# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| 03/15/2011 | Final | September 2009 - December 2010 |

**4. TITLE AND SUBTITLE**

A Dynamic Neural Network Approach to CBM

**5a. CONTRACT NUMBER**

W911NF-09-2-0036

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Dr. Margherita Zannini Phd
Dr. Lee Feldkamp PhD
Dr. Ken Marko PhD
Dr. John James PhD
Mr. Larry Maccani
Mr. Robert Wiebe

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

MIS2000 Global Defense Electronics, Inc.
21223 Hilltop St.
Southfield, MI 48033

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Army Research Lab
2800 Powder Mill Road
Adelphi, MD 20783

**10. SPONSOR/MONITOR'S ACRONYM(S)**

ARL

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited

**13. SUPPLEMENTARY NOTES**

(3 parts) Final Report, Appendix & Layout

**14. ABSTRACT**

This project was the continuation of an initial project regarding the use of Neural Networks as they related to Condition Based Maintenance of military vehicles.
The overall objective of the project was to investigate the use of prognostic algorithms and neural networks as they pertain to powertrain systems by creating relevant faults in the engine operating conditions related to fluid temperature, pressure, and flow, which produce performance loss and impact the vehicle health. Our investigation was carried out on the military version of the Caterpillar C7, 7.2L 6 cylinder engine mounted on an Eddy current dynamometer at the dynamometer facility of the Mobility Group at the Detroit Arsenal. The engine is a diesel in-line with a waste-gated turbocharger Faults were introduced by altering the normal response of some electromechanical components of the engine control system. Known malfunctions were generated in such a way that the faulty condition could be turned on and off without actually exchanging malfunctioning components.

**15. SUBJECT TERMS**

Neural Networks (NN), Condition Based Maintenance (CBM), Prognostic Algorithms

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Robert Wiebe |
| Unclassified | Unclassified | Unclassified | UU | 73 | 19b. TELEPHONE NUMBER *(include area code)* 248-353-0100 |

# FINAL REPORT

Cooperative Agreement Article

## W911-09-2-0036

A Dynamic Neural Network Approach to CBM


**Approved for public release; distribution is unlimited**

# Executive Summary

## Executive Summary

## Cooperative Agreement Article

## W911-09-2-0036

## A Dynamic Neural Network Approach to CBM

The concepts of Virtual Sensing, Condition Based Maintenance (CBM) and Prognostics have received significant attention for various applications due to potential benefits that include reduced cost by substituting expensive sensors with estimations, significantly reduced development time, greater time in service, and improved diagnostic and control properties. However, one problem with developing such technologies across this breadth of applications has been the need to develop special techniques for each problem domain, and perhaps for each individual problem.

This project was the continuation of an initial project that was done by MIS 2000 regarding the use of Neural Networks as they related to Condition Based Maintenance of military vehicles. In phase 1 of the project information was analyzed from two sources. The first source was Army Materiel Systems Analysis Activity (AMSAA) Data, this data was acquired from vehicles in routine operation in the field. The AMSAA Data was acquired by external data acquisition systems installed in vehicles used around a military base. The other information was obtained from an engine operated in a dynamometer cell being used for NATO durability test.

Phase 1 allowed for basically the training of the Neural Networks to see if they could accurately predict basic engine operation. Phase 2 of this project allowed for a dedicated dynamometer and engine that would allow for the induction of faults and structured testing.

The goal of this phase was to create relevant faults in the engine operating conditions related to fluid temperature, pressure, and flow, which produce performance loss and impact the vehicle health.

As mentioned earlier virtual sensors are also a part of CBM. The engine was also fitted with an encoder and a sensor on the flywheel housing so that the concept of virtual torque sensing could also be investigated during this phase of the project.

Faults were introduced (one at a time) by altering the normal response of some electromechanical components of the engine control system. Known malfunctions were generated in such a way that the faulty condition could be turned on and off without actually exchanging malfunctioning components. Only one component at the time was altered.

The sensor faults were accomplished by skewing the calibration of the temperature and pressure sensors.

Electrical connections between the Engine Control Unit (ECU) and the sensors were broken with a Break-out Box so that the sensor output could either be fed directly to the ECU with no changes or be altered before being returned to the ECU.

The engine was run at steady-states and selected speed/torque points ("Mini Maps") for periods of 1 to 3 minutes, as was also done in Phase 1. The perturbations were introduced continuously.

The main focus of this project was around the perturbations related to the fuel injection control pressure and turbo boost pressure. They were selected as likely candidates to affect the engine torque output by affecting either the fuel delivery system or the air charge with a potential effect in the engine torque output. Certain faults were seeded electronically by custom built circuits.

Data collection was coordinated between two systems. The first was the dyno cell data acquisition system which recorded at low rate but with a large number of signals from the temperature and pressure sensors with which the engine was instrumented and from the other instruments and controllers. The second system is a custom made system which acquired engine data broadcast on the communication bus (CAN bus) continuously so that the operating conditions perceived by the ECU could be compared to the actual engine behavior measured by laboratory instrumentation.

The set-up of the dynamometer is outlined in the final report in Section 2.

The data derived from the seeded faults experiments in this comprehensive testing facility constitute a rich set of engine conditions, under both normal and abnormal conditions, which can be used, after appropriate handling and reformatting, to develop and test Neural Networks models describing engine performance.

To simplify the networks development, the input data needs to be preprocessed as a time-aligned time series. Matlab was used as the program for processing the files. The processed Matlab files are designated with the "_genmod" suffix. These files were the sources for the training and testing sets and made the extraction process easy because the 2D data array, containing both CAN and Cell data and could be readily sliced using sorting and indexing criteria to combine test sections according to perturbation levels and reject data contaminated by unwanted experimental conditions.

Theoretically, if the training set is an overall faithful representation of all conditions met by the engine, any other set of data presented to the network as input would produce an output with equivalent fidelity.

The final report contains a greater explanation of the development and training of the Neural Networks. Below is an overview of the data used in NN development and testing.

Based on previous experience and the knowledge of the engine operating parameters, result in the selection of the parameters that were used to train the NN. The following data points were used for the Fuel Flow prediction:

- Engine Speed
- Load %
- Engine Oil Pressure
- Boost
- Injector Control Pressure
- Engine Coolant Temperature
- Intake Manifold Air Temperature
- Pedal %
- Desired Engine Speed
- Nominal Friction
- Load @ Speed

Virtual Torque Sensing (VTS) is a software alternative to detecting actual brake torque with a minimum investment in additional hardware (that is, not relying on new sensing technologies) at the expense of adding computational burden which can be either managed within the ECU or by another module interfacing with the ECU. The most promising VTS implementation is based on accurate engine speed measurements which are the input to a Neural Network model that calculates actual torque.

Section 6 contains additional details regarding virtual torque sensing.

This project was successful in showing that the Neural Networks are capable of predicting the operating characteristics of the engine.

A key element relates to the method of training dynamic neural networks so that they can attain their full theoretical capability and produce highly accurate models of complex dynamic systems. This approach is derived from combining a series of innovations developed by several groups over the past ten years and on improvements devised from the experience garnered in the application of these methods across a wide range of estimation, classification, virtual sensing and control applications.

MIS2000 believes that a model based reasoning approach, using dynamic neural networks to detect anomalies in system performance, will provide the Army with a sound solution for implementing CBM for military vehicles. A major benefit is the fast and efficient way of utilizing real time vehicle data to assess vehicle health. Since these algorithms are efficient, they do not necessitate extensive computing power and require a minimum hardware footprint to run.

The constraints of this approach are related to the data being fed into the model. The fact remains that good decisions cannot be made using bad or incomplete information. Therefore, an open systems approach will be used to allow for maximum integration of

additional sensor data as new technology becomes available. This information may be generated from virtual information i.e. Virtual Torque Sensors, or from additional external devices to augment real-time information acquired from the system via J-1939, Controller Area Network (CAN) bus or wireless interfaces.

# Outline

1. Introduction
2. Description of the Experimental Set-up
3. Description of the Experiments
4. Data Collection and Reduction
5. NN Analysis of Dynamometer Data (Models to Detect Faults)
6. NN Model to Estimate Torque (Virtual Torque Sensing)
7. Conclusions

Appendices (Appendix.doc)

A. Engine Specifications
B. Analog Signal Modifier Device
C. List of Acquired Signals

Additional Information in other files (Cell #2 Layout.pptx)

A. Pictures of the Dyno Cell Layout

# Section 1

## Introduction

## Introduction:

This project was the continuation of an initial project that was done by MIS 2000 regarding the use of Neural Networks as they related to Condition Based Maintenance of military vehicles. In phase 1 of the project information was analyzed from two sources. The first source was Army Material Systems Analysis Activity (AMSAA) Data, this data was acquired from vehicles in routine operation in the field. The AMSAA Data was acquired by external data acquisition systems installed in vehicles used around a military base. The other information was obtained from an engine operated in a dynamometer cell. One issue was that the data sources were not entirely under our control, due to the fact that:

1. AMSAA Data was acquired by others prior to the initiation of this project.

2. For the dynamometer portion of the testing for Phase 1 we were required to use an engine that was being run through a NATO Test to establish its durability. Therefore, perturbations to the engine could not be introduced because of the possibility of irreversibly affecting the engine. Access to the engine was limited to collecting baseline engine data during the scheduled maximum engine torque output evaluation at every 100 hrs and at five mid-range Speed/Torque points ("MiniMap")

Phase 1 allowed for evaluating how to construct and train Neural Networks to predict basic engine operation. Phase 2 of this project allowed for a dedicated dynamometer and engine that would allow for the induction of faults and structured testing.

## Phase II Objective

Create relevant faults in the engine operating conditions related to fluid temperature, pressure, and flow, which produce performance loss and impact the vehicle health.

Faults were introduced by altering the normal response of some electromechanical components of the engine control system. Known malfunctions were generated in such a way that the faulty condition could be turned on and off without actually exchanging malfunctioning components. Only one component at the time was altered.

The sensor faults were accomplished by skewing the calibration of the temperature and pressure sensors.

Electrical connections between the Engine Control Unit (ECU) and the sensors were broken with a Break-out Box so that the sensor output could either be fed directly to the ECU with no changes or be altered before being returned to the ECU.

The engine was operated at steady-states and selected speed/torque points ("Mini Maps") for periods of 1 to 2 minutes, as done in Phase 1. The perturbations were introduced continuously.

## Requirements for the induced sensor calibration changes

Different levels of perturbations were needed for each sensor. Initial estimates of perturbations were used that would not harm the engine while still creating detectable engine performance shifts.

## The sensors that were perturbed for this experiment

- **Pressure Sensors**
- **Engine Coolant Temperature Sensor**
- **Inlet Manifold Air Temperature Sensor**

The overall objective of the project is to develop prognostic algorithms and neural networks as they pertain to powertrain systems. The need for vehicle condition data requires the ability to implement prognostic algorithms that can be deployed on the vehicle. This requires a software approach that is both efficient and deployable within a real-time vehicle environment.

This effort was accomplished with two parallel tasks, one focused on engine dynamometer work, the other on analyzing seeded fault data with prognostic/diagnostic algorithms.

## Technology Introduction

The concepts of Virtual Sensing, Condition Based Maintenance (CBM) and Prognostics have received significant attention for various applications due to potential benefits that include reduced cost by substituting expensive sensors with estimations, significantly reduced development time, greater time in service, and improved diagnostic and control properties. However, one problem with developing such technologies across this breadth of applications has been the need to develop special techniques for each problem domain, and perhaps for each individual problem. Our strategy to deal with all of these problems involves the application of a single comprehensive approach involving machine learning using the most powerful representations for complex systems, combined with statistical analysis to produce the most accurate estimations and projections of system performance possible from the system models we create.

This approach provides a straightforward mean to accomplish these difficult tasks, but relies on proprietary technology to produce the machine learning algorithms which are keys to success.

A key element relates to the method of training dynamic neural networks so that they can attain their full theoretical capability and produce highly accurate models of complex dynamic systems. This approach is derived from combining a series of innovations developed by several groups over the past ten years and on improvements devised from the experience garnered in the application of these methods across a wide range of estimation, classification, virtual sensing and control applications. No other group in operation in the U.S. today has this combined experience in developing and deploying these applications in serious, difficult, and real-world problems. MIS2000 believes that a model based reasoning approach, using dynamic neural networks to detect anomalies in system performance, will provide the Army with the best solution for implementing CBM for military vehicles. A major benefit is the fast and efficient way of utilizing real time vehicle data to assess vehicle health. Since these algorithms are extremely efficient, they do not necessitate extensive computing power and require a minimum hardware footprint to run.

The constraints of this approach are related to the data being fed into the model. The fact remains that good decisions cannot be made using bad or incomplete information. Therefore, an open systems approach will be used to allow for maximum integration of additional sensor data as new technology becomes available. This information may be generated from virtual information i.e. Virtual Torque Sensors, or from additional external devices to augment real-time information acquired from the system via J-1939, Controller Area Network (CAN) bus or wireless interfaces

# Section 2

## Description of the Experimental Set-up

## Description of the Experimental Set-Up

### Background

This section describes the experimental set-up with which the Seeded Faults Experiments were carried out. Figure 2.1 shows schematically how several subsystems were combined together to build a comprehensive set-up for running an engine under well controlled conditions and for collecting highly granular and multifaceted data of the engine response. The way the engine was instrumented and the layout of the dynamometer cell was dictated by the need of maintaining commonality with testing conditions commonly used in other military projects for evaluating engines. However, several novel features were added to the standard equipment used in durability testing, such as the custom instrumentation to seed engine faults, a multi-functionality data acquisition system to record data broadcast from the engine ECU, and additional sensing devices, both high grade instruments and prototype sensors, to monitor the engine operating conditions. For overview of dyno set-up see (Attachment A).

Customized hardware was used so that specific malfunctions could be turned on and off in the engine operating conditions in a controlled fashion. For most cases, these faults were introduced by electronic means in order to avoid having to swap good parts with bad ones. This was achieved by perturbing the response of specific engine sensors, thus, tricking the ECU to compensate for perceived shifts in operating conditions. In the case of the Boost and the Fuel Pressure sensors, the changes induced by the control systems produced measureable changes in engine output. It was, therefore, possible to

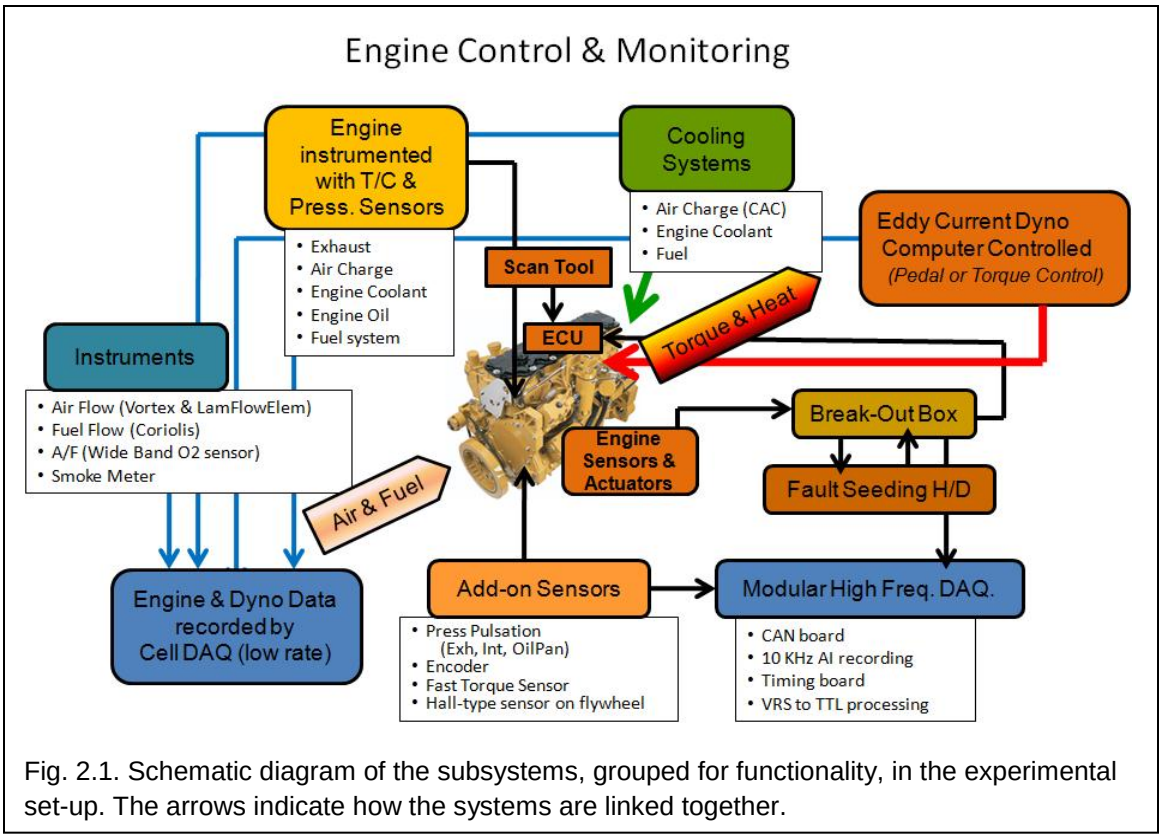



Fig. 2.1. Schematic diagram of the subsystems, grouped for functionality, in the experimental set-up. The arrows indicate how the systems are linked together.

alter engine performance in progressively larger steps, including decreasing and increasing engine torque output, to simulate engine deterioration. Restriction valves (not shown in Fig. 2.1) were also added in the intake and exhaust system as a way of externally perturb engine performance.

Data collection was coordinated between two systems. While the dyno cell data acquisition system (Cell DAQ in Fig. 2.1) recorded at low rate a large number of signals from the temperature and pressure sensors with which the engine was instrumented and from the other instruments and controllers, a custom made system (Modular High Freq DAQ in Fig. 2.1) acquired engine data broadcast on the communication bus (CAN bus) continuously so that the operating conditions perceived by the ECU could be compared to the actual engine behavior measured by laboratory instrumentation. Other types of data could also be recorded with this custom daq, for instance, TTL signals from timing sensors and analog signals from prototype sensors for monitoring combustion fluctuations. This system was capable of collecting data at much higher rate than possible with the dyno cell daq but high speed recording was practical only in burst mode ("snapshots"). Snapshots of about 20 seconds were sufficiently long to monitor the engine behavior during transitions between operating points, although the experiments in the dyno were designed for evaluating the engine mainly under steady state conditions.

The data derived from the seeded faults experiments in this comprehensive testing facility constitute a rich set of engine conditions, under both normal and abnormal conditions, which can be used, after appropriate handling and reformatting, to develop and test Neural Networks models describing engine performance.

## Engine

Our investigation was carried out on the military version of the Caterpillar C7 (for engine specifications see Appendix A) engine mounted on an Eddy current dynamometer at the dynamometer facility of the Mobility Group at the Detroit Arsenal. The engine is a diesel in-line 7.2L 6 cylinder engine with a waste-gated turbocharger, positioned at the mid-point of the exhaust manifold, and Hydraulic activated /Electronically controlled Unit Injectors (HEUI) with no EGR (Exhaust Gas Recirculation) and no exhaust after-treatment. The air pump driven by the engine is vented to air. The alternator is disconnected and power is supplied by a 24 V battery pack that is continuously trickled-charged while the engine is running.

