**Michigan
Technological
University**

Michigan Technological University
Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2020

DATA DRIVEN SENSOR FUSION FOR CYCLE-CYCLE IMEP ESTIMATION

Cooper Heyne Minehart
Michigan Technological University, cjheyne@mtu.edu

Copyright 2020 Cooper Heyne Minehart

Recommended Citation

Heyne Minehart, Cooper, "DATA DRIVEN SENSOR FUSION FOR CYCLE-CYCLE IMEP ESTIMATION", Open Access Master's Thesis, Michigan Technological University, 2020.
<https://doi.org/10.37099/mtu.dc.etdr/1115>

Follow this and additional works at: <https://digitalcommons.mtu.edu/etdr>

Part of the [Automotive Engineering Commons](#)

DATA DRIVEN SENSOR FUSION FOR CYCLE-CYCLE IMEP ESTIMATION

By

Cooper J. Heyne Minehart

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mechanical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2020

© 2020 Cooper Heyne Minehart

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. Wayne W. Weaver*

Department Chair: *Dr. William W. Predebon*

Table of Contents

List of figures	vii
List of tables.....	x
Author Contribution Statement.....	xi
Acknowledgements.....	xii
List of abbreviations	xiii
Abstract.....	xiv
1 Introduction.....	1
1.1 Goals and Objectives.....	1
1.2 Literature Review	2
1.3 Thesis Overview	3
2 Engines and Sensors	5
2.1 Dynamometer Engine.....	5
2.2 Vehicle Engine	7
3 Feature Extraction.....	11
3.1 Exhaust Pressure Feature Extraction.....	11

The exhaust data is the simplest of the angle-series measured inputs, so we will start with this. The exhaust data is collected from the exhaust pressure sensor and fed to the AVL DAQ every 0.5 crank angle degrees. This is shown as an angle-series pressure pulse, an example of which is shown in

Figure 3-1.	11
3.2 Crank Signal Feature Extraction	18
3.3 Ion Feature Extraction	27
4 Regularization and Network Structure.....	32
4.1 Rectifier Activation Function	32
4.2 Regularization	34
4.3 Network Structure	35
4.3.1 Required Network Layers	35
4.3.2 Fully Connected Layers	36
4.3.3 Connecting Layers and Network Depth.....	37
4.4 Neural Network Depth and Structure Decisions	38
4.5 Neural Network Implementation.....	39
4.6 Learning Rate and Drop Factor	40
5 Preprocessing Optimization	44
5.1 Stochastic Optimization	44

5.1.1	Pattern Search Algorithm.....	45
5.1.2	Reducing Standard Variation in Training.....	47
5.1.3	Brute Force - Grid Search.....	48
6	IMEP Model Error Analysis	52
6.1	Model Sensitivity	56
6.2	Model Error Division	62
6.3	Production Sensor Set	63
6.4	Feature Reduction: Backward Sequential Feature Selection	64
7	Conclusion and Future Work.....	68
7.1	Direction of Future Work.....	68
7.2	Conclusions	69
	References.....	70
	Appendices.....	73
	Appendix A: EcoBoost 2.0 Cam Phasing.....	73
	Appendix B: Crank Acceleration Correlation with IMEP.....	74
	Appendix C: Visual Grid Search Logic.....	75
	Appendix D: Map of MTU Drive Cycle.....	76
	Appendix E: Pattern Search Algorithm Code (MATLAB)	77

Appendix F: Kistler In Cylinder Pressure Sensor Specifications	79
Appendix G: Backward Sequential Feature Selection Code	100
Appendix H: Optimizable Windowing IMEP estimation system.....	103

List of figures

Figure 2-1: AVL DAQ and Associated Hardware.....	10
Figure 3-1 Raw Exhaust Pressure Signal.....	12
Figure 3-2: Exhaust Pressure Anti Alias Filter	14
Figure 3-3: Exhaust Pressure Signal with Filter Applied	15
Figure 3-4: Correlation of IMEP and exhaust phasing	16
Figure 3-5: Correlation of IMEP and exhaust phasing with EVO tracking.....	17
Figure 3-6: Raw Crank Signal	18
Figure 3-7: Raw Crank Signal	19
Figure 3-8: Crank signal filter response.....	21
Figure 3-9: Raw crank signal (left) Applied crank signal filter (right).....	22
Figure 3-10: Crank Acceleration with respect to time.....	24
Figure 3-11: Angle and Order domain response of crank window.....	25
Figure 3-12: Tukey window applied to Crank Acceleration data.....	26
Figure 3-13: 0th and 2nd order crank amplitude against IMEP.....	27

Figure 3-14: Ion Signal with Notation	29
Figure 3-15: Ion Signal Windowing	30
Figure 3-16: Ion Magnitude correlation with IMEP	31
Figure 3-17: Ion Phase correlation with IMEP	31
Figure 4-1: Rectifier Function	33
Figure 4-2: Sequence Input Layer with Fully Connected Layer.....	36
Figure 4-3: Simple Neural Network with Rectified units	37
Figure 4-4: Learning Rate and Drop Factor Grid Search	42
Figure 5-1: Structure of model to be optimized.....	44
Figure 5-2: Pattern Search Algorithm Example.....	47
Figure 5-3: Grid Search Optimization of Window Phasing.....	50
Figure 6-1: IMEP model Training and testing data	52
Figure 6-2: Section of IMEP testing data	53
Figure 6-3: Autocorrelation of Steady State IMEP.....	54
Figure 6-4: Cycle-Cycle Variation in IMEP	55
Figure 6-5: Cycle-Cycle Variation in IMEP	56

Figure 6-6: IAT vs Ambient Temperature	59
Figure 6-7: Estimating September 1st data set with single March training example	60
Figure 6-8: Estimating September 1st Dataset with Varied Training Data	61
Figure 6-9: Estimating September 1st Dataset with Complete Training Data.....	62

List of tables

Table 2-1: Specifications for vehicle and dynamometer engines	6
Table 2-2: Dynamometer Engine Sensors	7
Table 2-3: AVL DAQ Sensors and Sample Rate.....	8
Table 2-4: ATI Sensor Suite and Sample Rate	9
Table 3-1: Sensors and Input Signals.....	11
Table 3-2: Exhaust Anti Alias Filter	13
Table 5-1: Optimized Preprocessing Parameter Selection.....	49
Table 5-2: Selected Preprocessing Parameters	51
Table 6-1: Temperature Robustness Data Sets	58

Author Contribution Statement

This thesis covers the work done during my time as a master's student at Michigan Tech. My time was spent developing an IMEP estimation system for gasoline-direct-injection turbocharged engines. The results of the IMEP estimation system have been accepted to SAE World Congress 2021 as an abstract and the manuscript is under review. A journal paper is also planned, focusing on the signal processing and optimization of the IMEP model.

Acknowledgements

Professors Jeffrey Naber and Jason Blough mentored me through my entire graduate career at Michigan Tech. They taught me, challenged me, and pushed me through the ups and downs of higher education. Professor Jason Blough taught me the basics of rotating machinery digital signal processing and helped me through the more challenging applications through the program. Jeffrey Naber welcomed me as a student at the Advanced Power Systems Research Center, answered my many questions on combustion analysis, and supported me through my entrance to the academic world.

The staff of engineers at APSRC hold a wealth of knowledge and are always happy to share it. Joel Duncan, Tucker Alsup, and Paul Dice know everything about the lab, what your issue is, and how to fix it. I cannot stress enough how helpful and kind they are.

Ford proposed and funded the research into sensor fusion in conjunction with the projects being done by Amir Khameneian and Xin Wang. Chad Archer at Ford provided critical feedback and direction to the project. Xin Wang and Amir Khameneian taught me to use the dynamometer and ATI software and provided helpful feedback throughout our twice-weekly meetings.

Finally, thank you to my family, Walter, Jim, and Patty. Through a pandemic and graduate school, they have been a comfort in a new world.

List of abbreviations

IMEP	Indicated Mean Effective Pressure
EVO	Exhaust Valve Open
IVO	Intake Valve Open
EVC	Exhaust Valve Close
APSRC	Advanced Power Research Center
ICE	Internal Combustion Engine
CA	Crank Angle
ReLu	Rectified Linear Unit
IAT	Intake Air Temperature
RPM	Rotations Per Minute
TDC	Top Dead Center
BTDC	Before Top Dead Center
DAQ	Data Acquisition

Abstract

As the world searches for ways to reduce humanity's impact on the environment, the automotive industry looks to extend the viable use of the gasoline engine by improving efficiency. One way to improve engine efficiency is through more effective control – effective control systems require a feedback signal. Indicated mean effective pressure (IMEP) is a useful feedback signal for automotive control but is costly to measure directly.

Successful machine learning based sensor fusion requires effective feature extraction and model creation. Through a multistage application of machine learning to both the feature extraction process and the IMEP estimation process we are able to efficiently extract the useful data from angle-domain sensors and supply them to an estimation model.

Phasing exhaust and intake pressure signals with their respective cam shaft timing is shown to improve the correlation between pressure pulse phase and IMEP. Incremental improvements in preprocessing and network structure allow the model to estimate cycle-cycle variations in IMEP with an accuracy of 16 kPa rms error using a combination of existing and proposed cost-effective sensors.

1 Introduction

Gasoline engine efficiency is heavily dependent on a successful control system, which often requires sensor feedback. Indicated mean effective pressure (IMEP) is a useful feedback signal for an engine control system as it is directly related to engine torque, the control systems objective. Accurate estimation of the previous cycles IMEP allows for the use of advanced model predictive controllers [1].

Indicated mean effective pressure is measured directly using an in-cylinder pressure sensor. IMEP can vary from cycle to cycle even while the engine state (throttle position, camshaft timing, etc.) remains constant. This makes an accurate measure of IMEP from engine control unit commanded signals impossible, the measurement crank acceleration and exhaust pressure allows us to capture some of the stochastic portion of IMEP and improve our results beyond just mean value estimation.

1.1 Goals and Objectives

At the outset of this research, it was known from previous work by Prabhu and Nischal that exhaust pressure, crank acceleration, and ion signals all contained useful information for IMEP estimation [2], [3]. This project then hoped to find what information each sensor contained, how to optimize the feature extraction from each sensor, build an estimation system, and quantify the error in IMEP predictions over a range of engine states. As accurately estimating IMEP is critical in model predictive control and fault

detection, the model should be as accurate as possible with the sensors at hand. Model accuracy will be measured using the in-cylinder pressure based IMEP reading as truth.

Along with the standard suite of production engine sensors outlined in Table 2-4: ATI Sensor Suite and Sample Rate, it is possible that an exhaust pressure sensor and an ion probe could be used in production. The effectiveness of these sensors will be analyzed so that a cost-benefit analysis can be done.

1.2 Literature Review

Significant previous work has been done in the area of IMEP estimation using crank signal processing. Several patents exist for crank processing from partial and misfire combustion detection [4] to complete IMEP estimation [5]. A 2011 paper published by Morello, Blough, Naber, and Jia focused on time-domain frequency analysis of accelerometer signals to estimate combustion features in a 9 liter diesel engine [6]. The paper includes optimizing the time-domain window shape and length along with in depth frequency analysis of the accelerometer signals. The study found that the accelerometer signal alone was insufficient to accurately estimate IMEP. Jia et al. used the same experimental setup but applied a neural network with radial-basis activation functions to account for the non-linear transfer path between cylinder pressure and accelerometers [7]. Jia et al. also released an investigation into the abilities of accelerometer, ion, and crank based combustion metric estimation [8]. The study showed that calculating apparent heat release in a diesel engine was most effectively done using an

accelerometer. The Ion signal was shown to be too dependent on external conditions (fuel additives, engine speed, AFR, etc.) and the crank signal was shown to be inaccurate near TDC due to the zero crankshaft torque from combustion at TDC. Frequency domain work has been completed by Liu et al. by calculating a transfer function between the velocity of the crankshaft and in cylinder pressure; this system aimed to not only calculate IMEP but also the continuous cylinder pressure curve. This particular system used the frequency driven transfer function in conjunction with a physics based adiabatic model. These are all examples of systems using measured values to estimate some representation of in cylinder pressure.

Knock sensors are an important feedback signal and have been sensing in cylinder knock intensity reliably and in a production environment [9]. Rugland and Stenlaas showed that frequency based analysis of a broadband piezo-electric knock sensor can simulate an in cylinder pressure for a compression ignition engine [10]. Similar work with acceleration sensors was completed by Wagner et al. on spark ignition engines and using system identification [11]. The goal in this study is to improve IMEP estimation accuracy over previous studies through sensor fusion, improved training sets, optimized sensor windowing, and optimized network structure.

1.3 Thesis Overview

This thesis discusses the development and results of a data driven IMEP model using low cost production sensors. The model relies on the non-linear relation between indicated

net mean effective pressure (NMEP) and our sensor suite being captured by a neural network. Chapter 1 goes over the goals and objectives of this thesis, along with prior work done in the area. Chapter 0 discusses the engines and sensors that were used in this study. Chapter 0 goes over the signals that were collected and the processing that was completed to extract features for the neural network that is discussed in chapter 4. Chapter four also captures the structure of the neural network, the training/testing data, and the activation functions. Chapter 5 focuses on the optimization techniques that were used to phase sensor windows. Chapter 6 is based on the results of the IMEP model and how the error can be interpreted.

2 Engines and Sensors

Two engines of the same type were used for the tests in this project. They are both Ford 2.0L EcoBoost gasoline direct injected and reside at the at the Advanced Power Systems Research Center (APSRC) lab at Michigan Technological University. One of the engines is on an engine stand for the light duty dynamometer at the APSRC, while the other is in a Lincoln MKC. This section will describe each engine with its accompanying sensor suite.

2.1 Dynamometer Engine

The first engine in the setup is connected to a light duty AC dynamometer and uses a Computer Aided Solutions (CAS) data logger for high frequency engine pressures (cylinder, exhaust, intake) and crank speed. The CAS system allows pressures and speeds to be read every 0.5 crank angle degrees. ATI Vision software controls the ECU and reads control signals and stock temperature probes from the engine. A National Instruments DAQ is also present to read temperatures in the lab and mass airflow through a venturi before the intake. The dyno and vehicle engines are the same mechanically, and the specs are shown below in Table 2-1.

Specification	Value
Fuel	Gasoline
Displacement	2.0 Liters
Number of Valves	4 per cylinder
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
Firing Order	1-3-4-2
Intake Valve Timing	Variable
Intake Valve Timing	Variable

Table 2-1: Specifications for vehicle and dynamometer engines

The dynamometer engine is outfitted with the stock sensors for engine control equipped in a production Lincoln MKC (Table 2-4) along with the following lab grade sensors feeding the CAS data acquisition system.

Sensor	Quantit y	Make	Model	Samples/revolution
Crank Optical Encoder	1	BEI	H25	720
Exhaust Pressure	1	Omega	PX309	720
Exhaust Pressure	1	Kulite	ETL312M	720
Ion Probes	4	Custom	N/A	720
Cylinder pressure sensor	4	Kistler	PiezoStar 6125C	720

Table 2-2: Dynamometer Engine Sensors

The ATI ECU that controls the dynamometer engine also captures the signals shown in Table 2-4.

2.2 Vehicle Engine

The dynamometer engine is used for steady state testing and controlled temperature testing. The crank acceleration signal is effected by the inertia of the vehicle drivetrain, or in this case, the dynamometer inertia. In order to capture a crank signal representative of a vehicle drivetrain, a Lincoln MKC was outfitted with a similar data acquisition system as our dynamometer engine. The Lincoln MKC has the same engine and ECU as our dynamometer setup, and is fitted with similar sensors. The same ATI ECU is used to run the engine and log sensor data using ATI Vision software. An AVL DAQ is being

used to record cylinder, exhaust, and intake pressure for the vehicle along with the ion signals. The AVL DAQ can record up to 10 times per crank angle degree. A list of sensors and the respective sample rate is included below in Table 2-3.

