which was the best performance observed throughout.

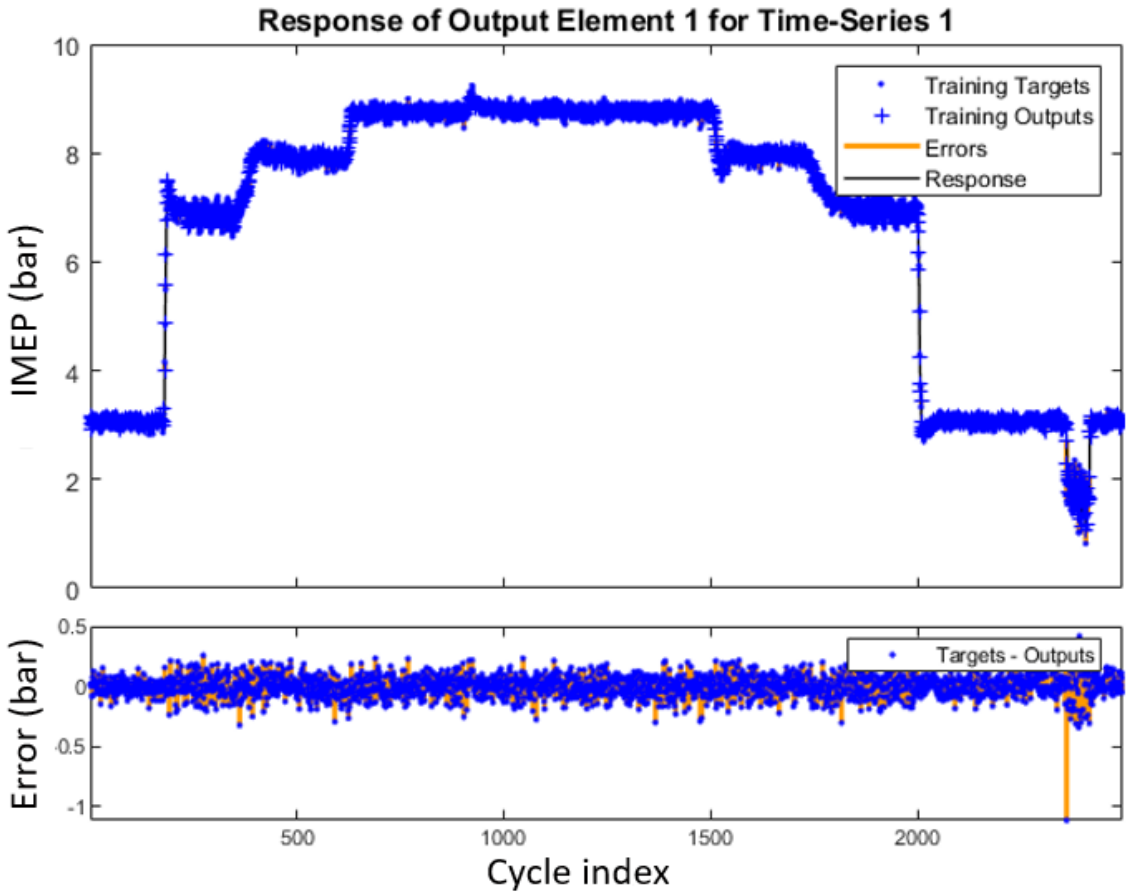


Figure 5.84: Network estimated and actual IMEP for pseudo-steady state tests

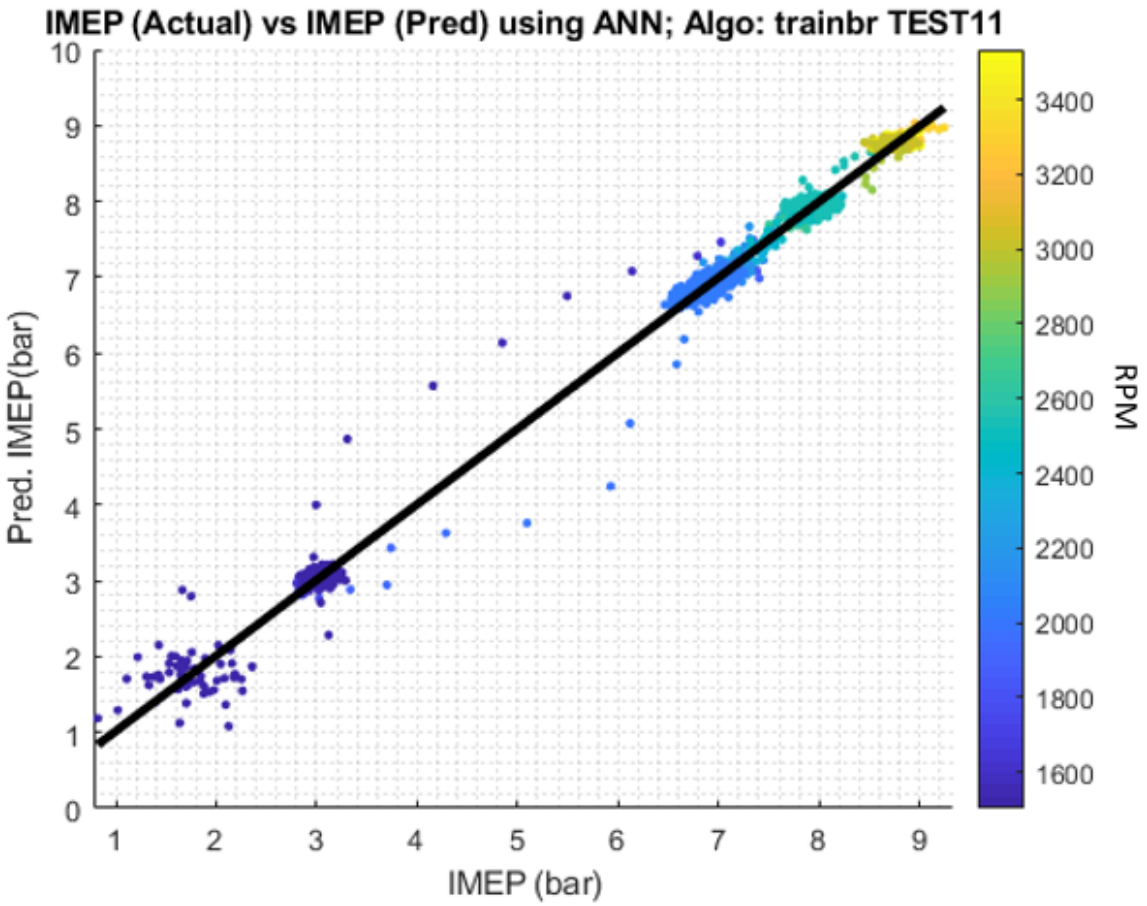


Figure 5.85: Error analysis of network

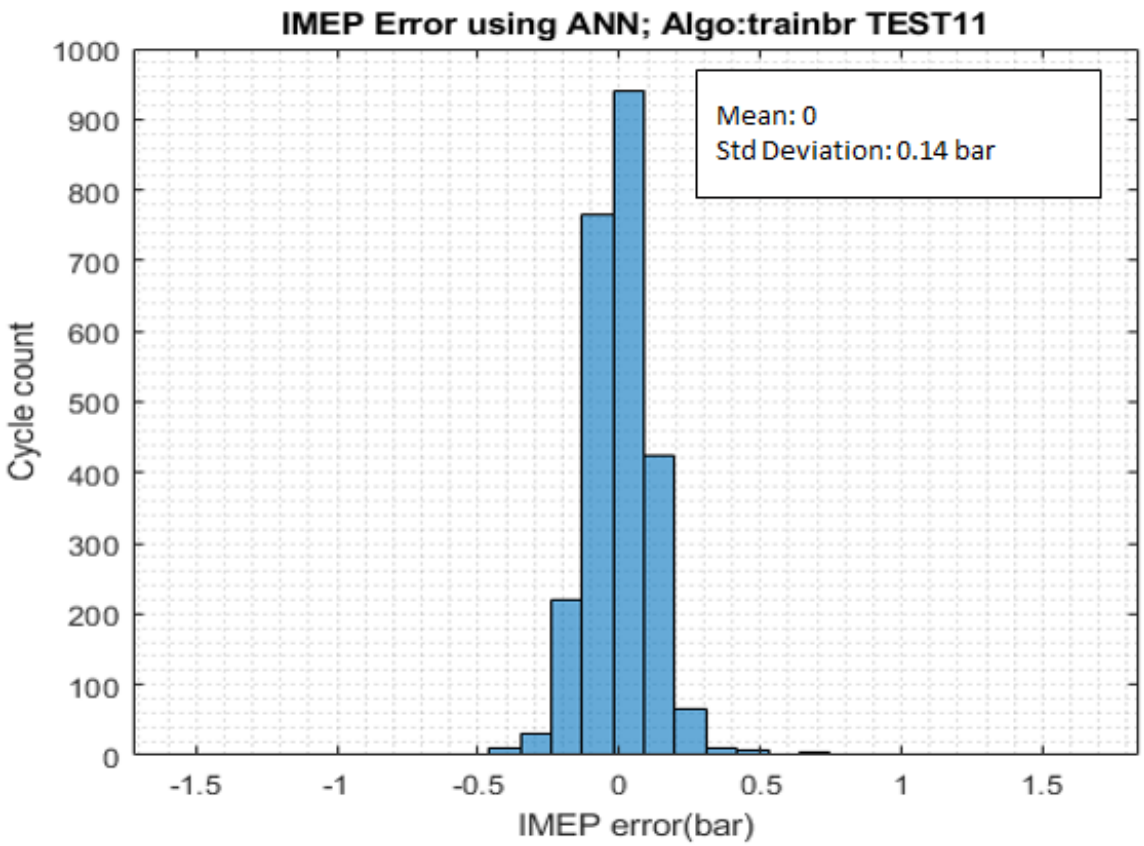


Figure 5.86: Distribution of error in estimation of IMEP in pseudo-steady state tests

The ability of the network to estimate IMEP on a cycle-by-cycle basis is shown in Figure 5.87 which is essentially a zoom-in of Figure 5.84. It can be seen that in some cycles the network estimated IMEP (shown in +) is the same as the actual IMEP (shown by .) and in some cycles the network estimates show a deviation. Figure 5.88 shows the results for the number of datapoints where the network estimated IMEP was within a certain percentage of the actual IMEP. It was observed that with the current ANN 90% of the IMEP estimates were within 3% of the actual IMEP and

98% of the IMEP estimates were within 5% of the actual IMEP. If a 3-5% variation in IMEP is attributed to stochasticity, then it can be stated that the network offers accurate IMEP estimates on a cyclic basis.

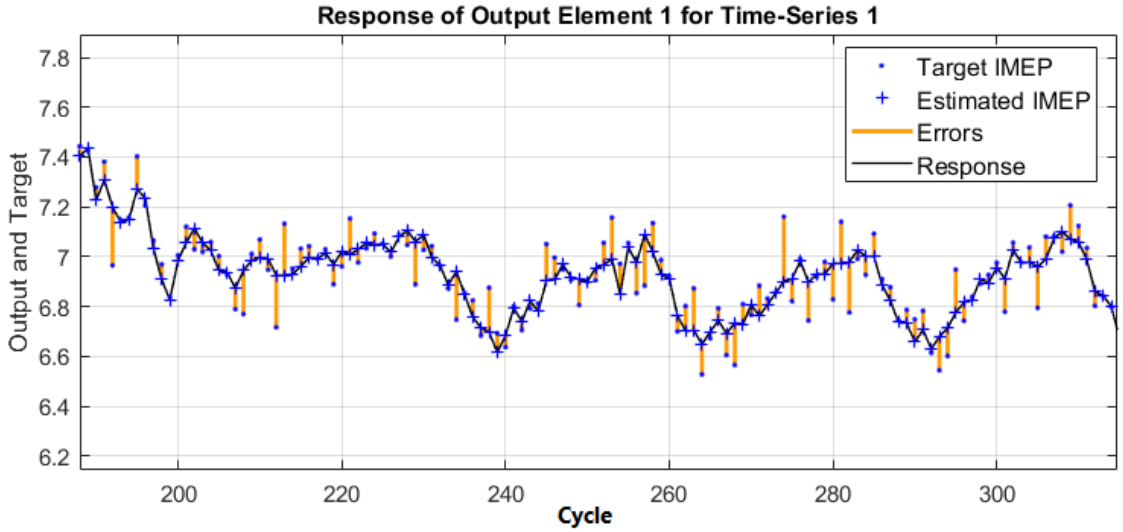


Figure 5.87: Estimation of IMEP on a cycle by cycle basis in pseudo-steady state tests

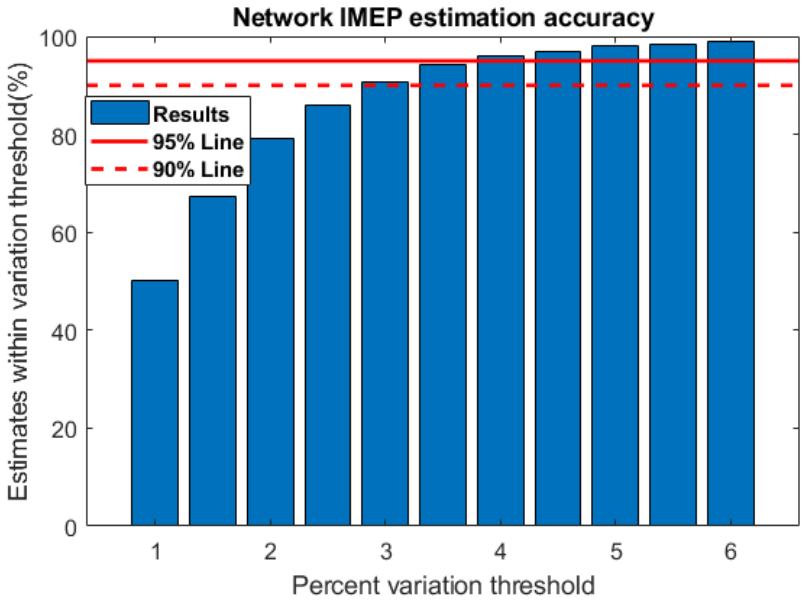


Figure 5.88: Accuracy of ANN IMEP estimation

CA50 estimation for pseudo-steady state test

The pesudo steady state dataset was also used to train a network to estimate CA50. The inputs to the network are mentioned in Table 5.24 and are similar to that previously used. The network developed and the specifications of the network are mentioned is Figure 5.89 and Table 5.25.