The air charge was cooled with a high efficiency water cooled heat exchanger positioned on the side of the engine. The air temperature was controlled at the desired set-point by regulating the inlet water flow in the heat exchanger. The temperature of the cooling water was not regulated. The typical set-point for the air charge temperature was 127 degF, as used in other durability tests carried out in these facilities. Because the heat exchanger controller was optimized for mid to high flow conditions, during idle

and low torque operations, the air charge temperature dropped and slowly recovered during mid torque operations. The engine coolant was also cooled externally with a similar set-up and the typical temperature set-point was 205 degF. Different set-point values could also be selected for both systems but the controllers response was optimized for these temperatures.

| Param | Units | Param | units |
|---|---|---|---|
| Speed | RPM | Intake Air Vortex | CFM |
| Torque | LB-FT | Fuel Flow | PPH |
| Throttle Position | % | Lambda | units |
| BMEP | psi | Smoke Meter | FSN |
| BHP | hp | LFE Dellta P | H2O |
| Air After Filter | Deg F | Torque Sensor | Ft/lb |
| Air After Compressor | Deg F | Air After Turbo | psig |
| Air Intake Manifold | Deg F | Air B4 Manifold | psig |
| Fuel Supply | Deg F | Air Inlet RSTR | H2O |
| Fuel Return | Deg F | Crankcase | H2O |
| Coolant B4 Engine | Deg F | Oil Gallery | psig |
| Coolant After Engine | Deg F | Exhaust B4 Turbo 1 | Psig |
| Oil Sump | Deg F | Exhaust B4 Turbo 2 | Psig |
| Oil Galley | Deg F | Exhaust Stack | H2O |
| Exh Port 1 | Deg F | Compressor Inlet | H2O |
| Exh Port 2 | Deg F | Coolant B4 Engine | Psig |
| Exh Port 3 | Deg F | Coolant after engine | psig |
| Exh Port 4 | Deg F | Coolant Cap | Psig |
| Exh Port 5 | Deg F | ECM 1--Boost | Volts |
| Exh Port 6 | Deg F | Sensor 1--Boost | Volts |
| Exh B4 Turbo 1 | Deg F | ECM 2-InjCtrlPres | Volts |
| Exh B4 Turbo 2 | Deg F | Sensor 2-InjCtrlPres | Volts |
| Exhaust Stack | Deg F | ECM 3-Eng Oil Pres | Volts |
| Coolant B4 CAC | Deg F | Sensor 3-Eng Oil Pres | Volts |
| Coolant After CAC | Deg F | ECM 4-EngCoolTemp | Volts |
| H20 Tower In | Deg F | Sensor 4-EngCoolTemp | Volts |
| H2O Tower Out | Deg F | ECM 5-IntManiAirTemp | Volts |
| CAC Flow | GPM | Sensor 5-IntManiAirTemp | Volts |
| Air B4 Filter | Deg F | Water Tower Flow | GPM |
| Air Cell Ambient | Deg F | Air Cleaner Out | H2O |
| Relative Humidity | % | Barometric Press. | In Hg |
| Relat. Hum. Temp | Deg F | Transducer Rack | Deg F |
| Dyno H2O In | Deg F | H2O Tower In | Psig |
| Dyno H2O Out | Deg F | Dyno H2O in | Psig |
| Air Test Cell Depressi | H20 | Fuel after filter | Psig |
| Fuel Regulator Supply | PSIG | Fuel supply | psig |
| Fuel Cart Return | Psig | Fuel return | H2O |

| Dyno | LabInstr | Eng | BoB | CellSyst |
|---|---|---|---|---|

Table 2.1. Example list of analog signals recorded by the Dyno Cell daq. As indicated in the legend, the color shading is meant to highlight signals deriving from different functional blocks as illustrated in Fig. 2.1. Dyno indicates values derived from the dynamometer controller; LabInstr refers to laboratory instruments added to measure engine inputs and outputs; Eng refers to T/Cs and Press Sensors with which the engine was instrumented; BoB refers to engine production sensors tapped at the Break-out Box; CellSyst indicates signals related to Cell environmental conditions and systems controlling fluids temperature.

## Instrumentation

Table 2.1 illustrates the variety of monitored signals (analog) derived from sensors, control systems and laboratory equipment. The list shows parameters of interest for our studies, grouped according to functionality as schematically shown in Fig. 2.1. It highlights whether the data derive from the dyno controller (Dyno), laboratory instruments (LabInstr) for measuring air and fuel inputs to the engine and exhaust, auxiliary engine sensors (Eng), voltage outputs of some engine sensors (BoB) tapped at the Break-out Box and the corresponding signal inputs to the ECU to track when sensor perturbations are introduced, and a number of environmental cell data and parameters associated to the systems that control the temperature of different fluids. A total of 138 parameters were recorded by the Dyno Cell daq (the full list is given in Appendix C), 5 associated with time information and dyno programming steps, 79 analog signals, and 54 status flags. With the exception of the BoB signals (10 in total) and the broad band Torque Sensor, this is the standard equipment configuration used for engine performance and durability studies. Details of the instrumentations are given below.

The engine was instrumented with a series of thermocouples and laboratory grade pressure sensors that monitor fluids temperature and pressure (Engine Coolant, Air Charge, Exhaust, Engine Oil, Fuel) at several locations in the engine and in the

external cooling systems. The exhaust gas temperature was measured at each exhaust port as an indication of mean combustion differences between cylinders. Pressure and temperature were also measured at the two inlet ports of the turbocharger, corresponding to the left half and right haft of the exhaust manifold, and downstream of the turbo in the exhaust duct. Additionally, the temperature and pressure of the air intake, before and after the air charge cooling system and before and after the turbo, were measured so that air handling system could be closely monitored. Pressure sensors were also inserted in the engine cooling system. In addition to this large number of monitoring devices related to the engine, other temperature and pressure sensors were used to monitor that the dynamometer, cell environmental conditions and the fluid-temperature control devices were operating within the desired range.

The engine speed and torque measurements were derived from the dynamometer controller instrumentation. The engine output was regulated by the dyno controller by means of an electromechanical device that actuates the engine pedal. Since Pedal Position corresponds to engine speed (Governor) rather than torque demand in the control strategy of the military version of the C7, the dynamometer could only be stably operated in two modes: in one case the Pedal Position was set while the engine speed was kept at a desired set-point by means of the dyno brake (“open loop case”, used to measure the engine output for a given driver demand, for instance, 100% pedal); in the other case (“closed loop”) engine speed and torque were maintained at the requested set-point by means of the dyno brake and by changing the pedal position with the servomechanism.

While engine performance and durability evaluation is commonly carried out with an Eddy current dyno with which the mean torque output is measured at steady state, this dynamometer cell was purposely equipped with a number of other high precision instruments to measure the inputs to the engine (air and fuel) and its output (torque, heat and crankshaft dynamics) especially suitable for studying engine performance changes due to subsystems perturbations. The intake air flow was measured by both a Laminar Flow device and a Vortex meter. The fuel consumption was measured by a differential Coriolis system with a response time of the order of 1s. The exhaust air-to-fuel ratio was detected with an ETAS Lambda Meter (made by ETAS). Soot production could also be monitored at steady state by an AVL Smoke Meter (made by AVL), but was not routinely used in our tests for practical reasons. Since these instruments, except the smoke meter, have relative fast response time (1s or better), measurements could be carried out during transition between two operating points to assess how quickly the engine output stabilized after the transient. Since the fault seeding experiments requires repeating a given test sequence several times, it is important to understand how quickly the engine output stabilizes after a transient and/or perturbation for expediting the experiments.

Engine speed and torque measurements obtained through the dynamometer instrumentation are usually filtered. Consequently, other instruments with higher frequency response were added to measure engine speed and torque fluctuations and evaluate combustion maldistribution between cylinders. An broad band torque sensor (strain-gage type, made by IRT) was mounted in-line between the engine and the dyno coupler to measure torque fluctuations due to combustion events and torque changes during transitions. Moreover, a high resolution laboratory encoder was mounted in front of the engine dampener for measuring crankshaft rotational speed accurately since speed fluctuations can be correlated to torque fluctuations. Additionally, a Hall-type sensor was mounted on the flywheel housing facing the ring-gear as another encoder at the back of the engine for investigating the effects of crankshaft torsional oscillations.

The pressure sensors with which the engine was instrumented were meant for mean value measurements. Thus, we have relied on a new type of low-cost piezoelectric device, potentially suitable for on-board application, to investigate the benefit of information derived from detecting pulsation variability in the intake and exhaust system related to uneven combustion events which should parallel fluctuations observed in torque and crankshaft acceleration. The device is commercially available and detects pressure fluctuations (ac component of pressure) in either the exhaust flow or a low pressure fuel line by contacting the fluid through a small orifice. It is typically used as a low-cost, easy to install diagnostic tool for identifying ignition and fuel system problems in a vehicle during repair in the shop. Three such sensors were employed for this project, one mounted in the intake system (after the Charge Air Cooler, CAC), one in the exhaust (post-turbo), and one attached to the oil dip-stick tube to detect blow-by.

| Electronically Seeded Faults |
|---|
| *Sensor Calibration Modification* |
| • Injection Control Pressure Sensor<br>• Boost Pressure Sensor<br>• Oil Pressure Sensor<br>• Intake Air Charge Temperature<br>• Engine Coolant Temperature |
| *Scan Tool* |
| • Cylinder Injection Cut-off |
| **Mechanical perturbations** |
| • Intake Flow restriction<br>• Exhaust Flow Restriction |

Table 2.2. List of Seeded Faults attempted in the project.

## Fault seeding

Table 2.2 shows the Seeded Faults pertaining to this project. They were selected as likely candidates to affect the engine torque output by affecting either the fuel delivery system or the air charge with a potential effect in the engine torque output. Since the selection of faults had to be done before starting the experiments without much prior knowledge of the engine control strategy, calibration and level of implemented diagnostic, evaluation of several cases were built in the plan knowing that the some of the resulting perturbation would turn out to be either insignificant in terms of engine output or could trigger engine operating modes outside the scope of the project (for engine protection derating). Certain faults were seeded electronically by custom built circuits

which modify the transfer function of certain engine sensors. A Break-out Box (BoB) was used to tap into the engine harness that connects the ECU to the engine sensors (such as timing, temperature and pressure) and actuators (injector solenoids and PWM (Pulse-Width Modulation) valves for controlling Boost and Injection Control pressure). The signal output and common lines of the sensors measuring Fuel Injection, Oil and Boost pressures were broken here and redirected to a custom designed analog device that modifies ("skews") the voltage signal output to simulate a change in the device gain (the gain multiplier ranges from 0.5 to 1.5) and/or bias (range from -1V to 1 V). The skewed output was then connected to the PCM harness at the BoB. Although the "Skewing" device is capable of changing the output of three sensors at the same time, we have only perturbed one sensor at the time in the experiments described in this paper. Information on the Analog Signal Modification Device (ASMD) used to modify the output of the pressure sensor is given in Appendix B. Similarly, the resistance of the thermistors for measuring the Engine Coolant and the Intake Air Charge temperatures was increased/decreased by a variable resistor network (Temperature Signal Modification Device or TSMD), inserted in the high signal line either in series or in parallel through the BoB, so that the PCM would detect a temperature lower or higher, respectively, than the true value by a selectable amount. The perturbation to either the pressure or temperature sensor could be remotely turned on and off from the dyno control room by means of relays embedded in the circuitry for ease of monitoring the effect of a perturbation.

Combustion instability was induced by interrupting fuel injection to one of the six cylinders at the time. This was achieved by either opening the line carrying the solenoid actuation current at the BoB or by means of the programmed functionality available in the Caterpillar Scan Tool used for assisting the technician to perform repairs. To avoid prolonged stress on the dyno joint, the second method was preferred since the perturbation could be introduced for short periods of time. Faults involving modifications of the response of other actuators (such as the Fuel High Pressure Regulator or the Waste-gate) were not included in the scope of this project because the respective valves are controlled by means of PWM signals.

To study potential engine output loss caused by added impedance in the intake air flow (that is, simulating a plugged air filter), a butterfly valve was placed downstream of the two air flow measuring instruments, approximately six feet upstream of the air inlet to the turbocharger. The valve closure could be changed in nine steps ranging from completely open to fully closed. Another butterfly valve was placed at the end of the exhaust pipe before the vent, roughly 20 feet from the turbo outlet. This valve was actuated by a stepping motor so that fine rotational settings of the valve (about 2 degrees) could be repeat ably selected.

Some of the experiments were carried out with DF2 fuel, then, repeated with JP8 that has lower energy content.

## Data acquisition

Two data acquisition systems were used to acquire data during the experiments. One was the Cell DAQ which is traditionally used for engine studies (performance/durability) by the dyno team. It could be programmed to record either continuously or during selected intervals all of the signals from the laboratory instruments, from the thermocouples and pressure sensors with which the engine was instrumented, from the five production engine sensors tapped at the BoB, from the dynamometer controller and from the other devices monitoring the cell operating conditions as listed in Appendix C. Because of the large number of channels, the maximum achieavable rate was about 0.7Hz. This data constitutes the Cell data set.

The other system was the Modular High Frequency DAQ used for acquiring analog signals with higher frequency content deriving from prototype sensors and event driven-signals (CAN and timing data). It was based on a National Instruments cRIO FPGA-based system that included a CAN board, a 16 bit analog board operated in differential mode, and a timing board with 100 ns resolution. The cRIO was operated through a Real-Time LabVIEW interface, custom developed to meet the requirements of this project, running on a host computer to which the data to be recorded was continuously streamed by Ethernet connection.

| CAN Param | Units | Rate (ms) | Res. | Cell Param | Units |
|---|---|---|---|---|---|
| EngSpd | RPM | 15 | 0.125 | EngSpd | RPM |
| Load | % | | 1 | Load | Lb*Ft |
| BoostPres | KPa | 500 | 2 | AirB4Mani | psi |
| InjCtrlPres | MPa | 500 | 0.004 | n.a. | |
| ManiIntAitT | degC | 500 | 1 | AirIntMani | degF |
| EngCoolT | degC | 1000 | 1 | CoolAftEng | degF |
| OilPres | KPa | 500 | 4 | OilGallery | psa |
| ElPot (Battery) | V | 1000 | 0.125 | n.a. | |
| Fuel Rate (ECU Commanded Fuel) | L/hr | 100 | | FuelFlow | PPH |
| Pedal | % | 50 | 0.25 | Throttle | % |
| LoadAtSpd | % | 50 | 0.125 | n.a. | |
| NomFric | % | 250 | 100% | n.a. | |
| DesEngSpd | RPM | 250 | 0.125 | n.a. | |

Table 2.3. Engine operating parameters available on the communication bus are listed with corresponding measurements derived by external instruments which were recorded by the cell data acquisition.

Messages broadcast on the engine communication bus (J1939 protocol at 250 KBauds) were continuously recorded so that engine data derived from the ECU could be compared to measurements done by the laboratory instruments. The engine operating parameters (CAN data) available on the bus for this engine configuration are listed in Table 2.3 with their rate and resolution. The table also shows whether the same parameter was measured independently with another device and recorded by the Cell DAQ. Notice that there was no independent measurement of the oil high pressure line which pressurizes the fuel in each injector.

Signals from analog sensors, containing high frequency information related to combustion events, were recorded at 10Khz by means of the cRIO analog board. They included: the IRT torque sensor, the three SenX

sensors, the primary (CAM1) and secondary (CAM2) variable reluctance sensors used by the ECU to adjust injection timing (there is no crankshaft sensor in this engine), an inductive current meter inserted in the Cylinder 1 actuation line at the BoB to monitor injection timing, and in some other instances, the injector driver signal for another cylinder. High frequency recording was enabled for short windows of time (snapshots) ranging from 1 to 30 s, user selectable according to what needed to be captured at different operating conditions. Each snapshot was triggered manually by the operator from the main panel of the LabVIEW application and the data were displayed at the end of the capture.

The cRIO timing board was used to record with 100 ns resolution the timestamps of pulse edges derived from 5 devices monitoring the crankshaft rotation, specifically, the laboratory-grade encoder, whose output was set at 36 pulses per revolution, the encoder index marking each revolution, the TTL signal from the Hall-type sensor mounted facing the ring gear, and the primary and secondary CAM sensors (resolution of “48 minus 1 teeth”, the missing tooth used for deriving TDC, Top Dead Center) after their voltage output was transformed into TTL signals by a custom-built Trigger Schmitt-type circuitry. This type of recording was started by the same manual trigger used for the analog recording and lasted the same length of time.

Since the analog signals are recorded an constant rate while the CAN messages and the timing data are event-based signals that are asynchronous, the data were saved in three different files that are precisely time aligned by the cRIO internal clock. Time alignment with the low-rate data recorded by the dyno cell data acquisition is done on post-processing on the basis of engine speed signatures.

Pictures showing the location of the cell layout, auxiliary sensors and of other instrumentation used in the project were taken by the dyno team (contractors need special permission for taking photos within the TARDEC facilities), organized and annotated in the file “Cell #2 Layout_distD.pptx” which contains a detailed description of the cell layout. This file is provided separately.

# Section 3

## Description of the Experiments

## Description of the experiments

### Background

This section describes the protocol developed for running the Seeded Faults experiments. A set of procedures were established in concert with the test cell team to maintain consistency between testing sessions and run experiments within constrains imposed by the dyno cell operations. The protocol includes procedures for warming the engine, both from room temperature and on warm restart, for tracking the stability of the max engine output over different testing days, and for studying the engine behavior with/without faults at selected speed/torque points with the engine coolant and air charge temperature maintained within repeatable limits. Test replicas with faults seeded at different levels had to be performed for generating a rich data set of data for training and testing the neural network models and for demonstrating that the induced engine performance shifts were statistically detectable. To fit the sizeable number of experiments within the project timelines and resources, tests were developed aiming at striking a balance between acquiring engine data at relatively stable air charge and exhaust temperature and keeping the length of any given experiment relatively short.

The majority of the experiments consisted in monitoring the engine behavior at a limited number of speed/torque operating points (MiniMap in Table 3.1) distributed over the engine operating range. The dyno, under computer control, stepped the engine through the selected points always in the same order at constant intervals, with either no fault or one fault seeded at the time at a constant level throughout the sequence. The same test sequence would then be repeated at several perturbation levels. Experiments were repeated on different testing sessions taking care to change the perturbation level order. Under speed/torque control conditions, the effects related to some seeding faults could be perceived as changes in engine fueling, exhaust temperature and pedal position. Conversely, other experiments were carried out with the dyno controlling only the engine speed and measuring the torque output. In this case the engine pedal position was set at 100% for the Performance test, which is the traditional way to evaluate the engine stability, while for the Pedal test the pedal position was stepped through the same settings observed in the MiniMap test with no faults. Thus, MiniMap and Pedal tests generate complementary information on engine behavior, the first being similar to studies of engine efficiency such as Brake Specific Fuel consumption, BSFC, the latter more representative of engine operations in a vehicle. Comparison of results obtained in the two modes may also provide insight on whether some torque fluctuations might be induced by the dyno/pedal control system.

## Testing protocol

Each testing session started with the engine at room temperature. It was agreed that engine warm-up would be done at 1500 RPM and 35% pedal position, consistent with the way that the C7 engine was operated in conjunction with other studies carried out by the dyno team, followed by a Performance test with no seeded faults, at the end of which the engine would be shut down following a short engine cool down.