Sensor	Quantity	Make	Model	Samples/ revolution
Crank Optical Encoder	1	AVL	366C	720
Exhaust Pressure	1	Omega	PX359	720
Exhaust Pressure	1	Kulite	ETL312M	720
Intake Manifold Pressure	1	Motorcraft	CX2391	720
Ion Probes	4	Custom	N/A	720
Cylinder pressure sensor	4	Kistler	PiezoStar 6125C	720

Table 2-3: AVL DAQ Sensors and Sample Rate

The ATI ECU sensors are recorded below in Table 2-4.

Sensor	Sample Rate
Cam Phasing	Firing rate
Engine Coolant Temp	Firing rate
Intake Air Temp	Firing rate
Wastegate Position	Firing rate
MAP	Firing rate

Fuel	Firing rate
Spark Advance	Firing rate
Engine Speed	Firing rate
Misfire flag	Firing rate
Rough road flag	Firing rate

Table 2-4: ATI Sensor Suite and Sample Rate

A diagram of the vehicle experimental setup is shown below in Figure 2-1.

Figure 2-1: Vehicle Experimental Setup

The AVL DAQ and its associated hardware (Kistler charge amplifier, power distribution, etc.) is shown in Figure 2-2 below.

Figure 2-2: AVL DAQ and Associated Hardware

3 Feature Extraction

The data acquisition system gives us 720 samples for each sensor for each combustion event, we need a way to efficiently extract the useful information and condense it into a smaller package. This process is called feature extraction, and often utilizes many digital signal processing techniques. The entire set of sensors and input signals to the model is shown below in Table 2-3.

Sensors Used	ECU Commands Used
AVL Optical Crank Encoder (simulating 60-2 crank wheel)	Fuel Injected
Intake Manifold Pressure	Spark Advance
Engine Coolant Temperature	Throttle Position
Intake Air Temperature	Wastegate Position
Exhaust Pressure Sensor	Cam Phasing
Spark Plug Based Ion Sensor	

Table 3-1: Sensors and Input Signals

3.1 Exhaust Pressure Feature Extraction

The exhaust data is the simplest of the angle-series measured inputs, so we will start with this. The exhaust data is collected from the exhaust pressure sensor and fed to the AVL DAQ every 0.5 crank angle degrees. This is shown as an angle-series pressure pulse, an example of which is shown in Figure 3-1.

Figure 3-1 Raw Exhaust Pressure Signal

The system has the exhaust pressure signal sampled at 720 samples per revolution. Appendix A shows that the exhaust valve duration is 224 crank angle degrees, over which 448 samples are taken. The goal is to estimate IMEP with these 448 samples. A neural network could have 448 input nodes and be trained to estimate IMEP using each piece of data. The network could then learn the relation between exhaust pressure and IMEP, even learning the Fourier transform if it were required [12]. This method would lead to a very wide network, with 3 angle-domain signals sampled at 720 samples per revolution we would have over 1500 input nodes. To reduce network size and have the potential to run this system on an embedded computer in real time; the useful information from each signal can be isolated.

The first step in reducing the size of the signal is down sampling. We will be looking at orders 1, 2, 4, and 8 of exhaust pressure for IMEP estimation. According to the Nyquist Theorem, we need to sample data at twice the frequency of the highest order of interest,

in order to accommodate the transition band of the anti-alias lowpass filter we will oversample for a 16th order signal. It follows that we must sample at 32nd order, or one sample every 11.25 crank angle degrees. In order to avoid aliasing we must lowpass filter to remove any frequency content above 8th order. Using the MATLAB filter designer, a finite impulse response filter is designed with the properties shown in table 3.1

Frequency Response	Low Pass
Impulse Response	Infinite Impulse Response
Passband Order	10
Stopband Order	19
Passband Ripple	5
Stopband Attenuation	10 dB
Design Method	Butterworth
Filter Order	5 th order

Table 3-2: Exhaust Anti Alias Filter

Finite Impulse Response filters have the advantage of being stable under all inputs, but the disadvantage of needing more taps for a similar attenuation Infinite Impulse Response filter and a larger phase delay response. Figure 3-2 shows the magnitude and phase response of the exhaust pressure filter.

Figure 3-2: Exhaust Pressure Anti Alias Filter

Figure 3-2 shows that there is a large linear phase delay associated with the signal. The large delay would be an issue if the signal were being compared to another, but it is only being used to train a neural network. The delay is consistent from one estimation to the next and the neural network will learn the relation between exhaust pressure and IMEP with this delay. This filter applied to the exhaust pressure signal shown in Figure 3-1.

Figure 3-3: Exhaust Pressure Signal with Filter Applied

We have now condensed the signal to 32 samples per revolution, still too many for a simple neural network. To further distill the information we are interested in, a Fourier transform is used. The Fast Fourier Transform (FFT) computes the discrete Fourier transform, and can be used here to find the order content of our angle domain data. Order domain analysis is often used for rotating machinery as it makes analysis of events that are periodic with rotation straight forward. The FFT is applied to the angle domain data with correct scaling, which yields order domain data. Second order with respect to crankshaft rotation is of interest here because the engine fires at second order -- an exhaust pressure pulse is present twice per engine revolution.

Now that information has been condensed from the signal, the utility of this information can be analyzed. A simple way of doing this is finding the linear correlation between the elements of this signal and IMEP. Figure 3-4 shows the correlation below.

Figure 3-4: Correlation of IMEP and 2nd order exhaust pressure phasing

A simple linear correlation shows the importance of exhaust pressure in IMEP estimation. There is a .953 root mean squared correlation coefficient between second order exhaust pressure phase and IMEP. The angle-domain signal is acquired from base EVO to base EVC. It can be seen that during periods of high cam phasing and high IMEP the correlation is poorer than during cycles with base cam timing. As the exhaust cam retards at higher IMEP, the order angle is retarded as well. The correlation may be improved by shifting the exhaust pressure window with the cam to always start with exhaust valve open (EVO) rather than to start at a fixed crank angle.

Using camshaft phasing information from the ATI and base camshaft timing (Appendix A: EcoBoost 2.0 Cam Phasing), the exhaust window start position was set to EVO. The correlation over the entire data set was slightly improved, and significantly improved during times of high cam phasing. This can be seen by comparing the data points in Figure 3-5 and Figure 3-4 as well as in the improved correlation coefficient.

Figure 3-5: Correlation of IMEP and 2nd order exhaust pressure – phased with EVO

This system extracts the 2nd, 4th and 8th order magnitude and phase from the exhaust pressure signal as well as the DC magnitude. The signal window is 224 degrees in

duration and is phased with EVO. Window length and base phasing will be optimized in the preprocessing optimization section.

3.2 Crank Signal Feature Extraction

The AVL DAQ gives the crank signal as microseconds per degree every 0.5 crank angle degrees. The raw signal looks like the one shown in Figure 3-6 below.

Figure 3-6: Raw Crank Signal

The crank signal has an error at 87 crank angle degrees before top dead center. The error spans three samples and is constant in position. The error is shown below in Figure 3-7.

Figure 3-7: Raw Crank Signal

There are several possible sources for this error. Electrical interference from fuel injectors or spark coils could induce electrical noise in the wires running to our DAQ. A physical issue with the encoder could leave a tooth out of place and cause the impulse error. Electrical interference is unlikely for several reasons: one - the error remains at 87 degrees BTDC no matter how the spark or fuel injectors are phased, and two – the signal from the encoder to the DAQ is a digital pulse train and insensitive to noise. The error only effects three samples and the signal can be lowpass filtered to our order of interest in order to remove the error. The anti-alias filter can be used to remove the error and allow for downsampling. The cutoff and passband frequencies were chosen based on the goal of extracting 1st, 2nd and 4th order amplitude and phase along with DC amplitude. The specifications for the signal are shown below in Table 3-3.

Frequency Response	Low Pass
--------------------	----------

Impulse Response	Infinite Impulse Response
Passband Order	5
Stopband Order	8
Passband Ripple	5
Stopband Attenuation	40 dB
Design Method	Butterworth
Filter Order	5 th order

Table 3-3: Crank Anti-Alias Specifications

The response of the filter is shown below in Figure 3-8.

Figure 3-8: Crank signal filter response

Ideally we would use the same filter for exhaust pressure signals and crank signals; they would have the same delay and could be directly compared. Here the exhaust pressure filter was optimized to extract orders 0 through 8 and the crank signal is optimized to remove the noise. Through backward sequential feature selection we found that orders 2 and 4 are the most important exhaust pressure indicators for IMEP. Due to this, in the future the crank anti alias filter can be applied to the exhaust as well. Orders 0 through 4 should not have their amplitude effected and the phase delay of the filter will be learned by the data fitting system (neural network). The applied filter and the raw signal are shown in Figure 3-9.

Figure 3-9: Raw crank signal (left) Applied crank signal filter (right)

The crank signal is currently in microseconds per degree, or the inverse of velocity. We would like to calculate crank acceleration, since crank acceleration and cylinder pressure can be related through Newton's second law. In order to calculate crank acceleration, first we need velocity, which can be found from our crank signal by taking the inverse and converting our units:

$$\frac{1}{\text{crank signal } (\frac{\mu s}{deg})} * \frac{10^6 \mu s}{s} = \frac{deg}{s}$$

Equation 3-1

Now that the signal is represented as a velocity, we can find the derivative of this signal. Simply taking the difference between each element of the signal will give us the change in velocity each .5 crank angle degrees:

$$\text{diff}(\dot{\theta}) = \frac{d\dot{\theta}}{d\theta}$$

Equation 3-2

In order to have acceleration with respect to time rather than crank angle, we can multiply by velocity $\left(\frac{d\theta}{dt}\right)$ which leaves $\ddot{\theta}$.

$$\frac{d\dot{\theta}}{d\theta} * \frac{d\theta}{dt} = \frac{d\dot{\theta}}{dt} = \ddot{\theta}$$

Equation 3-3

The next step is to apply a window to the data. For the FFT to function as desired, it requires that the signal is either perfectly periodic or a completely observed transient. The transients in crankshaft acceleration means that the signal is not perfectly periodic, and since the signal we are looking at only lasts 180 degrees, it is not a completely observed transient. Applying a Tukey window to the data forces the beginning and end of the signal to 0, simulating a completely observed transient. A series of 180-degree segments of crank acceleration signal are shown below before the window is applied.

Figure 3-10: Crank Acceleration with respect to time

A Tukey window with $\beta = .4$ is applied to each 180-degree segment of the signal. The window forces the beginning and end of the signal to zero. The window's angle and frequency response are shown in Figure 3-11 below.

Figure 3-11: Angle and Order domain response of crank window

The window is multiplied in the angle domain before the Fourier transform is taken. Point wise multiplication in the angle domain is the same as convolution in the order domain. The width of the main lobe on the right of Figure 3-11 is 5 orders, making leakage an issue. If we are looking at order 4 information, information from orders surrounding it will also be included. The main lobe dips to 5 dB of attenuation by 1.3 orders from the center, meaning that any information outside 1.3 orders from the order of interest is attenuated by 5 dB or more. The orders of interest are 2nd, 4th, and 6th. One of the advantages of the data driven estimation system is that it learns the relationship between the input variables and the objective variable. If this were a physics driven model we would need to isolate the energy of second order crank acceleration and find the relation to IMEP. Because the data driven system learns the relation, as long as the relation is consistent, we don't have to be as careful that we isolate only 2nd order (some leakage can be allowed). In Chapter 5.1.3 we extend window length by 180 degrees; this

will make the leakage an even smaller issue. The window applied in the angle domain is shown in Figure 3-12.

Figure 3-12: Tukey window applied to Crank Acceleration data

After taking the Fast Fourier Transform and correcting for window length and the Tukey amplitude, we can visualize the useful information by comparing it against the objective, IMEP. Order amplitude is the Euclidean norm of the order and order phase is the angle between the real and imaginary portion of the data. The correlation between IMEP and several orders of crank acceleration are shown below.

Figure 3-13: 0th and 2nd order crank amplitude against IMEP

Both DC and 2nd order crank amplitude appears to have direct correlation with IMEP.

The same is true for 4th and 6th order magnitude (shown in Appendix B: Crank Acceleration Correlation with IMEP), making them important parameters for estimating IMEP. None of the order phases show direct relationship with IMEP, however they will be kept as potential parameters as they may contain information and have a dependence on other information that can't be shown in this 2-dimensional plot. The dependence and significance of these variables will be resolved using backward sequential feature selection once the network has been created.

3.3 Ion Feature Extraction

Turbulent flow in the combustion chamber means that even with the engine operating at steady state, there are cycle-to-cycle variations in how combustion occurs and what the

IMEP is for that cycle. In order to capture the cycle to cycle variation we need to measure what is happening in the cylinder during combustion. The crank position sensor provides information during combustion, but the Ion sensor is our only sensor inside the combustion chamber.

Ion sensors measure the ionization of the gas between two electrodes. In this case, the spark plug anode and cathode are used as the two electrodes. The system measures the ionization of gasses by measuring a current passed between the electrodes at constant voltage. The gas being ionized is the air and fuel mixture being combusted. The coil-on-plug ignition system in the vehicle is also capable of reading the ion signal from the spark plug after ignition has occurred. The dyno engine has two ion sensors for each cylinder, one stand alone and one using the spark plug; for all tests here, we will only use the spark plug based sensor as it is constant between the vehicle engine and the dynamometer engine.

The ion sensor signal only starts reading after ignition has occurred. For this reason, the ion signal will be phased with spark advance, similar to how the exhaust pressure signal is phased with EVO. The ion signal is shown below in Figure 3-14, with end of ignition and the start of the ion signal marked. Between the two labels is the ‘ringing’ from the ignition.

Figure 3-14: Ion Signal with Notation

The spike that is present after ignition is the system ‘ringing’ from spark. A window is used to remove it from the signal and isolate the dynamics of the ion sensor; this is shown in Figure 3-15: Ion Signal Windowing

Figure 3-15: Ion Signal Windowing

After applying the window and the Fourier transform, we can find the correlation between Ion and IMEP. Figure 3-16 shows that 2nd and 4th order of the ion signal have poorer correlation with IMEP than the crank signal does. The signal can still be useful as crank based IMEP and misfire detection does not work as effectively at high rpm operation [13].

Figure 3-16: Ion Magnitude correlation with IMEP

The 2nd and 4th order ion phase also have an interesting non-linear relation with IMEP.

Figure 3-17: Ion Phase correlation with IMEP

Independently the information from the ION signal is not a strong predictor of IMEP.

Feature Selection will show possible dependencies on other signals.

4 Regularization and Network Structure

Many data driven models exist for estimating an output variable using multidimensional input data. Simple linear regression can be multidimensional and a robust approach to many data learning problems. As can be seen in Figure 3-17, some of our signals have a non-linear relation with IMEP; the nonlinear signals demand a model capable of coping with them. Neural networks are such a model, nonlinear activation functions allow the model to make nonlinear relations between input and output data.

4.1 Rectifier Activation Function

Neural Networks are a series of linear weights and biases separated by nonlinear activation functions. An activation function is a simple relation between an input and an output. The non-linear capabilities of neural networks are due to non-linear activation functions. Without activation functions, a neural network would be simple linear regression. The activation function can be any relation. Common activation functions include sigmoid functions, Rectified Linear functions (ReLus), or leaky rectifiers (leaky ReLus). The ReLu mimics a rectifier in electrical engineering, allowing positive values (positive half of AC voltage) to flow through while stopping negative values (negative half of AC). The ReLu activation function is shown in Figure 4-1: Rectifier Function.

Figure 4-1: Rectifier Function

The rectifier is a simple and efficient way of including nonlinearity in the system.

Sigmoid and Leaky Relu functions can perform just as well and can even reduce network size. The advantage of ReLu functions comes when training the network. ReLu functions are zero for all values less than zero. This introduces the idea of sparsity in a network, if one of your ReLu functions outputs zero for a section of data, it can disable entire sections of your network; this may seem like a disadvantage but compare it to linearization in the control world. During linearization a non-linear system is simplified around a point to make analysis easier. Sparse activation can perform similar operations; one section of the network will be active to make estimations on one linear operation area of the system. As the engine moves to a different operation area that is not linearly related to the first, the neural network activates an entirely different section of the network, disabling the first section and rendering it 'sparse'. The sparsity of ReLu

functions make the network much faster to train and execute, important when considering real time operation on embedded systems.