Table 5.24
Inputs to RNN to estimate CA50

ANN Inputs
Location of standalone ion peak
Location of flame front wrt standalone ion probe

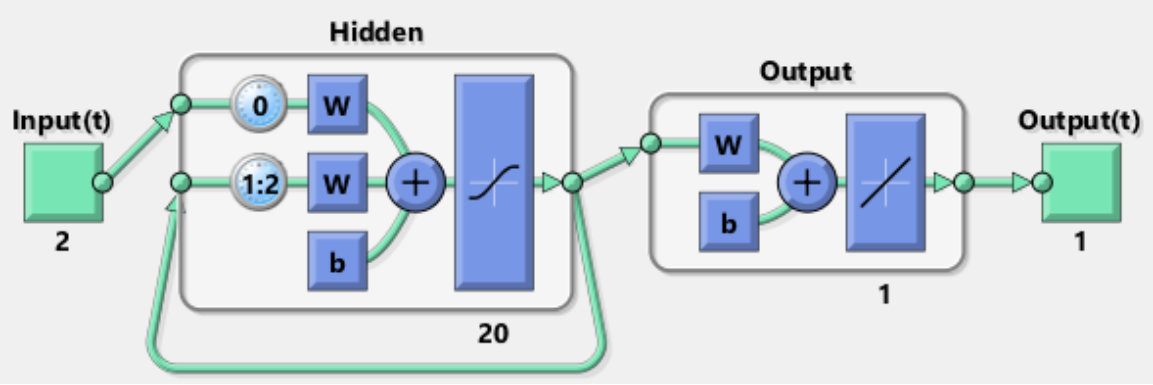


Figure 5.89: RNN to estimate CA50 in pseudo-steady state test

Table 5.25
Specifications of RNN to estimate CA50 in pseudo-steady state tests

Parameter	**Value**
Number of samples	2200
Samples in Training, Validation & Test set	70% , 15% & 15%
Number of Inputs	2
Number of Outputs	1
Number of Hidden Layers	20
Training Algorithm	Bayesian Regularization
Performance Metric	Least Squared Error

The results of the analysis are shown in Figures 5.90, 5.91 and 5.92. It can be seen that the network was able to offer good estimates of the CA50. Figure 5.91 shows that the deviations observed were not restricted to one particular operating regime. The distribution of errors shown in Figure 5.92 showed that the standard deviation of the errors in estimates was about 2.5 CAD which was the best results obtained throughout the study.

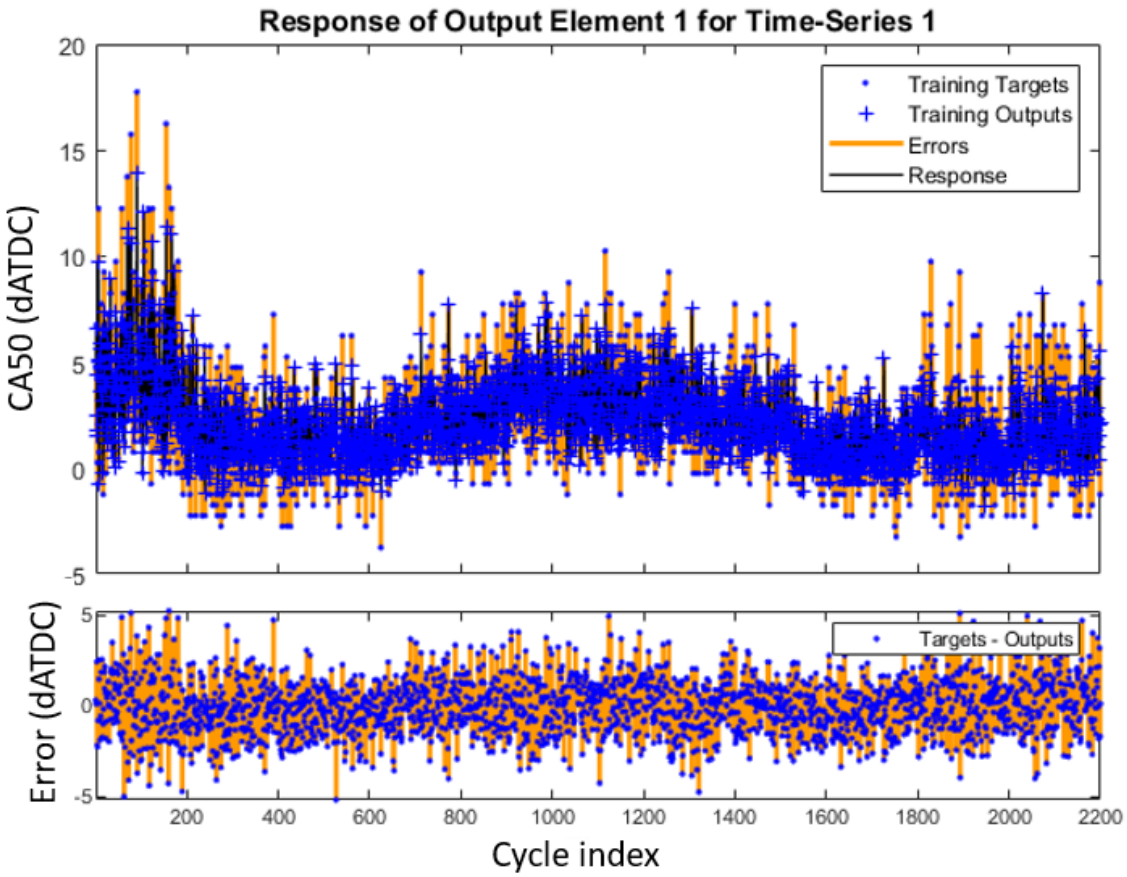


Figure 5.90: Network estimated and actual CA50 for pseudo-steady state tests

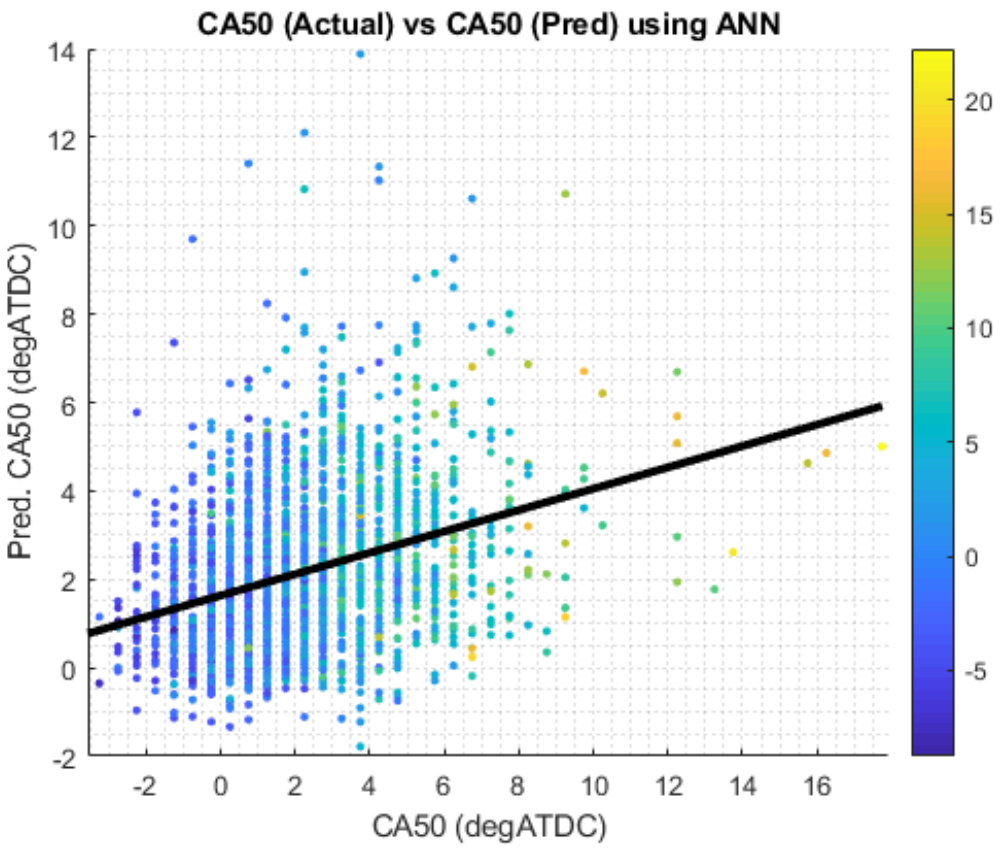


Figure 5.91: Error analysis of network

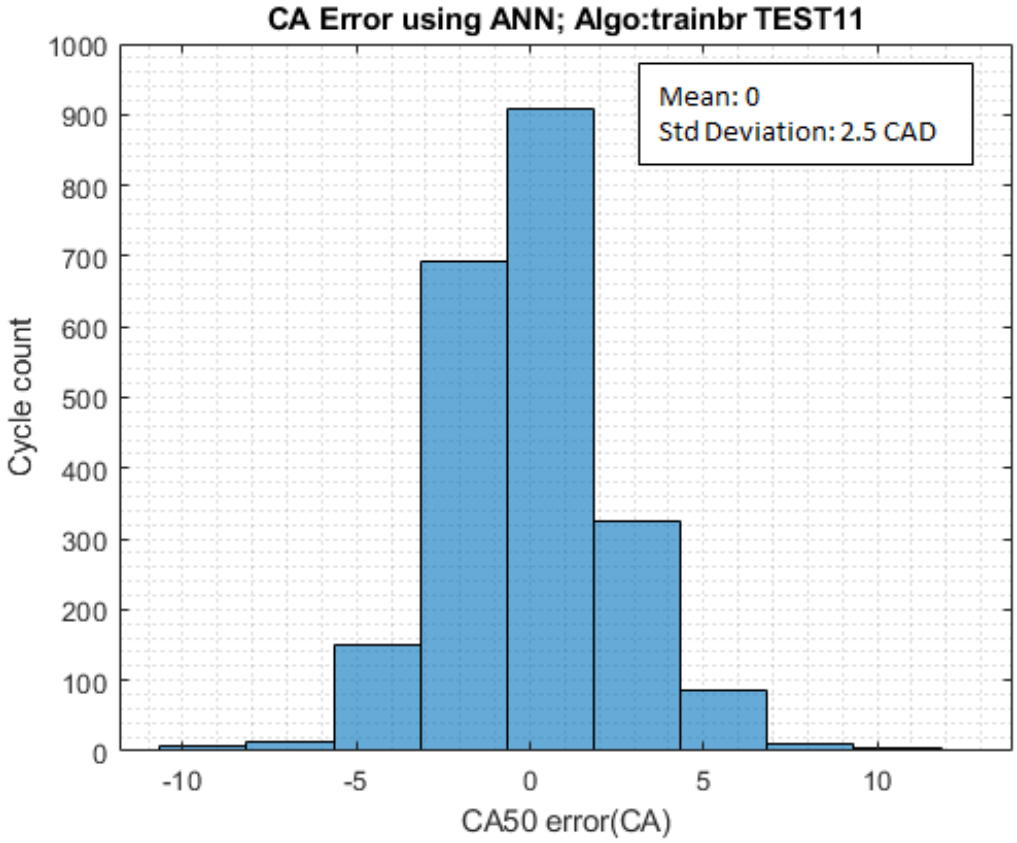


Figure 5.92: Distribution of error in estimation of CA50 in pseudo-steady state tests

Efforts were also made to evaluate the influence the number of hidden layers on the accuracy of estimation. Figure 5.93 shows the mean and standard deviation of the error in CA50 estimation for various number of hidden layers. It was observed that for the conditions evaluated, a network with 12 layers or 20 layers produced CA50 estimates with the least mean and standard deviation in errors. This study chose 20 layers due to repeatability of results.

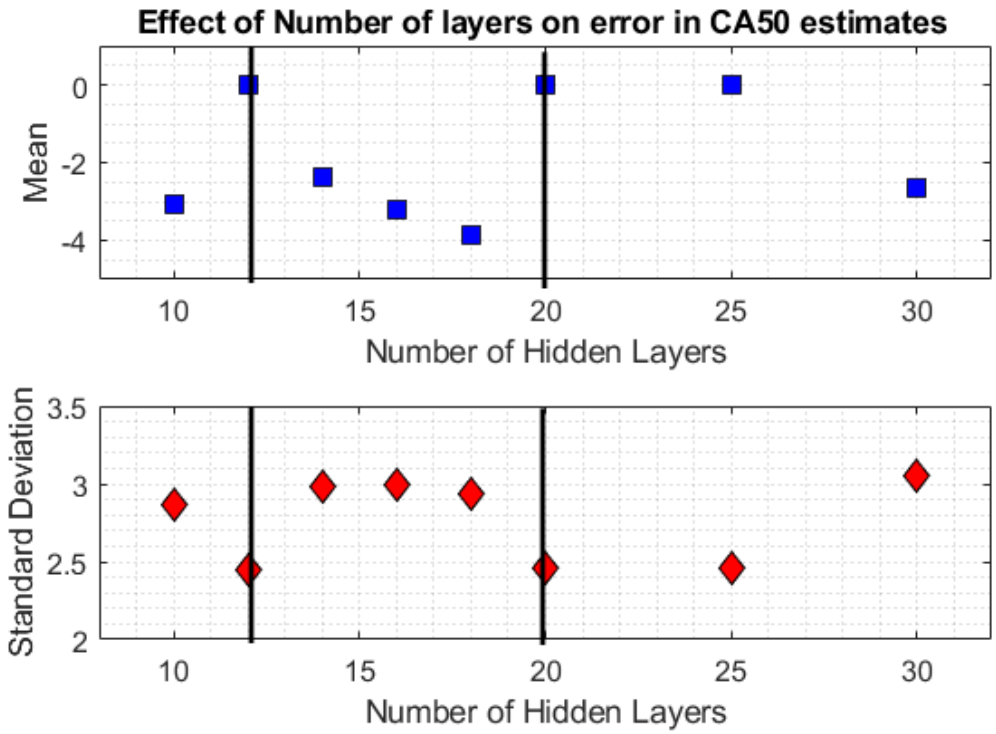


Figure 5.93: Effect on number of hidden layers on characteristics on CA50 estimation errors

5.5 Conclusion on neural network studies

The study initially utilized a simple feed forward network to estimate combustion metrics including IMEP and CA50 using numerous inputs from the various sensors discussed and several engine parameters as well. Progressively it was understood that a recursive approach would be more representative of capturing the combustion dynamics being studied. The inputs being given to the neural networks had to be refined and customized for obtaining good accuracy. The use of correlation studies

aided in uncovering the most critical inputs needed for estimation of IMEP and CA50. Finally the pesudo steady state test proved the ability of the neural network to provide reasonable estimates in the absence of heavy transience. Detailed conclusion are mentioned in Chapter 6

Chapter 6

Conclusion and Recommendations

This thesis offers a broad overview of the control and diagnostic capabilities of various sensors such as ion probes, exhaust pressure sensors and crankspeed sensors. The study also outlines a neural network based approach to estimate IMEP and CA50 by combining the information contained in the various sensors outputs. This study used a Ford 2.0L Ecoboost engine to conduct tests under various operating conditions as well as carry out transient testing. Additional testing was also conducted on an optical engine as well as in-vehicle. This study primarily focused on three sensors namely ion sensors, exhaust pressure sensors and crank position sensors. The prime objective of the study was to use the sensor suite to identify correlations with incylinder combustion as well as estimate combustion metrics including IMEP and CA50 under both transient and steady state conditions.