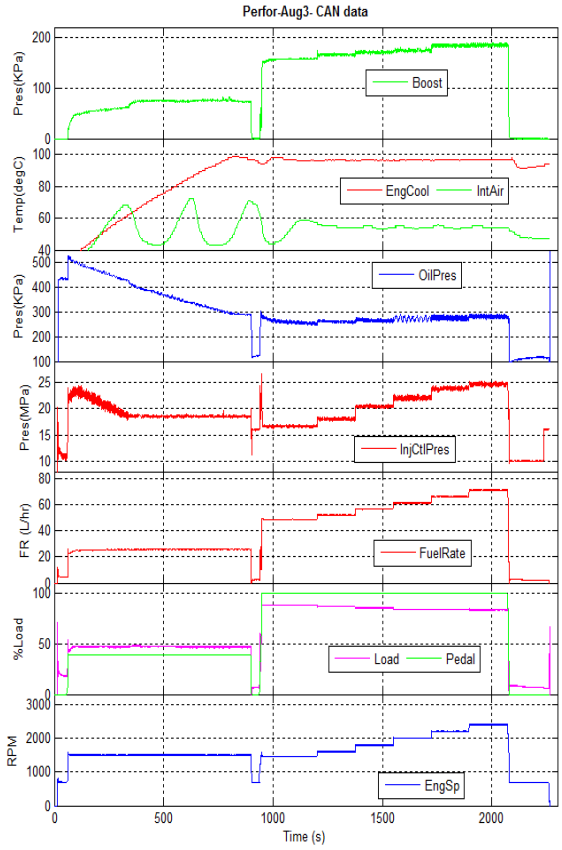



Fig. 3.1. Timeseries of the CAN parameters during the Warmup/Performance test.

Figure 3.1 shows CAN parameters plotted as a function of time to illustrate this type of run. The EngSpd trace shows the engine start (accomplished by means of the engine starter), followed by a short idle to ensure that the cRIO daq system and other equipment were operational, then a 15 mins warm-up time so that the engine coolant temperature stabilizes at 205 degF. After that, the computer stepped the engine through six engine speeds (1450, 1600, 1800, 2000, 2200, 2400 RPM) with the pedal position fixed at 100%, holding for 170 s at each step to collect the data for the engine stability analysis described later. The engine was then returned to idle conditions for the cool-down before shut-off. Notice the very large oscillations during warm-up of the air charge temperature since the CAC controller response was not optimized for these operating conditions. Thus, the first step in the sequence was extended from 170 to 260 s. Notice that the dyno system controlling the sequence of steps was tied to the dyno Cell daq, which was not able to complete a full acquisition cycle of the more than 130 parameters in less than 1s. As a consequence, the time at each step was determined by the daq cycle, thus, the actual holding time had a small variability (+/-1s).

The combined Warm-up/ Performance test was followed by experiments with/without the seeded faults. Since each engine restart required to reset the dyno computer and data acquisition in addition to stabilizing the engine coolant and air charge temperatures, for efficiency the dyno operations were programmed to repeat the same sequence of engine operating points several times, typically five. Thus, an experiment is

defined as the engine operations between each engine start and corresponding shut down, consisting of a short step to make sure the engine coolant is within the desired temperature limit, and repeating the same sequence of programmed operating points each at a different perturbation level of the same seeded fault, including no perturbation. It was not practical to change type of fault during a given experiment because it required opening two switches in the BoB for the high and common lines of the sensor associated with producing that perturbation and reconnecting the lines either to the ASMD (for pressure changes) or the TSMD (for temperature perturbations) input, output and common. On the other hand changing perturbation level was done by simply turning a rotary switch in either device. This change could only occur at idle because both devices were located in proximity of the Break-out-Box to limit the wiring length, and for safety, access inside the cell was only allowed when the engine was either off or at idle. However, relays had been incorporated in both devices to bypass their circuitry so that the perturbation could be turned on/off from the cell control room.

To optimize the data collection while working around the allowable engine running hours for a test day, the series of experiments carried out for each testing session was different. A typical experiment would take between 40 to 60 minutes, depending on test conditions. At times, experiments had to be cut short because of available test time or if unpredicted malfunction with the equipment or the engine occurred. Regardless, the data for each experiment were contained in a separate set of files identified by the date and test type which will be discussed in detail in the data structure section. The list of experiments is provided separately as an Excel file. For each testing session the log gives information on the test conditions of each experiment (fault type and perturbation levels) with the name of the files for the CAN, Cell, Analog and Digital data, the number of snapshots generated for each run and the containing the data reduced for analysis.

| ID | EngSpd | Torque |
|---|---|---|
| (#) | (RPM) | (Lb*Ft) |
| Stab | 1450 | 700 |
| P11 | 1450 | 400 |
| P12 | 1450 | 550 |
| P13 | 1450 | 700 |
| P21 | 1800 | 400 |
| P22 | 1800 | 550 |
| P23 | 1800 | 700 |
| P31 | 2200 | 400 |
| P32 | 2200 | 550 |
| P33 | 2200 | 700 |
| P41 | 1450 | 300 |
| P42 | 1800 | 300 |
| P43 | 2200 | 300 |
| Idle | 700 | 35 |

Table 3.1. Operating points for the MiniMap test sequence with the dyno controlling the engine in speed/torque control mode

## Engine Operating Points

Most of the experiments have been conducted stepping the engine through twelve selected operating points plus idle, representative of field conditions, and holding the engine at each step for a short period of time (typically 40 s). Table 3.1 lists in order the operating points of the MiniMap test sequence for which the dynamometer controls the engine speed and torque. Notice that the first point labeled Stab is used for engine stabilization and was approximately 140 s long. The three engine speed values were selected within the range recommended by the manufacturer for this engine (1440 to 2400 RPM) as representative of low, medium and high

speed operations. At each engine speed the torque is stepped through 3 values representing the mid/high torque range (this engine has a max output rating of 800 Lb*Ft at 1450 with JP8 fuel and a recommended top engine speed of 2400). Three low torque points complete the sequence. Considerations regarding the repeatability of the air charge temperature suggested grouping the low torque points all at the end of the sequence instead of interleaving them according to engine speed.

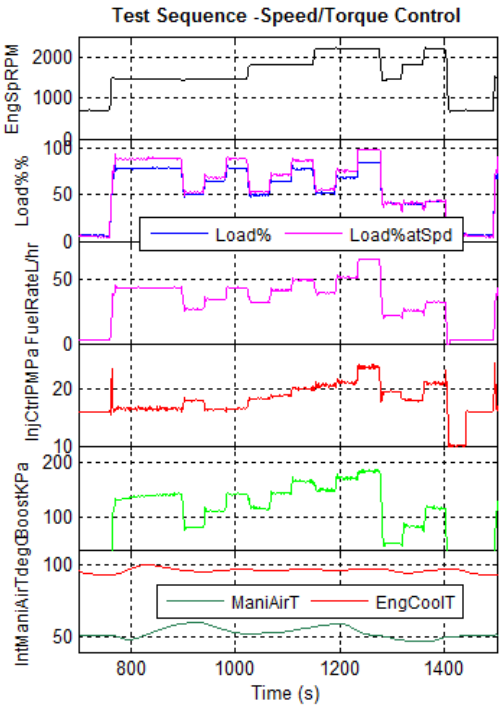



Figure 3.2. Engine operating parameter timeseries acquired from the CAN communication bus.

(205 degF), while the air charge temperature was slow to stabilize around 127 degF and was seen to drift within a 30 degF band because the CAC control parameters are optimized for the high torque range. Since in these tests the torque changes from 300 to 700 Lb*Ft, we have not attempted to achieve tight temperature stabilization, as done when measuring rated engine output or BSFC, because the test would become too long. Care was taken to prevent the air charge temperature from rising above 160 degF

Figure 3.2 shows plots of engine parameters acquired from the CAN communication bus as a function of time during such a MiniMap test sequence with no seeded faults. Notice the stepped behavior of the traces except for the case of the air charge temperature which increases/decreases very slowly, lagging the torque changes. The stabilization point was important to limit the first oscillation and reduce temperature variability. During the twelve step sequence,, the engine coolant remained within +/- 3 deg F of the set-point

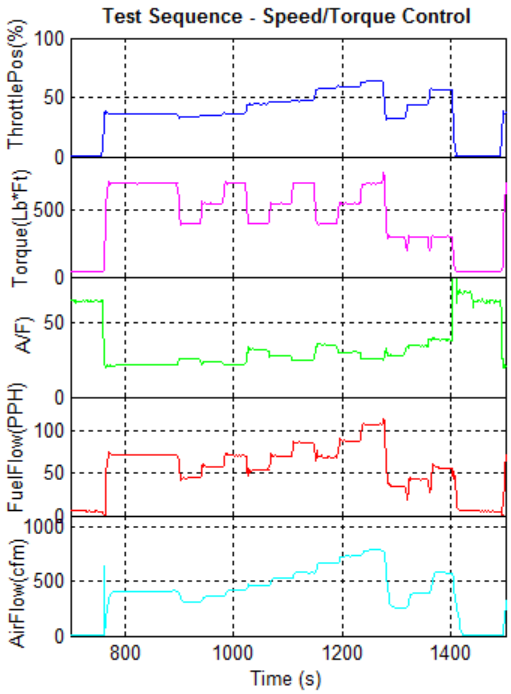



Figure 3.3. Timeseries of engine operating parameter from laboratory instrumentation and recorded with the Cell daq.

since the PCM progressively reduces fuel to protect the turbocharger against elevated exhaust temperature.

The CAN data timeseries show the engine operating conditions perceived by the PCM through the engine sensors and internal calculation. Such “internal” picture of the engine needs to be compared with that obtained from equivalent data acquired through the cell instrumentation. Fig. 3.3 shows timeseries of the Intake Air Flow, Fuel Flow, Air-to-Fuel ratio, Torque and Pedal Position (Throttle%) recorded by the cell daq at low rate. This is the “external” picture of the engine which depicts its true behavior.

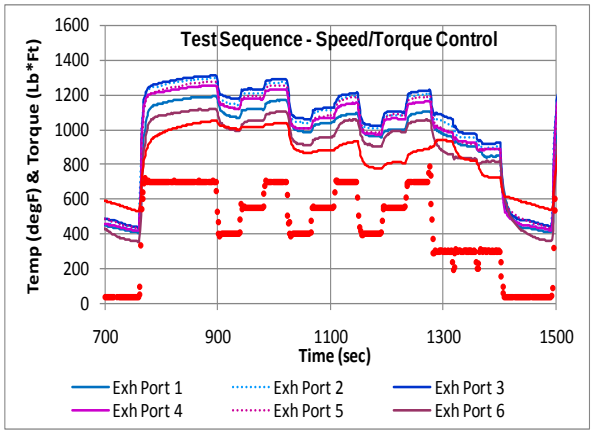



Figure 3.4. Traces of the exhaust temperature at different locations.

A more detailed picture of the engine operating condition is gained from analyzing additional parameters recorded through the the cell daq. For instance, the plots in Fig. 3.4 illustrate differences in temperature between the exhaust ports and downstream the turbocharger (Stack) while Fig. 3.5 shows the exhaust pressure upstream and downstream the turbocharger together with the intake pressure.

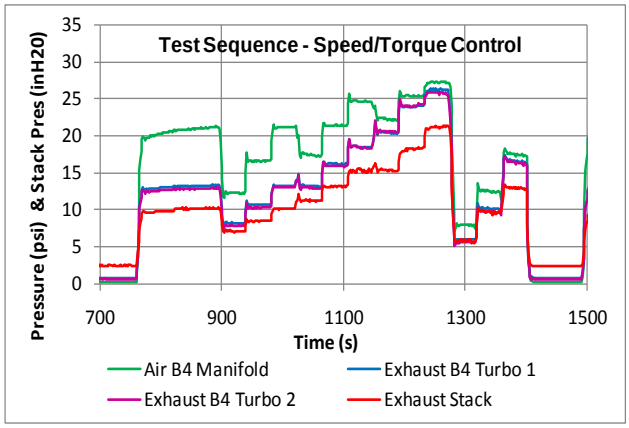



Figure 3.5. Traces of pressure signals in the intake and exhaust system.

These plots are useful to assess engine stabilization although slow drifts in temperature are due to the exhaust system walls equilibrating in temperature. The exhaust temperature is an important parameter since it affects the pressure on the turbocharger, thus, the induction process. Also, the turbocharger needs to be protected from over temperature. Notice that the temperature traces show a faint drift after the first rapid change due to the torque transition between steps. Notice, however, in the plots of Fig. 3.5 that pressure appears to stabilize more quickly than with temperature.

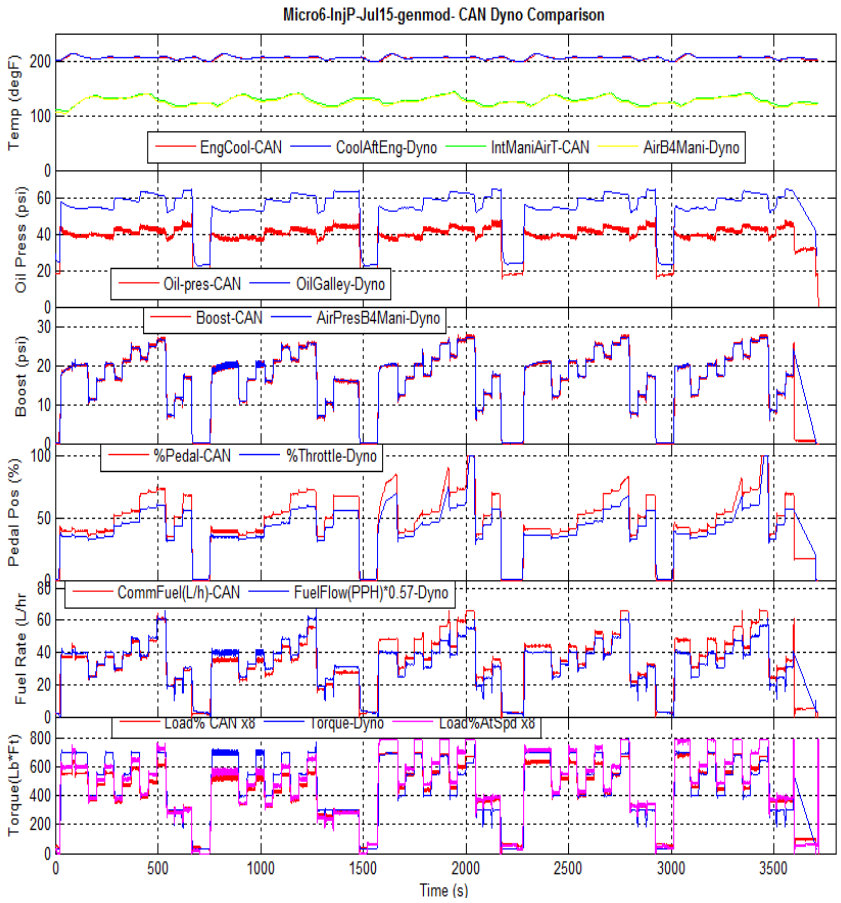
Notice also from Figs. 3.4 that exhaust temperature in the port increases as a function of torque (more heat is generated because more fuel is burnt) but decreases as a

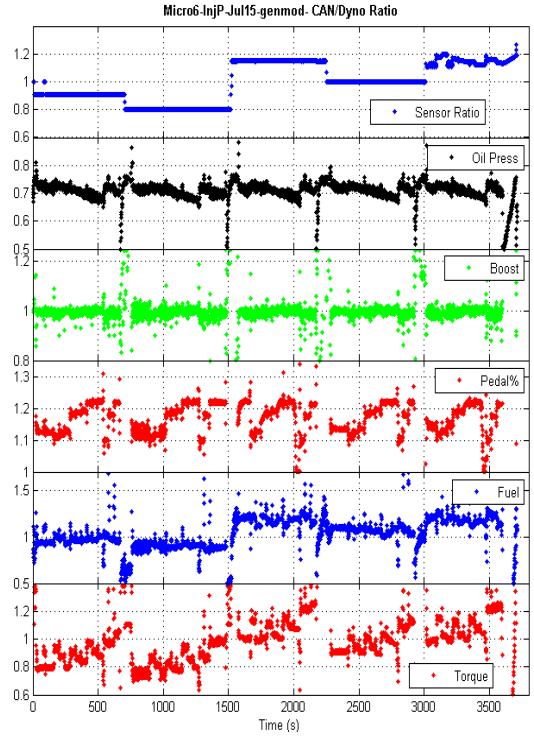
function of engine speed (more flow). When either the fuel system or the induction system malfunctions, the temperature signature could be used as another diagnostic. Unfortunately, there is no exhaust temperature sensor in this application as it would be found in a platform with after-treatment. The data set collected in this project may be useful to evaluate other sensing methods for engine performance abnormalities.

Temperature differences between exhaust ports may be partly due to geometric effect but they may also reflect combustion differences between cylinders. A small pressure difference (less than 0.5 psi) is observed between the two inlet ports to the turbo under some operating conditions consistent with the observed difference in temperature.

## Results

Fig. 3.6 gives an example of time series derived during an experiment in which the injection pressure sensor gain was changed. The dyno was in speed/torque mode. Each subplot in Fig 3.6a show the time series of a CAN parameter together with the corresponding one measured with an external device. Temperatures and Boost pressure are in very good agreement, while Pedal%, Commanded Fuel and Load% (CAN data) deviates from actual measurements of %Throttle, Fuel Flow and Torque (Dyno data) even in the 4 repeats with no perturbation applied. The deviations change depending on the perturbation level. Ratios of these time series are helpful to highlight differences induced by the seeded fault as shown in Fig. 3.6b. The fuel ratio is similar to the sensor ratio, although noisier because of the combined variability from the two measurements, especially during step transitions when time delay between the two types of measurements play is an important factor.




(a) (b)

Fig 3.6. (a) Composite plots of CAN and Dyno data for an experiment in which the 12 point sequence of Table 3.1 was repeated 5 times at gain perturbations level of x0.9, x0.8, x1.15, x1, and bias offset of+0.4V. (b) Ratios plots of the CAN/Dyno timeseries. Changes in the ratios from sequence to sequence highlights the effect of the

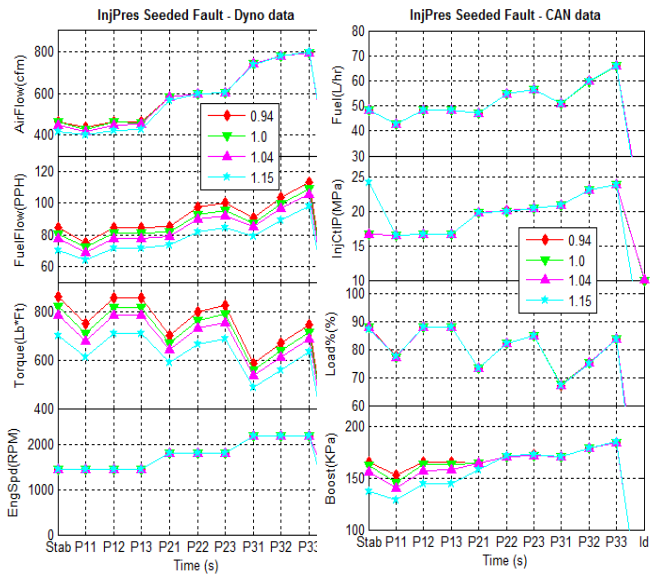



Fig. 3.7. Mean values of engine operating parameters at different speed/torque points are plotted in the order with which the data was acquired during a test sequence similar to that of Table 3.1 (the low torque points have been omitted). Each plot corresponds to a test during which gain of the Injection Control Pressure sensor was altered by a factor indicated in the legend. For these tests the dyno was operated in "open loop"; torque is only measured but not held constant, while the Pedal was fixed.