All variables are regularized to a mean of 0 with a standard deviation of 1 before entering the network, and learned weights and biases allow the system to adjust what values enter the rectifier and therefore what values are excluded by it. The rectifier is called an activation function because it is only 'active' above input values of 0. Ideally, to speed up a network, different areas of the network are activated for different inputs, this is called sparse activation. Sparse activation mimics linearization at many points, the network is activated based on inputs and the network at those inputs approximates a linear function with the output.

A sigmoid activation function or a leaky rectifier activation function are also both useful in the system but are more computationally costly as they are active for all inputs - $-\infty$ to ∞ , whereas ReLu layers are active only for $x > 0$.

4.2 Regularization

Input data for a neural network can have any range. The engine speed and fuel delivered are measured in rpm and grams respectively. Engine speed can vary from 600 to 6000 rpm, while grams of fuel injected will never surpass 1. At the beginning of network training, rpm will have a much larger effect on the engine than fuel because of its large value. The network will have to learn to scale down the engine speed or scale up the fuel,

and this takes iterations. This scaling can take so many iterations to learn that the network may hit the maximum iterations before converging on a solution.

To speed up training, we can give each input the same importance from the beginning. This is done by subtracting the mean from the variable and dividing by the standard deviation. This will result in zero-mean and 1 standard deviation data. This also improves the utility of our ReLu activation functions as they require inputs to have both positive and negative values to have any effect.

4.3 Network Structure

Neural networks can range broadly in size. Shallow networks can be used broadly for regression problems, while deeper networks are effective for image classification and audio recognition. The widely used GoogLeNet model is a 22 layer network structure used for image classification [14]. Models like this use large 2-dimensional input arrays that represent images, our data is one dimensional and our model structure can be smaller because of this.

4.3.1 Required Network Layers

There are some required network layers that cannot be modified. The first layer is an input layer to the network. Our data is fed to the network in 22 parameters, the first layer network size is determined by a sequence input layer. The sequence input layer takes the parameters and connects them to the next layer.

A regression layer computes the mean-square error for the network training. If the network is designed to predict multiple parameters, the name-value pairs can also be defined here in the MATLAB implementation. This layer necessarily follows a fully connected layer with number of elements equal to the number of responses (one in this case).

4.3.2 Fully Connected Layers

Fully connected layers take each parameter from the previous layer, multiply it by a learned weight and add a learned bias to create a new parameter, there are a set number of nodes in each fully connected layer. Figure 4-2 shows the structure of a sequence input layer on the left with a full connected layer on the right, with the learned weights and biases represented by the colored lines.

Figure 4-2: Sequence Input Layer with Fully Connected Layer

Each fully connected layer is essentially linear regression to several output nodes. In between the linear regression layers, rectified linear units allow the model to cope with non-linearities.

4.3.3 Connecting Layers and Network Depth

The simplest neural network design that can cope with nonlinearities is a fully connected layer followed by a ReLu layer followed by a fully connected layer. This is shown below in Figure 4-3.

Figure 4-3: Simple Neural Network with Rectified units

When counting network layers, the input layer is omitted, and the activation function is not counted. The above network has an input layer (not counted), a hidden layer with 3 neurons (layer 1) and 1 output layer (layer 2). A shallow network like this is easy to train, efficient in operation, and small to store.

4.4 Neural Network Depth and Structure Decisions

In our implementation, it was found that a shallow network worked well for IMEP estimation while the engine was running normally but performed poorly when tasked with estimating IMEP over a misfire cycle. This is likely due to the significant non0-linearity between a combustion cycle and a non-combustion cycle. Adding more layers to the network made it capable of estimating both healthy combustion cycles as well as misfire cycles. The network used for IMEP prediction has 2 fully connected hidden layers (each with ReLu activation functions) and one output layer, for a 3-layer network. Each hidden layer has 120 nodes and the output layer condenses to a single node. The network is created using the code:

```
reg.in.mean = mean(input)
reg.out.mean = mean(output)
reg.in.std = std(input)
reg.out.std = std(output)
input = (input - reg.in.mean)./reg.in.std;
output = (output - reg.out.mean)./reg.out.std;

featureDimension = size(input, 2)
numResponses = 1;
layers = [ ...
    sequenceInputLayer(featureDimension)
    fullyConnectedLayer(120)
    reluLayer
    fullyConnectedLayer(120)
    reluLayer
    fullyConnectedLayer(numResponses)
    regressionLayer];

options = trainingOptions('adam', ...
    'MaxEpochs', 400, ...
    'MiniBatchSize', 250, ...
    'InitialLearnRate', 0.003, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.75, ...
    'LearnRateDropPeriod', 80, ...
    'GradientThreshold', 1, ...
    'Shuffle', 'never', ...
```

```
        'Plots','none');  
std_net = trainNetwork(input', output',layers,options);
```

This code regularizes the input data (regularization parameters must be stored for future variables), creates a 3 level ReLu Feed Forward network, and trains it over 400 epochs with optimized learning rate dynamics (Chapter 4.6).

4.5 Neural Network Implementation

All neural networks used in this paper were designed with MATLAB's Deep Network designer. This method has the advantage of being simple and easy to modify network structure. The user defines the number and type of input (image, continuous, discrete), the number and type of hidden layers (activation functions, fully connected layers, convolutional layers, Long-Short Term Memory layers, etc.), and the output type (classification, continuous). The user defines a training scheme (Adam's is used here) along with learning hyperparameters and supplies training and testing data. The system is easy to use and fast to implement, but does have some draw backs. Other systems are more commonly used in programming circles, TensorFlow, PyTorch, Keras and more support for custom cost functions and layers may be available. For instance, it would be interesting to use experiment with implementing the physical equation between cylinder pressure and crank acceleration in a neural network activation function; while possible in MATLAB, far more support exists for TensorFlow. In addition, using TensorFlow tends to afford more legitimacy in computer science circles. For speed of development, MATLAB was used here.

4.6 Learning Rate and Drop Factor

Our neural network has 2 types of learned parameters, weights and biases. Each neuron connection has a weight and each neuron has a bias. In our system, sequence data is used as an input to the model and sequence data is the output. The difference between the truth and the model output is our error. As is often the case in optimization, the squared error is the cost function (negative error is as bad as positive error). Training the model is essentially optimizing each weight and bias over the set of training data to minimize error on the output. Optimization is a field of research on its own and many methods have been used to train neural networks. By far the most common is batch gradient descent with backpropagation, which is used here. Backpropagation calculates the gradient of the cost function for a single input. Gradient Descent then changes the weights based on each cost function gradient.

Gradient Descent makes steps based on gradient in the direction of minimum cost function. The size of those steps depends on the gradient and on the learning rate of the system. The learning rate is a user-set hyperparameter. Too large of a learning rate and the optimization can explode, too small and the optimization can be computationally costly to reach a solution [15]. Too small of a learning rate may also settle the system into a local minimum without searching the wider area for a global solution. It follows that the learning rate should be large when training begins, and smaller as training concludes. The large learning rate allows the system to find the most promising minima solutions, and the small learning rate allows the system to settle into the best solution.

This technique, variable learning rate, is widely used and has some alternatives. A cyclic learning rate follows a triangle wave shape and allows the system to explore many local minima [16]. Cyclic learning rates are often effective when a deep network has many local minima, and each should be explored to find the best. The number of local minima in a deep network increase exponentially with the number of parameters [17]. Our system is relatively simple, with less than 50 parameters (compared to an image classification network that can have thousands), so cyclic learning rate becomes less important.

A piecewise learning rate drops the learning rate by a specified factor over a specified number of epochs. Finding a learning rate, drop factor, and drop period that lead to efficient convergence is largely a trial and error process. With a base learning rate of .004 and a drop factor of .4, it was found that the network took around 80 epochs to converge on a loss function value. I set the drop period to 80 epochs and completed a 2 dimensional grid search on base learning rate and drop factor over the space shown in Figure 4-4.

Figure 4-4: Learning Rate and Drop Factor Grid Search

Figure 4-4 shows that the model error is fairly robust to learning rate and drop factor, so long as drop factor is maintained between .2 and .8 and learning rate is between 0.002 and 0.005. Finding the correct learning rate is important as different neural network problems have different cost function shapes, and one set of parameters will not work well for all problems.

Using a learning rate of .01 and a drop factor of .1, I found that training the model on the same data repeatably lead to a standard deviation in model error of 1.2 kPa. This large deviation makes optimization of the model difficult, as unless the improvement from optimization is significant, it is impossible to tell if the change in the model was due to altering preprocessing, or due to the deviation in model training.

Using a learning rate of .003 and a drop factor of .45, the standard deviation in model error was reduced to .35 kPa. Further reduction would be valuable, but this is the lowest standard deviation found by altering drop factor and learning rate. In future work, it would be interesting to see if a cyclic learning rate could be used to further reduce the deviation in model error, the thinking being that a cyclic learning rate might help the model settle into the same weights and biases each time rather than settle into the first optimum that it finds.

5 Preprocessing Optimization

As discussed in sections 3.1 and 3.2, window length and phasing for the crank and exhaust signal determine what portion of data is extracted. Finding the optimal window length and phasing is critical to ensuring that the system is successfully extracting all useful information from our sensors. There are four parameters to adjust, exhaust window length and phasing, and crank window length and phasing. The objective function to minimize is model error, which has a standard deviation of .35 kPa while training on the same data. The system to optimize is shown in Figure 5-1.

Figure 5-1: Structure of model to be optimized

5.1 Stochastic Optimization

The model error can change between two executions of the same function, this makes it a stochastic variable. Gradient Descent requires being able to compute the partial derivative of the cost function with respect to your optimization parameters. In our case of parameter optimization, finding the gradient at a single point is impossible due to the

random nature of our cost function, making optimization algorithms that depend on the gradient fail.

Genetic Algorithms have been getting a lot of attention as an efficient method of searching for a minimum in a large solution space of a function without a calculable derivative. Our problem meets many of these criteria but is also stochastic. This random aspect of our problem makes the genetic algorithm fails as it only queries a parameter choice once, making future parameter choice decisions based off of a single query with a random aspect.

5.1.1 Pattern Search Algorithm

Optimization systems exist for stochastic variables, pattern search algorithms and smoothed grid search algorithms come to mind. Pattern search algorithms start at a random point in the solution space and start a 4-point search surrounding the initial point. The first point queried that is lower than the initial point is our new central point, the search area is increased by a set step size and the search is completed again at the new point. If no point is found in the search that is smaller than the current central point, the search area is reduced by a set step size and the search is started again at the same central point. This is repeated until the search size is below a threshold. The pseudocode for this system in one dimension is shown below.


```

Initialize:
x0 = rand(solution_space)
f0 = function(x0)
while: step_size < threshold
    f1 = function(x0 + step_size)
    f2 = function(x0 - step_size)
    if f1 < f0
        x0 = x0 + step_size
        step_size = step_size*expansion_factor
    else if f2 < f0
        x0 = x0 - step_size
        step_size = step_size*expansion_factor
    else
        x0 = x0
        step_size = step_size*contraction_factor
end

```

This system can be optimized (store function returns rather than execute multiple times per grid search) and generalized to multiple dimensions. This code was written in MATLAB for our multidimensional optimization and was tested on known functions. (stored in appendix E). Figure 5-2 shows the algorithm converging on a known function minimum. The red cross is the known minimum and the blue stars represent the system converging. The code I wrote for this algorithm also computes the gradient between the current point and the last point, using the gradient and maximum iterations as an exit condition.

Figure 5-2: Pattern Search Algorithm Example

When used with a deterministic function, the pattern search algorithm is proven to guarantee global convergence [18]. However, when the data is stochastic, no such proof can be obtained, and if the data is stochastic enough, convergence cannot be obtained. In our case, it seems that the random portion of network results dominates the gains seen through parameter optimization. When the pattern search algorithm is run with a minimum step size of 5 degrees and an initial step size of 45 degrees for all four parameters, the algorithm does not converge before exceeding the maximum number of iterations set at 4,500.

5.1.2 Reducing Standard Variation in Training

The pattern search algorithm did not converge because the standard deviation of the objective function is too large when compared to the change in the objective function for a change in the preprocessing parameters (window length/phasing). To fix this, reduce

the standard deviation of the objective function (rms model error). To increase stability of the model from one training to the next (reduce standard deviation) we can change the training parameters (learning rate, learning drop factor, learning drop period).

Learning rate is the amount that network bias and weights change with each iteration.

Too small of a learning rate and the network will take a long time to converge on a solution. Too large of a learning rate and the network will bounce around the optimal solution without refining the search. A dynamic learning rate allows the network to use a large learning rate to quickly find the area surrounding an optimal solution and then reduce the learning rate to refine the search. The learning rate, drop period, and drop factor optimization was discussed in Chapter 4. Optimization reduced standard deviation in model error from 1.4 kPa to .35 kPa over the MTU drive cycle. Despite this reduction the objective function was still too stochastic for the pattern search algorithm to converge.

5.1.3 Brute Force - Grid Search

A grid search is a less elegant and efficient method of optimization than many but gives us a good idea of what the solution space looks like. Here we will define a grid over which the system will query each point at defined intervals. Base crank window phasing starts at 90 degrees before top dead center, and base exhaust window phasing starts at exhaust valve open. The exhaust signal window is 90 degrees while the crank window is 180 degrees. The phase modifiers were altered from 90 deg advance to 90 deg retard in 30 deg increments. The length modifiers were altered from 0 deg to 180 deg in 45 deg

increments. All permutations over this four dimensional solution space lead to 1764 queried points. After running the preprocessing and network training/estimation algorithm for each point, the 15 lowest error parameters were selected. These are shown in Table 5-1: Optimized Preprocessing Parameter Selection below.

Model Error (rms kPa)	Exhaust Window Retard	Crank Window Retard	Exhaust Length Modifier	Crank Length Modifier
15.9	0	60	0	180
16.1	0	90	180	180
16.1	-30	90	0	180
16.2	0	90	90	180
16.3	-30	60	180	180
16.3	0	0	45	180
16.3	-90	30	0	180
16.4	30	60	0	180
16.4	60	90	0	180
16.4	0	90	0	180
16.4	60	90	45	180
16.4	90	60	45	180
16.4	0	90	45	180
16.4	-30	0	90	180
16.4	30	0	180	0

Table 5-1: Optimized Preprocessing Parameter Selection

It is important to note that the model still has a standard deviation of .35 kPa, and the top 15 parameters vary by less than 2 standard deviations. Because of this variance, the best performing model is not necessarily the best model. The fact that the crank length modifier is 180 degrees in 14 of the 15 top performing models is a strong indicator that 360-degree window is superior to other window sizes (180 deg base window size + 180 deg modifier). Other parameters are less clear, Exhaust length modifier is always positive or zero with a mode of 0. Crank window retard has a mode of 90 degrees and exhaust

window retard a mode of 0. A crank window retard of 90 degrees suggests that the information in the crank signal before TDC is not useful (base window starts at 90-degrees BTDC, the 90-degree retard negates that). A zero-degree retard for exhaust signal is also intuitive as the base window starts at exhaust valve open.

In order to better understand the parameter optimization space, we can visualize 2 dimensions of the optimization with a surface plot. Locking the window lengths at base, an 81-point search was completed around different window phasing values, with the results being shown in Figure 5-3.

Figure 5-3: Grid Search Optimization of Window Phasing

With this analysis, it becomes apparent that the model is much more sensitive to changes in crank window phasing than changes in exhaust window phasing. This could be due to several reasons, either the exhaust contains the same content regardless of the window phasing, or more likely in my opinion, the exhaust pressure is a much less important parameter for IMEP estimation than the crank signal. Concavity exists in both dimensions in Figure 5-3, but the crank window axis is much steeper. Figure 5-3 matches the overall theme in the data, in that the lowest error occurs between 0- and 90-degree crank retard, and 0-degree exhaust retard.