The ion sensor studies conducted using the standalone ion probe instrumented onto the metal engine showed that under steady state conditions the amplitude and location of the pressure peak could be estimated using the ion peak amplitude and location. Across the operating conditions studied, the correlation of ion peak location with pressure peak location was more consistent (i.e $R > 0.6$) than the correlation of ion peak amplitude with pressure peak amplitude. The optical engine studies showed the influence of sensor location in obtaining good correlations with pressure metrics. It was seen that placing the sensor in front or rear positions (i.e sensor 3 and 4) yielded the best results in comparison to sensors 1 and 2 which were place near the intake and exhaust ports.

Further, the results of the knock tests showed that the integral of the ion signal followed a log normal distribution, similar to that of the integral of incylinder pressure. Additionally, of the various windowing techniques developed to obtain good correlation between the ion integral and pressure integral the modified adaptive windowing technique yielded the best results. The integral of the coil ion sensor output showed a correlation of 0.61 with pressure integral while the standalone ion sensor output showed a correlation of 0.63. Furthermore, visualizing the linear spectrum of the incylinder pressure and ion signals, it was seen that by using the modified adaptive windowing technique the knock frequency of 6kHz-7kHz was identifiable on the standalone ion probe. The coil ion probe signal however showed much poorer response. A possible cause for this could be the location of the sensor and the influence of ringing.

Moving onto exhaust sensor studies it was seen that under steady state conditions the exhaust pressure could be correlated with inclyinder pressure as well as detect misfires. However in order to be able to correlate exhaust pressure artifacts with combustion metrics it was essential to identify exhaust signatures and understand factors affecting them. Transient in-vehicle tests conducted, helped in waveform classification as it showed that exhaust waveforms could be classified based on engine load, speed, spark timing etc. The type I waveforms showed very good correlations with IMEP (R>0.9) and CA50 (R> 0.7) apart from just pressure metrics. The exhaust pressure signal of cycles under low load (Type II) with a prominent maxima showed good correlations with incylinder pressure. It was also observed that the type C cycles that consisted of cycles whose signals had a prominent trough always occurred during Tip-out/DFSO and consequently could not correlate with any of the pressure metrics.

Furthermore, the indices developed for detecting misfires/DFSO events showed that the methodology of using exhaust minima could help identify misfires with a high degree of accuracy. The exhaust studies also investigated the use of order tracking as an alternative means of extracting information from the exhaust pressure signal. The order analysis of the vehicle tests dataset showed the rich yet complex nature of information that can be extracted. There was need to isolate gearshift events and thus the use of transient test on an engine dynamometer was employed. The exhaust orders extracted namely 2, 3 and 4 were later used as inputs to the neural network. Similar processing was conducted on the crank data as well.

Lastly moving onto the neural network studies it was seen that using a simple feed forward network with 26 inputs from the various sensors and engine parameters it was seen that the network estimates of IMEP and CA50 followed the same trend as the actual IMEP and CA50. The error in estimates were nominal at about 0.4 bar for IMEP and 3 CAD for CA50 estimates. However the dimensionality of the input matrix was quite large. In an attempt to reduce input dimensionality feature scaling an PCA were implemented. Though PCA helped reduces the demensionality of the input matrix the accuracy of estimates were compromised.

Additionally in an attempt to use a network that would utilize prior cycle information to generate an estimate for a given cycle, a recursive network was utilized. Further, a separate network was used for IMEP and CA50 estimation. The inputs to these networks were refined and only parameters that correlated well with the combustion metric being estimated were used as inputs. In doing so, the estimates of the network improved such that the IMEP showed a standard deviation of about 0.47bar. CA50 estimation however displayed significant deviation. Finally to evaluate the performance of the neural network in the absence of heavy transience a pesudo transient test was developed. The performance of the network in the pseudo transient transient test was the best obtained with the standard deviation of IMEP found to be about 0.15 bar and CA50 about 2.5CAD.

Thus through the results shown in Chapter 5 it can be said that the study was

successful in using a sensor suite to identify abnormal combustion as well as estimate combustion metrics including IMEP and CA50 under both transient and steady state conditions.

Future work

This section entails a number of recommendations that could be used to develop this study further. In regard to knock detection using ion sensors, it was found that sensor location and ringing played a major role in inhibiting the coil ion sensor from accurately detecting knock. If ion sensors are to be used for knock detection then improving the coil ion sensors' capability in detecting knock would be pivotal. Coil integrated ion sensors do not provide a packaging hassle and require no additional machining to be implemented. Thus by improving the ability of the sensor to reject or avoid the ringing phase could help make ion sensors more viable in knock detection.

In this study there was limited usage of the pressure sensor placed at a standoff(i.e. Omega sensor). Results show that even the Omega sensor contains the necessary order information across various speed load regimes. The use of the Omega sensor as an input to the neural network instead of the Kulite sensor could be evaluated. Further, the effect of standoff distance on the analysis could also be investigated. Along with the use of the Omega sensor, the calculation and influence of the transport delay in

measurement would have to be conducted as well.

This study used neural networks as a means to fuse the information from the various sensors. The use of alternative techniques including Kalman filters could yield better results with reduced computational effort and is worth investigating.

The exhaust data used in the later part of this study involved a simplified dataset with the absence of gear shift events. It could prove worthwhile to conduct an analysis where the inputs to the neural network use sensor outputs from vehicle level tests.

Additional investigations into the applications of crank data would also need to be conducted. The influence of sampling rates and clock resolutions is a key study as well

A larger and more critical task in regard to this study would be to implement the findings and techniques developed in this study into the engine diagnostic and control strategy to evaluate the engine performance.

It was observed in this study that the network estimates showed large errors under highly transient conditions. One could consider using separate networks for steady state and highly transient operation in an effort to improve network performance.

The CA50 could at best be estimated to a accuracy of 2.5 CAD. Improvement of CA50 estimates using parameters like RGF, Torque etc. would be a worthy exercise.

The influence of using virtual sensors in addition to sensor suite could also be prove worthwhile.

Establishing a benchmark for acceptable errors in IMEP and CA50 estimates would also be important in assessing network performance.

References

[1] Kaushik Prabhu. Sensor fusion for spark-ignition engines. 2018.

[2] Yanyu Wang. The interaction of ignition and in-cylinder flow on flame kernel development and its impacts on combustion in an optically accessible direct injection engine. 2018.

[3] Libin Jia, Jeffrey D Naber, and Jason R Blough. Frequency response function adaptation for reconstruction of combustion signature in a 9-l diesel engine. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 229(17):3071–3083, 2015.

[4] Feilong Liu, Gehan AJ Amaratunga, and Nick Collings. A fourier analysis based synthetic method for in-cylinder pressure estimation. Technical report, SAE Technical Paper, 2006.

[5] Michael Wagner, Johann F Böhme, and Jürgen Förster. In-cylinder pressure

estimation from structure-borne sound. Technical report, SAE Technical Paper, 2000.

[6] Haris Hamedović, Franz Raichle, Jörg Breuninger, Wolfgang Fischer, Wolfgang Fishcer, Werner Dieterle, Martin Klenk, and Johann F Böhme. Imep-estimation and in-cylinder pressure reconstruction for multicylinder si-engine by combined processing of engine speed and one cylinder pressure. *SAE transactions*, pages 135–142, 2005.

[7] Bahram Bahri, Azhar Abdul Aziz, Mahdi Shahbakhti, and Mohd Farid Muhamad Said. Understanding and detecting misfire in an hcci engine fuelled with ethanol. *Applied Energy*, 108:24–33, 2013.

[8] Markus Willimowski and Rolf Isermann. A time domain based diagnostic system for misfire detection in spark-ignition engines by exhaust-gas pressure analysis. Technical report, SAE Technical Paper, 2000.

[9] V Giglio, G Police, N Rispoli, A di Gaeta, M Cecere, and L Della Ragione. Experimental investigation on the use of ion current on si engines for knock detection. Technical report, SAE Technical Paper, 2009.

[10] Dimitris Panousakis, Andreas Gazis, Jill Patterson, and Rui Chen. Analysis of si combustion diagnostics methods using ion-current sensing techniques. 2006.

[11] Nicolo Cavina, Poggio Luca, and Giovanni Sartoni. Misfire and partial burn

detection based on ion current measurement. *SAE International Journal of Engines*, 4(2):2451–2460, 2011.

[12] Shouvik Dev, Navjot Singh Sandhu, Mark Ives, Shui Yu, Ming Zheng, and Jimi Tjong. Ion current measurement of diluted combustion using a multi-electrode spark plug. Technical report, SAE Technical Paper, 2018.

[13] Abhijit Abhijit and Jeffrey Naber. Ionization signal response during combustion knock and comparison to cylinder pressure for si engines. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 1(2008-01-0981): 349–364, 2008.

[14] Wang Yinhui, Huang Kaisheng, Lin Zhihua, and Meng Fanbo. Advanced gasoline engine misfire diagnostic method based on crankshaft speed multiple filtering. In *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*, pages 1964–1968. IEEE, 2011.

[15] P Azzoni, G Cantoni, G Minelli, D Moro, Giorgio Rizzoni, M Ceccarani, and S Mazzetti. Measurement of engine misfire in the lamborghini 533 v-12 engine using crankshaft speed fluctuations. *SAE transactions*, 104:1423–1429, 1995.

[16] Syed Abbas Ali and Samir Saraswati. Cycle-by-cycle estimation of cylinder pressure and indicated torque waveform using crankshaft speed fluctuations. *Transactions of the Institute of Measurement and Control*, 37(6):813–825, 2015.

[17] Terrence S Brown and W Stuart Neill. Determination of engine cylinder pressures from crankshaft speed fluctuations. *SAE transactions*, pages 771–779, 1992.

[18] F Taglialatela, M Lavorgna, E Mancaruso, and BM Vaglieco. Determination of combustion parameters using engine crankshaft speed. *Mechanical Systems and Signal Processing*, 38(2):628–633, 2013.

[19] Samir Saraswati and Satish Chand. Reconstruction of cylinder pressure for si engine using recurrent neural network. *Neural Computing and Applications*, 19 (6):935–944, 2010.

[20] Jason R. Blough. MEEM 5700- Lecture Slides, Digital Signal Processing. *Michigan Technological University*, 2017.

[21] John B Heywood et al. Internal combustion engine fundamentals. 1988.

[22] Jeffrey Naber, Jason R Blough, Dave Frankowski, Monroe Goble, and John E Szpytman. Analysis of combustion knock metrics in spark-ignition engines. Technical report, SAE Technical Paper, 2006.

[23] NG Andrew. Lecture Slides, Machine Learning. *Stanford University*, 2018.

[24] Jeffrey Naber, Jason R Blough, Dave Frankowski, Monroe Goble, and John E Szpytman. Analysis of combustion knock metrics in spark-ignition engines. Technical report, SAE Technical Paper, 2006.

[25] N Cesario, F Tagliatatela, and M Lavorgna. Methodology for misfire and partial burning diagnosis in si engines. *IFAC Proceedings Volumes*, 39(16):1024–1028, 2006.