A quantitative analysis of the effects induced by a Seeded Fault can be better carried out in terms of mean values averaged over equal time windows to reduce noise. In this way the sensitivity of a parameter to the perturbation level can be established.

Fig. 3.7 shows composite plots of mean values of selected CAN and Dyno measurements plotted according to the order with which the points in the test sequence were stepped through as indicated in Table 3.1. The points at low torque have been omitted in these tests. The data refer to four replicas of the test sequence, one without the fault (Gain=1) and three with gain changes of x0.94, x1.04, x1.15, respectively, as indicated in the legend. Notice that in this experiment, torque is not controlled and the pedal position is set to the mean value found in previous tests under speed/control mode. However, the same torque value is not always reached after engine restart likely because of instability in the servomechanism that actuates the engine pedal.

The CAN data in Fig. 3.7 show that there is no apparent change in Injection Control Pressure since the PCM is able to compensate for the different sensor reading by means of the pressure control valve. No changes are seen in the commanded fuel, thus, Load% does not change since it is calculated from speed and fuel. On the other hand, a change in fuel delivered to the combustion chamber has occurred since a higher/lower control pressure translates into a higher/lower quantity of fuel injected in the cylinder. Notice that if the sensor is skewed high (gain x1.15, for instance) the PCM decreases the injection pressure, thus, the quantity of injected fuel decreases. Indeed, as shown by the Dyno data, the engine torque output measured by the dyno changes proportionally with fuel since the dyno is not trying to control torque. The fuel flow measured by the external instrument (the Coriolis fuel meter) also confirms that fueling has changed.

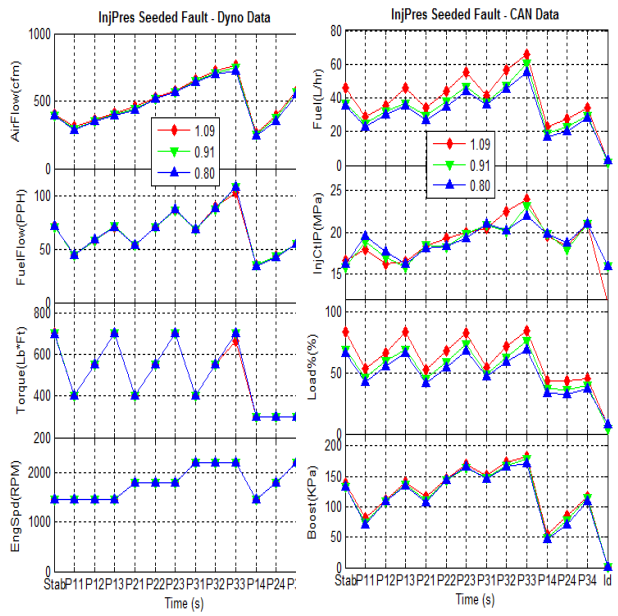



Fig. 3.8. Mean values of engine operating parameters at different speed/torque points as given in Table 3.1. The gain change applied to the Injection Control Pressure Sensor is indicated in the legend. These tests were carried out with the dyno operating in speed/torque control mode.

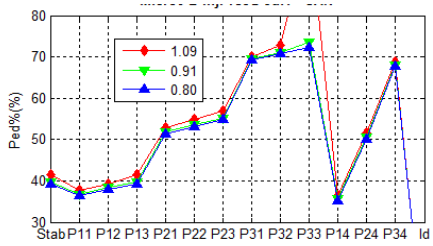



Fig. 3.9. Plots of the CAN Pedal% for the same tests illustrated in Fig. 3.8.

The plots in Fig. 3.8 show the opposite effect. In this case Torque is kept constant by the dyno. When the PCM adjusts the injection pressure by means of the bleed valve to correct for the pressure shift indicated by the skewed sensor, the engine output changes but the dyno corrects for the torque change by means of the throttle. Thus, the dyno cell instruments do not detect any change in either torque or fuel, but a small shift in Pedal can be observed (Fig.3.9). The PCM data, however, show changes both in Fuel and Load% due to the Pedal correction caused by the dyno controller. The plots in Fig. 3.9 show that the Pedal change is very small (of the order of 1 to 2%) because Pedal is insensitive to Torque within a certain range as indicated in Fig 8. If at that speed the engine is not able to produce enough torque, the pedal value climbs up toward 100%. For instance, this is seen happening at the third torque step at 2400 RPM.

We stress that, for the experiments with seeded faults, test-to-test variations of these features may provide supporting evidence that a malfunction has been induced. However, we need to establish first the detection limit for these measurements since the observed feature may be due to noise. Specifically, we need to establish the repeatability of these parameters and the stability of the engine over the time during

| EngSpd | Torque | Fuel Flow | AirFlow | A/F | Tturb1 | Tturb2 | Pturb1 | Pturb2 | AirB4M | CoolT | AirIntT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1450 | 0.8% | 0.6% | 1.2% | 1.6% | 0.9% | 1.0% | 1.0% | 1.2% | 0.6% | 0.1% | 2.2% |
| 1600 | 0.6% | 0.5% | 0.8% | 1.0% | 0.3% | 0.3% | 0.6% | 0.6% | 0.2% | 0.0% | 1.3% |
| 1800 | 0.4% | 0.4% | 0.8% | 1.1% | 0.4% | 0.3% | 0.9% | 0.5% | 0.2% | 0.1% | 1.3% |
| 2000 | 0.3% | 0.9% | 0.7% | 1.0% | 0.4% | 0.2% | 1.0% | 0.7% | 0.3% | 0.1% | 0.7% |
| 2200 | 0.4% | 0.7% | 0.7% | 1.0% | 0.4% | 0.2% | 0.9% | 0.5% | 0.1% | 0.1% | 0.6% |
| 2400 | 0.7% | 1.1% | 0.7% | 1.1% | 0.7% | 0.6% | 0.9% | 0.5% | 0.1% | 0.0% | 0.5% |

Table 3.2. The table reports the variability of engine parameters recorded by the Cell daq at different engine speed during 8 Performance tests (100% throttle) carried out on different days. The variability is given as the ratio of the mean standard deviation over the mean value of the measurements over 20 s.

which the experiments were conducted to prove that there may be a correlation between some features in the data and the seeded faults.

Table 3.2 gives the variability observed over 8 Performance tests carried out on 8 separate days for 11 engine parameters related to engine output. The measurements were done with the external sensors and instruments and recorded by the Cell daq. The variability, given in percentage, is calculated as the ratio of the standard deviation over the mean of measurements done over the last 20 s of each steady state step in the Performance Test (test carried out at 100% pedal).

| EngSpd | Load% | CmdFuel | Boost | InjCtlP | EngCoolT | ManAirT |
|---|---|---|---|---|---|---|
| 1450 | 0.6% | 0.2% | 0.8% | 1.0% | 0.3% | 0.3% |
| 1600 | 0.4% | 0.2% | 0.8% | 1.1% | 0.4% | 0.3% |
| 1800 | 0.3% | 0.3% | 0.7% | 1.0% | 0.4% | 0.2% |
| 2000 | 0.4% | 0.1% | 0.7% | 1.0% | 0.4% | 0.2% |
| 2200 | 0.7% | 0.1% | 0.7% | 1.1% | 0.7% | 0.6% |
| 2400 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |

Table 3.3. Variability of CAN parameters calculated for the same Performance tests used in the data of Table 3.2.

Similarly, Table 3.3 shows the variability for some of the CAN data calculated for the same tests. The data shows that the variability of most parameters is better than 1 percent. Similar variability values are obtained for other types of tests such as the one at mid/high torque described in Table 3.1.

## Effects of Seeded Faults

We have described earlier how changes in Fuel Flow (i.e., fuel commanded by the ECU to be injected in the cylinders) were detected when the Injection Control Pressure sensor was modified. This perturbation is useful to produce a set of data in which the actual Fuel Flow is different from what the ECU estimates. It also mimic the case in which the engine performance has changed without the ECU perceiving it. Similarly, a loss of engine output was produced by altering the calibration of the Boost Pressure sensor but in this case an appreciable torque loss was caused by the ECU derating the engine (i.e., decreased fuel) for protection of the turbocharger since the low pressure was misunderstood as a decrease in air flow which would have increased the exhaust temperature above the safe operating level. Thus, this seeded fault did not produce an engine torque change in progressively larger steps, rather it occurred when the Boost Pressure perturbation became sufficiently high that the ECU could no longer compensate the perceived low pressure condition by means of the Waste-gate. Therefore, Fuel Flow changed only when the ECU enable derating. The details of the affects of the Injection Fuel Pressure sensor and of the Boost Pressure sensor were reported in the paper presented at the 2010 GVSETS meeting.

Similarly, restricting the intake air flow or the exhaust flow caused the intake air flow to decrease without an appreciable loss in torque. Moreover, the ECU did not take any action in either case to change fuel since air flow is not measured directly. Torque loss was not easily detectable until the exhaust temperature increased over the safe limit for the turbocharger and these experiments had to be halted. The resulting data were not sufficient to develop an exhaust temperature model that could be used as a diagnostic tool.

Calibration changes for Manifold Intake Air temperature to read high triggered the ECU to derate fuel for safety because of the potential unsafe increase in exhaust temperature triggered by the elevated air charge temperature. Only selected experiments were carried out with this type of fault, mainly to record the effect but not to develop a diagnostic model.

The oil pressure sensor was first thought to affect operation of the high pressure pump that controls injection pressure. However, large perturbation in its calibration did not produce any effect in the measured fuel pressure. It was later determined that this sensor is used only to warn the driver of a low oil condition, thus, it was not a useful fault to study engine output changes.

## Information from Additional Sensors

We have discussed so far the information contained in the CAN and the Cell data showing that the effects of Seeded Faults are evaluated by looking for shifts in engine parameters measured by external devices and by comparing these measurements with those available to the ECU from production sensors or from other ECU. All these data reflect the cycle-averaged engine behavior and do not contain information related to distinct combustion events. To collect information on combustion stability and/or differences between cylinders in-cylinder pressure sensors have been traditionally used. These are expensive and mainly suitable for laboratory studies. Thus, during the experiments with seeded faults, we have incorporated low cost prototype devices (SenX sensors), potentially suitable for in vehicle use, that measure fluctuation of pressure in a gas flow. Since both intake and exhaust flows pulsate due to charge induction and combustion events, respectively, the SenX sensors could provide diagnostic information with affordable hardware relying on complex pattern recognition techniques.

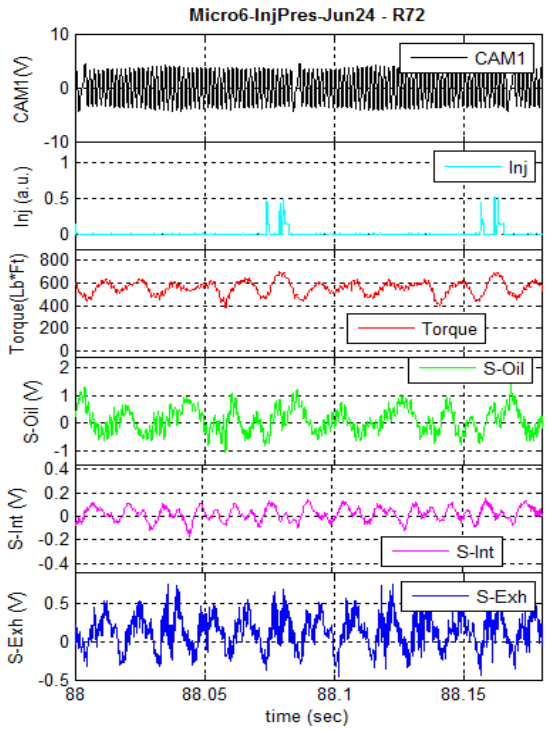



Fig.3.10. Example of Analog Signals recorded with the cRIO. Fluctuations in the SenX sensors in the Exhaust and in the Intake (S-Exh & S-Int) are correlated with those observed in the broad band Torque sensor. The signal of the sensor in the oil pan (S-Oil) is masked by noise. The primary CAM signal and the Inj #1 driver signal are shown to highlight the engine cycle.

Our interest in collecting data from these sensors during the Seeded Faults experiments was stimulated by the availability of the broad band Torque sensor capable of detecting torque fluctuations induced by combustion events and by the laboratory encoder to derive accurate engine speed measurements, thus, also angular acceleration to which Torque is related. As shown in Figure 3.10, the analog signals from three SenX sensors (one in the exhaust, one in the intake, and one in the oil pan) were recorded with the analog board in the cRIO at 10 KHz to provide sufficient bandwidth to analyze signatures associated with combustion events even at 2400 RPM. For practical reasons, the high frequency recording was limited to a short time window, typically 20 s. The time axis of the plots is referenced to the CAN timing. As shown in the figure, in addition to the broad band sensor, the primary CAM signal (variable reluctance sensor) was also recorded to provide a

timing reference and cylinder identification capabilities. In some experiments, the signal driver for Injector #1 was also recorded.

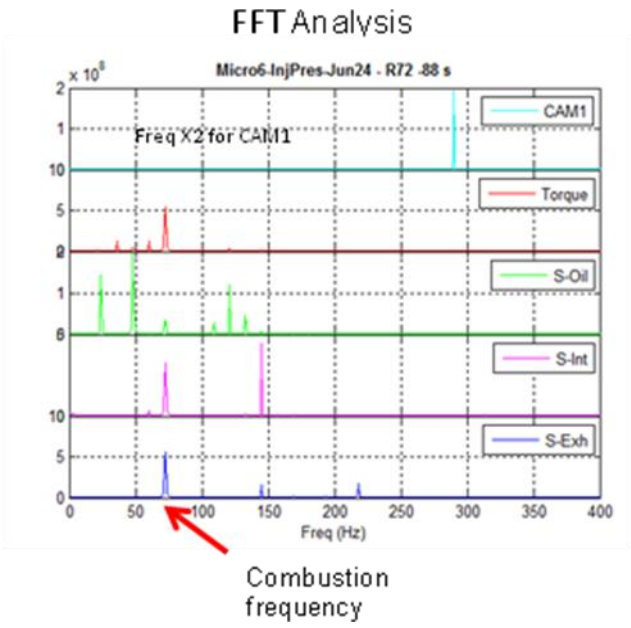



Fig. 3.11. Fourier analysis of the signals plotted in Fig. 3.10. The FFT was done over a 1s window.

Figure 3.11 shows Fourier analysis of the SenX sensor traces reported in Figure 3.10 over 1 s window. This analysis was attempted to evaluate whether the frequency spectrum would change when either the Injection Pressure or the Boost Pressure sensor were perturbed, indicating that the seeded fault was inducing some level of cylinder maldistribution, but no detectable effect could be reliably observed. Therefore, the only analog data that were used for model development were those from the broad band Torque sensor used to train the Virtual Torque Sensor model as described in detail in Section 6.

# Section 4
## Data Collection and Reduction

## Data Collection and Reduction

This section describes the structure of the data acquired during the experiments and the post-processing that was required to generate input data in a form suitable for inputs to different Neural Network models. The challenge with the data collection for this project was to balance the need for acquiring a relatively small number of asynchronous data from three different sources (CAN messages, high frequency analog sensors and timing signals for monitoring the crankshaft speed) while still collecting a very large number of analog signals (Cell data, consisting of mostly slow varying signals related to engine and fluid temperature control systems) to maintain consistency with the procedures used by the dyno team in their studies (see Section 2). Only a few of these analog signals are actually used during development of the NN models, some others are only used to verify the effect of the perturbations, and most are checked to make sure all of the systems were working properly.

The substantial differences in data sources and information content and legacy considerations dictated that two data acquisition systems (Dyno Cell DAQ and cRIO based DAQ) be employed and the data for each experiment be recorded in four types of data files reflecting the different characteristics of the data, as schematically illustrated in Figure 4.1. While the Cell data were recorded in engineering unit as time-series in an ASCII file (text file with .DAT extension), the cRIO saved the data in LabVIEW binary format which needed to be converted into ASCII. As described in detail later in this section, further post-processing was required to reduce the CAN and the timing data into time-series in engineering units (that is, two column arrays each containing timestamps and the physical values of the parameters) and perform the time alignment with the Cell data. Since CAN data were broadcast at different rate, the last step was to

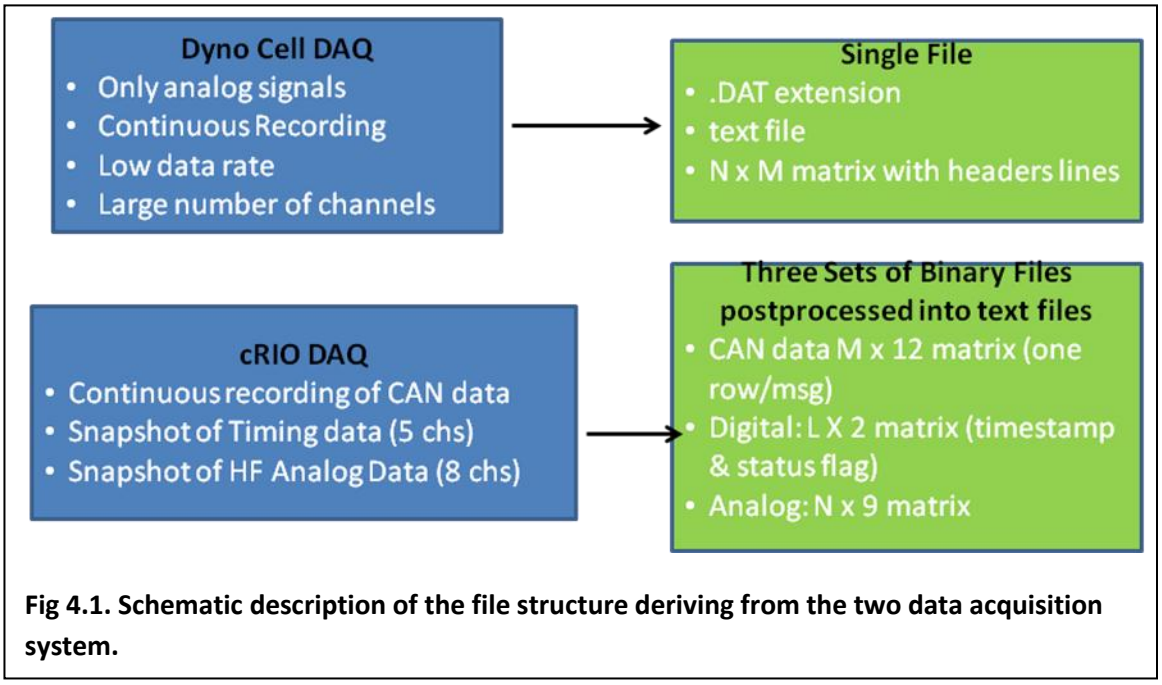



**Fig 4.1. Schematic description of the file structure deriving from the two data acquisition system.**

reduce CAN and Cell data time-series into a generalized matrix format (a single multicolumn 2D array) with the same time base so that the information could be efficiently used as inputs for both the NN models development and the analysis of the effect of the perturbations.

In our implementation the existent Cell daq was recording continuously, at approximately 0.7 Hz and with some filtering, all of the data derived from the laboratory instrumentation and the sensors with which the engine was instrumented (see list of signals in Appendix C). The cRIO based daq, custom tailored to this project, handled the other acquisition modes which necessitated substantially higher data rates but were limited to few signals, specifically, CAN, timing, and high frequency analog signals which included the broad band Torque sensor, prototype SenX sensors, the engine timing sensors (variable reluctance type), and the driver signal for Injector #1.

