FINAL PROJECT REPORT

NSF GRANT NUMBER ECS-8821340
Dear Dr. Hazelrigg,

Enclosed is the completed text of the final project report for our NSF grant, to accompany NSF Form 98a which was previously mailed. The report provides complete details of all work on the project, and presents our results.

We are very grateful for the support of the NSF for this work. In addition to any possible contribution made to the body of knowledge in the experimental area, four undergraduate students and one graduate student gained valuable research experience.

Your feedback or questions about the report or the project are welcome. (805 756-2317, 2781 or 5061).

Respectfully submitted,

Carl A. MacCarley, Ph.D., PE.
Associate Professor and Principal Investigator

encl.
Carl A. MacCarley
Electronic & Electrical Engineering
Cal Poly St University Fdn
San Luis Obispo  CA  93401
NSF Grant Conditions (Article 17, GC-1, and Article 9, FDP-II) require submission of a Final Project Report (NSF Form 98A) to the NSF program officer no later than 90 days after the expiration of the award. Final Project Reports for expired awards must be received before new awards can be made (NSF Grants Policy Manual Section 677).

Below, or on a separate page, provide a summary of the completed projects and technical information and attach it to this form. Be sure to include your name and award number on each separate page. See below for more instructions.

**PART II - SUMMARY OF COMPLETED PROJECT (for public use)**

The summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the project title, it should begin with a topic sentence starting the project's major thesis. The summary should include, if pertinent to the project being described, the following items:

1. The visible smoke emissions typical of diesel automobiles during acceleration are a direct result of inaccurate fuel quantity control. A significant improvement in accuracy could be achieved if it were possible to sense the actual amount of fuel being injected into the diesel engine, and control the fuel metering based upon feedback of this information. No suitable sensor has been found which is capable of directly measuring the instantaneous fuel delivery.

2. The objective of the project was the development and experimental assessment of a new sensing technique for diesel fuel injection flow. The technique involves the real-time analysis of the motion of the needle valve in the fuel injection nozzle to indirectly sense fuel flow through the nozzle. A needle position sensor, fuel temperature sensor, and a sensor processing computer are used to generate an estimate of the injected fuel quantity, which may be used by the main engine control computer for closed-loop fuel metering.

3. Algorithms were developed for the required signal analysis task, and the technique was tested on a laboratory apparatus and on a test engine. The technique showed good accuracy on the laboratory apparatus. Absolute accuracy during preliminary engine tests was poor, although relative accuracy and repeatability were good. The need for improved algorithms based on more sophisticated injector models is indicated.

4. Closed-loop fuel control employing this method has the potential to improve fuel control accuracy, and thereby reduce exhaust smoke emissions and increasing fuel efficiency.

**PART III - TECHNICAL INFORMATION (for program management use)**

List references to publications resulting from this award and briefly describe primary data, samples, physical collections, inventions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available to the research community.

The complete project report is attached, describing in detail the data, experimental procedures, and results of the research. No patents were filed for inventions derived from this work.

Data and results from this project have been published by the Society of Automotive Engineers, at the 1990 SAE International Congress, Detroit, MI. Paper Number: 900494, "An Indirect Sensing Technique for Closed-Loop Diesel Fuel Quantity Control". This paper also appears in the 1990 SAE Transactions. Other publications based on this research are in progress.

---

**IMPORTANT: MAILING INSTRUCTIONS**

Return this Entire packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the Attention line on the envelope.
The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant. Do not enter information for individuals working less than 40 hours in any calendar year.

<table>
<thead>
<tr>
<th></th>
<th>Senior Staff</th>
<th>Post-Doctorals</th>
<th>Graduate Students</th>
<th>Under-Graduates</th>
<th>Other Participants(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Total, U.S. Citizens</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B. Total, Permanent Residents</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U.S. Citizens or Permanent Residents(^2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>American Indian or Alaskan Native</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Black, Not of Hispanic Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hispanic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pacific Islander</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>White, Not of Hispanic Origin</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>C. Total, Other Non-U.S. Citizens</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify Country</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D. Total, All participants(^\text{A+B+C})</strong></td>
<td>3</td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Disabled(^3)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

\(^1\) Category includes, for example, college and precollege teachers, conference and workshop participants.

\(^2\) Use the category that best describes the ethnic/racial status for all U.S. Citizens and Non-citizens with Permanent Residency. (If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)

\(^3\) A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. (Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens").

**AMERICAN INDIAN OR ALASKAN NATIVE:** A person having origins in any of the original peoples of North America, and who maintain cultural identification through tribal affiliation or community recognition.

**ASIAN:** A person having origins in any of the original peoples of East Asia, Southeast Asia and the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

**BLACK, NOT OF HISPANIC ORIGIN:** A person having origins in any of the black racial groups of Africa.

**HISPANIC:** A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

**PACIFIC ISLANDER:** A person having origins in any of the original peoples of Hawaii; the U.S. Pacific Territories of Guam, American Samoa, or the Northern Marianas; the U.S. Trust Territory of Palau; the islands of Micronesia or Melanesia; or the Philippines.

**WHITE, NOT OF HISPANIC ORIGIN:** A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

 THIS PART WILL BE PHYSICALLY SEPARATED FROM THE FINAL PROJECT REPORT AND USED AS A COMPUTER SOURCE DOCUMENT. DO NOT DUPLICATE IT ON THE REVERSE OF ANY OTHER PART OF THE FINAL REPORT.

NSF Form 98A (Rev. 5/90)
Final Project Report

The Development of a Sensing and Signal Processing Technique to Facilitate Closed-Loop Fuel Injection Control for Diesel Engines

December 15, 1990

NSF Grant Number ECS-8821340

Carl A. MacCarley, Principle Investigator
Electronic and Electrical Engineering Department
California State Polytechnic University
San Luis Obispo, California
# TABLE OF CONTENTS

1. EXECUTIVE SUMMARY ................................................................................. 1
2. INTRODUCTION ............................................................................................. 2
3. BACKGROUND AND PROBLEM DESCRIPTION ............................................ 2
4. EXPERIMENTAL METHOD ............................................................................. 5
5. SPECIFIC PROJECT TASKS ............................................................................ 6
6. PRELIMINARY SIMULATION STUDIES .......................................................... 7
7. CONSTRUCTION OF A REFERENCE RATE MEASUREMENT APPARATUS ...... 10
8. DEVELOPMENT OF HYDROMECHANICAL MODELS FOR INJECTORS .......... 12
   8.1 Static Nonlinear Model ........................................................................ 13
   8.2 Dynamic Models ................................................................................ 16
9. DESIGN OF THE SENSOR PROCESSOR AND INTERFACE CONTROLLER .... 18
   9.1 Signal Processing Requirements ............................................................ 18
   9.2 Hardware Design Considerations .......................................................... 19
      9.2.1 Microprocessor Selection ................................................................. 19
      9.2.2 A/D Conversion ............................................................................. 20
      9.2.3 Memory ........................................................................................ 20
      9.2.4 Interface and Timing Support .......................................................... 21
      9.2.5 Additional Design Issues for a Multi-processor Engine Controller ... 21
         9.2.5.1 Task Assignment ..................................................................... 22
         9.2.5.2 Architectural Relationship ....................................................... 22
         9.2.5.3 Interprocessor Communications .............................................. 22
10. OPERATION OF THE PROTOTYPE SENSOR PROCESSOR ........................... 23
11. INTERFACE/CONTROL COMPUTER OPERATION ......................................... 24
1. EXECUTIVE SUMMARY

The visible smoke emissions typical of diesel automobiles during acceleration are a direct result of inaccurate fuel quantity control. A significant improvement in accuracy could be achieved if it were possible to sense the actual amount of fuel being injected into the diesel engine, and control the fuel metering based upon feedback of this information. No suitable sensor has been found which is capable of directly measuring the instantaneous fuel delivery.

The objective of the project was the development and experimental assessment of a new sensing technique for diesel fuel injection flow. The technique involves the real-time analysis of the motion of the needle valve in the fuel injection nozzle to indirectly sense fuel flow through the nozzle. A needle position sensor, fuel temperature sensor, and a sensor processing computer are used to generate an estimate of the injected fuel quantity, which may be used by the main engine control computer for closed-loop fuel metering.

Algorithms were developed for the required signal analysis task, and the technique was tested on a laboratory apparatus and on a test engine. The technique showed good accuracy on the laboratory apparatus. Absolute accuracy during preliminary engine tests was poor, although relative accuracy and repeatability were good. The need for improved algorithms based on more sophisticated injector models is indicated.

Closed-loop fuel control employing this method has the potential to improve fuel control accuracy, and thereby reduce exhaust smoke emissions and increasing fuel efficiency.
2. INTRODUCTION

Despite significant advances in electronic control technology applied to diesel engines, commercially available injection systems for automotive diesel engines remain limited by the open-loop mapping of the injection pump. An initial calibration is relied upon to translate a fuel delivery command to an actual fuel quantity. In practice, however, these two variables may be substantially different due to the effects of mechanical wear, repair, and the wide range of operating conditions. Possible ramifications of this discrepancy are excessive exhaust smoke due to overfueling, increased fuel consumption, degraded drivability, and poor idle characteristics. The major obstacle to closing the fuel control loop is the lack of a suitable sensor for instantaneous fuel delivery from the injector.