Appendix A

Sample Code

The code below is used for ANN studies, specifically for the pseudo transient test

A.1 ANN for Pseduo-Transient code

```
% ↩
   ----------------------------------------------------------------------


% FILE: ANN_pesudo_transient.m
%
% AUTHOR(S): Nischal Muralidhar, Jeff Naber and Jason R. ↩
   Blough
%
% DESCRIPTION: Processes sensor data of ion, exhaust ↩
   pressure and crank
% sensor and then uses them to train a neural network to ↩
   predict IMEP or
% CA50
% Written specifically for and tested on Ford 2.0L ↩
   Ecoboost engine
% ↩
   ----------------------------------------------------------------------
```

```
%%
%Look out for samplig rate of Pressure trace, exhaust and↩
    ion signal, a
%number of parameters change based on that
%THIS CODE IS SPECIFIC TO THE DATASET MENTIONED
%V2 uses MAP, Fuel mass,Spark timing and RPM from CAS ↩
   instead of ATI as in V1. Cam
%timing,wastegate position,
clear all;
close all;
clc;
%% Read data from NI, ATI and CAS
Test_no= ' TEST11'; %UPDATE WHEN DATA SET CHANGES . No ↩
   space in text
test_num = 11;
test_date = 20190115;
dt = num2str(test_date);
%% Load ATI Data
     folder_loc = strcat('D:\',dt,'\ATI');
     cd (folder_loc);
    TEST_ATI = sprintf('TEST%d.xlsx', test_num);
```

```
 Data_ATI = xlsread(TEST_ATI);
time_ATI=Data_ATI(38:end,1); %timestamp
 IGN_ATI = Data_ATI(38:end,4);
 N_ATI   = Data_ATI(38:end,2);
 eng_tq = Data_ATI(38:end,6); %TORQUE SOURCE
 eng_rpm = Data_ATI(38:end,2); %RPM
 eng_ld  = Data_ATI(38:end,3); %LOAD
 eng_evt = Data_ATI(38:end,24); %VCT_angle_exh i.e. ↩
    exhaust adv or rtd
 eng_ivt = Data_ATI(38:end,25); %VCT_angle_int i.e ↩
    intake adv or rtd
 eng_spk_ca = Data_ATI(38:end,4); %Spark SAFTOT
 eng_spk_src=Data_ATI(38:end,5); %Spark source
 eng_fuel_src=Data_ATI(38:end,64); %fuel source
 Cyl2_IGN_ATI=Data_ATI(38:end,33); %Cylinder 2 SA
  eng_fuel_mg=Data_ATI(38:end,30).*453592; %fuel ↩
     supplied to Cyl2 in mg (original lbm)
  RGF=Data_ATI(38:end,117); % RGF(%) computed using ↩
     AIR_RESD and CYL_AIR_CHG
%% ATI Synchronization
 l_N_ATI = length(N_ATI);
```

```
k=0;
ATI_cal1=120/max(diff(time_ATI));
Cycle_ATI(1)=0;
for k=2:l_N_ATI
Cycle_ATI(k) = (N_ATI(k,1)/ATI_cal1)+Cycle_ATI(k-1);
end


k=0;
i=1;
for k=1:l_N_ATI
    if Cycle_ATI(k)>=i
        N_ATI_cycle(i)=N_ATI(k);
        IGN_ATI_cycle(i) = IGN_ATI(k);
        eng_tq_cycle(i) = eng_tq(k);
        eng_rpm_cycle(i) = eng_rpm(k);
        eng_ld_cycle(i)  = eng_ld(k);
        eng_evt_cycle(i) = eng_evt(k);
        eng_ivt_cycle(i) = eng_ivt(k);
        eng_spk_ca_cycle(i) = eng_spk_ca(k);
        eng_spk_src_cycle(i)=eng_spk_src(k);
        eng_fuel_src_cycle(i)=eng_fuel_src(k);
```

```
            Cyl2_SA_cycle(i)= Cyl2_IGN_ATI(k);
            eng_fuel_mg_cycle(i)=eng_fuel_mg(k);
            RGF_cycle(i)=RGF(k);
           i=i+1;
        end
    end

    ati_idx=2490;
    ATI_trigger_idx = find( abs(IGN_ATI_cycle(ati_idx:end↩
       ) - IGN_ATI_cycle(ati_idx))>=3, 1, 'first')+↩
       ati_idx;

%% Load CAS Data
    folder_loc = strcat('D:\',dt,'\CAS');       %Change ↩
       as per your location
    cd (folder_loc);
    s = num2str(test_num);                  % Converts ↩
       number to string

    basepath    = strcat(folder_loc,'\TEST',s);
    cd(basepath)                           % Goto Basepath
```

```
foo   = load('Trace.cpdc.mat');
CAS_sync = getfield(foo,char(fieldnames(foo))); %Do ↩
   not delete even if not used
clear foo;

foo   = load('CAIGN.Cyl1.EST.mat');
IGN_CAS = getfield(foo,char(fieldnames(foo))); %Do ↩
   not delete even if not used
clear foo;
eng_spk_CAS = IGN_CAS;

foo   = load('RPM.Timer.mat');
N_CAS = getfield(foo,char(fieldnames(foo)));
clear foo;
eng_rpm_CAS=N_CAS;

foo   = load('IMEP.Cyl2.mat');
Cyl2_IMEP = getfield(foo,char(fieldnames(foo)));
clear foo;
```

```
foo   = load('Trace.IonSensingCyl2.mat'); %Standalone↩
    Ion
IonSig_s = getfield(foo,char(fieldnames(foo)));
clear foo;
IonSig_s=detrend(IonSig_s);

foo   = load('Trace.Ion_Coil.mat');
IonSig_c = getfield(foo,char(fieldnames(foo)));
clear foo;

foo   = load('tqca1.mat');  % Crank angle for ↩
   cylinder 1
tqca1 = getfield(foo,char(fieldnames(foo)));
clear foo;

tqca_ion=(-179.967666625977 :0.5:539.532348632813)'; ↩
   %Ion and Exhaust sampled at 0.5CAD
tqca_cycle=(-179.967666625977:0.5:(180+900));
foo   = load('PresTrace.Cyl2.mat');
Cyl2_P = getfield(foo,char(fieldnames(foo)));
clear foo;
```

```
foo   = load('CA50.Cyl2.mat');
Cyl2_CA50 = getfield(foo,char(fieldnames(foo)));
clear foo;

foo   = load('Peak Loc.Cyl2.mat');
Cyl2_Pk_loc = getfield(foo,char(fieldnames(foo)));
clear foo;

foo   = load('Peak.Cyl2.mat');
Cyl2_Pk_amp = getfield(foo,char(fieldnames(foo)));
clear foo;

foo   = load('Knock Intensity.Cyl2.mat');
Cyl2_KI = getfield(foo,char(fieldnames(foo)));
clear foo;

foo   = load('Trace.ExhaustPressure.mat'); %Kulite
Exh_kulite = getfield(foo,char(fieldnames(foo)));
clear foo;
```

```
    foo    = load('Average.MAP.mat');
    eng_map = getfield(foo,char(fieldnames(foo)));
    clear foo;

    foo    = load('Average.Fuel_Flow.mat');
    eng_mf_CAS = getfield(foo,char(fieldnames(foo)));
    clear foo;

%Loading the crank timing data
load('trtime1.mat') % for Cyl1
load('trtime4.mat') % for Cyl3
load('trtime5.mat') % for Cyl4
load('trtime3.mat') % for Cyl2
n_cyc= size(trtime3,2); % number of cycles

%% CAS & ATI & NI Synchronization index (cycle number ↩
   allign)
    cas_idx=1990;
    CAS_ATI_trigger_idx = find( abs(IGN_CAS(cas_idx:end) ↩
       - IGN_CAS(cas_idx))>=3, 1, 'first')-1+cas_idx;
```

```
    cycle_diff_CAS_ATI = ATI_trigger_idx- ↩
       CAS_ATI_trigger_idx ;
    CAS_end_idx = length(IGN_CAS);
    resh_CPDCIn = reshape(CAS_sync,[],1);     % reshaping↩
        in one column
    CAS_tirgger_idx = find(resh_CPDCIn>0.4, 1, 'first')↩
       -1;    % index for first trigger
    CAS_tirgger_cycle = fix(CAS_tirgger_idx/size(CAS_sync↩
       ,1));  % Cycle for first trigger

ati_cycles=cycle_diff_CAS_ATI:(cycle_diff_CAS_ATI+size(↩
   IGN_CAS,2)-1); %ATI cycles that line up with CAS

%% Synchronization Check
figure;
subplot(2,1,1)
plot(-IGN_CAS, 'r', 'linewidth', 2)
hold on
plot(IGN_ATI_cycle,'b', 'linewidth', 2)
hold off
grid on;
```

```
set(gca,'FontSize',14)
title('CAS and ATI data Synchronization');
ylabel('Ignition Timing (CAATDC)')
legend('CAS','ATI')

subplot(2,1,2)
plot(-IGN_CAS,'r', 'linewidth', 2)
hold on
plot(IGN_ATI_cycle(ati_cycles),'b', 'linewidth', 2)
hold off
grid on;
set(gca,'FontSize',14)
ylabel('Ignition Timing (CAATDC)')
xlabel('Cycle')
legend('CAS','ATI')

%% Cycle wise all ATI parameters
%ASSUMPTION : ATI PARAMETERS SAME FOR ALL CYLINDERS in a ↩
   given cycle
eng_tq_cycle = eng_tq_cycle(ati_cycles);
eng_rpm_cycle = eng_rpm_cycle(ati_cycles);
```

```
eng_ld_cycle = eng_ld_cycle(ati_cycles);
eng_evt_cycle = eng_evt_cycle(ati_cycles);
eng_ivt_cycle = eng_ivt_cycle(ati_cycles);
eng_spk_ca_cycle = eng_spk_ca_cycle(ati_cycles);
eng_spk_src_cycle=eng_spk_src_cycle(ati_cycles);
eng_fuel_src_cycle=eng_fuel_src_cycle(ati_cycles);
Cyl2_SA_cycle=Cyl2_SA_cycle(ati_cycles);
eng_fuel_mg_cycle=eng_fuel_mg_cycle(ati_cycles);
RGF_cycle=RGF_cycle(ati_cycles);
clear eng_tq  eng_ld  eng_evt  eng_ivt  eng_spk_ca  wg_dc↩
    eng_spk_src  Cyl2_IGN_ATI eng_fuel_mg;
%% plot drive cycle
figure;
title(strcat('Drive profile for MTU transient test -', ↩
  num2str(test_date), num2str(Test_no)));
xlabel('Cycle number');
yyaxis left
plot(eng_rpm_cycle,'Linewidth',2);
ylabel('Engine RPM');
hold on
yyaxis right
```

```
plot(eng_ld_cycle,'Linewidth',2);
ylabel('Norm. Engine load');
legend('Speed','Load');
grid minor;

%% %Sorting Crank data
% Plotting Crank angle vs time
cyc=1;
figure;
plot (trtime1(:,cyc),tqca1)
hold on
plot (trtime4(:,cyc),tqca1)
plot (trtime5(:,cyc),tqca1)
plot (trtime3(:,cyc),tqca1)
hold off
set(gca,'fontsize',20);
ylabel ('Crankangle [deg]');
xlabel ('time [sec]');
legend ('Encd Cyl1','Encd Cyl3','Encd Cyl4','Encd Cyl2')
grid minor
title ('Crankangle vs. time')
```

```
title (['Crankangle vs. time for Cycle :' num2str(cyc) ' ↩
   Test ' num2str(Test_no)]);

%% Calculate Velocity signal for Cylinder 2
Cyl= 'Cylinder 2';
TRTIME= trtime3; % change according to above line

% Computing the rpm (slope of the above plot)
dCA= diff(tqca1);
dTime1= diff (TRTIME);
dCA_rep= repmat(dCA,1,n_cyc);
crnk_rpm= (dCA_rep./dTime1)*60/360;
cyc_rpm=1500;%cycle for which RPM is plotted

figure;
plot (tqca_ion(1:end-1),crnk_rpm(:,cyc_rpm))
ylabel ('rpm');
grid minor
title (['rpm vs. time for ' num2str(Cyl) ' Cycle :' ↩
   num2str(cyc_rpm) ' Test ' num2str(Test_no)]);
```

```
%% %Order analysis of crank data
crnk_rpm=[crnk_rpm; crnk_rpm(end,:)];%insert an element ↩
   in the last row of all columns to make length 1440
signal=crnk_rpm;
block_count=size(crnk_rpm,2);
N=360*4;
D_Theta= 0.5;
fs= 360/ D_Theta; %Osample
T=N*D_Theta; %R
delf=fs/N; %del_o
win_len=N;
win=hanning(win_len);
ACF=1/mean(win);
win_mat=repmat(win,1,block_count);
sig_fft=fft(signal.*win_mat);
pos_fft=(1/N).*[sig_fft(1,:); (2*sig_fft(2:N/2,:))].*ACF;
amp_sig=abs(pos_fft');
phase_sig=angle(pos_fft');
freq=(0:length(pos_fft(:,1))-1)'*fs/N; %Order resolution ↩
   is 0.5
%%
```

```
figure;
subplot(2,1,1)
label='start-end ';
pcolor(freq,1:block_count,log10(amp_sig));
colormap default;
shading('interp');
set(gca,'fontsize',12)
xlabel('Order');
ylabel('Event index');
set(gca,'Ydir','reverse')
zlabel('Amplitude');
xlim([0 20]);
set(gca,'XTick',0:2:20);
title([strcat('Cyl2 crnk RPM, Cyc: ',label ,', ', num2str↩
   (test_date),' ',num2str(Test_no))]);
view([-90 90]);

subplot(2,1,2)
plot(eng_rpm_cycle,'Linewidth',2);
title(['Drive profile for MTU transient test -' num2str(↩
   Test_no)]);
```

```
xlabel('Cycle number');
ylabel('Engine RPM');
hold on
yyaxis right
plot(eng_ld_cycle,'Linewidth',2);
ylabel('Norm. Engine load');
legend('Speed','Load');
grid minor
xlim([1 block_count])
%%
%Amplitude and phase of 2nd,3rd and 4th order, cylinder 2↩
    crank RPM data.
%CROSS CHECK ORIENTATION OF POS FFT. SUM LINEAR SPECTRA, ↩
   THEN TAKE ABS OR
%PHASE
pos_fft=pos_fft'; %each cycle in a row
%Amplitude
ANN_crnk_rpm_2or_amp=abs(sum(pos_fft(:,5),2));
ANN_crnk_rpm_3or_amp=abs(sum(pos_fft(:,7),2));
ANN_crnk_rpm_4or_amp=abs(sum(pos_fft(:,9),2));
ANN_crnk_rpm_6or_amp=abs(sum(pos_fft(:,13),2));
```

```
%Phase
ANN_crnk_rpm_2or_ang=angle(sum(pos_fft(:,5),2));
ANN_crnk_rpm_3or_ang=angle(sum(pos_fft(:,7),2));
ANN_crnk_rpm_4or_ang=angle(sum(pos_fft(:,9),2));
ANN_crnk_rpm_6or_ang=angle(sum(pos_fft(:,13),2));
%% %Ion first peak and peak location

%flame front and ringing detection
end_angle =110;
thresh_SA=1; %V
thresh_SC=1;
k=1;
for i=1:n_cyc
[~,str_idx]= (min(abs(tqca1-(-1*eng_spk_ca_cycle(cyc)))))↩
   ;
[~,end_idx]= (min(abs(tqca1-end_angle)));

[temp,~] = find(IonSig_s(str_idx:end_idx,i)>thresh_SA,1,'↩
   first'); %find the first instance in the define CA ↩
   range when ion rise above 0.5V
    if isempty(temp)
```

```
    rise_idx_SA(i)=0; %if no element is found assign ↩
       spark timing so eventually we''l have pointer at ↩
       Spk timing for this cyc
     miss_cyc_SA(k)=i; %missing cyc
    k=k+1;
    else
       rise_idx_SA(i)= temp;
    end

[temp,~] = find(IonSig_c(str_idx:end_idx,i)>thresh_SC,1,'↩
   first');
    if isempty(temp)
    rise_idx_SC(i)=0; %if no element is found assign ↩
       spark timing so eventually we''l have pointer at ↩
       Spk timing for this cyc
    miss_cyc_SC(k)=i; %missing cyc
    k=k+1;

    else
        rise_idx_SC(i)= temp;
    end
```

```
rise_idx_SA(i)=rise_idx_SA(i)+str_idx; %Get the correct ↩
   index wrt to tqca1
ANN_rise_angle_SA(i)=tqca1(rise_idx_SA(i));

rise_idx_SC(i)=rise_idx_SC(i)+str_idx; %Get the correct ↩
   index wrt to tqca1
ANN_rise_angle_SC(i)=tqca1(rise_idx_SC(i));

end

%% %Standalone Ion - find peaks in cycles with normal ↩
   combustion
%Location and amplitude of Ion peak
for cyc=1:n_cyc
      flag=ismember(cyc,miss_cyc_SA);

    if flag==1 %if abnomral cycle assign zero
     ANN_Ion_s_pk_amp(cyc)=0;
    ANN_Ion_s_pk_loc(cyc)=0;
```

```
    else
    [pkt,lct] = findpeaks(IonSig_s(:,cyc),tqca1,'NPeak↩
       ',1,'MinPeakWidth',3,'MinPeakHeight',1,'↩
       MinPeakProminence',0.5,'WidthReference','halfprom↩
       ');
    ANN_Ion_s_pk_amp(cyc)=pkt;
    ANN_Ion_s_pk_loc(cyc)=lct;


    end


end
%% %For coil ion start from opposite side and find a peak↩
   . First flip the
%signal then find a peak and location, then convert ↩
   location to usual conv
%Location and amplitude of Ion peak
k=1;
for cyc=1:n_cyc
      flag=ismember(cyc,miss_cyc_SA);
      [~,str_idx]= (min(abs(tqca1-(-1*eng_spk_ca_cycle(↩
         cyc)))));
```

```
if flag==1 %if abnomral cycle assign zero
 ANN_Ion_c_pk_amp(cyc)=0;
ANN_Ion_c_pk_loc(cyc)=0;

else
   [pkt,lct] = findpeaks(IonSig_c(str_idx:end,cyc),'↩
     MinPeakHeight',1.5,'MinPeakWidth',5,'↩
     MinPeakProminence',1,'SortStr','descend');