In practice, the Cell data provide an overall comprehensive picture of the “mean” engine behavior, specifically of the “true” engine operating conditions since they are measured externally, while the CAN data represent a somewhat limited “internal” picture of the engine behavior as detected by the ECU but with increased granularity respect to time (higher rate). Since CAN and Cell data streams are continuously recorded, it is possible to time align them on the basis of the well-defined engine speed steps produced by the way the experiments were performed. Some of the signals recorded with the analog and digital boards in the cRIO also refer to measurements of “true instantaneous” engine behavior because they derive from external sensors that detect signatures associated with each combustion event (i.e., broad band Torque sensor and prototype SenX sensors).

Continuous recording of the CAN data was very important for comparison with the external data and could be easily managed within the cRIO FPGA resources. On the other hand storing continuously either the timing or the high frequency data was not only unfeasible but the incremental gain in information over what could be gathered in burst mode recording (“snapshots”) appeared minimal since our experiments were not focused on detecting random spikes but rather repeatable features in the signals. Thus, a LabVIEW custom application was developed to control the cRIO so that only “snapshots” of timing and analog signals would be acquired simultaneously for short windows of time of selectable length and triggered manually by the operator. Each snapshot generated two files, one for the synchronous data (analog, 10 KHz recording) and one for asynchronous data (timing or “digital”). Both acquisition modes relied on the same common clock used by the CAN board so that the snapshots and the continuous data are intrinsically time aligned.

## CAN and Dyno data reduction

| msgID | Rate (ms) | param | Conversion | Units |
|---|---|---|---|---|
| 2CF00400 | 15 | engsp | (B5*256+B4)/8 | RPM |
| | 15 | %load | B3-125 | % |
| 38FEEF00 | 500 | EngOilPres | B4*4 | KPa |
| 38FEEE00 | 1000 | EngCool | B1-40 | degC |
| 38FEF700 | 1000 | ElecPot | (B6*256+B5)*0.05 | V |
| 38FEF600 | 500 | Boost Pres | B2*2 | KPa |
| | 500 | Intake AIr Temp | B3-40 | degC |
| 38FEF200 | 100 | Fuel Rate | (B2*256+B1)*0.05 | L/hr |
| 38FEDB00 | 500 | InjControlPres | (B2*256+B1)/256 | MPa |
| 2CF00300 | 50 | Load at speed | B3 | % |
| | 50 | Pedal | B2*0.4 | % |
| 38FEDF00 | 250 | NominalFriction | B1-125 | % |
| | | DesEngineSpeed | (B3*256+B2)/8 | RPM |

**Table 4.1 CAN data format in the text files and conversion rules.**

For practicality, a binary CAN file would be closed during acquisition at 400,000 frames and a new one immediately created. While the converted text files were also kept as separate files, the data were merged during post-processing done in Matlab when conversion into engineering units was also carried out. Table 4.1 shows how to combine the 8 byte contained in the CAN frame to derive each parameter in engineering units.

Matlab files with the suffix "_ext" contains these 13 CAN parameters as 2D arrays, the first column being the timestamp, the second the engineering value. Parameter headers and units are also included in the file as string arrays.

Dyno Cell data were also imported in Matlab, not directly from the raw .DAT files, but from the Excel template used to summarize the results of every experiment. Import from the template offered a substantial speed advantage since it contained only the analog parameters of interest and the timestamps had already been converted to a numerical time series instead of being read character by character. Additionally, the parameter ordering was kept constant while over time the .DAT file structure had undergone changes. CAN and a subset of the imported Dyno data were time aligned using a graphical Matlab procedure consisting of progressive time shift to overlay the steps in engine speed. Although the final alignment determination required user judgment, the method was simple and the achieved alignment accuracy was of the order of 1s comparable with the time resolution of the Dyno Cell data. Afterwards CAN and Dyno Cell data were interpolated onto a 1Hz time-base. Dyno Cell data, as imported and and after interpolation, were also saved in the Matlab "_ext" files.

A subset of 30 Dyno Cell parameters was merged at the end with the CAN data to give a single 2D array which was saved with header information and other metadata into a Matlab file with the "_genmod" suffix. The parameter list in the 2D array is given in Table 4.2. The array could be easily exported as a text file. This generalized format contains more parameters than actually needed for the NN model development but is

useful because it makes it easy to extract subset of data for different purposes including rationality checks between different experiments.

| CAN data | Instr. & Dyno | Aux. Sensors | BoB |
|---|---|---|---|
| Time | Fuel Flow | T-IntAirMani | ECM1-Boost |
| EngSp | AirFlow | T-aftCompr | Sensor-Boost |
| Load% | A/F | CoolAftEngine | ECM1-InjPres |
| EngOilP | BB-Torque-Sen | T-ExhB4Turbo1 | Sensor-InjPres |
| Boost | Speed | T-ExhB4Turbo2 | ECM1-OilPres |
| InjCtrlP | Torque | T-ExhStack | Sensor-OilPres |
| EngCoolT | Throttle Pos | P-AirB4Mani | ECM1-EngCoolT |
| IntManiAirT | | P-aftTurbo | Sensor-EngCoolT |
| Pedal% | | P-ExhB4Turbo1 | ECM1-AirIntMani |
| ElPot | | P-ExhB4Turbo2 | Sensor-AirIntMani |
| FuelRate | | P-ExhStack | |
| DesEngSp | | T-OilGalley | |
| NomFric% | | P-OilGalley | |
| Load@Sp | | | |
| **CAN** | **Cell** | | |

Table 4.2. Parameters interpolated on a 1Hz time base saved in the Matlab file with the suffix "_genmod". The params highlighted in purple are the CAN data while the ones in blue are Cell data. The latter are listed in three subsets corresponding to Laboratory Instruments and Dyno Controller derived measurements, signals from thermocouples and pressuse sensors on the engine, and signals tapped at the Break-out Box to track the level of perturbation applied to the engine sensors for seeding faults.

### Input data reduction for the NN Models based on the CAN data

This section discusses how data sets for the NN model development were generated. Two examples are shown below, one pertaining to the information needed to generate estimates of actual Boost Pressure, the other for actual Fuel Flow delivered to the cylinders. These models were designed to estimate the mean value of an engine parameter as a function of time based on time dependent inputs as monitored during the MiniMap experiments. Data interpolated at 1Hz were, thus, suitable to develop these models since they were not expected to estimate the cycle-by-cycle engine response. While these models can follow the transition from one MiniMap step to the next, they are not meant to closely reproduce transients since the bulk of the data was acquired during quasi steady-state conditions for the purpose of detecting engine response shift caused by the seeded faults. More transient data would be needed to construct and train models well describing transients. Section 5 explains the methodology for constructing Neural Networks models based on CAN data and deals

with the overall considerations that need to be taken into account to design the Network architecture. However, we have taken the liberty of adding the results of the estimates of these NN models in this section concurrently with describing the input data since in crude terms these models can be thought of as deriving coefficients ("weights") for a best fit of known data (that is, when both the input parameters and the actual estimated parameters are "measured"). Once these "weights" are derived during training, the model is checked for consistency on a new set of the data ("blind test"), in which case only the inputs data are provided to the network.

| Input Parameters | |
|---|---|
| **Fuel Flow Model** | **Boost Pres Model** |
| EngSp | EngSp |
| Load% | Load% |
| EngOilP | EngOilP |
| Boost | Cmd Fuel |
| InjCtrlP | InjCtrlP |
| EngCoolT | EngCoolT |
| IntManiAirT | IntManiAirT |
| Pedal% | Pedal% |
| DesEngSp | DesEngSp |
| NomFric% | NomFric% |
| Load@Sp | Load@Sp |
| FuelFlow | Pres B4 Int Mani |

**Table 4.3. List of CAN parameters for input to the network are listed in the blue cells. The parmeters shown in red are the laboratory measurements needed as input during training.**

In each case the data were derived from experiments during which the voltage output of a sensor out was "skewed" so that the ECU compensates for the perceived shift. Specifically, for the Boost case, the Boost pressure sensor was altered causing the Waste-gate to adjust, within the controller allowed limits, in order to maintain the desired boosting action, therefore causing a change in the expected air charge delivered to the cylinders, while for the Fuel Flow case the Injection Pressure sensor was altered causing the Pressure Control Valve to adjust to maintain the desired pressure, thus causing more/less fuel to be injected.

Data from several experiments containing both normal and abnormal testing conditions (Seeded Fault) were combined to construct input data in the form of one training set and one testing set ("blind"). Both data set contain the values of those engine operating parameters that are always known when the model is deployed (that is, the inputs are "internal" engine data as perceived by the ECU). As shown in Table 4.3, for the present project, it was assumed to rely only on the CAN data as known inputs, although the information on the communication bus is somewhat limited compared to what would be available with direct access to the ECU. The training set needs to include the values measured by an external instrument ("true values") of the parameter that is estimated by the model. The "true" values derive from the Cell data.

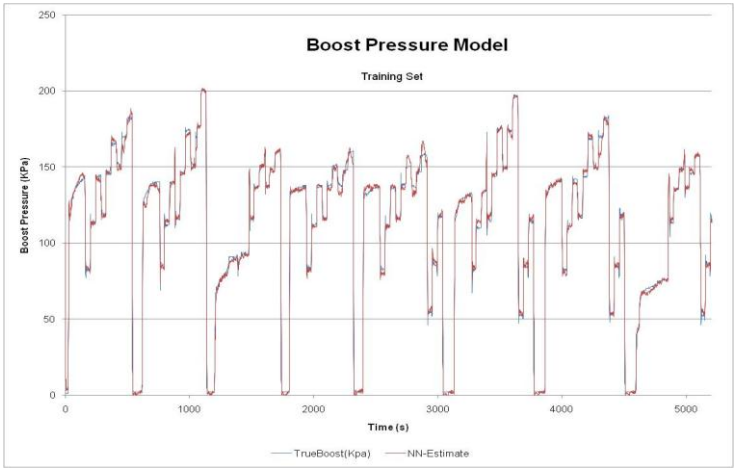



**Fig. 4.6. Example of sequences used as training set illustrated by means of the Boost timeseries. Blue indicates the measured input data (Dyno param AirPresB4Mani), red the model output generated during training.**

To simplify the networks development, the input data needed to be preprocessed in the form of time-aligned time series. The Matlab file designated with the "_genmod" suffix were the sources for the training and testing sets. The data extraction process was simplified because the 2D data array in that file could be readily sliced horizontally (i.e., referenced to time) using an index (not shown in Table 4.2) used to link every data to a specific operating point in the experiment sequence (MiniMap). The array was sliced to combine test sections according to perturbation levels identified by the BoB related params and to reject data contaminated by unwanted experimental conditions. Given the standard ordering of the parameters in the 2D array, selecting the different parameters for each model was also straightforward (vertical slicing). In practice, data slices representing a MiniMap sequence at a given perturbation level were chained together. Slices were taken purposely selected from different experiments, with at least 2 slices per perturbation level, including no perturbations, and they were ordered in a way to mix perturbation levels. As a result, the input data set is in the form of N x L matrix, N being the number of independent measurements and L the number of parameters utilized by the model as given in Table 4.3, L being a subset of CAN parameters. Figure 4.6 illustrates the combination of sequences as input data for a Boost

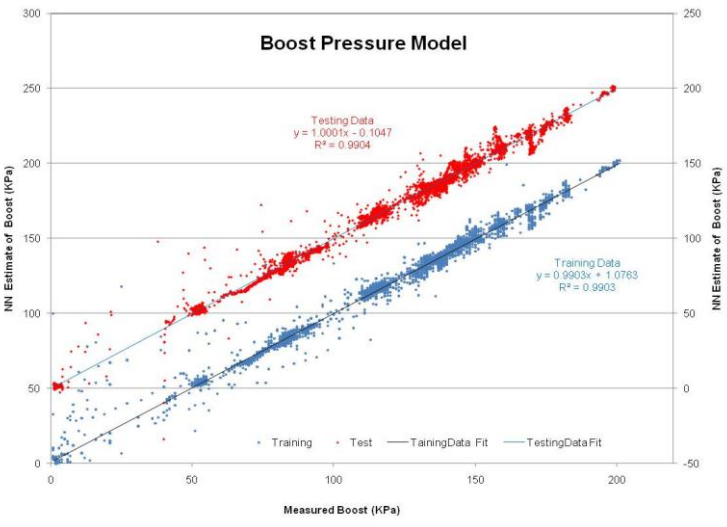



**Fig. 4.7. Scatter plots of the NN estimate of Boost pressure vs measured value.**

Pressure model (only the Boost time series is shown in the plot). The Model estimate during the training is generated in the form of a similar time series of equal length which is also shown in Figure 4.6.

Theoretically, if the training set is an overall faithful representation of all conditions met by the engine, any other set of data presented to the network as input would produce an output with equivalent fidelity. In practice, this is tested independently using another data set obtained in similar conditions but not used during the training. The testing set includes similar slices taken from different experiments at similar perturbation levels and two slices from the training set for consistency checks. The input testing matrix, however, does not contain the information on the true measurements ("blind set").

The agreement between estimated and actual Boost Pressure for both the training and the blind set can be evaluated by means of correlation plots as shown in Fig. 4.7. Since the data derive from experiments run under speed/torque control mode, some degree of clustering is observed, but the response of Boost action to the seeded fault as measured by the external sensor, builds in more gradually than observed in the fuel system perturbation case. The vertical spread in the scatter plots reflects a combination of variability in the data input and model quality. Notice that the variability between Training and Test data is very similar confirming the fidelity of the model. The standard deviation of the differences between estimated and measured values could also be used as a comparison criteria.

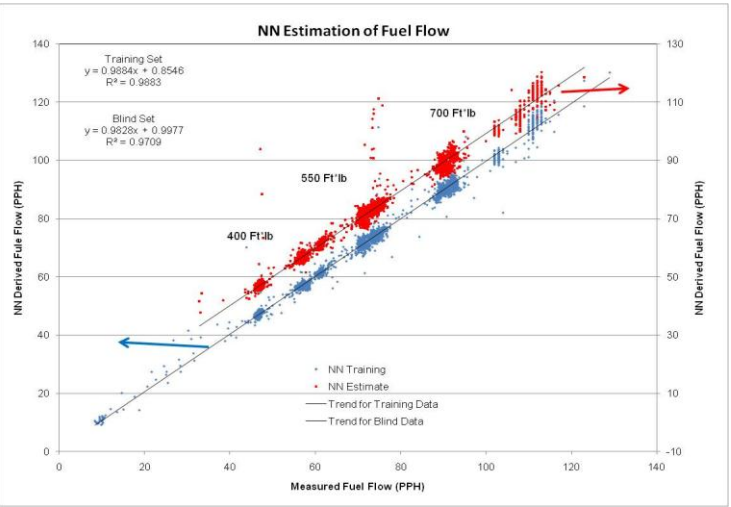



**Fig. 4.8. The plots show the correlation between NN model estimates and the Actual Fuel consumed at the 12 engspd/load testing points used in the Injection Pressure Seeded Fault (blue dots for Training run, red dots for Testing run). The data clusters represent the different Torque values at which the experiment was run.**

Input data sets for the Fuel Flow model were similarly created. Figure 4.8 shows equivalent correlation plots obtained for the Fuel Flow model both for the training and testing set.

## Timing data for the Torque Model

This section describes how we have monitored the crankshaft acceleration/deceleration behavior (dynamics) observed during an engine cycle which can provide an indirect measure of engine torque. The magnitude of the peak-to-peak oscillations in engine speed during an engine cycle (that is, two crankshaft rotations) are caused by the periodically positive torque generated during each combustion event, but the relationship between torque and crankshaft acceleration is complicated by system inertia, dampening, higher harmonics generation and torsional oscillations. Since the speed changes in a cycle are typically of the order of a few percents of the cycled-averaged engine speed, speed measurements need to be accurate. In out tests, four independent measurements were made relying on devices mounted at different locations and with different characteristics for assessing potential improvements in the quality of the data: two relied on existing engine sensors, one on a laboratory device (AVL encoder), and one on an auxiliary production type sensor (Hall-type).

The crankshaft rotational speed is measured by detecting the time for the crankshaft to turn by a constant number of degrees (angular step). This type of measurement requires a mechanical element, connected with the crankshaft with minimum backlash, with markings (encoding) identifying such steps, and a sensor, mounted rigidly with the engine, that produces a signal with a unique signature corresponding to the markings. Every modern engine requires at least one such device to control injection timing. Since an engine would not be able to run without timing information, the C7 engine relies on two Variable Reluctance Sensors (VRS) for robustness, a Primary and a Secondary CAM sensor, both mounted in such a way to detect equally spaced protuberances ("teeth") on the face of the gear which is connected to the crankshaft gear train to actuate the camshaft. To generate a reference position, one of the 24 protuberances is omitted. The VRS generates a modulated voltage output derived by the perturbation of the magnetic field perturbation induced by each protuberance sweeping the face of the sensor, as illustrated in the plot of Fig 4.2. Since the CAM gear makes one revolution every two crankshaft turns, the voltage signal plotted as a function of time shows only 47 peaks and one "break" ("missing tooth"). Thus one crankshaft revolution takes 24* Δt, where Δt can be measured as the time difference between two adjacent

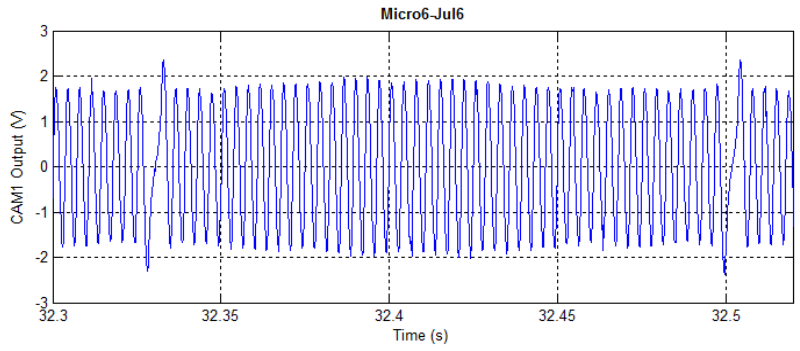



**Fig 4.2. Voltage output for the CAM1 VRS sensor plotted as a function of time. The mean engine speed is 700 RPM.**

zero-crossings of either the rising or the falling signal wave

The trace in Fig. 4.2 is the analog recording of the Primary CAM sensor (CAM1) tapped through the Break-out-Box carried out at 10 KHz with the analog board included in the cRIO. The board had to be operated in differential mode not to perturb the signals to the PCM. Accurate measurements of these time segments rely on accurate detection of the zero-crossing which cannot be done relying on these analog signals (each measurements occurs every 100 microsec), even relying on interpolation approaches. The slow varying VRS voltage output were thus transformed into TTL signals which can be acquired by means of a timing board (the analog-to TTL transformation was done by means of a Schmitt-Trigger-based device built for another similar application). Thus, each low-to high transition of the TTL signal (or high-to-low) points to the time when the shaft has rotated by the coded number of degrees. Both the Hall-type device and the encoder provided TTL compatible outputs.