An indirect fuel delivery sensing mechanism based upon the use of the injector needle lift in conjunction with the fuel temperature was developed and evaluated. An estimation of the injection rate characteristic is determined from real-time analysis of the needle lift signal using a high-speed sensor processor. Integration of the rate characteristic followed by temperature correction yields a total mass delivery estimate for use as a feedback quantity for closed-loop fuel control. Signal processing algorithms are derived from computer models of the injector and verified experimentally. The method is then evaluated for its emission-reduction potential on a dynamometer-mounted test engine. Possible long-term degradation in accuracy due to nozzle coking is studied. Advantages and limitations of the technique are identified.

A description of this work and details of this indirect sensing technique were published at the 1990 Society of Automotive Engineers International Congress, in Detroit, MI [MacCaryley90]. This paper is included in Appendix 1 of this report.

3. BACKGROUND AND PROBLEM DESCRIPTION

Although diesel engines and injection systems represent a mature technology, it is only in recent years that electronic controls have been successfully applied. The majority of diesel applications are still mechanically controlled, with no electronics involved other than the fuel shutoff solenoid valve control.

The potential benefits of microprocessor-based control applied to diesel engines have been well established [Reams82, Kihara83, Martinsons82, Trenne82, Kawai84]. However, many of the improvements made possible by advanced electronic control are dependent upon exact knowledge by the controller of the hydraulic characteristics of the injection system components.

This dependency is particularly important in the case of small-displacement automotive diesel engines, which use low-cost distributor pumps which must accurately meter very small (less than 50 mm³ per injection) fuel quantities at high speeds.
Major incentives for more accurate fuel control have appeared in the form of recent regulatory pressures for "cleaner" diesels along with the demands of the automotive market for driving characteristics more like those of gasoline fueled engines. Increasing concerns about rising gasoline costs may be expected to produce a renewed interest in automotive diesels.

Available and currently envisioned electronically controlled distributor-type fuel injection pumps for automotive diesel engines generally operate with a map-based translation between commanded fuel quantity and actual fuel delivery. Typically, a ROM-stored multidimensional map is accessed with inputs which include commanded fuel volume and pump speed. Fuel temperature may be used to modify the table output to yield a corrected fuel sleeve or rack position corresponding to a given commanded fuel mass. The map and correction factor(s) are generated experimentally from pump test data, typically using a reference pump. A fully specified map of adequate resolution and valid correction factors may require the acquisition of a large number of data points on a pump test stand.

There are several limitations of this open loop fuel control method. The generation of individual calibration maps for each pump, and subsequent storage of individual maps in ROM, is impractical in production. Usually, the open-loop maps are based upon data collected from a single reference pump. At best, a linear correction for pump misalignment is performed during final checkout of individual production pumps, using either an external resistor network [Stumpp83] or final PROM programming procedure. However, minor machining differences, even within manufacturing tolerances, can cause noticeable differences between the calibration maps of individual production pumps and the test pump. This difference is compounded by the synergistic relationship between the injectors and the pump. The injectors fitted to a particular production pump may differ slightly in their flow characteristics from those used with the test pump during the master calibration, thus changing the overall calibration. One or more injectors might also be replaced or readjusted at a later date.

Possibly more important is the problem that the injection system often operates under conditions much different than those that existed during the master calibration tests. The calibration also changes over the course of time due to normal wear and corrosion of pump and injector internal components, and the accumulation of carbon deposits in the injector nozzle (nozzle coking).

In actual service, the delivered fuel quantity may differ substantially from the mapped quantity. Possible effects of this discrepancy are excessive exhaust smoke due to overfueling, inaccurate torque limiting, increased fuel consumption, degraded drivability, and unstable or noisy idle characteristics.

If the actual fuel delivery per injection were directly sensible, closed-loop control of the fuel quantity would be possible as a means for improving the fuel control accuracy without the need for further mechanical refinements and tighter tolerances in the pump. The major obstacle to closing the fuel control loop appears to be the lack of a suitable sensor for the fuel quantity.
The need for quantity measurement on an individual injection basis is a matter of some debate. If injection-to-injection differences are not significant, it probably is not necessary to determine the delivered fuel quantity on an individual injection basis. Averaging-type flow sensors might be adequate for the task. We know of no production averaging-type flow sensors, but at least one concept for such a sensor has been suggested in the literature (Challen88). However, real-time control usually is most effective if the most current information obtainable is used for the feedback quantities. Furthermore, it is desirable to also have available information on the injection timing and rate characteristics, which can only be determined on an individual injection basis. Therefore, there may be considerable advantage to a sensor capable of resolving the delivered fuel quantity and rate characteristic on an individual injection basis.

Real-time monitoring of the actual fuel delivery (per individual injection or average flow) is difficult. A number of factors may be cited in relation to this technical obstacle. For a typical small displacement (i.e., under two liter) diesel engine, the fuel delivery volume is very small (in the range of from 5 to 50 mm$^3$ per stroke), and the duration very brief (on the order of 1.0 ms). The repetition rate per cylinder is typically from 5 to 50 injections per second, with line pressure fluctuating from the delivery valve opening pressure to the injection peak pressure (as high as 100 MPa) at this repetition rate. The static volume in each fuel injection line and the secondary passages of the pump may exceed the fuel delivery per injection by a large factor. Fluid compressibility, inertial effects, tubing strain, and internal leakage in the pump make the process non-ideal, so that the actual fuel delivery often differs significantly from the metered plunger displacement in the pump. Mass transport in this medium is characterized by propagation of a pressure wave between the pumping chamber and the injector nozzle. Computer simulation of the injection pump hydraulics using finite difference methods is often relied upon to predict the delivered quantity and injection characteristics [Oren83, Kumar83, Sharma83].

While suitable sensors for the rate characteristic have been suggested [Bosch66, Komaroff66, Thoma74] for use in test bench calibration of pumps, a practical sensor suitable for use during actual engine operation as a real-time feedback control device is not, to the best of our knowledge, currently available. Methods utilizing injection pulse duration in conjunction with engine speed to estimate fuel usage have been investigated [Wolff86]. And as previously mentioned, a thermal convection based averaging fuel flow sensor has also been suggested [Challen88].

The advantages of closed-loop control in general are well established. Efforts to close the control loop on engine torque [Ribbens81, Fleming82, Sood84], combustion luminescence [Bunting84], and cylinder pressure [Challen88] have been important recent contributions to diesel control technology. All of these techniques may be considered as indirect indicators of the injected fuel quantity, which for some performance metrics (i.e., emissions) is the target variable of primary importance.
4. EXPERIMENTAL METHOD

The reported work investigated the feasibility of using information contained in the motion of the injector needle in combination with other sensor signals, to infer the delivered fuel quantity, rate characteristic, and timing of critical portions of the delivery schedule on an individual injection basis. It was hypothesized that the availability of these metrics in real time could facilitate improvements in engine control, with resultant improvements possible in engine emissions, efficiency, drivability and noise.

Needle valve position has often been used as an approximate qualitative indicator of the injection rate characteristic [Burman62(1), Hiroyasu80, Obert68(1)]. The beginning and end of needle lift are recognized as the respective start and end of the injection pulse. Start of needle lift is usually used as the injection timing reference event. The use of a needle lift sensor for closed loop injection timing control is common practice [Stumpp83, Ives84, Wolff82]. Pre- and post- injections are also identified by the needle lift. The duration of the injection period, measured using the needle lift signal, has been used as an approximation for the fuel quantity [Stumpp83], and in conjunction with an engine speed has been used to map fuel consumption for a specific engine and pump [Wolff86]. However, the displacement of the needle is, at best, a very nonlinear indicator of actual fuel flow rate through the injection nozzle. Inertial effects on both the moving parts and the fluid also distort the displacement-flow relationship.

Our studies indicate that for a restricted class of injectors and subject to certain restrictions applied to the injector design, it appears possible that the needle lift trace can be used to characterize the injection rate history, and therefore the total fuel delivery. However, the needle-lift vs flow relationship is a complex one, requiring signal processing algorithms which inverse-model the injector, in the engine operating environment.

The high-speed signal analysis needed to accomplish this indirect sensing task in real-time requires specialized data acquisition and signal analysis hardware. Specific hardware requirements, and the design of a prototype "sensor processor" for this task will be described later in this report. All preliminary model development and validation tasks were performed off-line using advanced UNIX workstations, or a Cray YMP supercomputer. Algorithms developed off-line were then written into the firmware of the sensor processor for real-time sensing during actual engine tests.

Two types of common automotive IDI injectors were instrumented for needle lift and fuel temperature, as illustrated in the cross-sectional diagram of Figure 1. Several possible injector flow models were derived, based on practical as well as accuracy considerations. Experiments were conducted on a pump test stand to calibrate and test the models, directed toward correlating instantaneous nozzle flow with the needle motion and fuel temperature. An apparatus was constructed for reference measurement of the fuel injection rate history, to serve as a tool for model development and evaluation. High-speed data acquisition and signal analysis equipment were used to calibrate and test the static and dynamic flow models for
the test injectors over the range of operational conditions. A block diagram of the experimental apparatus is shown in Figure 2. A finite-element hydraulic computer simulation of the pump and injector was also developed to aid in the modeling process of this highly nonlinear system.

The objective was to determine if an inverse injector model could be generally found, which when implemented as a signal processing algorithm could be used to determine some or all of the above-stated metrics with sufficient accuracy and reliability for use as feedback control signals. Engine control algorithms were also studied that could optimally utilize the estimated metrics. A potential closed loop control strategy is suggested in Figure 3, which includes both real-time error reduction, and long term adaptation using nonvolatile random access memory.