    if isempty(pkt)%cycles where actual coil peak in ↩
      negligible use ringing peak
    clear pkt lct;
    ANN_Ion_c_pk_amp(cyc)=IonSig_c(rise_idx_SC(cyc));
    ANN_Ion_c_pk_loc(cyc)=tqca1(rise_idx_SC(cyc));
    mark(k)=cyc;
    k=k+1;

    elseif size(pkt,1)==2
    ANN_Ion_c_pk_amp(cyc)=pkt(2); %use the largest ↩
      peak as odds are this is first peak that isn't↩
       ringing
    ANN_Ion_c_pk_loc(cyc)=tqca1(str_idx+lct(2));
```

```
        elseif size(pkt,1)==1
        ANN_Ion_c_pk_amp(cyc)=pkt(1); %use the largest ↩
           peak as odds are this is first peak that isn't↩
            ringing
        ANN_Ion_c_pk_loc(cyc)=tqca1(str_idx+lct(1));


        end


    end

end

%% %Exhaust Pressure data
Exh_1col=Exh_kulite(:); %puts one cycle after the other

% This section puts the exhaust data in a proper form
% Reshape the exhaust pressure data into one column for ↩
   each cycle. From the start we next
% 7 blocks of data to see the exhuast event for cycle 1. ↩
   Each block has data
```

```
% for 180 degrees. however we cannot see the last cycle ↩
   exhaust data
k=1;
for i=1:size(Exh_kulite,2)-1

ExhPres_K_EVO_cycle(:,i)=Exh_1col(k:k+7*180*2-1); %need 7↩
    blocksbut for safe measure taking an 8th block
k=k+(4*180*2);
end

%%% Building a filter
D_Theta=0.5;
filt_order= 8; % order of filter to be built
fc = 15; % cut-off frequency, selected based on linear ↩
   spectrum plot
fs = 360/ D_Theta; % sampling rate
[b,a] = butter(filt_order,fc/(fs/2)); % building the ↩
   filter
figure
freqz(b,a) % plotting the filter
%%% Applying the filter
```

```
for ii=1:size (ExhPres_K_EVO_cycle,2)
    % Two-way filtering for each cycle
ExhPres_filt(:,ii) = filtfilt(b,a,ExhPres_K_EVO_cycle(:,↩
   ii));
end
%%% Changing variable name to use filtered signal
ExhPres_K_EVO_cycle_abs=ExhPres_filt;
ExhPres_K_EVO_cycle=detrend(ExhPres_K_EVO_cycle_abs); %↩
   detrend removes any trend lines and offsets in signal
%the for loop below tries to center the signal about zero↩
    so as to ease
%processing in misfire identification and waveform ↩
   classification

%Arrange exh such that each column has 360 data points. ↩
   start from
%cylinder 1 EVO of first cycle in dataset which is the ↩
   648th data point i.e
% to get to TDC 180 pts, EVO is 144 from there, sampling ↩
   rate is 0.5CAD so
% 2(144+180)=648
```

```
raw_exh_one_col=Exh_1col(2*(180+144):end);
max_iter=floor(size(raw_exh_one_col,1)/(2*2*360));

%cut into blocks of 1 cycles
k=1;
for i=1:max_iter

raw_exh_oa_all(:,i)=raw_exh_one_col(k:k+2*2*360-1); ↩
   %2*360 will take each block from Cyl1 EVO to just ↩
   before...
%start of Cyl 1 EVO in next cycle
k=k+(2*2*360);
end
%%

%Stitch them back together and cut into blocks of size ↩
   360*4*2 or 2 cycles
%sampled at 0.5CAD
raw_exh_one_col=raw_exh_oa_all(:);
max_iter=floor(size(raw_exh_one_col,1)/(4*2*360));
k=1;
```

```
raw_exh_oa=[];
for i=1:max_iter

raw_exh_oa(:,i)=raw_exh_one_col(k:k+4*2*360-1);
k=k+(4*2*360);
end

%Stitch them back together and cut into blocks of size ↩
   360*2*2 or 1 cycle
%sampled at 0.5CAD
raw_exh_one_col=raw_exh_oa_all(:);
max_iter=floor(size(raw_exh_one_col,1)/(2*2*360));
k=1;
raw_exh_oa2=[];
for i=1:max_iter

raw_exh_oa2(:,i)=raw_exh_one_col(k:k+2*2*360-1);
k=k+(2*2*360);
end

%%
```

```
%Create blocks such that window lines up with peak of ↩
   exhaust pressure
%Each block need to be 2 cycles long. if sampling at 0.5↩
   CA thats 2880pts
cycles=1:ati_cycles(end);
cyc_name='1:end';
label=' start to finish';
cont_signal=raw_exh_oa(:); %Puts one column below the ↩
   other so exhaust events will be continuous
rpm_2=eng_rpm(cycles)'; %Each row has the Cyl RPM in ↩
   firing order
 %Vector of RPM in firing order
i=[];
k=1;
for i=1:size(cycles,2)
    cont_rpm(k:k+4)=eng_rpm(cycles(i));
    k=k+4;
end

%We will be using bocks of size 2880. In the very first ↩
   block to make the window line up with exhaut peak of
```

```
%Cylinder 1 we need to start block from 333 data point in↩
    cont_signal. The next block should have the window
%line up with next exhaust peak,to achive this we shift ↩
   the block captured by 360 pts and take 2880 data point
cont_signal=cont_signal(324:end);
%First lets find out how many block we can get from the ↩
   cycles
block_count=fix((length(cont_signal)-2880)/(180*2))+1; %↩
   block count is the number of block of data we can get
%from cont_signal. fix gives the quotient. 180*2 cuz we'↩
   re sampling at
%0.5CAD and 360 datapoints will be exhaust event of one ↩
   cylinder

%For loop below puts each block in a cloumn, then shifts ↩
   180 pts and takes
%another block
k=0;
signal=zeros(2880,block_count);
for i=1:block_count %for loop goes from second block
   str_idx=(180*2*k);
```

```
    if str_idx==0
      signal(:,i)=cont_signal(1:2880);%first block
    else
    block=cont_signal(str_idx:str_idx+2879);
    signal(:,i)=block; %Each block will go into a column
    end
    k=k+1;
end
%%

N=360*4*2;
D_Theta= 0.5;
fs= 360/ D_Theta; %Osample
T=N*D_Theta; %R
delf=fs/N; %del_o
win_len=N;
%win=hanning(win_len);
win=tukeywin(win_len,0.5);
%win=[zeros(869+13+10,1); tukeywin(1084,0.5); zeros↩
   (927-13-10,1)];
ACF=1/mean(win);
```

```
win_mat=repmat(win,1,block_count);
sig_fft=fft(signal.*win_mat);
pos_fft=(1/N).*[sig_fft(1,:); (2*sig_fft(2:N/2,:))].*ACF;
amp_sig=abs(pos_fft');
phase_sig=angle(pos_fft');
freq=(0:length(pos_fft(:,1))-1)'*fs/N;
k=1;
%NOTE : Look at the next figure, if the exhaust block is ↩
   first cut starting
%from Cyl1 EVo and we go 333 points in as done in code ↩
   above, the window
%will line up wih the peak of cylinder 1
%thus for loop below is 1-3-4-2
%Sort cylinderwise amp
cyc=1;
for i=1:4:block_count
  if cyc<=floor(block_count/4)
    %In firing order
   Cyl1_exh_ls(:,k)=pos_fft(:,i);
   Cyl1_oa(k,:)=amp_sig(i,:);
```

```
    Cyl3_exh_ls(:,k)=pos_fft(:,i+1);
    Cyl3_oa(k,:)=amp_sig(i+1,:);

    Cyl4_exh_ls(:,k)=pos_fft(:,i+2);
    Cyl4_oa(k,:)=amp_sig(i+2,:);

    Cyl2_exh_ls(:,k)=pos_fft(:,i+3);
    Cyl2_oa(k,:)=amp_sig(i+3,:);

    k=k+1;
    cyc=cyc+1;
    end
end

%%
%Plot a block and also the window applied
index=(1:1:N)';
figure;
yyaxis left
plot(index,signal(:,3200),'k');
hold on
```

```
plot(index,signal(:,3200),'b');
ylabel('Exh Pressure(kPa)');
xlabel('index');
xlim([1 2880]);
yyaxis right
plot(index,win,'red');
ylabel('Window amplitude');
title('Block and window used in order analysis');
blah=[zeros(869+13+10,1); hann(1084); zeros(927-13-10,1)↩
   ];
plot(index,blah,'-black');
legend('Cyl1, Cycle 757','Cyl2, Cycle 757','Window-new','↩
   Window-old');

%%
%Cyl2  waterfall
figure;
waterfall(freq,1:floor(block_count/4),log10(Cyl2_oa));
xlabel('Order')
ylabel('Event index');
zlabel('Amplitude');
```

```
xlim([0 20]);
set(gca,'XTick',0:2:20);
title([strcat('Cylinder 2-Cycles: ', num2str(cyc_name),↩
   label)]);

%%
%Cyl 2 colormap in order of occurence

cycles=1:ati_cycles(end);
cyc_name='1:end';
label=' start to finish';

figure;
subplot(2,1,1)
pcolor(freq,1:block_count/4,log10(Cyl2_oa));
colormap default;
shading('interp');
set(gca,'fontsize',12)
xlabel('Order');
ylabel('Event index');
set(gca,'Ydir','reverse')
```

```
zlabel('Amplitude');
xlim([0 20]);
set(gca,'XTick',0:2:20);
title([strcat('Cyl2 Exh Colormap,Cycles: ', num2str(↩
   cyc_name),label)]);
view([-90 90]);


xval=1:n_cyc;
subplot(2,1,2)
plot(xval,eng_rpm_cycle,'Linewidth',2);
title(['Drive profile for MTU transient test -' num2str(↩
   Test_no)]);
xlabel('Cycle number');
ylabel('Engine RPM');
xlim([1 n_cyc]);
hold on
yyaxis right
plot(xval,eng_ld_cycle,'Linewidth',2);
ylabel('Norm. Engine load');
legend('Speed','Load');
%%
```

```
%Plot amplitudes of orders 2,3,4 for all cylinders
figure;
subplot(3,1,1);
plot(log10(sum(Cyl1_oa(:,8:10),2)),'r'); %Cylinder1 order↩
    1, sum across 3 bins for all events
hold on
plot(log10(sum(Cyl3_oa(:,8:10),2)),'g');
plot(log10(sum(Cyl4_oa(:,8:10),2)),'b');
plot(log10(sum(Cyl2_oa(:,8:10),2)),'k');
legend('Cyl 1','Cyl 3','Cyl 4','Cyl 2');
title([strcat('2nd order.', label,' Cycles:',num2str(↩
   cyc_name))]);
ylabel('Amplitude (log)');
xlabel('Cycle number');
subplot(3,1,2);
plot(log10(sum(Cyl1_oa(:,12:14),2)),'r'); %Cylinder 1 ↩
   order 3
hold on
plot(log10(sum(Cyl3_oa(:,12:14),2)),'g');
plot(log10(sum(Cyl4_oa(:,12:14),2)),'b');
plot(log10(sum(Cyl2_oa(:,12:14),2)),'k');
```

```
legend('Cyl 1','Cyl 3','Cyl 4','Cyl 2');
title([strcat('3rd order.', label,' Cycles:',num2str(↩
   cyc_name))]);
ylabel('Amplitude (log)');
xlabel('Cycle number');
subplot(3,1,3);
plot(log10(sum(Cyl1_oa(:,16:18),2)),'r'); %Cylinder 1 ↩
   order 5
hold on
plot(log10(sum(Cyl3_oa(:,16:18),2)),'g');
plot(log10(sum(Cyl4_oa(:,16:18),2)),'b');
plot(log10(sum(Cyl2_oa(:,16:18),2)),'k');
legend('Cyl 1','Cyl 3','Cyl 4','Cyl 2');
title([strcat('4th order.', label,' Cycles:',num2str(↩
   cyc_name))]);
ylabel('Amplitude (log)');
xlabel('Cycle number');
%%
%Preprocess input for ANN - Done here only for Cylinder 2
% We input Engine Speed, load, MAP, SA, Intake and ↩
   Exhaust Cam phasing,
```

```
%location of waste gate, 2, 3, and 4 order amplitudes and↩
    phases of exhaust pressure
clear ('ANN_IP','ANN_OP');
%Inputs
ls=Cyl2_exh_ls';

%Amplitude
ANN_Cyl2_2or_amp=abs(sum(ls(:,8:10),2));
ANN_Cyl2_3or_amp=abs(sum(ls(:,12:14),2));
ANN_Cyl2_4or_amp=abs(sum(ls(:,16:18),2));
%phase
ANN_Cyl2_2or_ang=angle(sum(ls(:,8:10),2));
ANN_Cyl2_3or_ang=angle(sum(ls(:,12:14),2));
ANN_Cyl2_4or_ang=angle(sum(ls(:,16:18),2));

cyc_of_orders=size(Cyl2_oa,1);

ANN_eng_rpm=eng_rpm_CAS(2:cyc_of_orders+1)';
ANN_eng_ld=eng_ld_cycle(2:cyc_of_orders+1)';
ANN_eng_ivt=eng_ivt_cycle(2:cyc_of_orders+1)';
ANN_eng_evt=eng_evt_cycle(2:cyc_of_orders+1)';
```

```
ANN_cyl2_SA=eng_spk_CAS(2:cyc_of_orders+1)';
ANN_eng_map=eng_map(2:cyc_of_orders+1)';
ANN_eng_fuel_mg=eng_mf_CAS(2:cyc_of_orders+1)';

ANN_Ion_s_pk_amp=ANN_Ion_s_pk_amp(2:cyc_of_orders+1)';
ANN_Ion_s_pk_loc=ANN_Ion_s_pk_loc(2:cyc_of_orders+1)';
ANN_rise_angle_SC=ANN_rise_angle_SC(2:cyc_of_orders+1)';
ANN_rise_angle_SA=ANN_rise_angle_SA(2:cyc_of_orders+1)';

ANN_crnk_rpm_2or_amp=ANN_crnk_rpm_2or_amp(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_3or_amp=ANN_crnk_rpm_3or_amp(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_4or_amp=ANN_crnk_rpm_4or_amp(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_6or_amp=ANN_crnk_rpm_6or_amp(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_2or_ang=ANN_crnk_rpm_2or_ang(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_3or_ang=ANN_crnk_rpm_3or_ang(2:cyc_of_orders↩
   +1);
```

```
ANN_crnk_rpm_4or_ang=ANN_crnk_rpm_4or_ang(2:cyc_of_orders↩
   +1);
ANN_crnk_rpm_6or_ang=ANN_crnk_rpm_6or_ang(2:cyc_of_orders↩
   +1);
%%
% ANN_IP=[ANN_eng_rpm ANN_eng_map ANN_cyl2_SA ANN_eng_ivt↩
    ANN_eng_evt ANN_wg_dc ANN_eng_fuel_mg...
%     ANN_Cyl2_2or_amp ANN_Cyl2_3or_amp ANN_Cyl2_4or_amp ↩
   ...
%     ANN_Cyl2_2or_ang ANN_Cyl2_3or_ang ANN_Cyl2_4or_ang↩
   ...
%     ANN_Ion_s_pk_amp ANN_Ion_s_pk_loc ANN_rise_angle_SA↩
   ...
%     ANN_crnk_rpm_2or_amp ANN_crnk_rpm_3or_amp ↩
   ANN_crnk_rpm_4or_amp ANN_crnk_rpm_6or_amp...
%     ANN_crnk_rpm_2or_ang ANN_crnk_rpm_3or_ang ↩
   ANN_crnk_rpm_4or_ang ANN_crnk_rpm_6or_ang];

ANN_IP=[ANN_Ion_s_pk_loc ANN_rise_angle_SA];
%%
%Output
```

```
ANN_Cyl2_IMEP=Cyl2_IMEP(2:cyc_of_orders+1)';
ANN_Cyl2_CA50=Cyl2_CA50(2:cyc_of_orders+1)';

%ANN_OP=[ANN_Cyl2_IMEP ANN_Cyl2_CA50];

ANN_OP=[ANN_Cyl2_IMEP ANN_Cyl2_CA50];
%% %Remove abmormal cycles

miss_cyc_SA_corr=miss_cyc_SA-1;%becasue the latest matrix↩
    starts from cycles two, the cycle index in ↩
   miss_cyc_SA
%will be one index less than actual index
for cyc=1:cyc_of_orders
flag=ismember(cyc,miss_cyc_SA_corr);
if flag==1
    ANN_IP(cyc,:)=[];
    ANN_eng_rpm(cyc)=[]; %needed for color coding later
    ANN_OP(cyc,:)=[];
end
end
```

```
%% Recursive neural net

%ANN_IP=[ANN_Cyl2_2or_amp ANN_Cyl2_3or_amp ↩
   ANN_Cyl2_4or_amp ...
%    ANN_Cyl2_2or_ang ANN_Cyl2_3or_ang ANN_Cyl2_4or_ang];
%ANN_OP=[ANN_Cyl2_IMEP];

x = tonndata(ANN_IP,false,false);
t = tonndata(ANN_OP,false,false);

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for ↩
   challenging problems.
% 'trainscg' uses less memory. Suitable in low memory ↩
   situations.
trainFcn = 'trainbr';  % Bayesian Regularization ↩
   backpropagation.

net = layrecnet(1:2,20,trainFcn);
```

```
[Xs,Xi,Ai,Ts] = preparets(net,x,t);
net = train(net,Xs,Ts,Xi,Ai);

% Setup Division of Data for Training, Validation, ↩
   Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

ANN_OP_resul2 = net(Xs,Xi,Ai);
ANN_OP_resul=cell2mat(ANN_OP_resul2);
perf = perform(net,ANN_OP_resul2,Ts);

% Test the Network
ANN_error=ANN_OP(1:end-2,:)'-ANN_OP_resul;

% View the Network
view(net)
```

```
% Plots

figure, plotperform(tr)

%figure, plottrainstate(tr)

%figure, ploterrhist(ANN_error)

%figure, plotregression(t,ANN_OP_resul)

%figure, plotfit(net,x,t)

cycles=1:size(ANN_OP_resul,2);
```