Using the data from Table 5-1, The mode of the lowest 15 error parameters were chosen as our optimal set. This is shown in Table 5-2.

Exhaust Window Retard	Crank Window Retard	Exhaust Length Modifier	Crank Length Modifier
0	90	0	180

Table 5-2: Selected Preprocessing Parameters

6 IMEP Model Error Analysis

The model is capable of estimating cycle-cycle variations in IMEP over both steady and transient drive cycles. This includes estimating IMEP of misfire cycles. For all results in this section the model was trained on 33,148 healthy combustion cycles with 300 misfire cycles included in that. The results are based on 15,571 cycles that the network did not see during training. Figure 6-1 shows the engine speed and load conditions that were in the training data.

Figure 6-1: IMEP model Training and testing data

The training data covers the engine operating condition from idle to over 5000 rpm. The data contains only warm-engine information and the training and testing data were both

taken on the same day (March 3rd, 2020). The weather was 2 degrees Celsius and the data was taken over the MTU drive cycle which is shown in Appendix D: Map of MTU Drive Cycle. The error over the entire test drive cycle is shown below in Figure 6-2: Section of IMEP testing data.

Figure 6-2: Section of IMEP testing data

Much of the data in the testing set is non-fueled cycles below 0 kPa IMEP. This means that the engine is not injecting fuel, combustion is not occurring, and complex control is not required. During these cycles we do not need to predict IMEP. For this reason, they are removed from future error analysis.

If an engine is running in steady state, the IMEP fluctuates from one cycle to the next. This is referred to as cycle-cycle variation. Steady state IMEP has little to no

autocorrelation, meaning that one cycle does not affect the next (there is no pattern). This is shown below in Figure 6-3: Autocorrelation of Steady State IMEP. The autocorrelation multiplies the signal by a delayed value of itself and finds the sum of values. This is used to find a pattern in the cycle-cycle variation of IMEP.

Figure 6-3: Autocorrelation of Steady State IMEP

The cycle-cycle variation in IMEP is stochastic as it varies randomly with the same engine control inputs. Because the IMEP can vary from cycle to cycle while the engine controls are not changed, we need to measure some aspect of combustion to estimate cycle-cycle variation. Our model uses crank acceleration and exhaust pressure measurements before and after combustion to capture this cycle-cycle variation. In order to show that our model estimates this cycle-cycle variation we need to show how much the IMEP varies from cycle to cycle. To do this I have low pass filtered the measured IMEP, we can use that as an estimate of the deterministic portion of IMEP. If the data

driven model can estimate IMEP more accurately than the deterministic estimate of IMEP from the measured signal, the model is capturing cycle to cycle variation.

The measured IMEP and a low pass filtered version of the same data is shown below.

The low-pass data represents the average IMEP over a section, what the IMEP would be if cycle-cycle variation did not exist. If our model is more accurate than this low-pass data, it is capturing cycle-cycle variation. The model-estimated, measured, and low-pass filtered IMEP are shown below in Figure 6-4.

Figure 6-4: Cycle-Cycle Variation in IMEP

Over the section shown in Figure 6-4 the root-mean-square of data driven model error is 16 kPa, while the error in the lowpass measured data is 19.0 kPa. Over this section the model is capable of estimating some element of cycle-cycle variation. Over the entire 15,571 cycle data set, the model error is 19 kPa and the low-pass measured error is 36 kPa, suggesting that the model is estimating cycle-cycle variation. The cycle-cycle

estimation is easier to visualize over a shorter period, in Figure 6-5 it is clear that the model is estimating the cycle-cycle variation.

Figure 6-5: Cycle-Cycle Variation in IMEP

6.1 Model Sensitivity

Model sensitivity refers to the model's robustness in the presence of noise. In our case noise can be anything that hasn't been factored into the model. This can include electrical noise (ignition system, injectors, alternator, radio, etc.), rough roads (interferes with crank signal), changes to ambient conditions, road grade, or one of many other factors [9]. One of the disadvantages of a data driven model is that the model cannot extrapolate for data it does not have. The model can interpolate and extrapolate between and beyond data points that the model has but does not know the relation. For instance, a physics-based model knows the relation between cylinder pressure, force on the piston, acceleration of the rotating assembly, and vehicle acceleration. The data driven model

must learn these relations and does not know the rules to extrapolate beyond the data provided. This can also be an issue for changes in coolant temperature and ambient conditions. A physics-based model can estimate the heat transfer to the cylinder walls in a cold engine, but the data driven model must learn this. Because of this the model can be made more robust by training on a more diverse set of data. This issue was made apparent when the model was trained on data from mid-day on March 3rd, 2020, where the ambient temperature was 2 degrees Celsius and tested on data from the same day but that morning where the temperature was -7 degrees Celsius. Testing and training (different sets) on the mid-day data, the model had an rms error of 18 kPa. When the model was tested on the same drive cycle but from several hours prior with a temperature of -7 degrees Celsius, the model showed significant DC offset error. Over the entire drive cycle the error was 140 kPa, well above the 16 kPa seen previously. In order to correct that, data was taken at various ambient temperatures ranging from -9.4 degrees Celsius to 62 degrees Celsius. The data sets taken for temperature robustness testing is shown below in Table 6-1: Temperature Robustness Data Sets

Date	Test Number	Subtest Section	Route	# of cycles	note	Ambient Temperature
March 3, 2020	1	A	MTU Drive Cycle	6,204		-1.6° C
March 3, 2020	1	B	MTU Drive Cycle	15,490		-1.6° C
March 3, 2020	1	C	MTU Drive Cycle	17,658		-1.6° C
March 19, 2020	2	A	MTU Drive Cycle	23,560		1.7° C
March 19, 2020	2	B	MTU Drive Cycle	11,315		1.7° C
March 19, 2020	2	C	MTU Drive Cycle	2,618		1.7° C
March 19, 2020	2	D	MTU Drive Cycle	6,274		1.7° C
August 31, 2020	3	A	MTU Drive Cycle	9,408	Urban Portion	16.7° C
September 1, 2020	4	A	MTU Drive Cycle	22,027	APSRC to Campus	12.8° C
September 1, 2020	5	A	WOT acceleration		Flat road	20.6° C

Table 6-1: Temperature Robustness Data Sets

The data set ambient temperature is plotted below against the average intake air temperature. To test the robustness of the model to changes in ambient temperature, I removed the August 31st data set at 16.7° C from the training set, trained the model on the rest of the data, and tested the model on the data from August 31st.

Figure 6-6: IAT vs Ambient Temperature

The testing data has an ambient temperature between two training data points. This should allow the model to interpolate between the known data points and make a prediction on the training data with a temperature it is unfamiliar with. First, we will show that if the model has not seen a variety of ambient temperatures, the model cannot predict data with a different temperature. We will train the model on data from March 3rd

and test it on the MTU drive cycle from September 1st. This results in the predictions shown in Figure 6-7.

Figure 6-7: Estimating September 1st data set with single March training example

The model has an rms error of 44 kPa over the drive cycle. The average intake air temperature (IAT) over the training data was 11 degrees Celsius, while the average IAT over the testing data was 26.7 degrees Celsius. Including training data with ambient temperatures surrounding the testing data, we can show both the dependence of the model on ambient conditions, as well as its ability to account for them. The same testing data (September 1st, 12.8° C) was predicted with a network trained on both March 3rd data (-1.6° C) and august 31st data (16.7° C). The results are shown in Figure 6-8.

Figure 6-8: Estimating September 1st Dataset with Varied Training Data

Including a training data with a similar ambient temperature has significantly improved results. The model error has dropped from 44 kPa to 25 kPa over the September 1st (12.8° C) training data. Results were further improved by including all data (exclusive of September 1st) and the model error dropped to 23 kPa, and the results are shown below in Figure 6-9.

Figure 6-9: Estimating September 1st Dataset with Complete Training Data

Through the use of a more diverse training set I was able to make the model more robust to changes in ambient temperature conditions. Other noise sources such as changes in road grade, gear shifts, and vehicle accessories were included in the data sets and do not appear to be a problem for the model.

6.2 Model Error Division

Different sensors pick up different information as engine speed and load shift. One of the advantages of sensor fusion is the ability to exploit different sensors over different operating points. In order to see the models ability to estimate IMEP over different engine operating points, the model error was split between different load and speed regions. The model error for a variety of engine conditions is shown below for the model with the full feature set.

Figure 6-10: Model Error with Full Feature Set

The table can be read as if it were an engine tuning table, with speed increasing from left to right and load increasing from bottom to top. The model error is lowest below 2200 rpm and below 12 bar IMEP; within this region the model can estimate cycle-cycle variation. Above that level the model is still capable of estimating mean IMEP but does not capture cycle-cycle variation.

6.3 Production Sensor Set

The accuracy of the model using only production sensors is also of interest. The model above uses the entire feature set shown in Table 3-1. The only two sensors that are not available in the current production vehicle are the ion and exhaust pressure sensors. With these sensors removed, the model could be run on any production MKC. Figure 6-11 shows the error in the model with production sensors and signals.

Figure 6-11: Model Error with Production Feature Set

Model performance is not significantly affected, and even improves over some operating regions. The model is still capable of estimating with cycle-cycle accuracy below 2200 rpm.

The data used for this model was processed using the code available in Appendix H: Optimizable Windowing IMEP estimation system. The processed data was tested and trained using the optimized neural network developed in the chapter 4.6.

6.4 Feature Reduction: Backward Sequential Feature Selection

Up to this point we have used all of our available sensors and extracted information as inputs to the model. In order to reduce network size, unhelpful or redundant signals can

be removed from the system. A coherence study could be used to find features with similar frequency information to our objective value, IMEP. Coherence is calculated using cross powers and auto powers and is a measure of similarity between two signals. Cross powers and auto powers are both linear operations, meaning coherence is also a linear operation. The relation between input signals and IMEP is not necessarily linear, meaning that coherence may not give us the full picture. A more robust approach to feature importance analysis is backward sequential feature selection (BSFS). BSFS is a ranking algorithm used to find the importance of features to a neural network (or any other model). BSFS begins with the entire feature set, for this case say we have one crank feature, one exhaust pressure feature, and one intake pressure feature (for three total features). The model is run three times for the first iteration, once with the crank feature removed, once with the exhaust feature removed, and once with the intake feature removed. The error in each model is recorded, and the lowest error from these three iterations is found. The lowest error is associated with the least important feature, since removing that feature increased model error the least. For examples sake, say exhaust pressure was the least important feature found in iteration 1. For following iterations, exhaust pressure will be removed. Iteration two then continues, the model first runs with exhaust and crank removed (only intake feature remains) and then run with exhaust and intake removed (only crank feature remains). The lowest model error is again found (for example, say removing crank resulted in the lowest error), and that feature is removed. The only remaining feature is then intake pressure. This gives us a range of features from least important to most important to choose from with the corresponding model

capability. Depending on availability of that feature (whether the sensor already exists, computational cost to acquire) the user can then find the desired balance between model accuracy and required sensors. This can also be used to reduce network size as it shows redundant signals and signals that do not carry information useful for IMEP estimation. This BSFS process was completed for the list of features shown below in Figure 6-12.

Figure 6-12: Backward Sequential Feature Selection Results

The BSFS process shows which features contain information most useful to IMEP estimation. The system also gives us some idea of feature dependency. Figure 6-12 can be read as if each feature were an element of a horizontal bar chart, with the length of the

bar corresponding to feature importance. The white space represents a feature that is no longer included in the set. For example, in Figure 6-12, 2nd order ion amplitude was found to be the least important variable in the first iteration (farthest left) and was removed for future iterations. Fuel injected (reported by ATI ECU) was found to be the most important feature. The figure can also show feature redundancy. It appears that ECU reported intake manifold pressure contains information similar to the fuel injected signal. This result is drawn from the fact that the importance of the fuel feature increases significantly after the MAP signal is removed. It should be noted that the entire set of testing data was considered when completing this feature selection. Completing a new BSFS for each of the operating conditions shown in Figure 6-10 would show the importance of each feature over different engine operating regions.

The code for this process was written from scratch in MATLAB and can be used as a feature selection algorithm for any multi-input data learning model. The code feeds a user-defined function (objective function) a series of training and testing data. The objective function then returns the rms error of the model over the testing data. The rms error of the model is recorded and the BSFS process proceeds. The code is included in Appendix G: Backward Sequential Feature Selection Code.

7 Conclusion and Future Work

When developing a data driven model it is helpful to understand the physics behind what is happening and the relation between inputs and outputs. Here we know that indicated mean effective pressure is our output, and that it is related to crank acceleration, exhaust manifold pressure, and intake manifold pressure. Other inputs, such as the ion sensors, do not have such a direct relation with the output. This is where the data driven model is helpful, we do not need to know the relation between ion signal and IMEP for the signal to be useful. In this paper I have developed an IMEP model using low cost production possible sensors and showed that the system is robust to ambient condition changes and is capable of estimating cycle to cycle changes in IMEP. I have also shown that using my optimized preprocessing and model structure, the system is capable of estimating IMEP across the engine operating range using only production sensors.

7.1 Direction of Future Work

In order to implement this model in a production setting, it must be executable on an engine control unit (ECU). The model calculates three Fourier transforms for every combustion event (or 1 for the reduced feature set model) and runs the data through a 3-layer neural network, all of this must be implementable on a production ECU. Computational load could be additionally reduced by calculating the Discrete Fourier Transform for only the orders of interest.

Further testing should also be completed at varying elevation levels, to ensure that the model is capable of coping with changes in ambient pressure. The data used to test and train the model contain slight changes elevation, but only spans the range of 195 meters to 265 meters above sea level. More significant changes would show the models ability to cope with changes in ambient pressure.

All testing for this model was completed on an automatic transmission vehicle with a torque converter. If the model were to be implemented on a manual transmission vehicle or automated clutch vehicle, the crank dynamics may change and would need to be investigated.

7.2 Conclusions

In this paper I have shown that it is possible to estimate cycle to cycle variation in a turbocharged gasoline engine using production sensors. I have developed a model and tuned the preprocessing techniques and the model structure. The results of this study make model predictive control using IMEP as a feedback signal possible without the need to sense in-cylinder pressure.

References

- [1] X. Wang *et al.*, “DSCC2019-9073 PREDICTION WITH RECURSIVE-LEAST-SQUARE BASED ADAPTATION FOR CLOSED-,” pp. 1–10, 2019.
- [2] K. Prabhu, “Sensor Fusion for Spark-Ignition Engines,” Michigan Technological University, Houghton, Michigan, 2020.
- [3] M. Nischal, “Application of Sensor Fusion for SI Engine Diagnostics and Combustion Feedback,” Michigan Technological University, Houghton, Michigan, 2019.
- [4] Louis Yizhang Liu, D. P. Kultgen, J. D. Naber, and Hanlong Yang, “MISFIRE DETECTION IN AN INTERNAL COMBUSTION ENGINE,” US2005/0033501A1, 2005.
- [5] C. I. Rachmil, D. L. McKay, and G. W. Malaczynski, “Method for Estimation of Indicated Mean Effective Pressure for Individual Cylinders From Crankshaft Acceleration,” US 7.623,955B1, 2009.
- [6] A. J. Morello, J. R. Blough, J. Naber, and L. Jia, “Signal Processing Parameters for Estimation of the Diesel Engine Combustion Signature,” *SAE Int. J. Passeng. Cars - Mech. Syst.*, vol. 4, no. 2, pp. 1201–1215, 2011, doi: 10.4271/2011-01-1649.
- [7] L. Jia, J. Naber, J. Blough, and S. A. Zekavat, “Accelerometer-based combustion

- metrics reconstruction with radial basis function neural network for a 9 L diesel engine,” *J. Eng. Gas Turbines Power*, vol. 136, no. 3, pp. 1–9, 2014, doi: 10.1115/1.4025886.
- [8] L. Jia, J. D. Naber, and J. R. Blough, “Review of Sensing Methodologies for Estimation of Combustion Metrics,” *J. Combust.*, vol. 2016, 2016, doi: 10.1155/2016/8593523.
- [9] V. T. Nascimento, L. R. Pumputis, C. Q. Pimenta, I. C. Trevas, and W. R. Navegantes, “SAE TECHNICAL Knock sensor background analysis and their relation with combustion pressure curve Knock Sensor Background Analysis and Their Relation with Combustion Pressure Curve,” 2020.
- [10] C. Rugland and O. Stenlaas, “Knock Sensor Based Virtual Cylinder Pressure Sensor,” *SAE Tech. Pap.*, vol. 2019-Janua, no. January, pp. 1–11, 2019, doi: 10.4271/2019-01-0040.
- [11] M. Wagner, J. F. Böhme, and J. Förster, “In-cylinder pressure estimation from structure-borne sound,” *SAE Tech. Pap.*, no. 724, 2000, doi: 10.4271/2000-01-0930.
- [12] K. Rao and P. Yip, *The Transform and Data Compression Handbook*, 1st ed. New York: CRC Press, 2000.
- [13] K. Iida, K. Akishino, and K. Kido, *IMEP Estimation from Instantaneous*

Crankshaft Torque Variation. 1990.