On-engine tests of engine controls based on this method have been performed. For the purposes of this study, only limited-authority fuel limit controls were evaluated, which could be implemented without internal modification of the existing engine control computer. This was necessary in order to permit an exact comparison between engine operation with and without the closed-loop fuel control. Further benefits in terms of engine controllability may be expected if more sophisticated control algorithms (e.g., adaptive) are used, which require custom design of the main engine control computer to utilize directly the fuel delivery feedback information provided by the sensor processor.

Also of interest, is the long-term reliability of this sensing method, in view of possible degradation of the injector/sensor due to nozzle carbon formation (coking) and internal wear. A simple experiment was conducted to assess the effects of nozzle coking on the accuracy of the sensing method. However, adequate study to evaluate this possible problem would require the accumulation of several hundred hours of engine operation time for each individual test, which is beyond the scope of this project. Cylinder-to-cylinder variations in the injection characteristics, and the relative benefit of instrumenting more than one cylinder for fuel flow should also be assessed.

5. SPECIFIC PROJECT TASKS

Specific development and experimental tasks performed during the course of this project were as follows:
1. Development of a super-computer simulation of the injection system hydraulic and mechanical behavior, including detailed pump and injector dynamic mass-transport models.
2. Construction, calibration and refinement of a laboratory reference injection rate measurement apparatus.
3. Characterization and flow mapping of sample fuel injectors.
4. Analysis of injector flow vs needle position relationships and formulation of flow models.
6. Iterative improvement of both the experimental methods and the resulting accuracy of the injector models and calibration data.
7. Modification and recalibration of fuel injector assemblies, when required for compatibility with this sensing method.
8. Limited study of temperature dependency of flow models.
9. Development of signal processing algorithms to translate arbitrary needle lift histories into fuel mass estimates.
10. Design, fabrication and debugging of a special purpose sensor processor computer for real-time analysis of needle lift signals.
11. Modification of a commercial single board computer to serve as an interface control computer, for communications with existing engine control computer and terminal for user input and data display. Design, fabrication, and testing of analog interface electronics.
12. Software and firmware development for sensor processor and interface control computer.
13. Setup and instrumentation of diesel test engine in dynamometer test cell.
15. Baseline engine tests to determine open-loop fuel control behavior and emissions for as-delivered engine and injection pump.
16. Comparative engine tests with sensor processor fuel control.
17. Data reduction from engine tests to assess accuracy of sensing method during actual engine operation, and efficacy in exhaust smoke reduction.
19. Preparation of final project report.

A schedule of project activities was submitted with the monthly reports from this project. It was continuously updated over the course of the project, and is included in Appendix 2 of this report.

6. PRELIMINARY SIMULATION STUDIES

Preliminary to model development work, we retraced and extended the work of several prior researchers in the area of computer simulation of injection systems, in an effort to better understand the behavior of high pressure pulsed hydraulics, and the synergy between the injection nozzles, the pump, and the injector lines.
A finite difference model of a generic distributor-type diesel fuel injection system was developed, similar to that suggested by K. Kumar [Kumar83] and others. The complete system model includes a detailed hydraulic model of the pump, the injector lines, and the injection nozzle. Twenty three parameters are used to characterize the complete model.

The injection pump models use cam profiles obtained from physical measurements of Robert Bosch VE Series pumps. This is a rotary-type distributor pump, incorporating a single common plunger for all injection outlets. The cam profile is implemented as a third degree polynomial approximation, or as a mapped function with second order interpolation between specified cam profile points. The pump plunger, upon contact with the cam, compresses the fluid in the pumping chamber, causing a pressure wave to propagate down the injector line. The propagation of this pressure wave is modeled using a finite difference fuel element algorithm, which may be derived directly from basic Newtonian relationships, discretized using a backward difference approximation.

\[
\begin{align*}
q_i^{n+1} &= A_i(\Delta t)^2 \left[ \frac{1}{\rho_0 A_0} \left( p_i^n - p_{i-1}^n \right) + 2q_{i-1}^n - q_{i}^{n-1} \right] \\
p_i^n &= p_r + \frac{K}{A_i \Delta l} \left[ q_i^n - q_{i+1}^n \right]
\end{align*}
\]

where:
- \( i \) = node index
- \( n \) = time step
- \( A_i \) = cross-sectional area in injection line [m²]
- \( K \) = bulk compressibility modulus of diesel fuel [Pa]
- \( q_i^n \) = cumulative discharge past node \( i \) at time step \( n \) [m³]
- \( p_i^n \) = pressure in the \( i \)th node on the \( n \)th time step [Pa]
- \( p_r \) = residual (quiescent) pressure in line
- \( \Delta l \) = length of a node [m]
- \( \Delta t \) = time increment [s]

Once the pressure wave reaches the injector, the behavior of the injection nozzle is modeled by a second order force-mass dynamic model, described in detail later in Section 8 (Model Development). This simple model treats the injector needle assembly as a lumped mass, subject to inertial and frictional forces, as well as the spring force opposing the fluid pressure force acting on the piston area of the needle. Fluid discharge from the nozzle is inferred from the instantaneous nozzle area, the instantaneous nozzle discharge coefficient, and the differential pressure across the orifice. Both the nozzle area and discharge coefficient are nonlinear functions of the needle position.

Working with the model revealed a strong dependence of the time increment on the volume (therefore \( \Delta l \)) of the finite fluid element, for numeric convergence and plausible results. For a given time increment, the incremental volume of fluid entering the element
must be a small percentage of the element volume. Since the fluid is nearly incompressible, the incremental pressure rise during each iteration rises very quickly with increased incremental flow into the element. Thus, a small time increment permitted fine spatial quantization, while a longer time increment required coarser spatial quantization. The plausibility of the results from a finite element model ultimately depend on the spatial quantization; generally, the smaller the better.

The injection system components we simulated required division of the injection lines and internal passages of the pump and injector into 1000 finite fluid elements (simulation nodes). We established a criteria based upon a maximum allowable incremental pressure rise per element per iteration of ten percent. A maximum allowable time increment of one nanosecond was necessary to achieve this criteria. A typical complete injection event spans at least one millisecond. Therefore, a minimum of one million time steps, each requiring the processing of one thousand nodes is required for each simulation run. Approximately 20 FORTRAN statements are executed for each node. Thus at least 20 billion statements must be executed for simulation of a single injection pulse.

Preliminary simulation development was done on a Sun 386i UNIX workstation. But the execution of the simulation required over four hours of run time on the workstation. Alternatively, we employed the NSF-sponsored Cray XMP (later upgraded to a YMP) at the San Diego Supercomputer Center. Simulation run time was reduced to under ten minutes on this platform.

The complete FORTRAN source code for the final version of the simulation, along with one complete output listing, are included in Appendix 3.

Our initial objective for the simulation work was to attempt to get the simulation to ideally mimic the behavior of a particular complete injection system based upon a VE pump. The resulting simulation would then be used as a basis for development and preliminary validation of candidate injector flow models. We dissected a VE series pump, and measured all accessible internal component dimensions to determine model parameters. Other model parameters which were based upon physical properties of the fluid or environment were also determined, either from appropriate references [Bosch76, Kumar73, Henein85, Bosch66, Komoroff66] or by experimental measurement (eg., fluid density). Following considerable effort in determining the most appropriate model parameters, the instantaneous nozzle flow rate and line pressure histories predicted by the simulation eventually agreed with the actual data measured using the reference rate measurement apparatus (discussed in the next section) to a first approximation. However, the subtle pressure or rate inflections observed in the actual system response were not duplicated by the simulation. This was especially noticeable when considered over a wide range of simulated pump speed and fuel deliver settings. It is unclear from our survey of the literature on injection system simulation if this problem has been resolved by other investigators. Typically, published simulation results have been validated based upon comparison with only a limited number of operational conditions, usually from slower speed, larger displacement injection systems. We have not encountered pub-
lished simulation results with confirmed validity over the complete range of operation for a high speed, small displacement distributor pump. We surmise that our results are probably consistent with those of other investigators.

We have concluded that the benefit of the simulation work was not in providing a precise system simulation for later model development work, but rather in the increased understanding it gave us of the complexity of the high pressure pulsed hydromechanical system we were dealing with. Although the success of the proposed sensing method required only the accurate modeling of the injection nozzle (not the pump or injection lines), the complex flow characteristics we encountered for the complete system indicated the need for a more sophisticated injector model than originally hypothesized.

7. CONSTRUCTION OF A REFERENCE RATE MEASUREMENT APPARATUS

The model development and evaluation process required an accurate reference for the measurement of the injection rate history. A laboratory apparatus was proposed by Wilhelm Bosch [Bosch66] which has been used by other investigators as an accepted standard for injection rate measurement [Kumar83, Kaminoto78]. We fabricated an apparatus conforming to Bosch's specifications for reference rate measurement.

The Bosch apparatus operates on the principle of propagation of a pressure wave through a fluid column. The injector discharges directly into a fluid-filled tube of constant diameter and known length. The passage of the resulting pressure wave past a pressure transducer section of the tube provides a pressure signal that is representative of the instantaneous injection flowrate into the column, delayed slightly by the propagation time from the nozzle to the transducer section. By locating the pressure transducer section close to the nozzle, the delay is negligible. The pressure wave is attenuated as it is reflected back and forth along the length. Fluid injected into the measurement tube is relieved through a needle valve and pressure relief valve at the end of the tube. The relationship between the pressure of the wave and the injector flowrate is:

\[ q = \frac{f_t P}{10 a \rho} \text{ cm}^3/\text{sec} \]  

(1)

where

- \( f_t \) = flow area of measurement tube (cm²)
- \( a \) = acoustic velocity in diesel fuel (m/sec)
- \( \rho \) = density of diesel fuel (gm/cm³)
- \( P \) = differential pressure (Pa)

The product \( a \rho \) is referred to as the acoustic impedance. For diesel fuel at the experimental conditions we used \( a = 1250 \text{ m/sec} \) and \( \rho = 0.84 \text{ gm/cm}^3 \). The inner flow area \( f_t \) of
the measurement tube was 0.317 cm². The pressure transducer must not cause any reflections, so pressure is measured by a strain gauge attached to the outside of a thin-wall section of the measurement tube, referred to as the pressure transducer section. Hoop strain in the tube serves as the mechanism for measurement of the passing wavefront pressure. The quiescent pressure in the measurement tube is maintained by the pressure relief valve at approximately peak cylinder compression pressure to simulate the actual backpressure encountered in an engine installation of the injector.