The encoder was mounted on the front of the engine before the dampener. Thus, the speed measurements derived from the encoder and CAM sensors could be affected by torsional oscillations induced in the crankshaft. For evaluating this potential problem, the Hall-type sensor was mounted at the front of the engine relying on the ring-gear as the encoding wheel since it was not possible to add another suitable element. The ring-gear signal had the drawback of providing 134 purses/rev, which is non modular with 6, substantially complicating the analysis as discussed later. An additional output from the encoder (1 pulse/rev) was fed into the timing board to provide a reference position in addition to that deriving from the CAM sensors. As a result 5 signals had to be monitored with the timing board. The encoder output was chosen to generate 36 pulse/rev, which was a trade-off between resolution and the need for not over tasking the cRIO acquisition system.

The timing board recorded its clock time and the logical state of the input signals (high or low) every time that one of the five TTL signals switched. Because of considerations of speed and memory space allocations in the cRIO FPGA (the core of the data acquisition system), the time was recorded in clock ticks and each transition was recorded as a logic state in bit form. As an example, 01010 would mean that at the moment of the transition the TTL signals in Channels 0, 2 and 4 were low, while those in Channels 1 and 3 were low. Contrasting the method of recording analog data at constant rate, the data recording with the timing board is event-driven (asynchronous) and the resolution with which each event is time-stamped can be very high without the burden of storing a very large number of readings. The time accuracy, however, depends on the “jitter” of the generated TTL signal and the board speed.

The timing data were recorded as snapshots during the same time intervals of time the analog recording was activated. A file with the DI extension (called “digital” files because of the nature of the data) was generated for each snapshot. The text version of

the file (converted from the native LabVIEW binary format) is formatted as a two-column file, the first being the timestamp of the event in clock ticks (2^32), the second the decimal representation of the logical state. Therefore post-processing is needed to extract the time difference between corresponding events from which to calculate the crankshaft rotational speed (that is, the length of time taken by the shaft to rotate by the amount specified by the specific encoder). was written to extract such time differences. The algorithm converts the clock ticks to a continuous time base expressed in ms from the beginning of the recording by detecting clock rollovers (every 107 sec approximately).

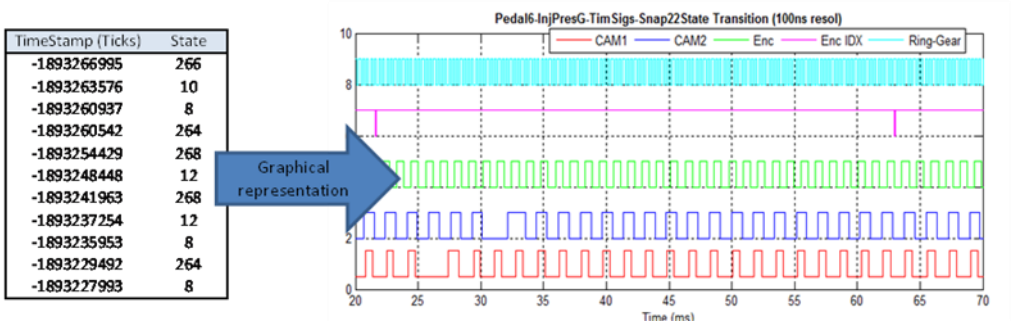



**Fig 4.3. The structure of the digital files in text format is shown. The stacked plots show the five TTL signals monitored by the timing board as reconstructed with the Matlab decoding program.**

The time difference calculation is based on extracting the logical bit pattern from the decimal representation, keeping track of the timestamp when the bit corresponding to a given channel switches low for the case of measurements done by means of the signal falling edge, and subtracting timestamps for adjacent events. Fig. 4.3 illustrates the format of the digital files and shows stacked plots of the high/low state for each channel as a function of time for one engine revolution. Fig 4.4 illustrates the calculated time differences which are tabulated as an array. The traces show the same data plotted as a function of the time corresponding to the next edge transition (falling edges in this case).The data give the inverse of the crankshaft speed measured over a rotation defined by the encoder resolution. Notice the oscillations observable in each signal, three for each shaft rotation identified by the black markers. Notice that the waveform

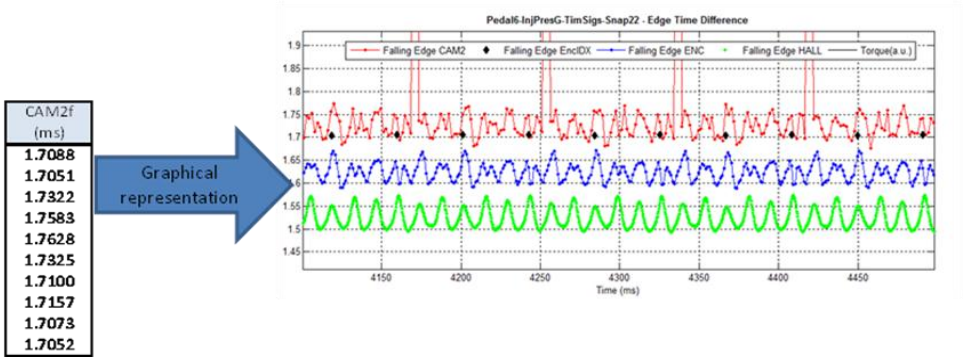



**Fig. 4.4. The traces illustrate the time differences derived for the CAM2, Encoder and Hall-type sensor. The vertical axis is in ms and applies to the CAM data (1450 RPM). The traces for the encoder and Hall sensor have been shifted. Their corresponding mean value is 2/3 and 24/134, respectively, of that of the CAM trace.**

for the Hall sensor is smoother than that of the encoder while the one for the Cam signal is less repeatable. The spike in the Cam-related signal every 2 rotations is an artifact of the "missing tooth". Notice that traces are not expected to be sinusoidal and line shape feature.

The data corresponding to the time differences are given as inputs to the Neural Network model that infers Torque. This model also requires the corresponding Torque data for training. Once the model has been developed and trained, only the timing data are needed as input.

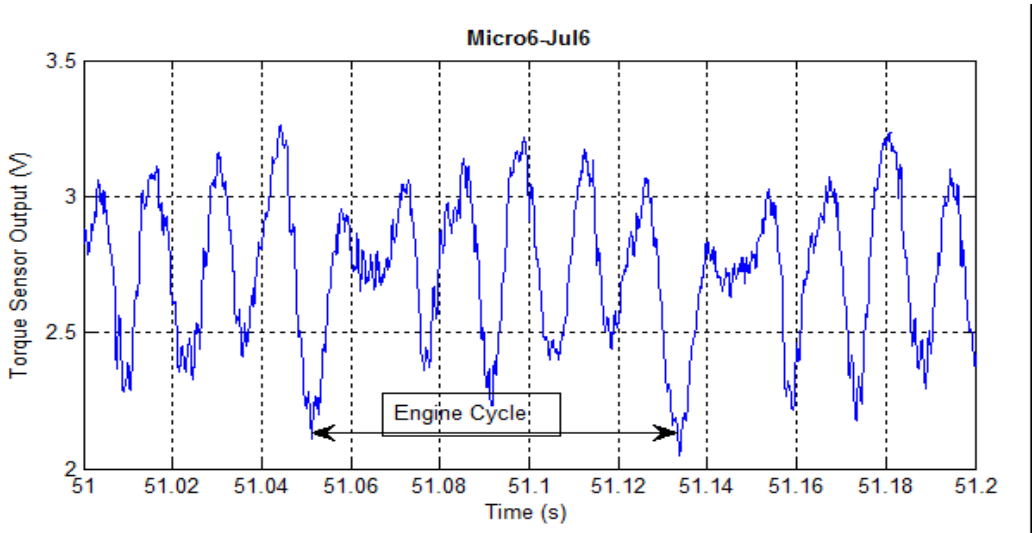



**Fig. 4.5. The plot shows the torque sensor signal recorded with the broad band torque sensor mounted at the rear of the engine. The signal is recorded with the analog board. The scaling is 264 Lb*Ft/V.**

The Torque data are found in the analog file with the corresponding filename. Time alignment of the two recordings derives from the fact that they are triggered at the same time by the cRIO with an uncertainty estimated to be of the order of a few microseconds related to the clock ticks required for the data acquisition loop to initiate both recordings. Since the torque values are averaged over a cycle in the model, the alignment error is negligible and does not impact the model.

Snapshots were mainly acquired during transition from one operating point to the other.

# Section 5

## NN Analysis of Dynamometer Data

## Introduction:

To accurately conduct an experiment it is important that there be control over the type and quantity of the data that is collected.

The reason for the direction of the second phase was because in phase I of the project, data was acquired from various vehicles in routine operation in the field. Typically, the data sources were not entirely under our control, due to the fact that

1. Data was acquired by others prior to the initiation of this project and we could not influence its composition, frequency of collection or volume.
2. Data was collected by sensors which supplemented the sensor suite usually installed to operate and monitor the engines, and as such, the data was connected asynchronously with the engine control computer.
3. Data was collected from the engine control computer, but since the operation of such computers involves material proprietary to the manufacturers, we were not able to completely specify independent requirements for data acquisition and composition.

## Methodology:

It was our goal to develop methods which were universal and broadly applicable to the task of data analysis, so that our procedures and processes would form a coherent, singular approach to the assigned tasks (performance analysis, diagnostics and prognostics) which could be deployed in any of the following ways:

1. On-going analysis of pre-established data bases using historical data
2. Modeling of systems based upon historical data.
3. Analysis of real-time data acquired from vehicles
4. Analysis of real-time data on-board the vehicles
5. Projections of future behavior based upon one or all of 1-4.
6. Capability assessments of vehicles with respect to each other or a pre-defined standard.

A key element of our strategy was the use of machine learning methods to build high-fidelity data models which provided important capabilities:

1. The models enabled us to review historical and real-time data to discern deviations from nominal performance or behavior. Applied with appropriate caution to the data under analysis, it permitted us to detect errors and inconsistencies in the data and to remove the data from the modeling effort in order to establish high-quality information databases to construct even more accurate models. This process, often is described as bootstrapping, has resulted in the extraction of high-quality data from data known or suspected to be contaminated by errors.
2. The models are compact representations of the data, and permit the analysis of systems to be carried out efficiently using the empirical models rather than the large, but frequently incomplete databases that they represent.

3. Models of many dynamic systems permit both interpolation and extrapolation of data in regions from which data has not been acquired.
4. Fixed parameter models of multiple dynamic systems can be captured in a single dynamic network and used to analyze (and control) systems whose dynamics are not stationary.
5. High-fidelity models overall operating conditions allow the introduction of model-based reasoning which compares system performance to nominal or expected performance and can optimally detect deviations.

We have developed powerful training algorithms for dynamic neural networks based upon the Kalman filtering methods applied to simultaneous weight updates in recurrent networks. This method, along with the series of innovative procedures applied during the training process produces neural networks which are capable of learning the behavior of complex non-linear systems, or even systems of such systems. Once a network has been trained to emulate the behavior of such systems, a very compact computational model of these systems can be used in lieu of the real systems. These schemes are easier to use and more efficient in many cases than efforts to create first principle models of such systems, or less accurate and efficient models by other approximation methods, for the purpose of diagnosing, controlling or estimating the state of such systems.

Neural network models were our primary recourse in developing the system models because our experience with many problems related to diagnostics and control in vehicle systems resulted in successful model development for every problem with which we were confronted. Clearly, other modeling schemes can, and have been, used successfully on some similar problems, but generally rely on expertise with the specific techniques. Our software is unique, but other schemes to train dynamic neural networks are widely available and have been used with some success on these problems. For the complexity of models needed for analysis of this data, MATLAB NN Toolbox, with Elman networks is a popular, widely available, well documented means of attacking similar problems and is packaged in a framework which provides the means to do the data cleansing and validation necessary for success.

**Data Selection**

Experience with many problems similar to the ones addressed here provided insight into the selection of the parameters necessary to construct appropriate models. However, the choice of the parameters to measure and record for any vehicle system is largely determined by the architecture of the control system. In order to provide stable control of a complex system, control theory provides guidelines for achieving controllability and robustness, meaning that the input-output variables of the controller can be expected to provide a “spanning set” of the information necessary for effective diagnostics. In some cases, information is redundant, and some reduction of

complexity for diagnostics can be achieved through more frugal monitoring strategies. However, it is our view that if all the I/O information from a controller is recorded, successful diagnostic strategies can be developed. Many problems can be solved with much less information, if the expertise is available to either decide *a priori* what should be monitored, or deduce from *a posteriori* analysis of all the recorded data, the minimum set required for specific tasks.

In general, since we must prepare for ANY problem that might arise, we have chosen to capture all the data available, to handle any contingency.

**Data Cleansing and Validation**

Data is often contaminated with errors introduced from many different sources, and the data with errors are captured in the time series information we encounter from real physical systems. Since our goal is to model as accurately as possible the behavior of these systems, we do not wish to include the anomalous information in building the model.

We can provide several examples to exhibit measurement data which has been contaminated by calibration adjustments inadvertently included in the data samples. When the original data is plotted and visually examined the outlier data distorts the scaling and obscures the relevant dynamics of the system. In these cases, the auto scaling features of the learning system can obscure information required for the proper "learning" of the dynamics since the data is scaled for analysis and the real dynamic range of the information is contained within 0-100 units rather than in -100,000 to 600,000 units range covered by the erroneous data. Since machine learning methods rely on error feedback, reducing the dynamic range of the observed signal to its actual range rather than the pathological factor of more than **10exp6** can dramatically improve the learning process.

After the flagrant outliers are removed, the data is used to produce a system model which is then retested on the same data. Since the NN schemes we rely on are parsimonious in the use of computational resources (number of internals nodes and layers), the networks do not make adjustments for small anomalies in the data. Consequently, data segments which do not fit the overall dynamics of the system do not match the models. A reexamination of the data to resolve these discrepancies is undertaken to determine if the anomalous data should be removed from the training or retained. Unless a good cause for removal is found, the data will be retained. This strategy can be used because the networks, even in the presence of low incidences of anomalous data still learn the overall true dynamics. This observation has been tested with analytic simulations of complex dynamics which are purposely contaminated with anomalies. The networks were observed to learn the true dynamics and the comparison of the model behavior and the generated data showed the areas where the contamination was introduced.

Typically, with our data in this project and across many other investigations, we have found that the error rates in the data bases to be generally small, with rates on the order of 1% or so, after the major outliers are removed. We have general chosen to remove the erroneous data for cause in order to improve the models, and to more easily separate true anomalies due to system failures from spurious anomalies due to measurement errors. We note however, that quite effective models and performance in model based reasoning can still be obtained even if the small anomalies are not excluded. We therefore recommend a very judicious use of data removal during the modeling-training phase of the process.

**Neural Network Synopsis**

The Time Lagged Recurrent Neural Networks (TLRNN's), which form the basis of our modeling efforts, are an extension of conventional static networks. The TLRNN's include time-lagged internal nodes connecting the neurons in the hidden layers to themselves and other nodes in the same layer. With this structure, the network develops a "memory" in that prior information circulates in these time-lagged loops until the information is no longer needed. In fact, during the training process, the networks learn how long to keep information in storage by themselves, without requiring that information to be provided by the user. Thus, they are able to account for past inputs and outputs so that they can properly deal with dynamic systems. The figure below describes a Multiple-Input Single-Output (MISO) form of these dynamic networks, with variable numbers of hidden layers and variable numbers of nodes in the hidden layers, so one may understand the difference between traditional static networks and the dynamic networks used here.

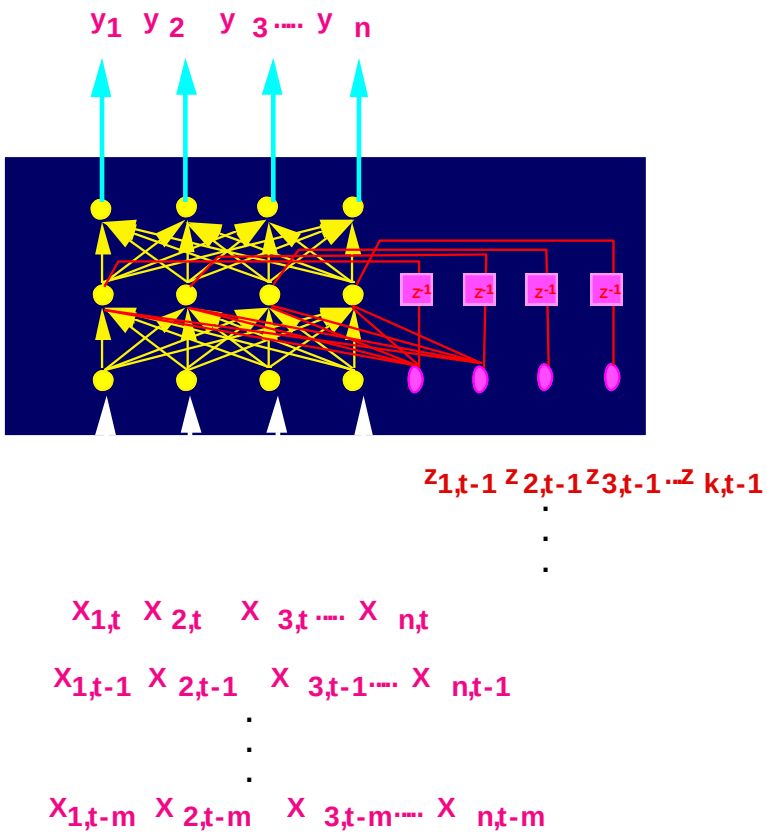



**Figure. 5.1 In this network, the nodes in the hidden layers are not only connected to the nodes in the following layer, but also to themselves through a one-step time delay. That time delay feeds back the prior value of the node(s) output(s) to the same hidden layer. This feature gives this form of network a "memory" of the past. Thus, the neural network output is a function of both the current inputs at time t, and past inputs at times t-1, t-2, t-3,…**

**The trainable weights in the network therefore are both the feedforward (instantaneous) weights and the recurrent weights (memory). Although the network structure provides a very powerful means of representing complex systems, its utility for modeling complex systems was limited due to the difficulty on training such a device by strategies that worked with static networks. The Kalman procedures have overcome that obstacle.**

## Neural Network Training Strategy

- Kalman Learning Algorithm adapted to train neural networks
    - Most powerful training algorithm currently known
- Multi-stream learning – trains on many operating conditions at the same time
- Data randomization – want networks to provide correct responses no matter how the system is operated.
- Comprehensive data set – provide examples of most operating conditions

- Automatic configuration of learning rates and neural network architecture for each problem. – Neural network solves the problem, not the user.

The prime ingredient in the success of our approach was the introduction of the Generalized Extended Kalman Filter (GEKF) in 1994, which removed unnecessary approximations from the computation. This improvement of course was not computationally practical until computer memory and speed had increased to be able to handle the complexity of the GEKF computation. That step, followed by several other important developments including specialized code for matrix operations, optimization of learning rates, and concurrent presentation of multiple dynamics for complex problems has produced the learning strategy used in our work.

It is important to note that the learning algorithm and the off-line training contain basically all the complexity of this entire process. However, these elements of the procedure are packaged in a software kit that handles virtually all the network set-up, preparation, training and testing without heavy reliance on end-user expertise for those purposes. This approach is far different from the standard AI machine learning packages which offer many types of networks and algorithms to end-user and rely heavily on user gained expertise for each problem to produce an optimal solution by properly choosing 10-20 parameters to obtain viable solutions.

When properly and completely trained, a dynamic neural network is a compact model of the complex system it was trained to emulate. As such, it can be operated in real-time, to produce to reproduce, almost exactly, the behavior of the real system. This behavior can then be utilized to construct model-based diagnostics as described below.