- [14] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *J. Chem. Technol. Biotechnol.*, 2015, doi: 10.1002/jctb.4820.
- [15] M. A. Nielsen, *Neural Networks and Deep Learning*. 2015.
- [16] L. N. Smith, “Cyclical Learning Rates for Training Neural Networks,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2017, pp. 464–472, doi: 10.1109/WACV.2017.58.
- [17] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, “Snapshot ensembles: Train 1, get M for free,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–14, 2017.
- [18] V. Torczon, “On the convergence of pattern search algorithms,” *SIAM J. Optim.*, vol. 7, no. 1, pp. 1–25, 1997, doi: 10.1137/S1052623493250780.
- [19] “Google Maps.” Houghton, Michigan, 2020, [Online]. Available: <https://www.google.com/maps/@47.139409,-88.5697384,13.17z/data=!5m1!1e4>.

Appendices

Appendix A: EcoBoost 2.0 Cam Phasing

Appendix B: Crank Acceleration Correlation with IMEP

Appendix C: Visual Grid Search Logic

Appendix D: Map of MTU Drive Cycle

[19]

Appendix E: Pattern Search Algorithm Code (MATLAB)

```
syms x y
% f = @(y, z) sin(y) * sin(z);
f = @(x,y) -(exp(-(x/3).^2-(y/3).^2) + exp(-(x+2).^2-(y+2).^2))+.04*sin(2000*x);

% f = sin(x) + cos(y);

% visualized pattern search
x0 = [2,2];
n = numel(x0)

step = 10
step0 = step
clc
clf
min_sol_prev = 10;
sol0 = 100
max_iterations = 100;
% fig stuff
h = figure(1);
axis tight manual % this ensures that getframe() returns a consistent
size
filename = 'testAnimated.gif';
%
for k = 1:max_iterations
    sol0_prev = sol0;
    [A,B] = meshgrid([x0(1)-step x0(1)+step],[x0(2)-step x0(2)+step]);
    c=cat(2,A',B');
    d=reshape(c,[],2)';
    %     clf
    %     fcontour(f)
    ezsurf(f)
    hold on
    for i = 1:length(d)
        sol_temp(i) = f(d(1,i), d(2,i));
    end

    plot3(d(1,:), d(2,:), sol_temp,'b*')

    title('pattern search algorithm example')
    xlabel('preprocessing parameter 1')
    ylabel('preprocessing parameter 2')

    %     plot(x0(1), x0(2), 'r*')

    extra = 1;
    %     xlim([-10 10])
    %     ylim([-10 10])
    %     view(208, -25)
```



```

view(90,90)
set(gcf,'color','w');
drawnow
    % Capture the plot as an image
frame = getframe(h);
im = frame2im(frame);
[imind,cm] = rgb2ind(im,256);
pause(.1)
sol0 = f(x0(1), x0(2));

for i = 1:length(d)
    sol(i) = f(d(1,i), d(2,i));
    if sol(i) < sol0 % Check if the most recent objective function
                    % evaluation is less than the current best sol
        x0 = d(:,i);
        break
    end
end
if min(sol) > sol0
    step = step/2 %+rand(1)*step/4;
    if abs((max(sol_temp)/min_sol_prev - 1)) < .001 & step <
step0/10
        fprintf('\nextit condition met: solution gradient and step
below threshold \n')
        fprintf('Number of iterations: %3.0f, ',k)
        fprintf('solution:\n [%3.2f  %3.2f]\n\n', x0(1), x0(2))
        fprintf('minimum objective function value found: %3.2f,
',sol0)
        plot(x0(1), x0(2), 'rx')
        break
    end
else
    step = step*2-rand(1)*step/4;
end
min_sol_prev = min(sol_temp);
if k == max_iterations
    'exit condition met: Maximum iterations'
end
% Write to the GIF File
if k == 1
    imwrite(imind,cm,filename,'gif', 'Loopcount',inf);
else
    imwrite(imind,cm,filename,'gif','WriteMode','append');
end
end
end

```

Appendix F: Kistler In Cylinder Pressure Sensor Specifications

PiezoStar® Pressure Sensor

Type 6125C...

Ground isolated, pluggable for Pressures up to 300 bar

Ground-isolated high-temperature pressure sensor with integral connecting cable for measuring cylinder pressures in combustion engines.

- Ground-isolated
- Without additional cooling
- Low thermal shock error, very low load-change drift
- High sensitivity using new PiezoStar crystal
- Available with oil-proof cable Type 1983AC1

Description

The use of new PiezoStar crystals in the piezoelectric pressure sensor Type 6125C... made it possible to boost sensitivity to 37 pC/bar and the pressure range to 300 bar. This doubling of sensitivity compared with the previous generation product and isolation of the sensing element prevent of noise currents and allow interference-free measurements, when potential differences exist between engine and measuring system.

This new design also reduces the change in sensitivity with temperature. The new sensor is characterized by minimal thermal shock and outstanding linearity. Despite all these improvements its mounting dimensions remain fully compatible with superseded Types 6123..., 6125A... and 6125B...

This PiezoStar sensor is available with high-temperature connecting cable Type 1967A1 in insulated metal sheathing or oil-proof fluoropolymer cable Type 1983A.

The pressure sensor Type 6125C... is also available with PiezoSmart®. This is an active system for automatic identification of individual pressure sensors and provides automatic parameter setting of measuring chains (see description of PiezoSmart system for more information).

Applications

The uncooled sensor Type 6125C... is suitable for accurate measurement in gasoline and diesel engines. Its ground-isolated design makes it ideal for test stands with ground loop problems. The shoulder sealing of this sensor allows it to be mounted with its front flush in the cylinder head. A wide variety of accessories allow alternative configurations; for example use of a sleeve permits mounting the sensor through a water-cooling channel.

Type 6125C...U20 Version

For applications mainly in the knocking range or at very high pressure rises, use of Type 6125C...U20 with reinforced diaphragm (heavy duty version) is recommended.

Technical Data

Measuring range	bar	0 ... 300
Calibrated partial ranges	bar	0 ... 100, 0 ... 200, 0 ... 250, 0 ... 300
Overload	bar	300
Sensitivity at RT	pC/bar	~36
Natural frequency	kHz	>70
Linearity, all ranges (at 23 °C)	%FSO	≤±0,4
Acceleration sensitivity		
axial	bar/g	<0,003
radial	bar/g	<0,0005
Operating temperature range	°C	-20 ... 350
Temperature, min./max.	°C	-50/400
Sensitivity change		
250 °C ±100	%	±1
RT ... 350 °C	%	±2
Thermal shock error		
at 1 500 1/min, IMEP = 9 bar		
Δp (short-term drift)	bar	≤±0,3
ΔIMEP	%	<±1,5
Δp _{max}	%	<±1
Insulation resistance at 30 °C	Ω	≥10 ¹³
Ground isolation 30 °C	Ω	≥10 ⁸
Torque wrench setting	N-m	10
Weight without cable	g	30
Connector, ceramic insulator	—	KIA G 10-32

Page 1/4

This information corresponds to the current state of knowledge. Kistler reserves the right to make technical changes. Liability for consequential damage resulting from the use of Kistler products is excluded.

©2008 ... 2015, Kistler Group, Eulachstrasse 22, 8408 Winterthur, Switzerland
 Tel. +41 52 224 11 11, Fax +41 52 224 14 14, info@kistler.com, www.kistler.com
 Kistler is a registered trademark of Kistler Holding AG.

6125C_000-695e-10.15

Type 6125C...U20 (other specifications as for Type 6125C...)

Sensitivity	pC/bar	≈-33
Measuring range	bar	0 ... 300
Calibrated partial ranges	bar	0 ... 100, 0 ... 200, 0 ... 300
Overload	bar	350
Sensitivity shift 250 °C ±100	%	±1,2
Thermal shock error (at 1 500 1/min, IMEP = 9 bar)		
Δp (short time drift)	bar	≤±0,4
ΔIMEP	%	≤±2
Δp _{max}	%	≤±1,5

The pressure sensors Type 6125C... can be mounted flush with the wall of the combustion chamber or recessed in an

Fig. 1: Direct mounting of sensor Type 6125C1... with mounting nut

Fig. 2: Mounting sensor Type 6125C... with mounting sleeve

Fig. 1a: Bore for direct mounting

Fig. 2a: Bore for mounting with mounting sleeve

6125C_000-695e-10.15

Fig. 8: Step drill Typ 1337, 1337A, 1337A2

Included Accessories

- Sensor Type 6125C... with mounting nut and cable as specified in ordering key
- Adapter, KIAG 10-32 neg. – BNC pos.
- Cu seal
- Ni seal

Type/Art. No.

6125C...
 1721
 1102
 1102A

Optional Accessories

- Torque wrench 5 ... 40 N-m
- Tubular socket wrench SW8
- Screw tap M10x1
- Extraction tool for sensor
- Extraction tool for sensor dummy Type 6469A1 and Type 6469A2
- Mounting sleeve, M10x1 incl. O-ring
- Mounting sleeve, 3/8x24 UNF incl. O-ring
- O-ring for mounting sleeve
- Mounting sleeve M10x1
- Mounting nut 3/8x24 UNF
- Clamping ring for mounting nut
- Cu seal
- Ni seal
- Mounting sleeve M10x1
- Flame guard
- Spare cable, KIAG 10-32, L = 1 m
- Spare cable, fluoropolymer KIAG 10-32, L = 1 m
- Adapter M10x1 for pressure generator Type 6906A
- Adapter 3/8x24 UNF for pressure generator Type 6906A
- Dummy sensor M10x1
- Dummy sensor 3/8 – 24 UNF
- Engine adapter M14x1,25/M10x1
- Engine adapter M14x1,25/M10x1 retracted
- Engine adapter M14x1,25/3/8x24 UNF
- Step drill for mounting nut M10x1
- Step drill for mounting nut 3/8x24 UNF
- Finishing tool/reamer

Type/Art. No.

1371B...
 1373
 1353
 1317
 1319
 6433A...
 6434A...
 1169
 6473
 6474
 1141B...
 1102
 1102A
 6523A...
 6539A3
 1967A1
 1983AC1
 6952A1
 6952A2
 6469A1
 6469A2
 6583
 6583Q04
 6584
 1337A
 1337A2
 1337

Ordering Key

Without mounting nut	0
With mounting nut M10x1	1
With mounting nut 3/8x24 UNF	2

Cable Arrangement

Without cable	0
With cable Type 1967A1, L = 1 m	1
With fluoropolymer cable Type 1983AC1, L = 1 m	2

Without PiezoSmart	–
With PiezoSmart	5

Version

Standard	–
Reinforced diaphragm	U20

Type 6125C

Detailed information about PiezoSmart sensor identification may be found in the PiezoSmart brochure.

Ordering Example

Version with cable Type 1967A1 and PiezoSmart
 Version with M10x1 mounting nut and fluoropolymer cable Type 1983AC1

Type

6125C01S
 6125C12

Kistler In Cylinder Pressure Transducer

											Location		
#	Model	Serial Number	Sensitivity	pC/bar	Bar 0-250	Calibration Date	Program	In Stock	Sent for Calibration	Dyno Cell	Vehicle Tag	Model	Engine Cylinder
1	6125A	953357	15.75	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 1
2	6125A	953382	15.67	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 2
3	6125A	953383	15.86	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 3
4	6125A	953385	15.69	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 4
5	6125A	1003068	15.88	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 5
6	6125A	1003070	15.3	pC/bar	250	1/7/2010	LSPI				061W833	F150 Truck	Cyl 6
7	6125A	953380	15.89	pC/bar	250	2/14/2011					061W039	MKC	Cyl 1
8	6125A	953384	15.4	pC/bar	250	2/14/2011					061W039	MKC	Cyl 2A
9	6125A	1001965	16.1	pC/bar	250	3/12/2012					061W039	MKC	Cyl 2
10	6125A	1003066	15.77	pC/bar	250	1/4/2011					061W039	MKC	Cyl 3
11	6125A	1003067	15.94	pC/bar	250	1/4/2011					061W039	MKC	Cyl 4A
12	6125A	1003069	15.92	pC/bar	250	1/4/2011					061W039	MKC	Cyl 4
13	6125A	615358	15.6	pC/bar	250	3/12/2012					MTU	2.0 GTDI	Cyl 1
14	6125A	Transducer Plug									MTU	2.0 GTDI	Cyl 2A
15	6125A	615359	15.77	pC/bar	250	3/12/2012					MTU	2.0 GTDI	Cyl 2
16	6125A	953358	15.65	pC/bar	250	2/14/2011					MTU	2.0 GTDI	Cyl 3
17	6125A	Transducer Plug									MTU	2.0 GTDI	Cyl 4A
18	6125A	641982	15.59	pC/bar	250	2/14/2011					MTU	2.0 GTDI	Cyl 4
17	6125A	993934	15.46	pC/bar	250	1/4/2011	Sent to MTU						
18	6125A	904247	15.83	pC/bar	250	1/4/2011	Sent to MTU						
19	6125A	904252	15.62	pC/bar	250	1/4/2011	Sent to MTU						
20	6125A	1003063	15.87	pC/bar	250	1/4/2011	Sent to MTU						
21	6125A	1003064	15.74	pC/bar	250	1/4/2011	Sent to MTU						
22	6125A	1003065	15.82	pC/bar	250	1/4/2011	Sent to MTU						
23	6125A	546330	14.99	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
24	6125A	546332	14.91	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
25	6125A	546333	15.38	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
26	6125A	546335	14.85	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
27	6125A	546336	14.95	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
28	6125A	546337	15.17	pC/bar	250	1/4/2011	Ryan Baker project				061W110	Edge	
29	6125A	993907	15.5	pC/bar	250	5/26/2010	Ryan Baker project				061W747	Explorer	
30	6125A	1003071	15.93	pC/bar	250	5/26/2010	Ryan Baker project				061W747	Explorer	
31	6125A	641985	15.69	pC/bar	250	8/20/2010	Ryan Baker project				061W747	Explorer	
32	6125A	645194	15.98	pC/bar	250	8/20/2010	Ryan Baker project				061W747	Explorer	

Pressure Transducers In Stock

33	6125A	546329	15.29	pC/bar	250	8/20/2010		x					
34	6125A	546331	15.2	pC/bar	250	8/20/2010		x					
35	6125A	645197	15.9	pC/bar	250	8/20/2010		x					
36	6125A	641963	15.57	pC/bar	250	2/14/2011		x					
37	6125A	904246	15.89	pC/bar	250	3/12/2012		x					
38	6125A	612734	15.02	pC/bar	250	3/12/2012		x					
39	6125A	612735	15.67	pC/bar	250	3/12/2012		x					
40	6125A	904255	15.77	pC/bar	250	3/12/2012		x					
41	6125A	577118	15.95	pC/bar	250	3/12/2012		x					
42	6125A	1053077	16.38	pC/bar	250	3/12/2012		x					
43	6125A	577119	16.09	pC/bar	250	3/12/2012		x					
44	6125A	1067884	16.72	pC/bar	250	3/12/2012		x					
45	6125A	577121	15.99	pC/bar	250	3/12/2012		x					
46	6125A	546339	15.43	pC/bar	250	3/12/2012		x					
47	6125A	573756	16	pC/bar	250	3/12/2012		x					
48	6125A	573787	16.04	pC/bar	250	3/12/2012		x					
49	6125A	966884	15.32	pC/bar	250	3/12/2012		x					

New Pressure Transducers

50	6125C	1829709	34.05	pC/bar	200	3/1/2010		x					
51	6125C	1829711	34.34	pC/bar	200	3/1/2010		x					
52	6125C	1829713	35.01	pC/bar	200	3/1/2010		x					
53	6125C	1829714	34.45	pC/bar	200	3/1/2010		x					
54	6125C	1829716	34.36	pC/bar	200	3/1/2010		x					
55	6125C	1829717	33.93	pC/bar	200	3/1/2010		x					
56	6125C	1829718	34.27	pC/bar	200	3/1/2010		x					
57	6125C	1829719	34.72	pC/bar	200	3/1/2010		x					
58	6125C	1829720	33.47	pC/bar	200	3/1/2010		x					
59	6125C	1829681	35.5	pC/bar	200	3/1/2010		x

AVL 366C ANGLE ENCODER

Know exactly where you are

THE SOLUTION

The Angle Encoder Unit with integrated marker disk and mounted optical cable and electronics is suitable for mounting on a free shaft end for flange mounting. The electronics can be connected directly to the AVL-X-ion series. As the measuring principle is based on optical transmission, the angle encoder is extremely robust against the high temperatures, electrical interference and vibrations of an engine.