The length of the measurement tube is critical, since reflected pulses can interfere with the primary wavefront. Interference is selective with pump speed. Bosch prescribed the use of two different length measurement tubes to handle the complete RPM range, 4.67 and 9.34 meters. In order to provide sufficient attenuation of the primary reflex pulse to avoid interference at all speeds, we found it necessary to use a tube 80.16 meters long.

Our apparatus was fitted to accept two types of nozzle holders: Robert Bosch KCA series (RB) and Diesel-Kiki Type 71-1280 (DK). The nozzle holders were fitted internally with Hall-effect needle position sensors manufactured by Wolff Controls. Both nozzle holders work with pintle-type nozzles intended for indirect injection applications. All tests were performed using a Diesel-Kiki (Robert Bosch licensed) NP-VE4 injection pump mounted on a Bacharach Type YYQ pump test stand. Figures 4a and 4b are photographs of the reference rate measurement apparatus and associated instrumentation with the test stand.

Total fuel delivery per injection may be calculated by integration of the rate characteristic. By integrating and comparing with the volume delivery measured by the test stand burette, Bosch reported an error of 3.3% for a single selected rate characteristic sensed by the apparatus. Using this same method, we observed an average absolute error of 4.82% (RB injector) or 6.52% (DK injector) for the Bosch rate apparatus over the operational range of the pump/injector, with a peak error of 14.6% (RB) or 15.1% (DK) at a high delivery conditions. Figure 5 is a matrix indicating the accuracy of the Bosch-type reference rate apparatus with the RB injector over the operational speed and throttle angle range of the pump.

A fundamental assumption of the Bosch apparatus is that the instantaneous pressure of the passing wavefront is linearly proportional to the instantaneous rate of entry of fluid into the column (the injection rate). After extensive analysis of preliminary data from the apparatus, we concluded that this assumption is not valid over the range of flow rates that we were attempting to measure. The exact mechanism of this nonlinearity is not certain, but one possible contributing factor could be that at higher flow rates, the downstream pressure at the flow orifice (injector nozzle) increases, thus reducing the nozzle flow. This is considered a basic limitation of any apparatus in which injection into a fluid-filled chamber is substituted for the actual injection environment encountered in the combustion chamber of the engine.

A nonlinear correction function, as shown in Figure 6 was experimentally determined in order to correct the hypothesized nonlinear flow-pressure relationship. Employing this correction function, an improvement in the accuracy of the Bosch apparatus was observed when processing the accumulated injection history data. We observed an average absolute
error for the corrected Bosch apparatus of 2.93% (RB injector) over the operational range of the pump/injector, with a peak error of 8.1%. The accumulated DK injector data was not reprocessed (this is a very labor-intensive task). All subsequent data reduction, and the resulting static model $f_1$ function maps (next section) were generated by preprocessing the Bosch apparatus rate data through the corrective function of Figure 6.

The significance of this nonlinear corrective function to the accuracy of the rate Bosch rate measurement apparatus is illustrated in Figure 7, which shows a typical injection rate trace, with and without the correction. The integrated volume under the two curves would differ significantly.

8. DEVELOPMENT OF HYDROMECHANICAL MODELS FOR INJECTORS

The basis of the method was the hypothesis that it is possible to infer instantaneous fluid flow through the injector nozzle from knowledge of the motion of the injector needle (in conjunction with knowledge of fluid temperature and possibly fluid pressure). Therefore, one of the key tasks of the project was the development of one or more accurate mechanical-hydraulic models which would relate the motion of the needle to the injection rate. Total fuel deliver per injection is then simply determined by integration of the rate characteristic during the injection event.

The approach we took was to start with the simplest possible model, and increase the level of sophistication as validation tests indicate. The first class of models we studied are referred to as "static" models since they relate the instantaneous nozzle flow rate to the instantaneous position of the needle, neglecting all derivatives of the needle position. The second class of models are referred to as "dynamic" since they include the first and second derivatives of the needle position in the flowrate equation.

Prior computer simulation results for the entire pump/line/injector hydraulic system provided considerable insight into the behavior of the injector in response to a pressure pulse from the pump, propagated down the injection lines. However, the complexity of the injector itself required actual tuning and validation of each proposed model using the pump/injector test stand and reference rate measurement apparatus, and collection of large quantities of data. It was recognized that the hydromechanical behavior of the injector is far more complex than could be precisely modeled by any real-time algorithm executable on the sensor processor. Our objective, rather, was to model predominant static and dynamic characteristics in order to achieve adequate fuel quantity sensing accuracy.
8.1. Static Nonlinear Model

A simplified injector flow model described in earlier work [MacCarley87] was used which essentially ignores dynamic effects and assigns a one-to-one mapping between needle position and instantaneous flow rate, modified only by a density factor dependent on fuel temperature. A basic requirement of this model is that the needle (pintle, poppet) be unrestricted by a physical stop in its fully open position. The Bobert Bosch KCA injectors that we evaluated met this requirement unmodified. It was necessary to modify the Diesel Kiki 78-1280 injectors slightly, to extend the unrestricted range of needle travel from 0.6 to 0.8 mm.

To compensate for any possible increase in flow (relative to the open loop calibration of the existing pump electronics), the nozzle opening pressure was increased slightly by increasing the shim thickness in the nozzle holder. The injectors were iteratively recalibrated on the pump test stand until their flow maps were identical to their prior-to-modification conditions, and to each other.

A basic nozzle flow relationship is commonly used to describe the instantaneous mass fuel flow rate through an injector nozzle, as illustrated by the diagram of Figure 8. [Streeter71, Oren83, Obert68(2), Kumar83, Burman62(2), Sharma83]:

\[ \dot{m} = 10^3A_{no}(x)C_d(x,\dot{m})\sqrt{2(p_1 - p_2)}\rho(T) \]  

(2)

\( \dot{m} \) = mass flow rate [mg/sec]
\( A_{no} \) = nozzle area [mm²]
\( C_d \) = nozzle discharge coefficient [unitless]
\( p_1 \) = injection pressure, immediately upstream of the orifice [kPa]
\( p_2 \) = downstream cylinder pressure [kPa]
\( \rho \) = local fuel density [mg/mm³]
\( x \) = needle displacement from closed position, [mm]
\( T \) = fuel temperature [°C]

\( A_{no} \) is a unique function of the needle position \( x \). \( C_d \) is both a function of \( x \) and of the flow velocity, so that normally an iterative solution is necessary for the flow problem. It is reasonable to assume \( \rho \) to be dependent upon temperature \( T \) only, since fluid compressibility affects it only slightly, even at the high pressures involved [Bosch66]. It is impossible to determine the time history of \( p_2 \) without the use of a cylinder pressure transducer. If a pressure transducer is available to the feedback control system, exact knowledge of \( p_2 \) can be used in this calculation. Lacking such a transducer, a constant average value \( \bar{p}_2 \) based upon test measurements may be used [Obert68(2)], with some loss of accuracy.

The linear movement of the needle may be modeled by:

\[ p_1A_1 - (p_1 - p_2)A_2(x) = 10^{-3}M\ddot{x} + 10^{3}k_\tau\dot{x} + 10^{3}(k_{spr}x + f_0) \]  

(3)

\( M \) = combined mass of the needle, spindle, and part of the spring mass [gm].
\( k_f = \) combined linear coefficient of friction [Newtons–sec/mm]
\( k_{spr} = \) differential linear spring coefficient for small displacements [Newtons/mm]
\( f_0 = \) nozzle opening force [Newtons]
\( A_1 = \) upper valve piston area [mm²]
\( A_2(x) = \) effective lower (counteracting) valve piston area [mm²]
\( x = \) needle position, \( x \geq 0 \) [mm]
\( \dot{x} = \) needle velocity [mm/sec]
\( \ddot{x} = \) needle acceleration [mm/sec²]

A dynamic model for the nozzle flow must take into account the inertial and frictional effects on the moving parts, as well as the fact that the flowrate is rapidly changing.

However, the use of (3) to calculate these effects requires generation of the first and second derivative from samples of the needle position. It was observed experimentally that noise amplification obscured the true dynamics, requiring filtering techniques to yield even an approximation to \( \dot{x} \) or \( \ddot{x} \).

A considerable simplification is possible by neglecting the inertial and frictional effects, so that pressure \( p_1 \) is characterized by \( x \) alone, and none of its derivatives. This is not unreasonable considering the extremely high force to mass ratio acting on the needle assembly. Errors introduced by this assumption tend to be self-canceling when the rate curve is integrated over the entire injection interval, especially if the curve is close to symmetric. The error is also reduced by minimization of the needle assembly mass and frictional contact area.

The final assumption that

\[ A_1 \gg A_2(x) \quad (4) \]

allows the slight effect of the cylinder pressure to be ignored in the needle position equation. (Assumes fluid pressure acting on upper piston area \( A_1 \) only.) This is actually valid only after the nozzle is flowing, so that some ambiguity occurs during initial and final flow, reducing accuracy particularly at small deliveries.