A.2 Ion knock detection code

```
% FILE: Ion_Knock_Detection.m
% AUTHOR(S): Nischal Muralidhar, Jeff Naber and Jason R. ↩
   Blough
% DESCRIPTION: Processes Ion sensor to evaluated knock ↩
   detection capability
% Written specifically for and tested on Ford 2.0L ↩
   Ecoboost engine
close all; clear all; clc
%% Loading the data
Test_no= ' TEST 2'; %UPDATE WHEN DATA SET CHANGES . No ↩
   space in text
test_num = 2;
test_date = 20181011;
dt = num2str(test_date);
%% ATI Data for Cyl 2 Ign
dt = num2str(test_date);
```

```
    folder_loc = strcat('M:\jnaber_ford\2.0L Ford Metal ↩
       Engine\Test Data\',dt,'\ATI');       %Change as ↩
       per your location
    cd (folder_loc);
    TEST_ATI = sprintf('TEST%d.xlsx', test_num);
    Data_ATI = xlsread(TEST_ATI);
    IGN_ATI = Data_ATI(38:end,6); %Cyl1 SPK_ADV
    Cyl2_IGN_ATI=Data_ATI(38:end,46); %CHECK COLUMN for ↩
       SPK_ADV[3] in ATI file %Cyl 2 SPK_ADV Another ↩
       Option is Column 54
%%
% Load CAS Data
    folder_loc = strcat('M:\jnaber_ford\2.0L Ford Metal ↩
       Engine\Test Data\',dt,'\CAS');       %Change as ↩
       per your location
    s = num2str(test_num);                 % Converts ↩
       number to string
    basepath    = strcat(folder_loc,'\TEST',s);
    cd(basepath)
load('Trace.IonSensingCyl2.mat');
load('Trace.IonCoil.mat');
```

```
load('tqca3.mat'); %CHECK which tqca size matches
tqca1=tqca3-540;
load('IMEP.Cyl2.mat');
load('CA50.Cyl2.mat');
load('Knock.Cyl2.mat');
load('PresTrace.Cyl2.mat');
load('Knock FKI4.Cyl2.mat') % Knock FKI4 for cylinder 2
load('Knock Intensity.Cyl2.mat') % Knock intensity for ↩
   cylinder 2
load('Knock Amplitude.Cyl2.mat') % Knock amplitude for ↩
   cylinder 2
%% _s - standalone ion probe, _c - coil ion probe
IonSig_ssf= IonSensingCyl2Trace;
IonSig_csf= IonCoilTrace;
PresTrace= Cyl2PresTrace;
IMEP= Cyl2IMEP;
CA50=Cyl2CA50;
Knock_inten=Cyl2KnockIntensity;
Knock_amp=Cyl2KnockAmplitudeRT;
Knock=Cyl2Knock;
num_cyc= size(IMEP,2);
```

```
cycles= 1: num_cyc;
%MAx and Min Knock intensity cycles
[min_knock_inten,min_knock_cyc]= min(Knock_inten);
[max_knock_inten,max_knock_cyc]= max(Knock_inten);
%Max and Min Knock Amplitudes
[min_knock_amp,min_knock_amp_cyc]= min(Knock_amp);
[max_knock_amp,max_knock_amp_cyc]= max(Knock_amp);
%% Display Test Parameters
load('IMEP.Cyl1.mat')
load('CA50.Cyl1.mat')
load('CAIGN.Cyl1.EST.mat')
load('RPM.Timer.mat');
load('IMEP.EA.mat') % for COV calc
Rpm= mean(RPM);
Rpm= round(Rpm/500)*500
IMEP= mean(Cyl2IMEP(1:end))
STD_IMEP= std(Cyl2IMEP(1:end))
CA50= mean(Cyl2CA50(1:end))
ST= mean (Cyl2_IGN_ATI) % Use Cyl2 Spark from ATI for ↩
   Knock tests
mean_IMEP=mean(EAIMEP);
```

```
std_IMEP= std(EAIMEP);
COV= std_IMEP/mean_IMEP*100
basepath=strcat('\\homes.mtu.edu\home\Desktop\Sensor ↩
   Fusion_2018\Knock_studies_Nis');
cd(basepath);
%% Plotting
cyc= 144 %max_knock_cyc; % UPDATE BASED ON WHICH CYCLE ↩
   YOU WANT THE FUNC TO PLOT
%Had to create an encoder signal for ion data as the ↩
   tqca1 has 1840 point to
%account for heigher resolution around the knock region
tqca_ion=(-179.967666625977 :0.5:539.532348632813)';
spark_pt=mean(ST); %If Knock Test use Cyl 2 Spark from ↩
   ATI not Cyl1 spark from CAS
[~,spark_pt_idx]= (min(abs(tqca1-(-1*spark_pt))));
figure;
subplot(2,1,1);
plot(tqca1,PresTrace(:,cyc));
ylabel ('In-cylinder Pressure (bar)');
xlim([-30 70])
set(gca,'XTick',[-180 : 15 : 540]);
```

```
xlabel ('Crankangle [deg]');
title (['Raw Cyl Pressure. Data: ' num2str(Test_no) ' ↩
   Cycle :' num2str(cyc)]);
grid minor;
subplot(2,1,2);
plot(tqca1,IonSig_ssf(:,cyc),'-black');
hold on
plot(tqca1,IonSig_csf(:,cyc),'-red');
plot(tqca1(spark_pt_idx),0,'o','LineWidth',2,'↩
   MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize↩
   ',8);
ylim([0 10])
xlim([-30 70])
legend ('Standalone Ion probe signal','Coil Ion probe ↩
   signal','Spark timing:ATI');
set(gca,'XTick',[-180 : 15 : 540]);
xlabel ('Crankangle [deg]');
ylabel ('Ion signal - [V]');
title (['Raw Ion signal. Data: ' num2str(Test_no) ' Cycle↩
    :' num2str(cyc)]);
grid minor;
```

```
%% Log normal Probability density function for FKI4 and ↩
   Knock intensity
x=(sort(Cyl2KnockIntensity));
pct_95=prctile(x,95);
x=x./pct_95;
mu=mean(log(x));
sigma=std(log(x));
pdf=lognpdf(x,mu,sigma);
cdf=logncdf(x,mu,sigma);
figure;
histogram(x,30,'Normalization','pdf');
ylabel('PDF');
hold on;
plot(x,pdf,'LineWidth',2);
yyaxis right
plot(x,cdf,'LineWidth',2);
ylabel('CDF');
title(strcat('PDF of Knock Intensity :',Test_no));
xlabel('Knock Intensity (bar)');
clear('x','mu','sigma');
x=sort(Cyl2KnockFKI4RT);
```

```
mu=mean(x);
sigma=std(x);
FKI4_pdf=pdf('normal',x,mu,sigma);
figure
plot(x,FKI4_pdf,'LineWidth',2)
title(strcat('PDF of FKI4 :',Test_no));
xlabel('FKI4 rating');
ylabel('Probability density');
clear x;
%%
%Look where to place window so that flame front doesn't ↩
   mess with Knock detection in standalone ion
%Detection of Flame front on Standalone Ion
%Check Excel sheet "Variable settings for Knock test ↩
   codes.xlsx" for
%variable values for each test
str_angle =-15; %CAD ATDC
end_angle =60;
thresh_SA=2; %V
thresh_SC=2;
[~,str_idx]= (min(abs(tqca1-str_angle)));
```

```
[~,end_idx]= (min(abs(tqca1-end_angle)));
for i=1:num_cyc
[rise_idx_SA(i),~] = find(IonSig_ssf(str_idx:end_idx,i)>↩
   thresh_SA,1,'first'); %find the first instance in the ↩
   define CA range when ion rise above 0.5V
[rise_idx_SC(i),~] = find(IonSig_csf(str_idx:end_idx,i)>↩
   thresh_SC,1,'first');
end
rise_idx_SA=rise_idx_SA+str_idx; %Get the correct index ↩
   wrt to tqca1
rise_angle_SA=tqca1(rise_idx_SA);
rise_idx_SC=rise_idx_SC+str_idx; %Get the correct index ↩
   wrt to tqca1
rise_angle_SC=tqca1(rise_idx_SC);
%Look at the Cycles
plot_cyclewise_flamefront %Press any keep to look at ↩
   cycles in loop
%% Upsample of Pressure trace
idx_Hres_start= 301; % starting index of high res crank ↩
   data
idx_Hres_end= 701; % ending index of high res crank data
```

```
%Pressure
D_Theta_press= 0.5/2; % For Knocking data
Fs_press= 360/ D_Theta_press;
%Ion
D_Theta= 0.5;
Fs= 360/ D_Theta; % sampling rate = 0.5 deg so for 360 ↩
   degs (1 rev) we have 720 orders
% Upsampling PresTrace
PresTrace_up1= upsample(PresTrace(1:idx_Hres_start,:),2);↩
    % staring 0.5 deg interval
PresTrace_up2= upsample(PresTrace(idx_Hres_end:end,:),2);↩
    % ending 0.5 deg interval
PresTrace_up= [PresTrace_up1(1:end-2,:);PresTrace(↩
   idx_Hres_start:idx_Hres_end-1,:);PresTrace_up2(1:end↩
   ,:)]; %upsampled PresTrace
PresTrace_up_b = lowpass(PresTrace_up,0.45*Fs*Rpm./60,↩
   Fs_press*Rpm./60);
IonSig_ssf1= upsample(IonSig_ssf(1:idx_Hres_start,:),2); ↩
   % staring 0.5 deg interval
IonSig_ssf2= upsample(IonSig_ssf(idx_Hres_end:end,:),2); ↩
   % ending 0.5 deg interval
```

```
IonSig_ssf_up= [IonSig_ssf1(1:end-2,:);IonSig_ssf(↩
   idx_Hres_start:idx_Hres_end-1,:);IonSig_ssf2(1:end,:)↩
   ]; %upsampled PresTrace
IonSig_ssf_up_b = lowpass(IonSig_ssf_up,0.45*Fs*Rpm./60,↩
   Fs_press*Rpm./60);
IonSig_csf1= upsample(IonSig_csf(1:idx_Hres_start,:),2); ↩
   % staring 0.5 deg interval
IonSig_csf2= upsample(IonSig_csf(idx_Hres_end:end,:),2); ↩
   % ending 0.5 deg interval
IonSig_csf_up= [IonSig_csf1(1:end-2,:);IonSig_csf(↩
   idx_Hres_start:idx_Hres_end-1,:);IonSig_csf2(1:end,:)↩
   ]; %upsampled PresTrace
IonSig_csf_up_b = lowpass(IonSig_csf_up,0.45*Fs*Rpm./60,↩
   Fs_press*Rpm./60);
%% Band pass filter the IncylP, standalone Ion and coil ↩
   ion between 5-8kHz . Plot the bandpassed and origianl.↩
    Then do Frequcney analysis
tqca_p=-180:0.25:539.75;
tq_len=length(tqca_p);
winlen_CAD=35; %Length of window in CAD
```

```
win_offst_SA=8; %2; %CAD offset of window from flame ↩
   detection CAD
win_offst_SC=11; %6;
d_theta=0.25;
steps=1/d_theta;
recwin_str_idx=781; %index 15 deg ATDC in tqca_p
% find the crank angle where flame starts and build ↩
   appropriate window
for i=1:num_cyc
    %Standalone ion window
[~,flame_idx(i)]= (min(abs(tqca_p-rise_angle_SA(i)))); %↩
   find flame front index
s_win_str_idw(i)=win_offst_SA*steps+flame_idx(i);
str_zero=length(1:s_win_str_idw(i)); %Number of zeros to ↩
   pad at start
win_len=length(s_win_str_idw(i):s_win_str_idw(i)+steps*↩
   winlen_CAD); %Tukey window of defined size
fin_zero=tq_len-str_zero-win_len; %Number of zeros to pad↩
    at end
win_mat(:,i)= [zeros(str_zero,1);tukeywin(win_len,0.5);↩
   zeros(fin_zero,1)];
```

```
%Window for Coil Ion - similar to Standalone
[~,flame_idx(i)]= (min(abs(tqca_p-rise_angle_SC(i)))); %↩
   find flame front index
c_win_str_idw(i)= win_offst_SC*steps+flame_idx(i);
str_zero=length(1:c_win_str_idw(i)); %Number of zeros to ↩
   pad at start
win_len=length(c_win_str_idw(i):c_win_str_idw(i)+steps*↩
   winlen_CAD); %Tukey window of defined size
fin_zero=tq_len-str_zero-win_len; %Number of zeros to pad↩
    at end
win_mat_SC(:,i)= [zeros(str_zero,1);tukeywin(win_len,0.5)↩
   ;zeros(fin_zero,1)];
%Calc the size of the rect window for the Pressure signal↩
   . Rect win to
%extend from earliest flame detection CAD to a length of ↩
   60 CAD
    if flame_idx(i) < recwin_str_idx
    recwin_str_idx=flame_idx(i)+6*steps;
    end
end
%DND
```

```
int_len=50; %length in CAD across which to integrate for ↩
   PI and II later
str_zero=length(1:recwin_str_idx); %Number of zeros to ↩
   pad at start
win_len=length(recwin_str_idx:recwin_str_idx+steps*↩
   int_len); %Tukey window of defined size
fin_zero=tq_len-str_zero-win_len;
rect_win= [zeros(str_zero,1);ones(win_len,1);zeros(↩
   fin_zero,1)]; %upsampled PresTrace
%%
%win= [zeros(780,1);tukeywin(182,0.5);zeros(1918,1)]; %↩
   upsampled PresTrace
%win_mat=repmat(win,1,size(cycles,2));
PresTrace_up_bpass = bandpass(PresTrace_up_b,[5000 8000],↩
   Fs_press*Rpm./60); %Knock; %The CAS Knock signal is ↩
   the bandpassed pressure data
IonSig_s_bpass = bandpass(win_mat.*IonSig_ssf_up_b,[5000 ↩
   8000],Fs_press*Rpm./60);
IonSig_c_bpass = bandpass(win_mat_SC.*IonSig_csf_up_b↩
   ,[5000 8000],Fs_press*Rpm./60);
%Max ampltidue of bandpassed ion signals
```

```
max_ion_amp_s = max(IonSig_s_bpass);
max_ion_amp_c = max(IonSig_c_bpass);
%% Plotting the bandpassed data
%MAX and MIN using Knock Intensity
tqca_p=-180:0.25:539.75;
%Cycle with min knock
cyc= max_knock_cyc; %max_knock_cyc; % UPDATE BASED ON ↩
   WHICH CYCLE YOU WANT THE FUNC TO PLOT
plot_type= 'For a cycle';
legend_loc='northeast';
spark_pt=mean(Cyl2_IGN_ATI);
[~,spark_pt_idx]= (min(abs(tqca1-(-1*spark_pt))));
figure('pos',[500 300 900 600]);
subplot(2,1,1);
plot(tqca1,PresTrace(:,cyc));
ylabel ('In-cylinder Pressure (bar)');
xlim([-15 70])
set(gca,'XTick',[-180 : 15 : 540]);
xlabel ('Crankangle [deg]');
title (strcat('Raw Cyl Pressure.',plot_type,'. Data: ', ↩
   num2str(Test_no) ,' Cycle :', num2str(cyc)));
```

```
grid minor;
hold on
yyaxis right
ylim([-80 80]);
plot(tqca_p,rect_win.*PresTrace_up_bpass(:,cyc).*100);
ylabel ('Knock Pressure x 100 (bar)');
legend('Incyl Pressure', 'Bpass Incyl Pressure');
hold off
subplot(2,1,2);
plot(tqca1,IonSig_ssf(:,cyc),'-black',tqca1,IonSig_csf(:,↩
   cyc),'-Red');
hold on
plot(tqca1(spark_pt_idx),0,'o','LineWidth',2,'↩
   MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize↩
   ',8);
plot(tqca1(rise_idx_SA(cyc)),0,'^','LineWidth',2,'↩