The marker disc is equipped with 720 marks meaning the encoder provides a live resolution of 1/2 deg. CA. In addition to CDM and Trigger, the electronics provide information about the direction of rotation – right or left. In combination with AVL-X-ion for example, the actual angle position is available when stop-start-operation is performed.

THE APPLICATION

Engine tests are now even expanding to vehicle testing on the road (RDE).

AVL has developed a new optical encoder 366C that is much smaller, lighter and with less inertia than its predecessor. The 366C can be installed and operated in the vehicle – beyond this, the new encoder is equipped with fibre optical cables and a CDM-acquisition principle that generate a phase shift in signal evaluation for detection of rotation direction – either left or right.

For this reason, the absolute position of the crankshaft is known at any time in combination with X-ion and when the engine is shut down, the shaft can be positioned at any desired crank angle for start-stop applications.

THE CHALLENGE

Usage of angle encoder at the test cell is a standard application. Today's customer expectations are "plug and play": Easy installation of the encoder, precise and secure measurement of many different values in parallel and highly resolved and reliable data acquisition. This is required for all engine types – no matter whether it is a passenger car, heavy duty, racing or two-wheeler engine.

TECHNICAL DETAILS

- 720 crank degree marks (CDM) plus 1 trigger signal (TRG) per revolution as an LVDS-signal
- Precise measurement from $n = 0\text{rpm}$ to $n = 20,000\text{rpm}$
- Additional information about direction of rotation: rotation right - left, evaluation via AVL-X-ion
- Mounting situation (flange & supporting arm) compatible with 365C
- Additional signal-out-plug integrated in electronics for CDM transfer e.g. to the test cell (PUMA)
- Electrically compatible with Indicating devices of series AVL-Advanced GigaBit or AVL IndiMicro, but without information about direction of rotation.

TECHNICAL DATA

Type	Optical encoder 366C
Preferred acquisition	AVL X-ion™ (Gigabit possible)
Detection rotation direction	Yes, absolute angle available
Speed range	0rpm to 20,000rpm
Vibration resistance	500 • 9.81 m/s ² (500g)
Short term load	1,000g
Temperature range (mechanics)	-40°C to 120°C
Life time	10 ⁷ revs at 500g load
Signal type	LVDS, RS422
Compatibility mechanically	365C mounting situation, flange and supporting arm identical

FOR FURTHER INFORMATION PLEASE CONTACT:

AVL List GmbH, Hans-List-Platz 1, 8020 Graz, Austria
Phone: +43 316 787-0, fax: +43 316 787-400, email: info@avl.com, www.avl.com

July 2018, Classification Public

Rugged, General Purpose Transducer

Common Specifications for 100 mV, 0 to 5 Vdc, and 4 to 20 mA Outputs

- ✓ 1, 2 & 5 psi Low Pressure Ranges!
- ✓ All Stainless Steel Construction
- ✓ Gage or Absolute Pressure
- ✓ Rugged Solid State Design
- ✓ High Stability, Low Drift
- ✓ 0.25% Static Accuracy
- ✓ IP 65 Protection Class

OMEGA's PX309 Series models below 100 psi use a high-accuracy silicon sensor protected by an oil-filled stainless steel diaphragm. Units 100 psi and above use silicon strain gages molecularly bonded to the stainless steel diaphragm.

Common Specifications

Ranges: -15 to 50 psig, 0 to 1000 psia, 100 to 10,000 psig

Accuracy (Combined Linearity, Hysteresis and Repeatability): $\pm 0.25\%$ BSL, max

Minimum Resistance Between Transducer Body and Any Wire: 1M Ω @ 25 Vdc

Calibration: In vertical direction with fitting down

Pressure Cycles: 10 million, minimum

Pressure Overload: -15 to 50 psig and 0 to 1000 psia: 3 times rated pressure or 20 psi whichever is greater, 100 to 10,000 psig: 2 times rated pressure

Burst Pressure: -15 to 50 psig and 0 to 1000 psia: 4 times rated pressure or 25 psi whichever is greater, 100 to 10,000 psig: 5 times rated pressure

Long Term Stability (1 Year): $\pm 0.25\%$ of FS, typical

Operating Temperature: -40 to 85°C (-40 to 185°F)

Pressure Port: 1/4-18 MNPT

Pressure Port Material: -15 to 50 psig and 0 to 1000 psia: 316 SS, 100 to 10,000 psig: 17-4 PH SS

Bandwidth: DC to 1 kHz (typical)

CE: Compliant

Shock: 50 g, 11 ms half-sine

Vibration: ± 20 g

Response Time: <1 millisecond

Weight:

PX309: 154 g (5.4 oz),

PX319/329/359: 100 g (3.5 oz)

IP Rating: IP65

RoHS: Compliant

Order a snubber to protect your pressure transducer!

PS-4G, shown actual size.
Snubbers protect sensors from fluid hammers/spikes.

mV Output Wiring			
Wiring	Cable	M12 and mini DIN	Twist-Lock
Excitation (+)	Red	Pin 1	Pin A
Output (+)	White	Pin 3	Pin C
Output (-)	Green	Pin 4	Pin D
Excitation (-)	Black	Pin 2	Pin B
Spare	—	—	Pin E
Vent	—	—	Pin F

5 Vdc Output Wiring			
Wiring	Cable	M12 and mini DIN	Twist-Lock
Excitation (+)	Red	Pin 1	Pin A
Excitation (-)	Black	Pin 2	Pin B
Output (+)	White	Pin 3	Pin C
N/C†	—	Pin 4	Pin D
Spare	—	—	Pin E
Vent	—	—	Pin F

mA Output Wiring			
Wiring	Cable	M12 and mini DIN	Twist-Lock
Supply (+)	Red	Pin 1	Pin A
Supply (-)	Black	Pin 2	Pin B
N/C†	—	Pin 3	Pin C
N/C†	—	Pin 4	Pin D
Spare	—	—	Pin E
Vent	—	—	Pin F

† N/C: Do not connect any wires to this pin.

Dimensions: mm (inch)
 ϕ = diameter

OMEGA ENGINEERING INC.

PRESSURE TRANSDUCER FINAL CALIBRATION

0.00 - 100.00 PSIA
Excitation 28.000 Vdc

Job: WHS0021436 Serial: 110417I509
Model: PX359-100A5V Tested By: JANE
Date: 2/28/2018 Temperature Range: -20 to +85 C
Calibrated: 0.00 - 100.00 PSIA Specfile: PX359-5V+=15-=50G-=300A

Pressure PSIA	Unit Data Vdc
0.00	0.007
50.00	2.507
100.00	5.003
50.00	2.507
0.00	0.007

Balance 0.007 Vdc
Sensitivity 4.996 Vdc

ELECTRICAL LEAKAGE: PASS
PRESSURE CONNECTION/FITTING: 1/4-18 NPT Male
ELECTRICAL WIRING/CONNECTOR: PIN 1 = + EXCITATION
PIN 2 = - EXCITATION
PIN 3 = + SIGNAL
PIN 4 = N/C

This Calibration was performed using Instruments and Standards that are traceable to the United States National Institute of Standards Technology.

S/N	Description	Range	Reference	Cal Cert
41000FUN	AUTO Mensor 1200	0 - 100.00 PSIA	C-3051	C-3051
US37008245	HP 34970A DMM	Unit Under Test	C-2491	N/A

Q.A. Representative : *Jane Porter*

Date: 2/28/2018

This transducer is tested to & meets published specifications. After final calibration our products are stored in a controlled stock room & considered in bonded storage. Depending on environment & severity of use factory calibration is recommended every one to three years after initial service installation date.

Omega Engineering Inc., One Omega Drive, Stamford, CT 06907
http://www.omega.com email: info@omega.com phone (800) 826-6342

HIGH TEMPERATURE HIGH BANDWIDTH AMPLIFIED GAS TURBINE IS® PRESSURE TRANSDUCER

ETL-GTS-X-190 (M) SERIES

- Miniature, Robust Construction
- High Temperature
- Acceleration & Vibration Insensitive
- High Bandwidth Amplifier
- Patented Leadless Technology VIS®
- Separate DC and AC Outputs
- Dynamic and Static Capability
- Excellent Long Term Stability
- Coupled with High Accuracy

The ETL-GTS Series are high temperature, extremely rugged pressure transducers which are ideal for the measurement of instabilities and steady pressures in gas turbine combustors and compressors. They use a patented acceleration insensitive leadless sensing element that is more than 1000 times less sensitive to vibration than other Kulite transducers. The front of the transducers can withstand temperatures of 1000°F (538°C)* while the in-line amplifier can be placed in a cooler area. The amplifier is compatible with both existing and new data acquisition systems with a measurement bandwidth of up to 150KHz, specified by Option X.

CONSULT FACTORY FOR SPECS. ON SEALED GAGE
*NO SCREEN FOR 50 KHz (AND ABOVE) BANDWIDTH

CABLE
PIN 1 VIN ROD
PIN 2 RTD BLK
PIN 3 OUT (DC/AC) GREEN
PIN 4 OUT (AC) WHITE

PIN FUNCTIONS		P/N	T"
PIN	FUNCTION		
1	VIN	190	10-32 UNF-2A
2	RTD	190M	M 5 x .8
3	OUT (DC/AC)		
4	OUT (AC)		
5	SPARE		
6	SPARE		
7	SPARE

INPUT	3.5	7	14	21	35	70
Pressure Range	50	100	200	300	500	1000 PSI
Operational Mode	Absolute, Sealed Gage					
Over Pressure	2 Times Rated Pressure					
Burst Pressure	3 Times Rated Pressure					
Pressure Media	All Nonconductive, Noncorrosive Liquids or Gases (Most Conductive Liquids and Gases - Please Consult Factory)					
ELECTRICAL PERFORMANCE						
Rated Electrical Excitation	12 ± 4VDC or 28 ± 4VDC					
Maximum Electrical Current	25 mA					
Output Impedance	200 Ohms (Typ.)					
Output Range	0.5V to 5V ± 3% (2 Outputs)					
Bandwidth (-3dB)	Output 1: DC to 5kHz (Option A), 50kHz* (Option B) or 150kHz* (Option C) Output 2: AC 10Hz to 5kHz (Option A), 50kHz* (Option B) or 150kHz* (Option C) (X10 Additional gain)					
Output Filter	Low Pass Filter Available per Customer Specification					
Residual Unbalance	500mV ± 1%					
OUTPUT						
Resolution	Infinitesimal					
Natural Frequency (KHz) (Typ.)	Greater Than 1000 KHz					
Insulation Resistance	100 Megohm Min. @ 50 VDC					
ENVIRONMENTAL						
Operating Temperature Range	-65°F to +1000°F* (-55°C to +538°C) (Front End) -65°F to +257°F (-55°C to +125°C) (Connector + Amplifier)					
Compensated Temperature Range	+80°F to +850°F (+25°C to +454°C)					
Thermal Zero Shift	± 1.5% FS/100°F (Typ.)					
Thermal Sensitivity Shift	± 1.5% /100°F (Typ.)					
Linear Vibration	50g Peak, Sine 10 to 2000 Hz					
Humidity	100% Relative Humidity					
Mechanical Shock	100g half Sine Wave 11 msec. Duration					
PHYSICAL						
Electrical Connection	Burklin 70F 8251 Connector (Mating Connector Supplied)					
Weight	10 Grams (Nom.) Excluding Cable and Connector					
Pressure Sensing Principle	Fully Active Four Arm Wheatstone Bridge Dielectrically Isolated Silicon on Silicon Patented Leadless Technology					
Mounting Torque	15 Inch-Pounds (Max.) 1.7 N-m					

* Limited life above 850°F (454°C) dependent on operating conditions.

Note: Custom pressure ranges, accuracies, amplifiers and mechanical configurations available. Dimensions are in inches. Dimensions in parenthesis are in millimeters.

Continuous development and refinement of our products may result in specification changes without notice - all dimensions nominal. (L)

KULITE SEMICONDUCTOR PRODUCTS, INC. • One Willow Tree Road • Leonia, New Jersey 07605 • Tel: 201 461-0900 • Fax: 201 461-0990 • <http://www.kulite.com>

KULITE SEMICONDUCTOR PRODUCTS, INC.
 One Willow Tree Road • Leonia, New Jersey 07605
 Tel: 201.461.0900 • Fax: 201.461.0990 • www.kulite.com

CALIBRATION CERTIFICATE

Model No: ETL-GTS-B-190-100A

Serial No.: 8457-9-492

Customer: FORD MOTOR COMPANY

P.O.: AGC PO18000209

STANDARD ELECTRICAL CONNECTIONS:

N/A + Input N/A + Output
 N/A - Input N/A - Output

SPECIAL CONNECTIONS: X

SEE CONNECTIONS
 AT LEFT.

TEST CONDITIONS:

Rated Pressure: 100 PSIA

Operational Mode: ABSOLUTE

Maximum Pressure: 200 PSIA

Maximum Reference Pressure: N.A.

Tested At: 12.0 VDC Excitation

Maximum Excitation: 36.0 VDC

CALIBRATION

Sensitivity: 45.561 mV/PSI A

Zero Balance Output: 500 mV +/- 50 mV

Compensated Temperature Range: 25 DEG. C TO 300 DEG. C

Output Impedance: N/A Ohms

Input Impedance: N/A Ohms

B SCREEN / CRUSH RING SUPPLIED

PIN 1 :VIN

PIN 2 :RTN

PIN 3 :OUT(DC/AC)

PIN 4 :OUT(AC)

PIN 5 :SHIELD

PIN 6,7:SPARE

BANDWIDTH 500 kHz

REMARKS:

QUALITY ASSURANCE: calibration traceable to NIST

Pressure Source Id#: PG245

Model#: CPC 6000

Tested By: G. SAKAI

Inspected By:

Date: 3/19/2018

Date: MAR 19 2018

The calibration of Kulite Semiconductor Products, Inc.