The preceding assumptions allow \( p_1 \) to be expressed as a unique function of the needle position \( x \):

\[ p_1 = \frac{10^3}{A_1} k_{spr} x + p_{no} \quad x > 0, \quad p > p_{no} \quad (5) \]

where \( p_{no} = \) nozzle opening pressure [kPa].

This permits the discharge coefficient to also be expressed in terms of \( x \) alone:

\[ C_d(x, \dot{m}) = C_d(x) \quad (6) \]

so that (2) may be simplified to:

\[ \dot{m} = 10^3 A_{no}(x) C_d(x) \left[ 2(p_1(x) - p_{2}(T)) \right]^{\frac{1}{2}} \quad (7) \]
where $T_0$ = fuel temperature at which $f_1$ is measured.

Total fuel delivery is determined by time integration of $\dot{m}$ over the duration of the injection period. Since $T$ is slowly varying, it can be considered constant over the integration period.

\[
m = \int_{t_0}^{t_1} \dot{m} \, dt = \int_{t_0}^{t_1} f_1(x)f_2(T) \, dt
\]

\[
= f_2(T) \int_{t_0}^{t_1} f_1(x) \, dt
\]

In practice $f_1(x)$ is determined experimentally for a particular injector,

\[
f_1(x) = \frac{m(x)}{\int_{t_0}^{t_1} f_1(x) \, dt}
\]

while $f_2(T)$ is a known function for diesel fuel. This simple flow model is not valid in the case of a chattering or oscillating needle, known to occur under certain conditions in some injectors [Burman62(3)].

Data for injector model development were acquired and processed with the aid of a 100 MHz digitizing oscilloscope and a PC-based data acquisition system, communicating over an IEEE-488 serial bus. A complete description of the data acquisition and model development software developed for this application appears in the senior project report by Walter Clark, included in Appendix 4 of this report.

The reference rate measurement apparatus was used to generate the nonlinear map of $f_1$. The composite $f_1$ function is generated by averaging measured flow vs needle position data taken over the speed and throttle angle range of the pump. At each condition, 256 samples of the rate characteristic and needle lift curve are acquired and digitized. Eight throttle positions are tested at each of sixteen pump speeds, from 125 to 2000 RPM. Figure 9 illustrates the variation of the $f_1$ function with pump speed for a Diesel-Kiki 71-1280 injector. (The governor was operative, so that the fuel sleeve position was also changing with speed in Figure 9.) The rather minor variation in the $f_1$ manifold along the RPM axis is noted.

Figure 10 shows the overall average $f_1$ function for a Robert Bosch KCA30SD27/4 injector. Figure 11 shows the average $f_1$ for the Kiki injector. Numeric data for the static flow function of Figure 11 is included in Appendix 5.

This model requires only the nonlinear mapping of needle position into injector flowrate, temperature-density correction, and integration to yield total mass delivery per injection. Fig-
ures 12 through 15 show typical needle lift curves at various pump speeds, throttle settings, for both injectors, with corresponding rate curves derived from (1) the reference rate measurement apparatus, (2) the nonlinear mapping of the simplified static model, and (3) a dynamic model to be described in the next section. Comparison of the rate curves generated by the static model with that obtained from the reference rate apparatus indicate that the flow estimate lags slightly the movement of the needle, underestimating flow while the nozzle is opening, and overestimating while it is closing. The plots of these figures were computer generated directly from the sampled, digitized data collected with the laboratory test stand apparatus.

Figure 16 is an error matrix similar to that of Figure 5, comparing the integrated fuel quantity calculated by the static model from the needle lift signal, with the burette volume from the test stand. Over the operational range, an average absolute error of 4.46% (RB) or 4.72% (DK) is observed, with a peak error of 18.0% (RB) or 14.6% (DK) at medium-flow conditions in both cases. It may be concluded by comparison of this matrix with that of Figure 5 that the simplified static model provides a fuel quantity estimate of similar accuracy to that obtained from the reference rate apparatus (that was used to generate the $f_1$ function for the static model).

8.2. Dynamic Models

Several closed-form models were studied which incorporate inertial effects acting on the needle and fluid. The best results were obtained with a model developed as an modification of the simple static model.

From (3) and (4), $p_1$ may be written as the sum of static and dynamic contributions:

$$p_1 = \frac{10^3}{A_1} k_{spr} x + p_{no} + \left(\frac{10^{-3} M}{A_1} \frac{\dot{x}}{x} + \frac{10^3 k_f}{A_1} \ddot{x}\right)$$

$$= p_s(x) + p_d(x, \ddot{x})$$  \hspace{1cm} (11)

From (8),

$$\dot{m} = 10^3 A_{no}(x) C_d(x) \left[2\rho(T) \left[p_s(x) + p_d(x, \ddot{x}) - \bar{p}_2\right] \right]^{\frac{1}{2}} f_2(T)$$

$$= \left[10^3 A_{no}(x) C_d(x) \right]^{2} 2\rho(T) p_s(x) - \bar{p}_2 \left[10^3 A_{no}(x) C_d(x) \right]^{2} 2\rho(T) p_d(x, \ddot{x})$$

$$= \left[f_n(x) + f_d(x, \ddot{x})\right]^{\frac{1}{2}} f_2(T)$$  \hspace{1cm} (13)

where
The product $A_{no}(x)C_d(x)$ describes the nozzle as a function of the needle position. It is known to be a monotonic function that saturates at some maximum value of $x$. Experimental determination of $A_{no}(x)C_d(x)$ by continuous flow testing of the injector was unsuccessful due to the high flow at high pressure required for needle lift values above 0.5 mm (10 ml/sec., 20.7 MPa at 0.5 mm lift). An approximate function was used:

$$A_{no}(x)C_d(x) \approx \alpha x^\gamma \quad [\text{mm}^2]$$

(15)

where $\alpha$ and $\gamma$ are model parameters ($0 < \gamma < 1$).

With this assumption, it is possible to extend the static model previously derived to include some dynamic (velocity and acceleration) effects.

$$f_d(\dot{x}, \ddot{x}) = x^\gamma \left[ a\dot{x} + b\ddot{x} \right]^{1/\gamma}$$

(16)

where

$$a = \frac{2 \times 10^3 \alpha^2 \rho(T_0) k_f}{A_1} \left[ \frac{mg^2}{\text{mm} \cdot \text{sec}} \right]$$

$$b = \frac{2 \times 10^3 \alpha^2 \rho(T_0) M}{A_1} \left[ \frac{mg^2}{\text{mm}} \right]$$

The mass flow relationship is therefore

$$\dot{m} = \left[ f_1^2(x) + \dot{x} \left[ a\dot{x} + b\ddot{x} \right] \right]^{1/\gamma} f_2(T)$$

(17)

where $w = 2\gamma$, $0 < w < 2$.

$\dot{x}$ and $\ddot{x}$ are found by numeric differentiation of the needle position. A four-period average derivative is used to suppress the noise amplification effects of the differentiation process. The parameters $w$, $a$, and $b$ were optimally determined by least squares fitting the model to the acquired rate vs needle lift histories over the operational speed and throttle range of the pump. The Cray YMP computer at the NSF San Diego Supercomputer Center was used to run the very CPU-intensive optimization problems. Details of the optimization methods for model parameter calculation are given in Appendix 4 of this report. For the DK injector, optimum values were found to be $a = 3.99 \times 10^4$, $b = -4.80$, and $w = 0.366$.

Appendix 6 shows the relative accuracy of the model (for a sample set of model parameters) for each of the injection events recorded, and the overall average absolute accuracy over the entire data field. The optimization objective was the minimization of the average absolute error, which for this three dimensional highly nonlinear problem required considerable computer run time, even on the Cray YMP. All original data have been
archived on computer tape cassette (UNIX tar format) and are available to other investigators.

Figures 12 through 15 illustrate the performance of this dynamic model relative to the static model and the reference rate trace. A slightly better correlation with the reference trace is observed. For example, integration under the dynamic model rate curve of Figure 13 to calculate quantity yields 25.02 mm$^3$, which improves slightly upon the static model estimate (24.98 mm$^3$) of the measured burette volume fuel delivery (25.50 mm$^3$). For comparison, integration of the reference rate curve yielded a volume of 24.14 mm$^3$. Over the complete operational range for the DK injector, the dynamic model yielded an average absolute percentage error of 3.7%, an improvement over the 4.7% error of the static model. Peak error was reduced from 14.6% to 10.6%, both occurring at 2000 RPM, 135 deg throttle (measured volume 20.3 mm$^3$).

9. DESIGN OF THE SENSOR PROCESSOR AND INTERFACE CONTROLLER

The unique requirements of this sensing task dictate specialized hardware for the real-time analysis of the raw analog signal via the algorithm just described. The approach taken was the design of a dedicated "sensor processor" optimized for the rapid data acquisition and analysis task.

9.1. Signal Processing Requirements

A preliminary algorithm based upon the static nonlinear flow model discussed in the previous section was used to lower-bound the signal processing hardware and software requirements of this indirect sensing mechanism. A flowchart for the algorithm appears in Figure 17.

Implementation of the algorithm, based upon only needle lift and fuel temperature, requires first that voltage samples of the needle lift signal be periodically taken and stored over the duration of an "injection window", which is wide enough (in pump angle degrees) to fully contain any injection pulse.

The fuel temperature signal is then sampled and the value stored. After data acquisition, (or while it is in progress), three signal processing operations are performed:

1) **Nonlinear Translation** — A one dimensional table look-up for the $f_i(x)$ function is performed to translate needle position sensor voltage samples into estimated uncompensated flow rate points.