   MarkerEdgeColor','k','MarkerFaceColor','blue','↩
   MarkerSize',8)
plot(tqca1(rise_idx_SA(cyc)+win_offst_SA*steps),0,'^','↩
   LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','↩
   blue','MarkerSize',8);
```

```
plot(tqca1(rise_idx_SC(cyc)),0,'^','LineWidth',2,'↩
   MarkerEdgeColor','k','MarkerFaceColor','red','↩
   MarkerSize',8)
plot(tqca1(rise_idx_SC(cyc)+win_offst_SC*steps),0,'^','↩
   LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','↩
   red','MarkerSize',8);
ylim([0 10])
xlim([-15 70])
set(gca,'XTick',[-180 : 15 : 540]);
xlabel ('Crankangle [deg]');
ylabel ('Ion signal - [V]');
yyaxis right
plot(tqca_p,win_mat(:,cyc).*100,'-blue');
plot(tqca_p,win_mat_SC(:,cyc).*100,'-cyan');
ylim([-80 80]);
g=plot(tqca_p,IonSig_s_bpass(:,cyc).*1000,'-black')
plot(tqca_p,IonSig_c_bpass(:,cyc).*1000,'-Red');
set(g,'LineWidth',2);
ylabel ('BPass Ion signal - [mV]');
```

```
legend ('SA Ion','SC Ion','Spk timing:ATI','SA flame ↩
   front','SA win str','SC ringing','SC win str','SA ↩
   window','SC window', 'Bpass SA Ion','Bpass SC Ion','↩
   Location',legend_loc); %'Orientation','horizontal');
lgd.NumColumns = 2;
title (strcat('Raw Ion signal.',plot_type,'. Data: ', ↩
   num2str(Test_no) ,' Cycle :', num2str(cyc)));
grid minor;
%%
%Show that ion window always starts well before peak ↩
   knock
%find where peak occurs in knock signal
[~, pk_knock_idx]=max(rect_win.*PresTrace_up_bpass); %↩
   windowed bandpassed pressure signal used
s_win_diff=pk_knock_idx-s_win_str_idw;
c_win_diff=pk_knock_idx-c_win_str_idw;
[~, high_knock_cyc] =find(Knock_amp>=0.9);
figure;
subplot(2,1,1)
bar(s_win_diff/steps);
ylabel('crank angles');
```

```
title(strcat('Standalone Window start relative to knock ↩
   peak (CAD),',Test_no,',',num2str(test_date)));
subplot(2,1,2)
bar(c_win_diff/steps);
ylabel('crank angles');
title(strcat('Coil Window start relative to knock peak (↩
   CAD),',Test_no,',',num2str(test_date)));
%% Raw intensity in window of 15 to 60CAD
II_s_raw=IonSig_s_bpass(recwin_str_idx:recwin_str_idx+↩
   steps*int_len,:); %CAD of earliest flame detect to 60 ↩
   degrees after
II_c_raw=IonSig_c_bpass(recwin_str_idx:recwin_str_idx+↩
   steps*int_len,:);
PI_raw =PresTrace_up_bpass(recwin_str_idx:recwin_str_idx+↩
   steps*int_len,:); %sampled at 0.25CAD
N_ion = size(II_s_raw,1);
N_press = size(PI_raw,1); %numder of rows or crank angle ↩
   points
PI = sum(abs(PI_raw).*100)/N_press; %Change from bar to ↩
   kPa
```

```
II_s = sum(abs(II_s_raw).*1000)/N_ion; %Change from V to ↩
   mV
II_c = sum(abs(II_c_raw).*1000)/N_ion;
idx=find(PI <= prctile(PI,99)); %find values below the 95↩
   th percentile
%Correlation between PI and II
figure;
x=PI; y=II_c;
plot(x,y,'o');
coeffs = polyfit(x, y, 1);
% Get fitted values
fittedX = linspace(min(x), max(x), 200);
fittedY = polyval(coeffs, fittedX);
% Plot the fitted line
hold on;
plot(fittedX, fittedY, 'r-', 'LineWidth', 3);
r=corrcoef(x,y);
str=sprintf('r= %1.2f',r(1,2));
T = text(min(get(gca, 'xlim')), max(get(gca, 'ylim')), ↩
   str);
```

```
set(T, 'fontsize', 14, 'verticalalignment', 'top', '↩
   horizontalalignment', 'left');
title (strcat('Correlation PI vs Coil II. Data: ', ↩
   num2str(Test_no) ));
xlabel('Pressure intensity (kPa)');
ylabel('Coil Ion Intensity (mV)');
clear ('x','y','r');
figure;
x=PI; y=II_s;
plot(x,y,'o');
coeffs = polyfit(x, y, 1);
% Get fitted values
fittedX = linspace(min(x), max(x), 200);
fittedY = polyval(coeffs, fittedX);
% Plot the fitted line
hold on;
plot(fittedX, fittedY, 'r-', 'LineWidth', 3);
r=corrcoef(x,y);
str=sprintf('r= %1.2f',r(1,2));
T = text(min(get(gca, 'xlim')), max(get(gca, 'ylim')), ↩
   str);
```

```
set(T, 'fontsize', 14, 'verticalalignment', 'top', '↩
   horizontalalignment', 'left');
title (strcat('Correlation PI vs Standalone II. Data: ', ↩
   num2str(Test_no)));
xlabel('Pressure intensity (kPa)');
ylabel('Standalone Ion Intensity (mV)');
clear ('x','y','r');
figure;
x=II_c; y=II_s;
plot(x,y,'o');
coeffs = polyfit(x, y, 1);
% Get fitted values
fittedX = linspace(min(x), max(x), 200);
fittedY = polyval(coeffs, fittedX);
% Plot the fitted line
hold on;
plot(fittedX, fittedY, 'r-', 'LineWidth', 3);
r=corrcoef(x,y);
str=sprintf('r= %1.2f',r(1,2));
T = text(min(get(gca, 'xlim')), max(get(gca, 'ylim')), ↩
   str);
```

```
set(T, 'fontsize', 14, 'verticalalignment', 'top', '↩
   horizontalalignment', 'left');
title (strcat('Correlation Coil II vs Standalone II. Data↩
   : ', num2str(Test_no)));
xlabel('Coil Ion Intensity (mV)');
ylabel('Standalone Ion Intensity (mV)');
clear ('x','y','r');
figure;
x=PI; y=Knock_amp.*100;
plot(x,y,'o');
coeffs = polyfit(x, y, 1);
% Get fitted values
fittedX = linspace(min(x), max(x), 200);
fittedY = polyval(coeffs, fittedX);
% Plot the fitted line
hold on;
plot(fittedX, fittedY, 'r-', 'LineWidth', 3);
r=corrcoef(x,y);
str=sprintf('r= %1.2f',r(1,2));
T = text(min(get(gca, 'xlim')), max(get(gca, 'ylim')), ↩
   str);
```

```
set(T, 'fontsize', 14, 'verticalalignment', 'top', '↩
   horizontalalignment', 'left');
title (strcat('Correlation PI vs Knock Amp (Pk-Pk). Data:↩
    ', num2str(Test_no) ));
xlabel('Pressure intensity (kPa)');
ylabel('Knock Amplitude (Pk-Pk) (kPa)');
clear ('x','y','r');
%%
%Plot pdf
var=II_s;
xaxis='Standalone II (mV)'; %'Pressure Intensity(bar)' %'↩
   Standalone II (mV)';
plot_name='PDF of Standalone II';
x=(sort(var));
mu=mean(log(x));
sigma=std(log(x));
pdf=lognpdf(x,mu,sigma);
cdf=logncdf(x,mu,sigma);
figure;
histogram(x,30,'Normalization','pdf');
ylabel('PDF');
```

```
hold on;
plot(x,pdf,'LineWidth',2);
r=corrcoef(x,cdf);
str=sprintf('r= %1.2f',r(1,2));
T = text(min(get(gca, 'xlim')), max(get(gca, 'ylim')), ↩
   str);
set(T, 'fontsize', 14, 'verticalalignment', 'top', '↩
   horizontalalignment', 'left');
yyaxis right
plot(x,cdf,'LineWidth',2);
ylabel('CDF');
title(strcat(plot_name,':',Test_no));
xlabel(xaxis);
grid minor
clear('x','mu','sigma');
%%
%FFT the upsampled Incyl pressure and Ion signal
basepath=strcat('\\homes.mtu.edu\home\Desktop\Sensor ↩
   Fusion_2018\Knock_studies_Nis');
cd(basepath);
% FFT of the Pressure signal
```

```
% Data acquisition parameters
D_Theta_press= 0.25; % For Knocking data
Fs_press= 360/ D_Theta_press; % sampling rate = 0.25 deg ↩
   so for 360 degs (1 rev) we have 1440 orders
N=size(PresTrace_up_b,1);
num_cyc=cycles(end);
win_P = hann(N);
for ii=1:num_cyc
LS_PresTrace(:,ii)= Linear_Spectrum(PresTrace_up_b(:,ii),↩
    Fs_press, N, win_P); %Linear Spectrum for standalone ↩
   Ion sig
end
f_array_orders_Press= (0:(N/2))*Fs_press/(N); %X axis for↩
    plotting
% Converting orders to freq
Rpm= mean(RPM);
Rpm= round(Rpm/500)*500;
f_array_kHz_Press= f_array_orders_Press.*Rpm./60/1000; % ↩
   f_array in kHz so divide by 1000
%%
% FFT of the standalone ion signal
```

```
% Data acquisition parameters
D_Theta= 0.25;
Fs= 360/ D_Theta; % sampling rate = 0.5 deg so for 360 ↩
   degs (1 rev) we have 720 orders
N=size(IonSig_ssf_up_b,1);
for ii=1:num_cyc
win_I = win_mat(:,ii);
LS_IonSig_s(:,ii)= Linear_Spectrum(IonSig_ssf_up_b(:,ii),↩
    Fs, N, win_I); %Linear Spectrum for standalone Ion ↩
   sig
end
f_array_orders= (0:(N/2))*Fs/(N); %X axis for plotting
% Converting orders to freq
Rpm= mean(RPM);
Rpm= round(Rpm/500)*500;
f_array_kHz= f_array_orders.*Rpm./60/1000; % f_array in ↩
   kHz so divide by 1000
% FFT of the coil ion signal
% Data acquisition parameters
D_Theta= 0.25;
```

```
Fs= 360/ D_Theta; % sampling rate = 0.5 deg so for 360 ↩
   degs (1 rev) we have 720 orders
for ii=1:num_cyc
win_I = win_mat_SC(:,ii); %USe the custom window of the ↩
   cycle
LS_IonSig_c(:,ii)= Linear_Spectrum(IonSig_csf_up_b(:,ii),↩
    Fs, N, win_I); %Linear Spectrum for standalone Ion ↩
   sig
end
f_array_orders_coil= (0:(N/2))*Fs/(N); %X axis for ↩
   plotting
% Converting orders to freq
Rpm= mean(RPM);
Rpm= round(Rpm/500)*500;
f_array_kHz_coil= f_array_orders_coil.*Rpm./60/1000; % ↩
   f_array in kHz so divide by 1000
%% Plotting the Linear Spectrum
%Plotting the LS of Pressure trace
figure
subplot(3,1,1);
```

```
semilogy (f_array_kHz_Press,(abs(LS_PresTrace(:,↩
   min_knock_cyc))))
hold on
semilogy (f_array_kHz_Press,(abs(LS_PresTrace(:,↩
   max_knock_cyc))))
ylabel ('Linear Spectrum- Pressure data');
xlabel ('Frequency [kHz]');
legend ('Low Knock Cycle','High Knock Cycle')
grid minor
title (['Linear Spectrum for Knocking: ' num2str(↩
   max_knock_cyc) ' and Non-Knocking: ' num2str(↩
   min_knock_cyc) ' cycle (Pressure data)' Test_no]);
xlim([0 15]);
%Plot Standalone ion
%figure
subplot(3,1,2);
semilogy (f_array_kHz,(abs(LS_IonSig_s(:,min_knock_cyc)))↩
   )
hold on
semilogy (f_array_kHz,(abs(LS_IonSig_s(:,max_knock_cyc)))↩
   )
```

```
ylabel ('Linear Spectrum- Ion data');
xlabel ('Frequency [kHz]');
legend ('Low Knock Cycle','High Knock Cycle')
grid minor
title (['Linear Spectrum for Knocking: ' num2str(↩
   max_knock_cyc) ' & Non-Knocking: ' num2str(↩
   min_knock_cyc) ' cycle (SA, Custom win)' Test_no]);
xlim([0 15]);
%Plot coil ion
%figure
subplot(3,1,3);
semilogy (f_array_kHz_coil,(abs(LS_IonSig_c(:,↩
   min_knock_cyc))))
hold on
semilogy (f_array_kHz_coil,(abs(LS_IonSig_c(:,↩
   max_knock_cyc))))
ylabel ('Linear Spectrum- Coil Ion data');
xlabel ('Frequency [kHz]');
legend ('Low Knock Cycle','High Knock Cycle')
grid minor
```

```
title (['Linear Spectrum for Knocking: ' num2str(↩
   max_knock_cyc) ' & Non-Knocking: ' num2str(↩
   min_knock_cyc) ' cycle (SC, Custom win)' Test_no]);
xlim([0 15]);
%%
%PLot spectrogram of knocking and non-knocking cycle
specsig=IonSig_ssf(idx_Hres_start:idx_Hres_end,↩
   max_knock_cyc); %Raw Ion signal of max knock cycle in ↩
   knock window
specsig2=IonSig_ssf(idx_Hres_start:idx_Hres_end,↩
   min_knock_cyc); %Raw Ion signal of max knock cycle in ↩
   knock window
%Bpass Ion in hires win
[~,idx1]= (min(abs(tqca_p-(-30)))); %CA for high res ↩
   start
[~,idx2]= (min(abs(tqca_p-(70)))); %CA for high res end
specsigb=IonSig_s_bpass(idx1:idx2,max_knock_cyc);
specsig2b=IonSig_s_bpass(idx1:idx2,min_knock_cyc);
winlen=15;
overlap=12;
Fn=36000;%Check and change based on RPM of test
```

```
sampling_freq=2*Fn;
caxis_lim=[-100 -10];
x_axis=(-30:0.25:70);
figure('pos',[500 300 900 600]);
subplot(2,2,1)
spectrogram(specsig,tukeywin(winlen,0.5),overlap,8*winlen↩
   ,sampling_freq,'yaxis')
colorbar('off')
caxis(caxis_lim)
title(['Max knock cyc ', num2str(test_date),' :', Test_no↩
   ]);
set(gca,'YTick',[0:3:Fn/1000]);
set(gca,'XTicklabel',[]);
xlabel('Crank Angle');
subplot(2,2,3)
plot(x_axis,specsig)
ylabel('Ion signal(V)');
xlabel('Crank angle');
hold on
yyaxis right
g=plot(x_axis,specsigb.*1000)
```

```
ylim([-80 80]);
set(g,'LineWidth',1.2);
ylabel ('BPass Ion signal - [mV]');
grid minor
xlim([-30 70])
subplot(2,2,2)
spectrogram(specsig2,tukeywin(winlen,0.5),overlap,8*↩
   winlen,sampling_freq,'yaxis')
caxis(caxis_lim)
colorbar('off')
title(['Min knock cyc ', num2str(test_date),' :', Test_no↩
   ]);
set(gca,'YTick',[0:3:Fn/1000]);
set(gca,'XTicklabel',[]);
xlabel('Crank Angle');
subplot(2,2,4)
plot(x_axis,specsig2);
grid minor
ylabel('Ion signal(V)');
xlabel('Crank angle');
hold on
```

```
yyaxis right

g=plot(x_axis,specsig2b.*1000)

set(g,'LineWidth',1.2);

ylabel ('BPass Ion signal - [mV]');

ylim([-80 80]);

xlim([-30 70])
```