**Neural Network Training Tactics**

Once the overall strategy for training TLRNN's is understood, we must develop a practical means of implementing the training. The process is actually quite simple, and has several variations depending upon the outcomes desired.

We generally rely on "engineering judgment" of individuals working on specified systems to determine how much data should be acquired to provide a good representation of system behavior. The question of how much data is sufficient cannot usually be answered properly until we complete some initial studies with model building to determine how the model performance and the system data compare.

The training consists of taking a database, usually ranging in size from several thousands of data points to perhaps several million, cleansing the information as discussed above, and then compiling a scaled, normalized set of parameters for the inputs and outputs into a large data file. For our purposes, the data file consists of a large matrix whose rows contain all the inputs in the left-most columns, and the outputs in the right most columns. In general, any "clock" or indexing parameter which might

indicate the time or general order of the data is removed from the inputs, since we do not wish to have any spurious correlations of time and behavior which can be recognized and utilized.

The training consists of specifying the number of iterations of learning which will take place, the nature of the learning problem (classification or approximation), the number of independent streams to be used in each iteration and their length. The nature of the problem is the easiest to determine, since the choices are classification (for which we want the output to be one of two states, say 0 or 1 depending on the state of the inputs) or estimation (we want an analog value between 0 and 1 which represents the value of the output of the system). We have chosen to fix the number of data streams at 10, and leave the specification of the stream length to the operator. In general, it is necessary to have the stream length be long enough so that the dynamics of the system is represented within the chosen string length. For example, in a $4^{th}$ order system, inputs from 4 time intervals in the past can influence the current value, so the stream length should be at least 4. When the order of the system is unknown, an estimate must be made. In general, we operate with stream lengths between 10 and 20, which has proven sufficient for all the problems we have examined recently while still providing reasonable computational speed for the learning process.

Once the number of streams and stream lengths per iteration has been determined, the training software will extract from the database the following:

1. A number of start points equal to the number of strings selected.
2. A set of sequences beginning at each start point and continuing on for 1 string length.

   Thus, for a string number of 10 and a string length of 20, the program chooses 10 strings at 10 random start points in the data file, each of length 20. From the data matrix this means 10 groups of 20 rows of data. Consequently, one iteration of learning will provide 200 data points from the file for training.

In general, the number of training cycles should be large enough to allow the training to present all the data to the NN several times. So if we have N data points, and each iteration provides Y (200 in this case) data points, we would process N data points in Z iterations where Z is simply N/Y. Since the data selection is random, with the possibility of re-sampling a previously sampled data point, using precisely N/Y will not guarantee that all the data points are used. Consequently, the number of iterations is chosen to be about 3 to 5 times N/Y.

Training continues until the chosen number of iterations has been reached. For the real problems we have encountered, if the above guidelines are followed, the accuracy is at or near its asymptotic limit. We have constructed pathological cases of greater complexity for which further training is required, but that need has not been

manifest in our recent experimental data. If that situation should occur, the error measure of the fit, as evidenced by the RMSE of the residuals between the NN outputs and the Target Outputs can be monitored to be certain that the training is complete or nearly complete. For practical purposes, the accuracy of the models sought is an attainment of a model estimate which is differs from the true value by about 1%.

We can offer a great deal more insight into training methods and procedures for problems with widely different complexity, but that discussion is outside the scope of this report. When noise free simulations of complex systems are provided, model accuracies better than 1% are achievable, when proper attention is paid to the degree of training required, but since our data comes from production sensors with non-zero noise levels, that accuracy appears un-realizable with real data, and performance levels of about 1% have been obtained with cleansed, high-quality data for a variety of problems.

**Model Validation and Performance Analysis**

With extensive data sets, a traditional approach to model evaluation is to train to completion on a portion of a data set and test on an unseen "blind" set. In general, if the training is complete, and model generalization has been obtained, the model performance on the blind test is statistically equivalent to the training data. We have carried out blind testing to verify the performance of our trained models, and expanded the tests to include data which requires some degree of both interpolation and extrapolation by including analysis of data not represented in the training set.

In the process of model validation, it must be recognized that the techniques employed serve not only to validate the model, but also to determine the quality and sufficiency of the data. Exploratory analysis before data collection was complete was possible, and helped assure that the data was sufficient for our purposes and, indeed, of very high quality. This observation is not meant to imply that NN had a significant role in producing high quality data, but the NN analysis merely confirmed that the quality of the data was such that the NN estimate closely correlated the true value of the parameter to be estimated.

Prior to final blind testing, exploratory analysis and preliminary evaluations made with model building training were pursued. In these evaluations, the NN's were observed throughout the training process and it was noted that the behavior observed was consistent with performance expectations when training on "ideal" data sets. By "ideal" data sets we mean artificially constructed data sets which are accurate representation of complex simulation codes and which can be constructed, at will, to provide copious amounts of precise information, which can then be degraded with controllable Gaussian white noise. We assert, without detailed proof, except for reasonable expectation, that if the model evolution during training on real data mimics excellent performance on "ideal" data, then we can be confident that accurate models

are, in fact, being produced during training on real data. Consequently, during the acquisition and concurrent analysis of data, we may surmise that model development is proceeding as required, and that, in fact, before final tests (with blind data) unambiguously establish the final performance, that high performance with the data in hand will be obtained. The benefit of this tactic is that these methods frequently establish that planned data acquisition efforts often exceed the actual need for adequate information, and that some economies in testing costs can be achieved. We recognize that this observation may run counter to prior opinion about the need for voluminous data for neural network training, but recent work and development of novel training schemes suggests that these empirical model building efforts can be successful with far less data than earlier methods had required.

The validation process can be carried out in stages, since all the data necessary for final validation may not be available at the outset, and it is prudent to determine whether model building may indeed have a good chance of success as early in the process is possible. Thus we can break the evaluation and validation down into 6 steps, most of which can be completed with data readily available at some time in the development process.

1. Evaluation on the training set.
    a. During Training

    b. After Training

    c. This process is completed first in the development. During training the error reporting can be monitored, but only partially, since constant evaluation of performance on the entire dataset could dramatically slow down the training. However, the error measures on the data used for each iteration provide some insight into the training performance. If the error levels asymptotically reach acceptably (typically a few percent) low levels and do not exhibit large fluctuations, the training process is proceeding smoothly, and satisfactory results should invariably be obtained.

    d. After training, the error level on the entire dataset can be evaluated. If an accurate model for all the data has been obtained, the overall error level should be about the asymptote observed during training.

2. Evaluation on a Similar Sample of “Unseen” Data

   If additional data is available, or if portions of the original data can be spared from the training process, the second stage of evaluation can use data which is similar to but not identical to the data used for training. The goal is to establish that data from the same statistical distribution can be used to generate results which are similar to the results obtained on the training set. This step indicates that the data used for training is a satisfactory representation of a larger data set that can be drawn from the system under evaluation.

3. Evaluation on “Unseen” Data from a Different Period

   A further check is possible using data drawn from the system under investigation, but collected at a different time. Evaluation of performance on this data suggests that the data is valid for the system despite any changes that may have taken place in the time period between the original data extraction and the subsequent data collection. Similar performance indicates that aging is not a significant factor. Dissimilar performance may indicate that aging effects need to be considered, and that some incorporation of data from different epochs should be used in order to train a system to recognize the different performance levels of aging systems.

4. Evaluation of “Unseen” Data not contained in the Training Samples _ Interpolation

   This test is a much more severe test than any of the above. In this case, the performance is evaluated on operational data that is different than any used in training, requiring that the network learn to generalize or interpolate in order to handle circumstances which may not be covered in training. In general, passing this validation test should be a requirement in order to establish that the training data has been comprehensive enough to cover all expected operational modes. When performance on this validation exercise matches the levels seen in Steps 2 and 3, one may expect that the data acquired is an adequate representation of the dynamic performance of the system.

5. Evaluation of “Unseen” Data not contained in the Training Samples _ Extrapolation

   This evaluation utilizes data which is taken at operating conditions which are exterior to observed operating conditions and represent an extrapolation estimate from the empirical model. Again, this test is quite severe and often quite time-

consuming to perform. In general, for production systems, good design practice usually requires that models perform WITHIN observed parameter limits. For some of the models we have constructed, extrapolative analysis has shown that model accuracies are very good. However, it has been our strategy to avoid using this capability of the models by simply incorporating extrapolation data into an augmented training set.

6. General Testing on Very Large Samples of Novel Data

This final stage of evaluation is undertaken when the trained systems are actually deployed on production vehicles. Evaluation on these systems is impossible during development, since the production variability is usually unknown. Some evidence of likely performance can be obtained from simulation studies with developmental data to which artificial noise is added. Noise models may be available from prior experience with similar production systems in the past, but usually Gaussian white noise is used.

These evaluations can actually become the largest effort in the process when carried to completion, but they can be automated and distributed among many processors, unlike the scheme used during our development (two processors, non-automated, exploratory software). As noted previously, the model construction is usually straightforward and some simple DOE strategies can be used to optimize model performance for efficiency and accuracy. These evaluations are straightforward extensions of methods which are routinely incorporated in rigorous Design and Validation processes in the automotive industry.

## SUMMARY OF TRAINING AND TESTING

These processes are straightforward and can be adjusted to meet the needs established by requirements for individual problems. Residual noise in the estimations appears to be governed by the intrinsic noise from the production sensors and not by the model accuracy. This observation is based on experience with virtual sensors, which are produced by training the models to reproduce the signals of real sensors based upon redundant information from other sensors in the system. Such investigations revealed that the RMS noise from the virtual sensor and the real sensor were nearly identical, suggesting that the estimation performance was close to optimal. In the studies for this work, we attempted to provide performance levels capable of meeting both the needs for on-board diagnostics (real-time monitoring) and off-board analysis. The outline presented above pertains to the general procedures used throughout our work with both in-use field data and detailed laboratory measurements in

dynamometer cells. For specific performance details on the individual tasks undertaken, the results are detailed in other areas of this report. The ultimate performance levels attainable can be determined by a more exhaustive analysis of the data, supplemented by additional data from other production systems. Such an analysis is possible, but not warranted until performance standards and application scenarios are determined and accepted.

## Supplement

## Model-Based Diagnostics

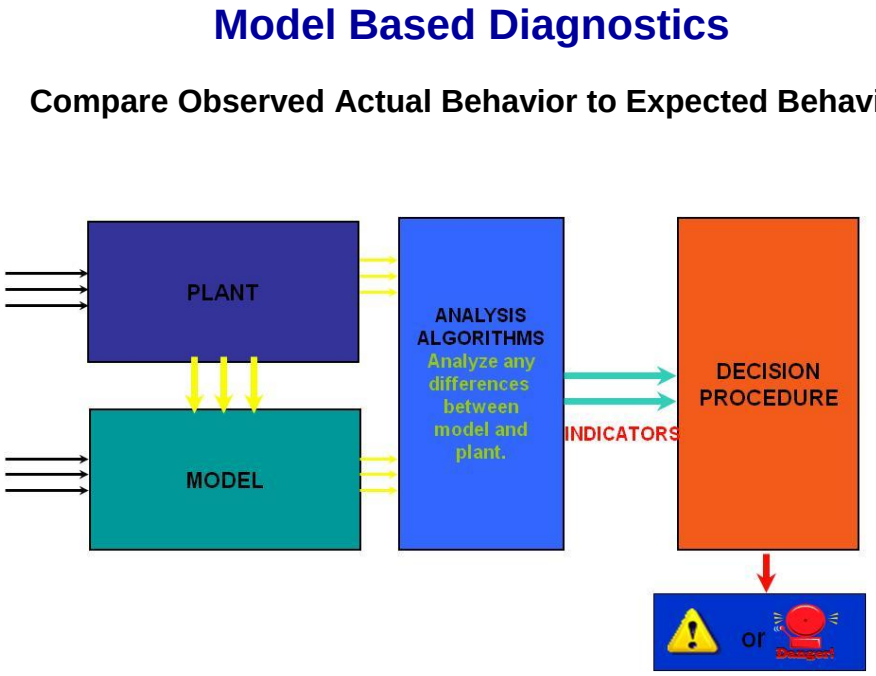



**Figure 5.2 A simple and reliable way to develop a diagnostic algorithm is to continuously compare the behavior of a real system to an accurate model of that system to see if the behavior of the system matches expectations. The expected system response is calculated concurrently on the basis of the trained neural network model. Diagnosis of anomalous behavior is based on devising some simple, or ultimately, optimal, strategies to compare the two data streams in order to reliably, regularly and efficiently provide recommendations (decisions) concerning system state-of health.**

Typically, models are developed as functional descriptor of the system based on a set of differential equations which calculates the system output from a set of measured inputs. These models are developed for specific systems and they need to be calibrated.

Another more powerful method is to build models by machine learning since this process does not require deep domain knowledge of the system response but only prior measurements of the system output for a broad range of inputs.

Neural networks are estimators of the system response but they are constructed using generalized procedures applicable to any type of system complexity. Since they are trained on a wide set of system operating conditions, these types of models are typically capable of describing the system independently of operator behavior and operating conditions. Additionally, they offer the opportunity to update the system knowledge with time, as more experience with real systems is gained.

Example: Fuel Flow Model

This model was constructed to estimate the actual Fuel Flow consumed regardless whether the engine operations were normal or abnormal (that is w/wo having perturbed the output of the Injection Fuel Pressure sensor.

Data inputs used for Fuel Flow prediction were:

- Engine Speed
- Load %
- Engine Oil Pressure
- Boost
- Injector Control Pressure
- Engine Coolant Temperature
- Intake Manifold Air Temperature
- Pedal %
- Desired Engine Speed
- Nominal Friction
- Load @ Speed

Eleven variables were used to predict the FUEL FLOW. None of the input variables had a simple correlation with the output target.

The neural network used was a TLRNN with two hidden layers between the 11 input nodes, and the single output node (Fuel Flow). The network was kept small to be certain that an efficient model was constructed, so that there were 23 nodes in the first layer and 11 nodes in the second layer.

The training was on a portion of the data with about half the information reserved for blind testing. The results are shown in Figure 5.3 in which the timeseries of the actual Fuel Flow and of the NN estimated fuel flow are overlaid

**Fuel Flow Model**

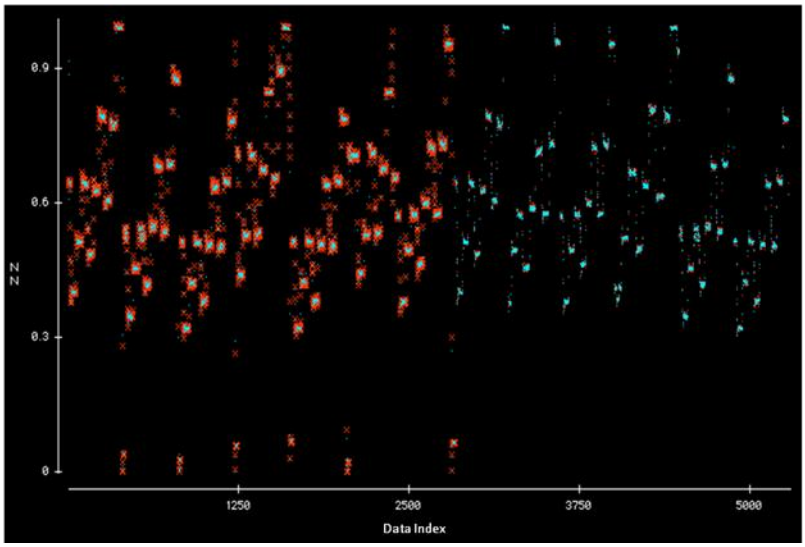



Fig. 5.3. Output of the Fuel Flow NN model (normalized in the 0-1 range). The measured data (blue dots) and the NN estimates (red dots) are overlaid. The red crosses indicate the subset of data used for training. The remaining data were used for model validation (blind set).

While many statistical tests are possible on the data to characterize performance, a simple statement of performance levels can be made in the context of the possible means such a model might be employed for diagnostics and prognostics. In an on-board application, a user might be interested in determining whether the power plant's fuel efficiency, at any operating condition, is within acceptable limits. Detailed examination of the data indicates that for observation periods of about 10-20 seconds, the measured fuel rate, and the estimated fuel rate on the blind test data agree to better than 1%.

# Section 6

## Virtual Torque Sensing

## Virtual Torque Sensing

### Introduction

Current electronically controlled diesel engines do not use an engine mounted sensor for determining the actual torque that an engine is producing. Physical torque sensors are expensive and not yet reliable for production applications and today's technology is mainly applicable for testing purposes rather than field use. The generation of appropriate torque output consistent with the driver demand, perceived as a pedal input, is the basis of the architecture of newer control strategies. The ECU performs complex calculations to command the amount of fuel to be delivered to the cylinders, adjust the injection timing and the boosting action so that engine torque output responds accordingly to the driver command, especially smoothing out torque transients. However, the torque output calculation carried out by the ECU does not necessarily provide an accurate estimate of the actual torque the engine produces, especially under transient conditions during which torque control can be very complex because of the dynamics of the systems involved in adjusting fueling and air induction. Furthermore, if any device in the control system deteriorates with time and shifts its operating behavior more than what is expected as normal aging, the torque output may deteriorate without being diagnosed.

Virtual Torque Sensing (VTS) is a software alternative to detecting actual brake torque with a minimum investment in additional hardware (that is, not relying on new sensing technologies) at the expense of adding computational burden which can be either managed within the ECU or by another module interfacing with the ECU. The most promising VTS implementation is based on accurate engine speed measurements which are the input to a Neural Network model that calculates actual torque. Figure 6.1 schematically illustrates this concept.

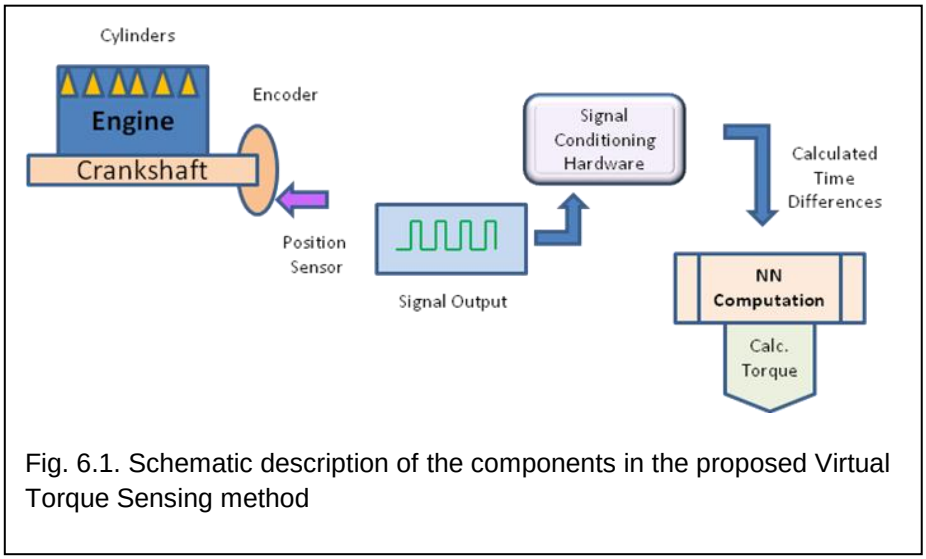



Fig. 6.1. Schematic description of the components in the proposed Virtual Torque Sensing method

VTS is, thus, an empirical method based on inferring engine torque relying on other type of measurements potentially already available from existing sensors (such as a crankshaft timing device) and a Neural Network processor to carry out the calculation. As schematically shown in Figure 6.1, it combines a Position Sensor and a wheel with constant angle marks (Encoder) for generating a periodic signal. The time elapsed between adjacent marks is extracted by a Signal Conditioning module which provides the input to the NN processor. The timing device could be either the same sensor/signal extraction electronics already existing in the production injection time control system, or an improved device/signal conditioning mounted in a location optimized for accurate speed measurements. Timing sensors are much more robust and cost effective than Torque sensors and have been used in productions for decades. If the production timing sensors can be used, VTS would only require software development to embed the NN model in the ECU.