2) **Time Integration** — The translated rate sample points are summed over the injection window period, multiplied by the sample period, and scaled accordingly, thus implementing a rectangular rule integration. The final integral value is the uncompensated total fuel delivery.
3) Temperature Compensation — The stored fuel temperature value is used to index a one-dimension table for the $f_2(T)$ function. The uncompensated fuel delivery is multiplied by this factor to yield the final fuel mass estimate.

Figure 18 illustrates these operations using a typical needle lift trace.

Data collection and analysis of other possible sensor inputs would occur concurrently with the needle lift and fuel temperature acquisition and processing. The minimum RAM storage required for adequate resolution, and the sampling rate were calculated as follows:

Sample over a 60 degree (engine rotation) injection window. Acquire at least 60 data points (1 per degree) under the worst conditions (highest RPM = 6000). At 6000 RPM:

$$\left(\frac{1}{6000\text{ RPM}}\right)\left(\frac{1}{360^\circ/\text{rev}}\right)\left(\frac{60^\circ/\text{sec}}{60^\circ/\text{min}}\right) = 27.78 \mu\text{sec/point}$$

How much RAM storage will be required to store all data points under the worst conditions (lowest RPM = 200)? At 200 RPM:

$$\left(\frac{1}{200\text{ RPM}}\right)\left(\frac{1}{360^\circ/\text{rev}}\right)\left(\frac{60\text{ sec}}{25\times10^{-6}\text{ sec/point}}\right) = 2000 \text{ pts max (each 16 bits)}$$

A minimum data storage area of 4000 bytes is required. The prototype sensor processor was therefore configured with 4K fast static RAM data storage memory.

9.2. Hardware Design Considerations

The major objectives in implementing the proposed sensor processor fell into two general areas: 1) rapid data acquisition and processing capability, and 2) efficient interprocessor communications. The hardware specifications were dictated in view of these objectives.

Figure 19 illustrates the major hardware features of the sensor processor. The key details of each hardware group are described below:

9.2.1. Microprocessor Selection

Consider the computational requirements of the sensor processor in the worst case scenario: maximum engine speed (6000 RPM). Upon command from the main controller, voltage samples are taken every 25 usec over a 60 degree period, which at 6000 RPM occupies 1.667 ms. Thus, 67 data points are sampled. After all data points are stored in memory, the processor has only 30 degrees of engine rotation (833 usec) to complete the three required signal processing operations. To perform the three signal processing operations, a minimum (not including any overhead operations) of four integer multiplies and seven adds per data point are required. For 67 data points, then, 268 multiplies and 469 adds (both using 16-bit words) must be performed within 833 usec. This is equivalent to an instruction
throughput of 0.9 MIPs (million instructions per second). It is possible that some of the signal processing operations could be performed concurrently with the data acquisition process, easing this requirement somewhat. More tightly bounded injection windows could also be used, and fewer data points taken, also reducing the throughput requirements. However, taking this as the minimum specification, it is clear that 8-bit microprocessors are out of the question for this task. With raw processing speed as the dominant requirement, and no need for any of the operating system support features offered by some 16 and 32-bit microprocessors, the Motorola MC68000 was selected for use as the sensor processor CPU for the prototype system. Its 16-bit data bus optimally handles 12-bit A/D output words, and its short instruction times for simple data manipulation operations are useful in this application.

9.2.2. A/D Conversion

The 25 usec maximum sampling period specification imposes special performance requirements on the A/D converter. An actual A/D conversion time significantly faster than this must be used due to the data storage and A/D addressing overhead time. For the prototype system, an Analog Devices ADC-EH12B3 ultrafast A/D converter was specified. This is a hybrid module with a 2.0 µsec 12-bit conversion time.

A 16:1 analog multiplexer (Siliconix DG-506) was employed to allow up to 16 analog sensor inputs to the sensor processor. Input channels are selectable by the CPU by writing the desired channel number to the A/D address (the MC68000 uses memory mapped I/O). The very rapid conversion time of the A/D converter, and the 16 channel analog multiplexer (MUX) allow up to 10 different sensors to be sampled during each 25 µsec sample period. This is more than adequate capability for any anticipated diesel engine/driveline control system.

9.2.3. Memory

EPROM (Erasable Programmable Read Only Memory) memory is used for storage of the firmware data acquisition, signal processing, initialization and communications programs. Although no more than 4K was anticipated for storage of the prototype software, a field of 16K was made contiguous in the address space using a pair of Intel 2764 8K×8 EPROM’s to allow more advanced future applications.

Since the RAM requirements were relatively small and could not justify the hardware overhead needed for dynamic RAM refresh, static RAM was chosen. Although a minimum of 4K RAM was called for by the signal processing requirements, a 16K RAM field was made available to accommodate up to four sampled analog sensor channels. Toshiba 2186 8K×8 static RAMs were used.

The use of nonvolatile RAM was considered as a possible substitute for both the RAM and EPROM. Nonvolatility or RAM storage is a valuable (but costly) feature for a diesel engine control system for two reasons:
(1) Power supply interruptions are common, but loss of RAM memory while an engine is running could have dangerous ramifications. A diesel engine, unlike a spark ignition engine, requires no electric power to operate and can continue to run uncontrolled while the control system is malfunctioning or dead.

(2) The storage of accumulated engine personality information over long periods of time allows much more effective and precise control strategies or adaptive signal processing methods. The ability to retain this accumulated information for future use, after the engine has been shut off, is a valuable asset to the control system. 2Kx8 nonvolatile (using an on-chip lithium battery) CMOS static RAM chips such as the Mostek MK 48702 or Dallas Semiconductor DS 1220 would be suitable.

9.2.4. Interface and Timing Support

In the prototype, the main interface functions are handled by an MC68230 PI/T (Parallel Interface Adapter/Timer), which provides three 8-bit communications ports, four handshake lines, and a 16-bit programmable timer. The communications functions will be discussed later. The timer provides a time base for periodically sampling the sensors and A/D synchronization.

In the design of the prototype, it was assumed that the main control processor was also an MC68000 or equivalent, due to its straightforward interface with the MC68230. This is not a binding restriction. The sensor processor is designed to work with any main control CPU via a well defined parallel communications protocol.

The prototype sensor processor is also equipped with a two digit LED display, driven and decoded from a PI/T parallel port. This provides operational diagnostics or output information, and is helpful in software debugging. It also allows the sensor processor to operate in stand-alone mode, as a test instrument, without a link to a main control processor.

In a production version, the display would be replaced by a diagnostic connector, for field installation of a display for calibration or service.

9.2.5. Additional Design Issues for a Multi-processor Engine Controller

The use of more than one central processing unit (CPU) in an engine control system involves several additional design issues, beyond those encountered in a conventional single processor controller. The addition of our sensor processor and interface control/computer to the existing single processor COVEC electronic control system creates a three-processor distribute controller. The general subject of multiple processor computer architectures for controllers is expansive, encompassing both hardware and software issues, and theoretical issues in distributed control. Only certain key issues were addressed in the context of the present application, and support the decisions made in the design of the prototype sensor processor and its interface to the main controller.
9.2.5.1. Task Assignment

The usable processing throughput of any multiple processor computer depends upon the optimum decomposition and allocation of tasks to the component processors. For a dedicated engine control computer employing two CPUs, task assignment is static and involves a number of design considerations.

In the prototype system, the separation of tasks between the sensor processor and the main control processor is reasonably well defined. The main controller handles all basic engine control and driver interface functions while the role of the sensor processor is to sample and analyze time varying signals from various sensors and report the results as soon as possible to the main controller.

For our prototype, the sensor processor handles crankshaft position sensing using its own programmable timer and interface to an optical timing disk attached to the injection pump drive gear. It determines the start and end of the injection sampling window, signaling the start and end of needle lift data acquisition. Our prototype sensor processor communicated via an 8-bit parallel link with an interface/control computer, which we constructed from a Motorola 68HC11 evaluation board. The interface/control handles the closed loop injection fuel control, overriding the authority of the existing COVEC pump controller when a speed-dependent fuel limit is exceeded. It also handles all user interface tasks, communicating via a serial (RS232C) link with an ASCII terminal at 9600 BPS.

In this configuration, the sensor processor functions fairly autonomously from the main controller, and communicates with it in a cyclic, deterministic protocol.

9.2.5.2. Architectural Relationship

Since the sensor processor functions normally under direct command from the main processor (except when reporting urgent sensor or system failure conditions), a predominantly master/slave architectural relationship exists between the two processors. The sensor processor might be characterized as an input handling subsystem with certain preprocessing capabilities.

9.2.5.3. Interprocessor Communications

In this application, speed of communication between the main and sensor processor is critical, since the sensor processor’s response to the start and stop data acquisition commands must be immediate. The immediacy requirement seemed to preclude serial communications due to its relatively slow transmission rate. Realistically, serial communications over a high speed link might still be fast enough to make command transmission seem instantaneous relative to engine rotation.

Serial communications has the distinct advantage of allowing the processors to be located a considerable distance (> 1 meter) apart, interfaced using a single wire pair or fiber optic cable. The pervasive use of electronics in automobiles has fueled industry interest in the standardization of a time multiplexed automotive LAN (Local Area Network) [Jurgen86,
Communications between a main controller and a distant slave (e.g., the sensor processor) might be accomplished using such a vehicular serial network. The main controller could be safely located inside the passenger compartment (such as under the dashboard) while the sensor processor might reside close to the engine to minimize the length of sensor input leads, thus reducing electromagnetic noise intrusion.