Appendix B

Letters of Permission

Michigan Tech

Nischal Muralidhar <[redacted]>

Permission to use figures in thesis

2 messages

Nischal Muralidhar <[redacted]> Tue, Mar 19, 2019 at 3:49 PM
To: Kaushik Prabhu <[redacted]@mtu.edu>

Hey kaushik,
Trust you are well. In regard to my thesis, I wanted to use a couple of figures pertaining to the engine setup and exhaust sensor studies from your work/thesis. I need to use them to help readers grasp some of the topics covered in my thesis. The list of figures I'd need permission for are listed below. Additionally, I might also have to include a few details from your presentations in my work. I assure you that I'd cite it appropriately. Please let me know if you agree to grant me permission to do so.

Thesis Title : SENSOR FUSION FOR SPARK-IGNITION ENGINES

- Figure 2.3 page 13
- Figure 3.5 page 29

Have a great day.

Looking forward to your response.

Regards,
Nischal

Kaushik Prabhu <[redacted]@mtu.edu> Tue, Mar 19, 2019 at 3:58 PM
To: Nischal Muralidhar <[redacted]@mtu.edu>

Hello Nischal,
I hope your thesis work is coming along well. In regard to your request, you have my permission to incorporate any information (including tables and figures) from my work that you would need to complete your thesis. This includes my thesis as well as my presentations that you've been given access to. Just make sure to cite them appropriately.

Do let me know in case you require any further assistance.

I wish you well in your future endeavours.

Regards

-Kaushik

Figure B.1: Letter of permission

Appendix C

Test Conditions

Date	TEST Num	Speed	CYL1 IMEP	CYL1 CA50 (ATDC)	vct_int	vct_exh	Lambda	Wastegate
20180109	1	1500	250	8	0	0	1	Auto
	2	3500	250	8	0	0	1	Auto
	3	3500	750	8	0	0	1	Auto
	4	1500	750	8	0	0	1	Auto
	5	1500	250	8	-35	35	0.9	Auto
	6	3500	250	8	-35	35	1	Auto
	7	3500	750	8	-35	35	1	Auto
20180908	TEST Num	Speed	CYL1 IMEP	CYL1 CA50 (ATDC)	vct_intake	vct_exhaust	Lambda	Wastegate
	1	1500	Auto	Auto	Auto	Auto	1	Auto
	2	2500	Auto	Auto	Auto	Auto	1	Auto
	3	Transient#1 (everythin auto with speed/load profile)						
	4	Transient#2 (everythin auto with speed/load profile)						
	5	1500	344	8.4	Auto	Auto	1	100% open
	6	1500	532	8.4	Auto	Auto	1	closed
	7	1500	532	8.4	Auto	Auto	1	open
	8	1500	344	8.4	-30	0	1	closed
	9	1500	532	8.5	0	20	1	open
	10	1500	532	8.5	0	20	1	closed
20181011	Test Num	Speed	Spark Adv	Fs in Knock win	SPA2	Throttle		
	1	1500	-3	0.25	3	4.7		
	2	1500	0	0.25	0	4.7		
	3	1500	3	0.25	-3	4.7		
	4	2500	-3	0.25	3	2.5		
	5	2500	0	0.25	0	2.5		
	6	2500	3	0.25	-3	2.7		
	7	2500	-3	0.25	3	2.7		
	8	2500	0	0.25	0	2.7		
	9	2500	3	0.25	-4	2.7		
20181029	TEST Num	Speed	CYL1 IMEP	CYL1 CA50 (ATDC)	vct_intake	vct_exhaust	Lambda	Wastegate
	1	Engine-off						
	2	Transient for knock						

	TEST Num	Speed	CYL1 IMEP	CYL1 CA50 (ATDC)	vct_intake	vct_exhaust	Lambda	Wastegate
20181121	1	Transient#1						
	2	Transient#2						
	3	Transient#3						
	7	1500	344	8.4	-30	0	1	Auto
	8	1500	532	8.5	0	20	1	Auto
	TEST Num	Speed	IMEP	CA50	Int Adv	Exh Rtd	Lambda	Wastegate
20190110	1	Change Transient						
	2	1500	250	8	-30	0	1	auto
	3	1500	250	8	-30	10	1	auto
	4	1500	250	8	-30	20	1	auto
	5	1500	250	8	-30	30	1	auto
	6	1500	250	8	0/-10/-20/-	30	1	auto
	7	1500	344	8.4	-30	0	1	Auto
	8	1500	532	8.5	0	20	1	Auto
	TEST Num	Speed	IMEP	CA50	Int Adv	Exh Rtd	Lambda	Wastegate
2019111	1	1500	250	8	0	30	1	auto
	2	1500	250	8	-10	30	1	auto
	3	1500	250	8	-20	30	1	auto
	4	1500	250	8	-30	0/10/20/30	1	auto
	5	1500	344	8.4	-30	0	1	Auto
	6	1500	532	8.5	0	20	1	Auto