Instrumentation is in conformance with ISO 10012

HIGH TEMPERATURE HIGH BANDWIDTH AMPLIFIED GAS TURBINE PRESSURE TRANSDUCER ETL-GTS-X-312 (M) SERIES

- Miniature, Robust Construction
- High Bandwidth Amplifier
- High Temperature
- Separate DC and AC Outputs
- Acceleration & Vibration Insensitive
- Dynamic and Static Capability
- Patented Leadless Technology VIS®

The ETL-GTS Series are high temperature, extremely rugged pressure transducers which are ideal for the measurement of instabilities and steady pressures in gas turbine combustors and compressors. They use a patented acceleration insensitive leadless sensing element that is more than 1000 times less sensitive to vibration than other Kulite transducers. The front of the transducers can withstand temperatures of 932°F (500°C)* while the in-line amplifier can be placed in a cooler area. The amplifier is compatible with both existing and new data acquisition systems with a measurement bandwidth of up to 150KHz, specified by Option *.

Pressure Range	3.5 50	7 100	14 200	21 300	35 500	70 BAR 1000 PSI
Operational Mode	Absolute, Sealed Gage					
Over Pressure	2 Times Rated Pressure to 1000 PSI (70 BAR) 1.5 Times Rated Pressure Above 1000 PSI to a Max. of 6000 PSI (415 BAR)					
Burst Pressure	3 Times Rated Pressure					
Pressure Media	All Nonconductive, Noncorrosive Liquids or Gases (Most Conductive Liquids and Gases - Please Consult Factory)					
Maximum Electrical Current	25 mA					
Rated Electrical Excitation	12 - 36 VDC					
Output Range	Output 1: 0.5V to 5V ± 3% (DC/AC) Output 2: 0.5V to 9.5V ± 3% (AC)					
Output Impedance	200 Ohms (Typ.)					
Amplifier Bandwidth (-3dB) Only	Output 1: DC to 5kHz (Option A), 50kHz* (Option B) or 150kHz* (Option C) Output 2: AC 10Hz to 5kHz (Option A), 50kHz* (Option B) or 150kHz* (Option C) (X10 Additional gain)					
Output Filter	Low Pass Filter Available per Customer Specification					
Residual Unbalance	Output 1: 500mV ± 50mV (DC/AC) Output 2: 5.0V ± 0.5V (AC)					
Resolution	Infinitesimal					
Natural Frequency of Sensor Without Screen (KHz) (Typ.)	Greater Than 300 KHz					
Insulation Resistance	100 Megohm Min. @ 50 VDC					
Operating Temperature Range	-65°F to +932°F** (-55°C to +500°C) (Front End) -40°F to +185°F (-40°C to +85°C) (Connector and Amplifier) -65°F to 662°F (-55°C to 350°C) (Cable)					
Compensated Temperature Range	+80°F to +572°F (+25°C to +300°C)					
Thermal Zero Shift	± 1.5% FS/100°F (Typ.)					
Thermal Sensitivity Shift	± 1.5% /100°F (Typ.)					
Linear Vibration	10-2,000 Hz Sine, 100g (Max.)					
Mechanical Shock	20g half Sine Wave 11 msec. Duration					
Electrical Connection	Burklin 70F 8251 Connector (Mating Connector Supplied Burklin 70F 8551)					
Weight	10 Grams (Nom.) Excluding Cable and Connector					
Pressure Sensing Principle	Fully Active Four Arm Wheatstone Bridge Dielectrically Isolated Silicon on Silicon Patented Leadless Technology					
Mounting Torque	75 Inch-Pounds (Max.) 8.5 Nm					

** Limited life above 850°F (454°C) dependent on operating conditions.
Note: Custom pressure ranges, accuracies and mechanical configurations available. Dimensions are in inches. Dimensions in parenthesis are in millimeters. All dimensions nominal. (P)
Continuous development and refinement of our products may result in specification changes without notice. Copyright © 2014 Kulite Semiconductor Products, Inc. All Rights Reserved.
Kulite miniature pressure transducers are intended for use in test and research and development programs and are not necessarily designed to be used in production applications. For products designed to be used in production programs, please consult the factory.

KULITE SEMICONDUCTOR PRODUCTS, INC. • One Willow Tree Road • Leonia, New Jersey 07605 • Tel: 201 461-0900 • Fax: 201 461-0990 • <http://www.kulite.com>

KULITE SEMICONDUCTOR PRODUCTS, INC.
 One Willow Tree Road • Leonia, New Jersey 07605
 Tel: 201.461.0900 • Fax: 201.461.0990 • www.kulite.com

CALIBRATION CERTIFICATE

Model No: ETL-GTS-C-312M-50A

Serial No.: 8195-4-71

Customer: FORD MOTOR COMPANY

P.O.: AGCPO18000209

STANDARD ELECTRICAL CONNECTIONS:

N/A	+ Input	N/A	+ Output
N/A	- Input	N/A	- Output

SPECIAL CONNECTIONS: X

SEE CONNECTIONS
 AT LEFT.

TEST CONDITIONS:

Rated Pressure: 50 PSIA

Operational Mode: ABSOLUTE

Maximum Pressure: 100 PSIA

Maximum Reference Pressure: N.A.

Tested At: 12.0 VDC Excitation

Maximum Excitation: 36.0 VDC

CALIBRATION

Sensitivity: 89.538 mV/PSI A

Zero Balance Output: 500 mV +/- 50 mV

Compensated Temperature Range: 25 DEG. C TO 300 DEG. C

Output Impedance: N/A Ohms

Input Impedance: N/A Ohms

B SCREEN / CRUSH RING SUPPLIED

PIN 1 : VIN

PIN 2 : RTN

PIN 3 : OUT (DC/AC)

PIN 4 : OUT (AC)

PIN 5 : SHIELD

PIN 6,7: SPARE

BANDWIDTH 150kHz

REMARKS:

QUALITY ASSURANCE: calibration traceable to NIST

Pressure Source Id#: PG224

Model#: CPC6000

Tested By: M. DeBellis

Inspected By:

Date: 3/19/2018

Date: MAR 19 2018

The calibration of Kulite Semiconductor Products, Inc.

Instrumentation is in conformance with ISO 10012

WATSON ENGINEERING INC.
TRANSDUCER CALIBRATION

Cal Date 6/5/2015
Re-Cal Date 6/5/2016

CHANNEL# 24 VEH # 061 W 039 SER# 110333

VOLTAGE	PRESSURE PSI
1.0019	0
1.4495	10
1.9964	20
2.4946	30
2.9925	40
3.4908	50
3.9892	60
4.4887	70
4.9878	80
5.4379	90
5.987	100

CAMPBELL
MULTIPLIER 0.200851007

CAMPBELL
OFFSET -19.93372469

R-CON 2X	
2.0038	0
2.899	10
3.9928	20
4.9892	30
5.985	40
6.9816	50
7.9784	60
8.9774	70
9.9756	80
10.8758	90
11.974	100

R-CON 2/3	
0.667933	0
0.966332	10
1.330932	20
1.663065	30
1.994998	40
2.327198	50
2.659464	60
2.992464	70
3.325197	80
3.625263	90
3.991329	100

CONVERSION CONSTANTS		
PSI TO	IN H2O	PSI X 27.677
PSI TO	IN HG	PSI X 2.036
IN HG TO	IN H2O	IN HG X 13.596
IN HG TO	PSI	IN HG X 0.4912
IN H2O TO	IN HG	IN H2O X 0.07355
IN H2O TO	PSI	IN H2O X 0.03613

Ser.#110333

Watson Engineering, Inc.
Transducer Calibration

Cal Date 6/5/2015
Re-Cal Date 6/5/2016

CHANNEL# 22 VEH # 061 W 039 SER# 083314

VOLTAGE	PRESSURE InHG
3.903	0
4.3972	5
4.8906	10
5.381	15
5.874	20
6.364	25
6.855	30
6.935	31

CAMPBELL
MULTIPLIER 0.101955074

CAMPBELL
OFFSET -39.8396989

R-CON 2X		R-CON 2/3		CONVERSION CONSTANTS		
7.806	0	2.601997	0	PSI TO	IN H2O	PSI X 27.677
8.7944	5	2.931464	5	PSI TO	IN HG	PSI X 2.036
9.7812	10	3.260397	10	IN HG TO	IN H2O	IN HG X 13.596
10.762	15	3.58733	15	IN HG TO	PSI	IN HG X 0.4912
11.748	20	3.915996	20	IN H2O TO	IN HG	IN H2O X 0.07355
12.728	25	4.242662	25	IN H2O TO	PSI	IN H2O X 0.03613
13.71	30	4.569995	30			
13.87	31	4.623329	31			
2	1	0.666666	1			
2	1	0.666666	1			
2	1	0.666666	1			

WATSON ENGINEERING INC.
TRANSDUCER CALIBRATION

Cal Date 5/6/2015
Re-Cal Date 5/6/2016

CHANNEL# 21 VEH # 061 W 039 SER# 2004489

VOLTAGE	PRESSURE PSI
0.9685	0
1.4727	5
1.9759	10
2.4788	15
2.9826	20
3.4857	25
3.9894	30
4.4929	35
4.9964	40
5.499	45
6.002	50

CAMPBELL
MULTIPLIER 0.099325426

CAMPBELL
OFFSET -9.623729829

R-CON 2X

1.937	0
2.9454	5
3.9518	10
4.9576	15
5.9652	20
6.9714	25
7.9788	30
8.9858	35
9.9928	40
10.998	45
12.004	50

R-CON 2/3

0.645666	0
0.981799	5
1.317265	10
1.652532	15
1.988398	20
2.323798	25
2.659597	30
2.995264	35
3.33093	40
3.665996	45
4.001329	50

CONVERSION CONSTANTS

PSI TO	IN H2O	PSI X 27.677
PSI TO	IN HG	PSI X 2.036
IN HG TO	IN H2O	IN HG X 13.596
IN HG TO	PSI	IN HG X 0.4912
IN H2O TO	IN HG	IN H2O X 0.07355
IN H2O TO	PSI	IN H2O X 0.03613

Ser.#02004489

WATSON ENGINEERING INC.
TRANSDUCER CALIBRATION

Cal Date 6/5/2015
Re-Cal Date 6/5/2016

CHANNEL# 19 VEH # 061 W 039 SER# 33670

VOLTAGE	PRESSURE PSI
1	0
1.025	10
1.0502	20
1.0751	30
1.1002	40
1.1253	50
1.1504	60
1.1754	70
1.2005	80
1.2258	90
1.251	100

CAMPBELL
MULTIPLIER 3.987718017

CAMPBELL
OFFSET -398.7492733

R-CON 2X

2	0
2.05	10
2.1004	20
2.1502	30
2.2004	40
2.2506	50
2.3008	60
2.3508	70
2.401	80
2.4516	90
2.502	100

R-CON 2/3

0.666666	0
0.683333	10
0.700133	20
0.716733	30
0.733466	40
0.750199	50
0.766933	60
0.783599	70
0.800333	80
0.817199	90
0.833999	100

CONVERSION CONSTANTS

PSI TO	IN H2O	PSI X 27.677
PSI TO	IN HG	PSI X 2.036
IN HG TO	IN H2O	IN HG X 13.596
IN HG TO	PSI	IN HG X 0.4912
IN H2O TO	IN HG	IN H2O X 0.07355
IN H2O TO	PSI	IN H2O X 0.03613

Ser.#033670

WATSON ENGINEERING INC.
TRANSDUCER CALIBRATION

Cal Date 5/6/2015
Re-Cal Date 5/6/2016

CHANNEL# 18 VEH # 061 W 039 SER# 53751

VOLTAGE	PRESSURE PSI
1.7165	0
2.2199	10
2.7221	20
3.2235	30
3.7233	40
4.2225	50
4.7207	60
5.2175	70
5.714	80
6.208	90
6.702	100

CAMPBELL
MULTIPLIER 0.200362834

CAMPBELL
OFFSET -34.5200016

R-CON 2X	
3.433	0
4.4398	10
5.4442	20
6.447	30
7.4466	40
8.445	50
9.4414	60
10.435	70
11.428	80
12.416	90
13.404	100

R-CON 2/3	
1.144332	0
1.479932	10
1.814732	20
2.148998	30
2.482198	40
2.814997	50
3.14713	60
3.47833	70
3.80933	80
4.138663	90
4.467996	100

CONVERSION CONSTANTS		
PSI TO	IN H2O	PSI X 27.677
PSI TO	IN HG	PSI X 2.036
IN HG TO	IN H2O	IN HG X 13.596
IN HG TO	PSI	IN HG X 0.4912
IN H2O TO	IN HG	IN H2O X 0.07355
IN H2O TO	PSI	IN H2O X 0.03613

SER#053751

WATSON ENGINEERING INC.
TRANSDUCER CALIBRATION

Cal Date 6/5/2015
Re-Cal Date 6/5/2016

CHANNEL# 17 VEH # 061 W 039 SER# 87020

VOLTAGE	PRESSURE PSI
1.0293	0
1.5268	2.5
2.024	5
2.5235	7.5
3.0228	10
3.522	12.5
4.0212	15
4.5202	17.5
5.02	20
5.519	22.5
6.017	25

CAMPBELL
MULTIPLIER 0.050101398

CAMPBELL
OFFSET -5.147586441

R-CON 2X	
2.0586	0
3.0536	2.5
4.048	5
5.047	7.5
6.0456	10
7.044	12.5
8.0424	15
9.0404	17.5
10.04	20
11.038	22.5
12.034	25

R-CON 2/3	
0.686199	0
1.017866	2.5
1.349332	5
1.682332	7.5
2.015198	10
2.347998	12.5
2.680797	15
3.013464	17.5
3.346663	20
3.67933	22.5
4.011329	25

CONVERSION CONSTANTS		
PSI TO	IN H2O	PSI X 27.677
PSI TO	IN HG	PSI X 2.036
IN HG TO	IN H2O	IN HG X 13.596
IN HG TO	PSI	IN HG X 0.4912
IN H2O TO	IN HG	IN H2O X 0.07355
IN H2O TO	PSI	IN H2O X 0.03613

Appendix G: Backward Sequential Feature Selection Code

```
% BSFS algorithm for data learning model (here called BSFS_network)
% with input data "train.in" and output data "train.out"

N = size(train.in,2); % number of input variables
error = 50*ones(N,N) % create a n by n array to store the error
idx = 1:N;           % list to know which input features have been
                    % tried and removed

for i = 1:N
    for k = 1:N
        temp = idx;
        try
            temp(k) = [];
            error(idx(k),i) = BSFS_network(train.in(:,temp),...
                                           train.out); % run the objective func
            idx;
        end
        imagesc(error) % plot the progress
        drawnow        % draw the plot (required to draw mid execution)
    end
    idx
    [val, place] = min(error(:,i)); % find the variable to remove

    idx(find(idx == place)) = [];
end

% the following code plots the above algorithm result
error(find(error == 50)) = -50;
error(12, 26) = -50;
imagesc(error)
% xticks(1:length(error))
% xticklabels(names)
yticks(1:length(error))
yticklabels(names(idx1))

ax=gca
mymap=colormap(ax);
mymap(1, :) = [1 1 1]; % paint removed features white
colormap(mymap);
colorbar
caxis([0 216])
h = colorbar;
set(h, 'ylim', [15 200])
caxis([10 100])

set(get(h, 'label'), 'string', 'Model Error (kPa)');
xlabel('iteration')
ylabel('Model Feature')
```