At the opposite speed extreme is the use of dual port RAM, either as a message passing mailbox or as a shared workspace between the processors. Since only one CPU at a time can access the RAM, arbitration logic is required to handle access contention. Only a very short distance is permitted between the processors, since the local buses must meet at the arbitration logic circuitry.

The communications method we applied in the prototype design takes a middle ground between these two extremes. Eight-bit parallel communications using the previously mentioned PI/T, paired with another PI/T in the interface/control computer, provides a command response approximately as fast as the interrupt service time of both CPUs, while permitting a moderate distance (up to ~3 meters) between processors without introduction of noise errors. Some interface circuitry is necessary for the 68230 PI/T, to provide increased drive capability and collision avoidance in the event of simultaneously transmitted messages from both processors. An asynchronous dialogue is conducted between the processors, using the handshake lines of the PI/Ts. Figure 20 illustrates the general relationship of the sensor processor and interface/control computer in a fully deployed intelligent sensor-based engine control system.

10. OPERATION OF THE PROTOTYPE SENSOR PROCESSOR

Figure is a flowchart of the sensor processor demonstration software. The operation over a single data acquisition, analysis, and communications cycle is as follows:

Upon either power-up or a reset command from the main processor, the sensor processor configures the PI/T ports and programmable timer, performs a self-diagnostic check, writes a diagnostic code to the display, and waits for a command from the main processor. While waiting, it continually refreshes the diagnostic display.

At the beginning of the injection window, an interrupt is generated by the interface electronics, which initiates the data collection cycle of the sensor processor. The sensor processor sets the MUX channel to the needle lift sensor, and A/D converts and stores a voltage value once every 25 μsec, scheduled by the programmable timer of the 68230.

Data acquisition continues until the end of the injection window. The maximum number of data points that can be acquired depends on the available RAM in the sensor processor. With 4K of RAM, 2000 points can be collected, allowing the full 60° injection window to be recorded down to 200 RPM.

Following completion of the data acquisition cycle, the MUX channel is changed to the temperature sensor input. This voltage is A/D converted and stored.
Once data collection has been completed, the sensor processor begins data analysis. Non-linear correction is performed on each data point by references to the ROM-stored $f_1(x)$ calibration table. The result is immediately time scaled and integrated to form a running sum. The integral result is complete when all data points have been processed.

A multiplicative temperature correction factor is determined via a table lookup for $f_2(T)$ using the stored fuel temperature value as an index. The integral sum is multiplied by this result to produce the total fuel delivery figure.

This result is scaled to a two byte word and prepared for transmission, prefaced with a start byte and a diagnostic word, and followed by a stop byte. Overflow of the integral sum, unreasonable data point values, out-of-range intermediate results, bus errors, and incorrectly passed interrupt vectors produce corresponding error codes in the diagnostic word, and signal processing is aborted at the point of error.

The result is written to the diagnostic display. A handshake-controlled data transfer routine is then executed in which the six bytes are asynchronously transferred to the control/interface processor via the parallel port. At the end of this sequence, the sensor processor is reset to its initial state, ready for another acquisition/analysis cycle.

A hardware watchdog timer, which is reset in the main wait/display refresh loop, will automatically cause a system reset in the event of timeout due to a program, communications, or noise related error.

A complete assembly language listing of the most recent version of the sensor processor firmware appears in Appendix 7.

11. INTERFACE/CONTROL COMPUTER OPERATION

The MC68HC11-based interface/control computer performs the dual function of closed loop fuel control and user interface. It receives the fuel quantity estimate data from the sensor processor, paced by its ability to accept this data. The interface control computer is slower than the sensor processor in this application, so that at high engine speeds, not all fuel quantity estimates are used by the controller.

It also displays output information and receives input from the operator (user) via a serial link to an ASCII terminal. The user is given the ability to change control parameters "on-the-fly" while the engine is running. This is a valuable aid to the tuning of the control algorithms.

A proportional-integral (PI) control algorithm with adjustable integrator reset rate is used for the most recent version of the limited authority fuel control. The integrator reset function is necessary due to the limited "override" fuel control authority. The control only has the ability to reduce the delivered fuel quantity relative to the quantity commanded by the COVEC controller; it cannot increase this amount. Therefore, the integrator of the PI control never sees a negative input which would "unload" the integration sum. An constant-rate reset "drain" for the integrator must be provided.
The electronic mechanism by which the override fuel control authority is implemented is described in the senior project report by Dan Need, in Appendix 8 of this report.

A complete assembly language code listing for the prototype interface/control computer appears in Appendix 9 of this report.

Figure 21 is a photograph of the complete prototype sensor processor and interface/control computer assembly, in an environmental enclosure. The analog interface electronics are in a separate enclosure to reduce the intrusion of digital noise into the analog sensor circuits.

12. EVALUATION OF METHOD ON TEST ENGINE

An Isuzu 4FB1 1.8L 4-cylinder automotive diesel engine was donated to this project by Isuzu Technical Center of America, in Cerritos, California. This is an indirect injection (IDI) high-speed (5000 RPM) engine employing a swirl-type combustion chamber. The power plant is equipped standard with the Diesel Kiki COVEC electronically controlled diesel injection pump (mechanically similar to the Robert Bosch VE4 pump). It was sold only in Japan, in 1982 and 83, installed in the Japanese equivalent of the USA-issue I-Mark automobile. To the best of our knowledge, it is the only automotive diesel engine actually sold in significant numbers to the public which used full-authority electronic controls. (It may be noted, however, that several manufacturers of diesel fuel injection equipment have developed and offered for sale full-authority electronically controlled pumps.) The existing electric actuators on the COVEC pump and the existing ECU considerably simplified our task of evaluating this control method on an electronically controlled diesel engine. The COVEC pump is no longer produced by Diesel Kiki, for lack of financial incentive to put electronics on small diesel engines (ref: personal conversation with Diesel Kiki representatives).

This engine, with a mechanically controlled (NP-VE4) version of the same pump is still in production and used in light trucks, although it cannot be sold in the California and some other states due to emissions standards. California restrictions on vehicular particulate emissions (0.08 gm./mi) enacted in 1986 have functionally prevented the sale of diesel automobiles or light trucks in the state since that year.

A phantom view of the COVEC pump showing its fuel sleeve and timing ring actuators is shown in Figure 22. The objectives of this experimental evaluation were best met by retaining as much of the existing COVEC electronic componentry as possible. This permitted rapid switching during engine tests between the "standard" open-loop fuel control and the sensor processor closed loop fuel control, for direct comparison under similar operating conditions. This also eliminated the need to redesign and fabricate electronic control functions such as timing control, duplicative of the existing fuel control computer (ECU). However, the degree of authority that the sensor-processor and interface controller could assert over the fuel delivery control was somewhat limited by this arrangement. Control of the fuel quantity
was implemented as a upper fuel limit control only. That is, the COVEC ECU maintains normal fuel control authority until the sensor processor determines, via rapid analysis of the needle lift signal, that a certain mapped fuel limit for that particular engine speed has been exceeded. This fuel limit map is stored in the PROM of the interface control computer, which also implements the fuel limit control. The map is experimentally determined to yield a particular exhaust smoke level corresponding to a true fuel quantity. When this fuel limit is exceeded, the interface control overrides the COVEC fuel control and limits any further increase in the fuel quantity.

The engine was mounted on a 250 BHP 5000 maximum RPM General Electric DC dynamometer. The exhaust is sampled, filtered, dessicated and then analyzed for selected gaseous components. Oxygen content is analyzed using a Beckman magnetic deflection-type oxygen analyzer. Exhaust carbon dioxide and carbon monoxide are individually measured by Beckman infrared absorption analyzers, tuned for each species. Broad spectrum hydrocarbons (HC) were measured by a Beckman flame ionization detector (FID) HC analyzer, calibrated using a 416 ppm hexane reference. Oxides of nitrogen (NO and NOx) are measured by a Beckman chemiluminescence NO/NOx analyzer.

An electronic (optical transmissive) exhaust opacity meter was used for modal (instantaneous) monitoring of exhaust smoke levels. It provided readouts in both percent opacity and Bosch smoke numbers. This instrument was provided on loan from Isuzu Technical Center.

The actual fuel flow rate was monitored on both a one second and ten second average basis by an AVL electronic balance-type automatic mass fuel flow meter. This provides an electronic readout in kg/hr.

Intake air flow was measured by a Merium laminar flow element air flow meter. Readout was in the form of inches water column of differential pressure across the element, which translates via a calibration curve to air flow rate at a given temperature.

The dynamometer load was manually controlled for all tests.

Figure 23 is a photograph of the engine on the dynamometer inside the test cell. Figure 24 is a photograph of the sensor processor and related electronics outside the test cell.

Baseline tests were run to confirm that the engine met manufacturer’s specifications with regard to maximum torque over the 500 to 5000 RPM usable engine speed range. No baseline particulate emissions specifications were available from the manufacturer. Apparently, since particulates (smoke) were not considered a regulated emission at the time of manufacture, there was no incentive for the manufacturer to test or accurately calibrate the maximum torque smoke limit. Parametric curves were then run at various indicated exhaust smoke levels. Brake torque and all emission species and flows were recorded. The estimated fuel quantity, as calculated in real-time by the sensor processor from the needle lift signal, was also recorded. The sensor processor used a static nonlinear model algorithm corresponding to a 6% average absolute error (test stand data) for all engine tests. Data from these tests are shown in Table 1.
Of special interest to us during these test runs was the five percent opacity fuel quantity. The sensor-processor-calculated fuel quantity corresponding to a 5 percent exhaust opacity at each engine speed was recorded and then later programmed into the PROM fuel delivery limit map to serve as the control reference for later emission reduction assessment tests.