The benefits of this technique would include:

- Minimum hardware and wiring modifications
- Improved engine response
- Increased fuel efficiency
- Increased transmission shifting efficiency
- Ability to determine powertrain health/reliability

The derivation of Torque output from Engine Speed measurements is based on the fact that the crankshaft does not rotate at constant velocity, but it accelerates during each combustion event and decelerates in between because of engine friction and the external resistance. As an example, in a six cylinder engine three combustion events occur per crankshaft rotation, thus the crankshaft oscillates at a main frequency corresponding to three times the engine speed as seen in plots in Section 4. The Torque output measured with a frequency torque sensor is also seen to oscillate in a corresponding fashion. However, developing a mechanical model that captures accurately the functional dependence of the speed variations as a function of torque is very complex because of the inertial characteristics of the engine. On the other hand, a pattern recognition approach based on machine learning can be used to link speed fluctuation features to torque. The section below explains the Neural Network model developed to infer mean brake torque from timing data and torque information. The data used to develop and test the Neural Network model were collected during the experiments described in earlier Sections. Time differences were then calculated from the raw timing data according to the procedures described in Section 5 and constitute a generalized input for developing different Neural Networks models.

**Model Inputs.**

The input measurements, in the form of time differences corresponding to the time taken by the crankshaft to rotate by a constant angle, are first converted into a network input vector which is constructed from a sequence of four phased summations, as detailed below, plus a rolling average. The network thus has five inputs. For camshaft data, time intervals are combined in pairs, while for encoder data three consecutive intervals are combined. This difference reflects the respective number of teeth encoded per revolution. In this set-up 24 pulses were encoded for cam data and 36 pulses from the encoder data. Each pulse or PIP interval contributes to every input vector. Consecutive input vectors correspond to the times that are 8 (cam) or 12 (encoder) teeth apart, so that in each case the network is three times per revolution.

For initial explorations, each phased summation included contributions from three consecutive PIP intervals. This was later revised for cam data to include contributions from only the current PIP and the next previous PIP. This avoids mixing in the same summation time interval values from combustion events associated with cylinders 1-2-3 and 4-5-6 which feed separately into the turbo (combustion order is 1-4-3-6-2-5 in this engine) with slightly different pressure values. Without attempting to justify the choice statistically, we merely note that the performance with the revised form appears to be at least as good as that of the original form.

The network input vectors are constructed as follow. As noted above, the basic clock unit is the timestamp interval, which we take to be the time between falling edges of the encoding device. In a six-cylinder engine, one PIP interval then consists of eight tooth intervals for cam data and 12 intervals for encoder data. Let us define t(0) to be the time stamp interval at a specified moment, and t(m) to be the time interval m intervals in the past. For cam data, the first four network inputs are constructed as follows:

$$in(0) = (t(0)+t(1)+ t(16)+t(17))/4 - tbar \quad (6.1)$$
$$in(1) = (t(2)+t(3)+ t(18)+t(19))/4 - tbar \quad (6.2)$$
$$in(2) = (t(4)+t(5)+ t(20)+t(21))/4 - tbar \quad (6.3)$$
$$in(3) = (t(6)+t(7)+ t(22)+t(23))/4 - tbar \quad (6.4)$$
$$in(4) = tbar \quad (6.5)$$

where tbar is an average over a suitably long interval

$$tbar = [t(0)+t(1)+\ldots+t(14)+t(15)]/16 \quad (6.6)$$

It should be noted that all input vectors have the same phase with respect to a fixed position in the cam rotation. In practice, this means that all teeth that represent time interval t(0) have the same angular distance, modulo 8 teeth, from a chosen fixed

reference position. It is quite likely that this condition is crucial.

For encoder data, the first four network inputs are

$$in(0) = (t(0)+t(1)+t(2)+ t(12)+t(13)+t(14)+t(24)+t(25)+t(26))/9 - tbar \quad (6.7)$$
$$in(1) = (t(3)+t(4)+t(5)+ t(13)+t(14)+t(15)+t(27)+t(28)+t(29))/9 - tbar \quad (6.8)$$
$$in(2) = (t(6)+t(7)+t(8)+ t(16)+t(17)+t(18)+t(30)+t(31)+t(32))/9 - tbar \quad (6.9)$$
$$in(3) = (t(9)+t(10)+t(11)+t(19)+t(20)+t(21)+t(33)+t(34)+t(35))/9 – tbar \quad (6.10)$$
$$in(4) = tbar \quad (6.11)$$

where $$tbar = [t(0)+t(1)+…+t(14)+t(15)]/16 \quad (6.12)$$

We investigated a small variation of the base inputs set, in which the average time interval was replaced by a scaled reciprocal of the interval. This replacement for input in(4) is equivalent to a speed rather than to a time interval. This change does not appear to have a significant effect.

The torque measurement contains a large amount of high frequency fluctuation, some of which is true noise but which contains real variation of torque related to the physical process of torque production. In the present case, we are not concerned with estimating torque at this time scale. Consequently, the measured torque values are smoothed somewhat prior to being used as targets for the network output.

**Neural Network Architecture and Training:**

The goal is to select a neural network architecture that can transform the stream of inputs derived from time intervals, such as those presented above, into a stream of torque values that approximate as closely as possible the stream of measured values. On the basis of previous work, the input set we have chosen together with the network architecture about to be described were expected to be capable of performing this task with reasonable accuracy.

We use the class of neural networks called time-lagged layered recurrent networks (also called dynamic networks, dynamic recurrent networks, and recurrent multilayer perceptrons). Networks of this class are capable of being trained to represent a wide range of dynamical systems, which then can be used as models, controllers, estimators, predictors, etc. The key to practical use of such networks is an effective training procedure. The procedure used here is described in considerable detail in IEEE publications coauthors by Feldkamp and Puskorius. That paper contains the recommendation that such networks be described for computational purposes as an ordered sequence of simple mathematical operations. This description, which does not

seem to be widely appreciated, gives rise to a rather simple way of executing a network of arbitrary complexity in a computer program and has served as the basis for embedding recurrent networks in high volume consumer products. The ordered network description is also advantageous when programming a training procedure.

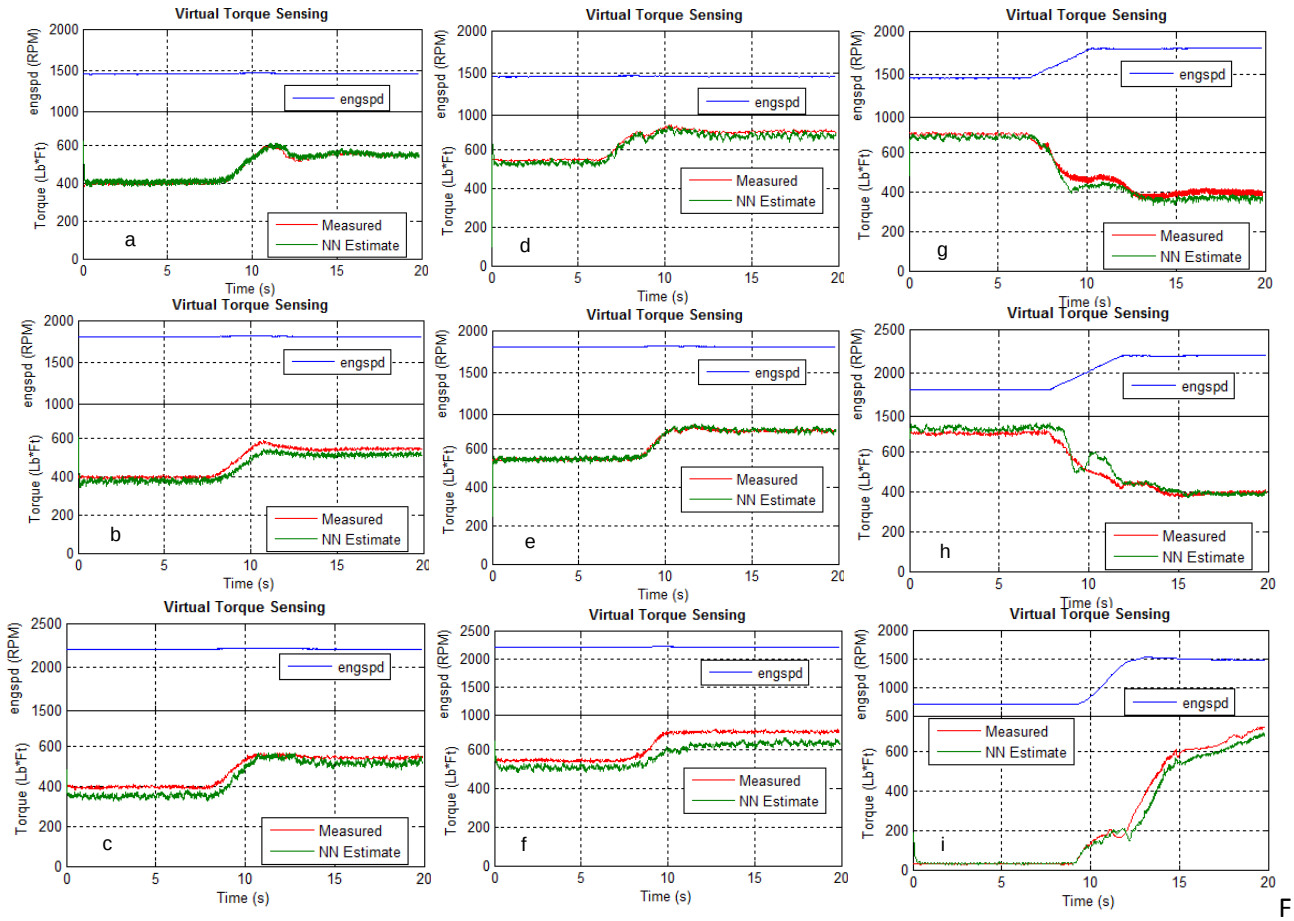
For this application, the detailed network architecture does not seem to be crucial, as long as the number of parameters does not encourage the training process to over fit the training data. For practical reasons, the amount of variation present in the data collected for this application was not as large as desired (though it seldom is!) and over fitting must not be disregarded. A fair amount of experimentation was performed to guide the choice of a specific architecture that would be capable of the estimation task while not being so flexible that fitting noise present in the data is easily done. If network resources are wasted on noise generated artifacts present in the data, generalization (roughly speaking, the ability of a parameterized model to perform well on data to which it was not exposed during training) will be compromised. In practice, this means that a more flexible network may well exhibit poorer generalization than a simpler network, even though its performance on the training data is noticeably superior.

A representative network for this application may be described with the notation 5-5r-3r-1L, i.e., 5 inputs, a recurrent layer of 5 fully interconnected nodes, a recurrent layer of 3 fully interconnected nodes, and a single linear output node. There are 86 weights. In this architecture, connections within a given layer have a one-time step delay, while connections from a layer to the next contain no delay. A plausible argument for the efficacy of recurrent networks is that the presence of computational layers of nonlinear nodes supplies the ability to fit a wide range of nonlinear functions, while the presence of distributed time delays supplies the ability to model dynamical functions. (It should be noted that the network description mentioned above is far more general than is required for such layered networks but exacts a very small computational premium.)

## Virtual Torque Model Quality

Figure 6.2 illustrates how closely the NN model based on the Encoder data estimates the actual torque output when the engine is stepped from one operating point in the MiniMap to the next one. The figure is a composite of nine graphs, each showing the NN estimated torque as a function of time and the corresponding measured torque trace. Each graph covers a 20 s window roughly centered on a torque transition since the input data to the model were derived from the Snapshot Timing data which last 20 s. These nine Torque changes illustrate different types of transitions between speed/load operating points in the MiniMap sequence (i.e., speed/torque control) according to which the experiments were carried out. Some of the plots refer to torque changes occurring at constant engine speed, others occurring with a concurrent speed change. For clarity, the engine speed traces are also shown in the graphs.

The Torque data (both measured and estimated) in the plots have averages over 12 engine cycles to filter fluctuations associated with combustion and noise. Ten training




ig. 6.2. Measured and NN estimated Torque traces for nine different types of Torque transition during a MiniMap with no Seeded Fault.

sessions were carried out to calculate the Network weigthts using 3 sets of snapshots acquired on different days in tests with no Seeded Fault. The estimated values shown in Figure 6.2 derive from the training session that produced the median result (or 5$^{th}$ best fit) in terms of RMS deviation between estimated and measured value. Notice that the results in Fig. 6.2 (a), (b), (d) and (e) show that the NN model reproduces closely the behavior of positive torque changes at constant speed of 1450, 1800 and 2200 RPM, and gives a good estimate of the torque equilibration value. However, the model seems to underestimate torque at the highest value at 2200 RPM. The model is not reproducing features occurring during negative torque changes possibly because the dyno controller dynamics are complicated by the concurrent engine speed change (the measured Torque trace shows a double transition). It is possible that training relying on additional data sets may improve the model and the observed large oscillations, not seen in the measured data, may be filtered out. The behavior of the Torque transition from Idle to 700 Ft*Lb/1450 RPM appears to be reproduced more closely than the downward transition.

Figure 6.3 shows similar results for the Model derived for the CAM2 data. The agreement between estimated and measured Torque is affected by unfiltered oscillations in the model output not observed in the Encoder-based model. Notice, however, that the Torque mean value is still qualitatively good. The oscillatory behavior in this model may be related to the higher noise observed in the time differences derived from the CAM2 sensors. The input data may be noisier because of the extra signal conditioning required in transforming the analog VRS signal into TTL, potentially introducing more “jitter” in the signal falling edges that are monitored. It is also possible that the noise is related to the gear-train that links the crankshaft to the camshaft.

**Model based on the CAM2 Data**

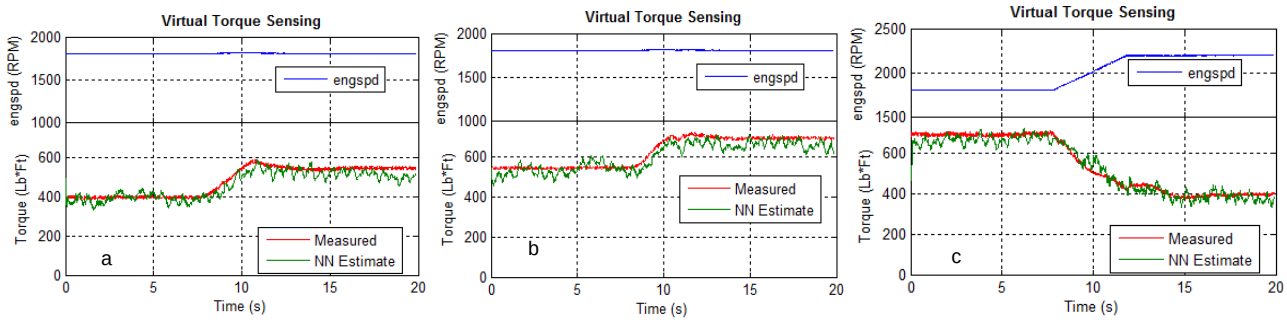



Fig. 6.3. Measured and NN estimated Torque traces for three of the same Torque transition illustrated in the plots of Fig. 6.2.

| Transition | | Error | |
|---|---|---|---|
| From | To | ENC | CAM2 |
| (RPM/Lb*Ft) | (RPM/Lb*Ft) | (Lb*Ft) | (Lb*Ft) |
| Idle | 1450/700 | 36.8 | 34.4 |
| 1450/400 | 1450/550 | 17.9 | 28.4 |
| 1450/550 | 1450/700 | 31.8 | 37.8 |
| 1450/700 | 1800/400 | 37.5 | 47.9 |
| 1800/400 | 1800/550 | 18.3 | 27.9 |
| 1800/550 | 1800/700 | 28.9 | 34.3 |
| 1800/700 | 2200/400 | 40.5 | 42.8 |
| 2200/400 | 2200/550 | 20.3 | 26.3 |
| 2200/550 | 2200/700 | 26.7 | 30.4 |
| 2200/700 | 1450/300 | 47.0 | 46.8 |
| 1450/300 | 1800/300 | 17.1 | 37.9 |
| 1800/300 | 2200/300 | 33.1 | 35.2 |
| 2200/300 | Idle | 95.3 | 44.2 |

Table 6.1. The standard deviation of the difference between measured and estimated torque for each snapshot is shown as a way of quantifying the model accuracy.

A measure of the quality of the NN Torque estimate can be obtained by calculating the standard deviation between measured and estimated values for each snapshot (Error). Table 6.1 shows these values for thirteen transitions in the MiniMap. The Error is roughly x1.5 larger for the CAM2 derived model, most likely due to the unfiltered oscillations. Since this is a cumulative measure of error over the snapshot, it is not expected to scale with the average Torque value because it could be biased by the largest deviations observed during the transition. Nevertheless, even considering that this may represent a worst case Error, the % error of Encoder-based NN model defined as the ratio of the Error over the Mean Torque ranges roughly between 5% and 10%, indicating that this method of Virtual Torque Sensing could be applicable to diagnosing engine performance changes at relatively steady-state of more than 10%, especially if averaging over different conditions can be done. The model does not identify the root cause for the performance change would but can issue a warning that the engine needs to be checked in the shop.

# Section 7

## Conclusions

**Conclusions:**

MIS2000 believes that a model based reasoning approach, using dynamic neural networks to detect anomalies in system performance, will provide the Army with a sound solution for implementing CBM for military vehicles. A major benefit is the fast and efficient way of utilizing real time vehicle data to assess vehicle health. Since these algorithms are efficient, they do not necessitate extensive computing power and require a minimum hardware footprint to run.

The constraints of this approach are related to the data being fed into the model. The fact remains that good decisions cannot be made using bad or incomplete information. Therefore, an open systems approach will be used to allow for maximum integration of additional sensor data as new technology becomes available. This information may be generated from virtual information i.e. Virtual Torque Sensors, or from additional external devices to augment real-time information acquired from the system via J-1939, Controller Area Network (CAN) bus or wireless interfaces.

The results from the testing related to the virtual torque sensor shows potential. As was discussed earlier, current electronically controlled diesel engines do not use an engine mounted sensor for determining the actual torque that an engine is producing. Physical torque sensors are expensive addition to an engine and why they are usually only used for testing purposes. Instead torque is calculated by throttle position and certain engine operating parameters. The calculation method does not necessarily provide an accurate measure of the torque that the engine is producing. This method of torque calculation does not take into account if there is a problem developing in the engine. Current diagnostics on diesel engines do not flag issues until there are operational problems with the engine or major component faults.

The next step would be to experiment with the virtual torque sensing on an actual vehicle. Virtual torque sensing could be an extremely important piece in the CBM puzzle. With the ever increasing cost of fuel virtual torque sensing could offer a low cost solution to provide for a more efficient operation of military vehicles. Ultimately leading to a savings in fuel consumption and vehicle maintenance costs.

In conjunction with the testing of the virtual torque sensing on a vehicle platform the other neural networks developed for this project could also be implemented on a vehicle platform.