The following code is the objective function

"BSFS_network.m"

```
function rms_error= BSFS_network(input, output)
    reg.in.mean = mean(input)
    reg.out.mean = mean(output)
    reg.in.std = std(input)
    reg.out.std = std(output)
    input = (input - reg.in.mean)./reg.in.std;
    output = (output- reg.out.mean)./reg.out.std;

    %%
    %     featureDimension = size(input, 2)
    %     numResponses = 1;
    %     layers = [ ...
    %         sequenceInputLayer(featureDimension)
    %         fullyConnectedLayer(120)
    %         reluLayer
    %         fullyConnectedLayer(120)
    %         reluLayer
    %         fullyConnectedLayer(120)
    %         fullyConnectedLayer(numResponses)
    %         regressionLayer];
    %
    %     options = trainingOptions('adam', ...
    %         'MaxEpochs',250, ...
    %         'MiniBatchSize', 250, ...
    %         'InitialLearnRate',0.003, ...
    %         'LearnRateSchedule','piecewise', ...
    %         'LearnRateDropFactor',0.5, ...
    %         'LearnRateDropPeriod',80, ...
    %         'GradientThreshold',1, ...
    %         'Shuffle','never', ...
    %         'Plots','training-progress');
    %     std_net = trainNetwork(input', output',layers,options);

    featureDimension = size(input, 2)
    numResponses = 1;
    layers = [ ...
        sequenceInputLayer(featureDimension)
    %         fullyConnectedLayer(120)
    %         reluLayer
        fullyConnectedLayer(40)
        reluLayer
        fullyConnectedLayer(40)
        reluLayer
        fullyConnectedLayer(numResponses)
        regressionLayer];

    options = trainingOptions('adam', ...
        'MaxEpochs',400, ...
        'MiniBatchSize', 250, ...
        'InitialLearnRate',0.003, ...
        'LearnRateSchedule','piecewise', ...
        'LearnRateDropFactor',0.75, ...
```



```

        'LearnRateDropPeriod',80, ...
        'GradientThreshold',1, ...
        'Shuffle','never', ...
        'Plots','none');
%     'Plots','training-progress');
std_net = trainNetwork(input', output',layers,options);
%% plot the training data - WCX
ypred = predict(std_net, input');

    error = ((ypred.*reg.out.std + reg.out.mean)-(output.*reg.out.std +
reg.out.mean)') ;
    rms_error = rms(error);
end

```

Appendix H: Optimizable Windowing IMEP estimation system

The following code is the parameterized IMEP estimation system that takes in 4 parameters, exhaust window length and phase and crank window length and phase. The function then returns processed data ready to be used in a neural network. The “BSFS_network.m” function takes this data as an input

```
% Only compiles data for cylinder 1.
function [input output names] =
cycle_wise_processing_fcn_opt(AVL_iFile_path, ATI_file_path )
    x = [0 60 0 180] % [exhaust phasing | crank phasing | exhaust
window length | crank window length]
    ATI = load(ATI_file_path);
    AVL = AVLiFileClass(AVL_iFile_path);

    ATI_spd = ATI.ENGINE_SPEED.signals.values; % find ATI engine speed
    AVL_spd = lowpass(((1./AVL.dTa.Values).*(1E6).*(60)./360), .1); %
Calculate A VL engine speed % replace this with premade filter
    AVL_spd = downsample(AVL_spd, 360); % Downsample AVL engine speed

    [acor,lag] = xcorr(AVL_spd, ATI_spd); % find the lag btwn the two
    [~,I] = max(abs(acor));
    lagDiff = lag(I); % the lag btwn the two in cycles

    ATI_data.engine_speed =
ATI.ENGINE_SPEED.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
    ATI_data.MAP =
ATI.MAP.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
    ATI_data.fuel =
ATI.MF_INJ_0_.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
    ATI_data.SA =
ATI.SAFTOT.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
    ATI_data.tps =
ATI.TPP_THETA_FIN.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
    ATI_data.VVT_exh =
ATI.VCT_ANGLE_EXH_0_.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
```

```

ATI_data.VVT_int =
ATI.VCT_ANGLE_INT_0_.signals.values(abs(lagDiff):length(AVL_spd)+abs(la
gDiff)-1);
ATI_data.WG =
ATI.WGC_DC.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-1);
ATI_data.ECT =
ATI.ECT_ENG.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-
1);
ATI_data.IAT =
ATI.IAT11_ENG.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDiff)-
1);
ATI_data.misgen_flag =
ATI.MIS_GEN_CUT_FLG.signals.values(abs(lagDiff):length(AVL_spd)+abs(lag
Diff)-1);
ATI_data.vspd = -
ATI.rVcVscCc_v_CcSetIntErr.signals.values(abs(lagDiff):length(AVL_spd)+
abs(lagDiff)-1);
% ATI_data.rough_road_flag = -
ATI.RRD_NDRV_FLG.signals.values(abs(lagDiff):length(AVL_spd)+abs(lagDif
f)-1);

%% Reshape exhaust data
cyl3 = 4+2:4:length(AVL.IMEP1)*4;
cyl4 = 4+3:4:length(AVL.IMEP1)*4;
cyl2 = 4+1:4:length(AVL.IMEP1)*4;
cyl1 = 4:4:length(AVL.IMEP1)*4;

% reshape kulite data
EA = x(1); % set the offset for the exhaust data (GA will optimize
this)
if EA > 0
    AVL_data.pexh_k = [zeros(abs(EA), 1) ;
    AVL.PEXH_K2.Values(1:end-abs(EA))];
else
    AVL_data.pexh_k = [AVL.PEXH_K2.Values(abs(EA)+1:end) ;
    zeros(abs(EA), 1)];
end

    AVL_data.pexh_k = reshape(AVL_data.pexh_k, length(AVL.PEXH_K2.CA),
length(AVL.PEXH_K2.Values)/(length(AVL.PEXH_K2.CA)));
    base_exh_window_start = 360;
    base_exh_window_end = 540 + x(3);
    AVL_data.exh.pexh_k1 =
    AVL_data.pexh_k(base_exh_window_start:base_exh_window_end,:);
%     plot(AVL_data.exh.pexh_k1(:,1001))
%     title('Base Exhaust Window Phasing')
%     xlabel('Crank Angle Degrees')/
%     ylabel('exhaust pressure (kPa)')
%     xticks(0:45:180)
%% Offset and reshape omega sensor data
EA = x(1); % set the offset for the exhaust data (GA will optimize
this)
if EA > 0

```

```

        AVL_data.pexh_o = [zeros(abs(EA), 1) ;
AVL.PEXH_O1.Values(1:end-abs(EA))];
    else
        AVL_data.pexh_o = [AVL.PEXH_O1.Values(abs(EA)+1:end) ;
zeros(abs(EA), 1)];
    end

    AVL_data.pexh_o = reshape(AVL_data.pexh_o,
length(AVL.PEXH_O1.CA)/4,
length(AVL.PEXH_O1.Values)/(length(AVL.PEXH_O1.CA)/4));
    AVL_data.exh.pexh_o2 = AVL_data.pexh_o(:,cyl1);
    AVL_data.exh.pexh_o3 = AVL_data.pexh_o(:,cyl2);
    AVL_data.exh.pexh_o4 = AVL_data.pexh_o(:,cyl3);
    AVL_data.exh.pexh_o1 = AVL_data.pexh_o(:,cyl4-4);
    %%

    AVL_data.exh.mean_1 = AVL_data.exh.pexh_k1(1,:);
    AVL_data.exh.pexh_k1 = AVL_data.exh.pexh_k1 - AVL_data.exh.mean_1;
    AVL_data.exh.mean_1 = mean(AVL_data.exh.pexh_o1);
    mis1 = find(ATI_data.misgen_flag(cyl1) ==1);
    mis2 = find(ATI_data.misgen_flag(cyl2) ==1);
    mis3 = find(ATI_data.misgen_flag(cyl3) ==1);
    mis4 = find(ATI_data.misgen_flag(cyl4) ==1);
    %% advance crank signal
    crank_advance = x(2); % set the offset for the exhaust data (GA
will optimize this)

    dta = AVL.dTa.Values;
    if crank_advance > 0
        dta = [dta(1:abs(crank_advance)) ; dta(1:end-
abs(crank_advance))];
    else
        dta = [dta(abs(crank_advance)+1:end) ; dta(end-
abs(crank_advance)+1:end)];
    end
    %%
    clc
    sig = dta;
    sig_down = downsample(dta, 36);
    CA = AVL.dTa.CA;
    CA_down = downsample(AVL.dTa.CA, 36);
    sig = reshape(sig, length(CA), length(sig)/length(CA));
    sig_down = reshape(sig_down, length(CA_down),
length(sig_down)/length(CA_down));

    dta_down = sig_down(:);
    passband = 8/20;
    stopband = 19/20;

    lpFilt = designfilt('lowpassfir','PassbandFrequency',passband, ...
        'StopbandFrequency',stopband,'PassbandRipple',5, ...
        'StopbandAttenuation',10,'DesignMethod','kaiserwin');
    dta = filtfilt(lpFilt, dta);
    %    clf

```

```

%      k = 5000
% %      plot(dta(k:k+80))
%      hold on
%      plot(dta_down(k:k+80))
%      legend('dta', 'down')
CA = AVL.dTa.CA;
vel = (1./dta)*166667;% sig = reshape(sig, length(CA),
length(sig)/length(CA));
%
%      subplot(2,1,1)
%      plot(vel(1:300))
%      subplot(2,1,2)
%      plot(vel(end-300:end))
%%
% find seconds per degree
%      dta = AVL.dTa.Values;
%      dt = dta./1E6; % seconds per deg
%      dt = dt*.5;      % duration of sample
dt = 1;
% find the acceleration
clf
accel = diff(vel);
accel = [accel(1) accel']'./dt;

%      % reshape accel
%      accel_r = reshape(accel, length(CA)/4,
length(accel)/(length(CA)/4));
%      accel1 = accel_r(:,cyl4-4);
%      accel2 = accel_r(:,cyl1);
%      accel3 = accel_r(:,cyl2);
%      accel4 = accel_r(:,cyl3);
%      AVL_data.crank.cyl1 = accel1;
%      AVL_data.crank.cyl2 = accel2;
%      AVL_data.crank.cyl3 = accel3;
%      AVL_data.crank.cyl4 = accel4;

% reshape accel
%      accel_r = reshape(accel, length(CA_down),
length(accel)/(length(CA_down)));
%      accel_r = reshape(accel, length(CA), length(accel)/(length(CA)));
%      base_crank_window_start = 720/(2);
%      base_crank_window_end   = 1080/(36*2) + (x(4)/36)*2;
base_crank_window_start = 720;
base_crank_window_end   = 1080+ x(4)*2;
accel_r = accel_r(base_crank_window_start:base_crank_window_end,:);

accel1 = accel_r(:,:);
AVL_data.crank.cyl1 = accel1;
%      plot(AVL_data.crank.cyl1(:,end))
%      plot(AVL_data.crank.cyl1(:,1))
%      title('Base Crank Window Phasing')
%      xlabel('Crank Angle Degrees')
%      ylabel('Crank Acceleration (normalized)')
%      xticks(0:45:180)

```

```

%% process ion A data
sig = AVL.IONA.Values;
CA = AVL.IONA.CA;
sig = reshape(sig, length(CA), length(sig)/length(CA));

SA = (ATI_data.SA(cyl1));
[~, idx] = min(abs(CA +SA'));

idx = idx-30;

sig = sig(idx:idx+359,:);

sig = sig-sig(1,:);

AVL_data.ion.A = sig;

%% process ion B data
sig = AVL.IONB.Values;
CA = AVL.IONA.CA;
k = 360;
sig = [sig(k+1:end) ; sig(end-k+1:end)];
sig = reshape(sig, length(CA), length(sig)/length(CA));

SA = (ATI_data.SA(cyl1));
[~, idx] = min(abs(CA +SA'));

idx = idx-30;
sig = sig(idx:idx+359,:);
sig = sig-sig(1,:);
AVL_data.ion.B = sig;

%% process ion C data
sig = AVL.IONC.Values;
CA = AVL.IONA.CA;
k = 720;
sig = [sig(k+1:end) ; sig(end-k+1:end)];

sig = reshape(sig, length(CA), length(sig)/length(CA));
size(sig);
SA = (ATI_data.SA(cyl1));

[~, idx] = min(abs(CA +SA'));

idx = idx-30;
sig = sig(idx:idx+359,:);
sig = sig-sig(1,:);

```

```

    AVL_data.ion.C = sig;
%     size(AVL_data.ion.C)
%     plot(AVL_data.ion.C(:,1))
    %% ION sensor for cyl D is not recording properly, only make
    predictions on cyl1
    % analyze ION data for synchronization
    %% FFT all the data
    % exhaust
    sig = [AVL_data.exh.pexh_k1 AVL_data.exh.pexh_k1(:,end)];
    sig = fft(sig);
    sig = sig(1:size(sig, 1)/2,:)/(2*size(sig, 1));
    sig_amp = abs(sig);
    sig_phase = angle(sig);
    AVL_data.exh.fft.cyl1.amp = sig_amp;
    AVL_data.exh.fft.cyl1.phase = sig_phase;

% Ion
    sig = [AVL_data.ion.A AVL_data.ion.A(:,end)];
    sig = fft(sig);
    sig = sig(1:size(sig, 1)/2,:)/(2*size(sig, 1));
    sig_amp = abs(sig);
    sig_phase = angle(sig);
    AVL_data.ion.fft.cyl1.amp = sig_amp;
    AVL_data.ion.fft.cyl1.phase = sig_phase;

% crank
    sig = [AVL_data.crank.cyl1 AVL_data.crank.cyl1(:,end)];
    f_s = 20;
    f_n = 10;
    R = 270/360;
    f = 0:1/R:f_n;
    size(f);
    sig = fft(sig);
    sig = sig(1:size(sig, 1)/2,:)/(2*size(sig, 1));
    sig_amp = abs(sig);
    sig_phase = angle(sig);
    AVL_data.crank.fft.cyl1.amp = sig_amp;
    AVL_data.crank.fft.cyl1.phase = sig_phase;
    %%
%     plot(sig_amp(:,10:100))
%     xticks(f)
%     xticklabels(string(0:7))
%     xlabel('Order With Respect to Crank Rotation')
%     ylabel('Crank Acceleration Amplitude (deg/s^2)')
    %%
%     k = 2027;
%     k = k-4:k+4;
%     clf
%     plot(AVL_data.crank.fft.cyl1.amp(1,k))
%     yyaxis right
%     plot(AVL.IMEP1(k))
%
    %% compile inputs

```

```

input = AVL_data.crank.fft.cyl1.amp(1:8,1:end-1); % first 10 orders
input = [input ; AVL_data.crank.fft.cyl1.phase(1:8,1:end-1)];

input = [input ; AVL_data.exh.fft.cyl1.amp(2:8,1:end-1)];
input = [input ; AVL_data.exh.fft.cyl1.phase(2:8,1:end-1)];
size(input);

input = [input ; AVL_data.ion.fft.cyl1.amp(1:8,1:end-1)];
input = [input ; AVL_data.ion.fft.cyl1.phase(1:8,1:end-1) ;
[AVL_data.exh.mean_1 AVL_data.exh.mean_1(end)]]';

input = [ATI_data.engine_speed(cyl1) ATI_data.MAP(cyl1)...
        ATI_data.fuel(cyl1) ATI_data.SA(cyl1)
ATI_data.tps(cyl1)...
        ATI_data.VVT_exh(cyl1) ATI_data.VVT_int(cyl1)
ATI_data.WG(cyl1)...
        ATI_data.ECT(cyl1) ATI_data.IAT(cyl1) input];
output = AVL.IMEP1;

%%
names = ["engine speed", "MAP", "fuel", "SA", "TPS", "VVT exh", "VVT
int", "WG", "ECT", "IAT",...
        "crank amp 1", "crank amp 2", "crank amp 3", "crank amp
4", "crank amp 5", "crank amp 6", "crank amp 7", "crank amp 8", ...
        "crank phase 1", "crank phase 2", "crank phase 3", "crank
phase 4", "crank phase 5", "crank phase 6", "crank phase 7", "crank
phase 8", ...
        "2nd order exhaust amplitude", "exhaust amp 3", "exhaust
amp 4", "exhaust amp 5", "exhaust amp 6", "exhaust amp 7", "exhaust amp
8", ...
        "exhaust phase 2", "exhaust phase 3", "exhaust phase 4",
"exhaust phase 5", "exhaust phase 6", "exhaust phase 7", "exhaust
phase 8", ...
        "ion amp 1", "ion amp 2", "ion amp 3", "ion amp 4", "ion
amp 5", "ion amp 6", "ion amp 7", "ion amp 8", ...
        "ion phase 1", "ion phase 2", "ion phase 3", "ion phase
4", "ion phase 5", "ion phase 6", "ion phase 7", "ion phase 8",
"exhaust_mean"];

end

```