The data of Table 1 indicted a high level of exhaust smoke in the full-power condition for the as-delivered COVEC pump. We must, however, allow for the possibility that the modification of the injectors may have increased the fuel delivery somewhat on the actual engine, even though this was not indicated in the injector calibration tests on the pump/injector test stand. If this were in fact the case, the observed high smoke levels may have been greater than normal for this engine and pump. However, it is unlikely that this factor alone could justify the very high smoke levels for the upper 50 percent of the throttle range at all engine speeds tested. This is most likely attributed to the actual open-loop fuel delivery calibration of the COVEC pump, as-delivered from the manufacturer.

The threshold of significant exhaust smoke seemed to correlate fairly consistently with a 2 per cent exhaust oxygen, confirming reasonably good combustion efficiency for this engine operating near stoichiometric conditions.

The estimated fuel quantity produced by the sensor process was used in conjunction with an assumed fuel density \( p = 0.81 \) to calculate an average mass fuel flow rate in \( \text{kg/hr} \), for direct comparison with the AVL fuel flow meter reading at each test condition. A reduced accuracy in fuel flow estimation was observed relative to the pump test stand data. A usually higher-than-actual fuel flow estimate was produced, by as much as 17.8% compared to the AVL average mass flowrate data.

This discrepancy lead us to run further tests to specifically study the accuracy of the fuel quantity reported by the sensor processor during on-engine tests. The data of Table 2 reflects comparative injector flow tests run on the engine and the pump test stand, under conditions as closely as possible identical.

On-engine fuel flow measurements are shown in the section "Fuel Flow Measurements from Test Engine" of Table 2. The pump and injectors were then removed from the engine and transferred to the pump test stand, complete with the electronic controls. The injectors were again flow-mapped at the same operational conditions as those just run on the engine. It was not possible to exactly match the fuel temperature conditions of the engine, but the test fuel temperatures reported by the DDS-II monitor were kept close to those recorded during the engine tests. The section "Fuel Flow Measurements from Test Stand" of Table 2 shows the results.

Oscilloscope photographs were also taken both on the engine and on the pump test stand, for direct comparison of the needle lift traces at identical speeds and throttle settings. Two comparison needle lift traces are shown in Figures 25a and 25b, representing identical operating conditions on the test stand vs the engine.

Based on the AVL average mass flow rate data, some differences in fuel flow were observed between the test stand and engine under identical conditions. These differences did
not exceed 5.3%. However, the fuel quantity estimates generated by the sensor processor differed quite significantly between the engine and test stand. In all cases, the fuel quantity reported during the engine tests exceeded the quantity reported on the test stand, by as much 49.8% (the 66.0% error for the 4500 half throttle condition was probably a recording error).

The comparative needle-lift traces of Figures 25a and 25b provide some insight as to the reason for these differences. Even though the actual fuel flow through the injector remained nearly equal, the shape of the needle lift trace seems to vary noticeable between the engine and the test stand.

It is interesting to observe, however, that the sensor processor and AVL fuel flow measurements agree much better with each other and the test stand burette data, than they do with the commanded fuel quantity reported by the DDS-II monitor for the COVEC controller. This provides some indication as to how far off the factory calibration map of the pump was.

Why the reduced accuracy of the sensor processor on the engine compared with the test stand? There were several possible explanations for this discrepancy:

1. The injectors may be flowing less fuel for the same needle-lift on the engine compared with the pump test stand. The sensor processor may, in fact, be accurately detecting this phenomena, and reporting high fuel quantities due to its calibration using pump test stand data.

2. Only the static fuel injector flow algorithm was actually test on the engine, due to noise problems encountered in the generation of the first and second derivatives for the dynamic algorithms. This algorithm is the least accurate of those tested on the injector test data. The dependency of the flow model on fuel temperature was estimated based on density change alone. Since it was not possible to locate a temperature sensor inside the fuel injector as originally planned, the local fuel temperature and the influence of this factor on the accuracy of the flow model are unknown. Application of an incorrectly assumed fuel density proportionally effects the comparison of the sensor processor fuel mass flow estimate with the AVL mass flow measurement.

3. Limitations in the coding of the algorithm on the experimental sensor processor in finite precision arithmetic and with limited throughput may have also reduced accuracy compared with the offline application of the same algorithm.

4. It is noted that although the same injectors were used on the engine as were used on the pump test stand for calibration of the sensor processor, different pumps were used. The pumps were presumed mechanically identical, since they were both Diesel Kiki NP-VE4 type. They should differ only in that the COVEC pump has electric actuators for fuel sleeve and timing, whereas the mechanical pump uses centrifugal advance and a Watt-type speed governner. Also, it was not possible to ascertain the timing curves of each pump. If the pumps did differ significantly with regard to their injection pressure characteristics or otherwise, it is possible that the relationship between the injector needle lift and the injector flow rate is not as independent of the pump and engine characteristics as presumed by the current algorithms. This conclusion would indicate that, in
actual engine operation, the sensing method is potentially in error by as much as 25 percent when the pump is changed and the operating conditions differ radically.

5. It is possible that the injector-to-injector flow calibration differed more significantly on the actual engine than was indicated by the pump test stand calibration, thus yielding a composite fuel flow for the four injectors that differs from the multiplication of the flow rate in the instrumented injector by four.

We investigated each of these possible explanations.

As mentioned previously, all data of Table 1 assumed a constant fuel density of 0.81 gm/ml. No correction for temperature differences between the engine test stand were possible, since measurement of the fuel temperature at the injection nozzle was not possible. The temperature effects are assumed to be significant, but to what degree is not certain.

Another significant difference between the two operating environments is the downstream (combustion chamber) pressure of the engine relative to the packed fluid column of the Bosch reference rate measurement apparatus used for the laboratory experiments on the test stand. Higher downstream pressures may be expected on a running engine, although the instantaneous pressures seen by the nozzle in the Bosch apparatus could be quite high due to fluid inertia in the packed fluid column. The fluid column in the Bosch apparatus is maintained by the pressure relief valve at an average pressure which is the same as the peak compression pressure of the engine. Suffice to note that the medium the nozzle is discharging into in each case is quite different. If the average combustion pressure of the engine does exceed the pressure encountered by the nozzle in the Bosch apparatus, the needle might withdraw more for the same instantaneous injection flowrate on the engine, giving the illusion (to the sensor processor) of increased fuel flowrate, an therefore a larger fuel quantity after integration. As pointed out earlier in the model development discussion, the assumption of a constant downstream pressure may not be valid when we seek to use the injector model to actually determine flowrate, rather than just understand it in a qualitative sense (as was the case for other investigators). If this is indeed the predominant reason of the observed differences, a cylinder pressure sensor, or some means for inferring instantaneous cylinder pressure (possibly heat release from the injected fuel) would be required for accuracy (in an absolute sense) of this method on an engine.

Injector-to-injector (cylinder-to-cylinder) differences were also retested on the test stand. The instrumented injector (cyl #1) did flow more than the injectors on the other cylinders under some conditions, but the differences were small. The volume delivery data from the test stand burettes for each individual cylinder's injector are summarized in Table 3.

Another possible source of error which might have exacerbated the problem was the interface electronics. Low-pass analog and digital filters that were necessary to block signal noise on the needle lift signal, introduced non-trivial distortion of the needle lift signal, thus altering the fuel quantity estimate generated by the sensor processor. The noise apparently originates from the injector-mounted Hall-effect needle lift sensor, and is not easily detected from visual inspection of oscilloscope traces. Removal of the filters was not possible without
even more severe distortion of the signal by the noise. We estimate that the distortion of the needle lift signal by either the noise or the filters introduced an error as great as 10% relative to off-line processing of sampled needle lift data taken on the test stand. This represented a source of error during both test stand tests (in which the sensor processor was used as opposed to off-line processing) and engine operation, but the the sense of the induced error in each case could be different. Redesign of the interface electronics or sensor processor, might have reduced this effect, but we believe the contribution of this error source was minor, at worst. It may be noted that the errors introduced by the filters are at least repeatable, whereas noise-induced errors are not.

Aside from the test stand vs engine differences, it was observed that the sensor processor fuel delivery estimate using the static flow algorithm was consistent. It was repeatable both in its comparison with the AVL flow measurement, and the smoke level to which it corresponded. This was confirmed during engine emissions tests in which the closed loop fuel control could be switched in or out while the engine was in operation. The fuel limit in these tests was programmed in the sensor processor to yield a 5% maximum smoke opacity level. With the closed loop fuel control engaged, the 5% smoke limit was consistently maintained within plus or minus two percent opacity, even though the maximum engine torque varied by as much as ten per cent at reference points of 2000, 3000 and 4000 RPM.

Figure 26 illustrates the relative full-power smoke opacity levels, with and without the sensor processor control. Figure 26 also illustrates the corresponding engine torque for these data, with and without the sensor processor control. The smoke limit was consistently maintained during five repetitions of these engine tests over a year-long period. It may also be noted that the significant reduction in exhaust smoke is achieved with only a negligible reduction in maximum torque. A concomitant increase in thermal efficiency is realized, since the excess fuel delivered during the overfuelling conditions (without the sensor processor) does not produce any significant increase in energy output once the smoke limit has been exceeded.

These data appear to verify the utility of the sensing method to implement feedback controlled fuel limiting for elimination of full-throttle smoke, although the need for consideration of additional factors in the injector flow model is clearly indicated. A possible improvement in the model generation and calibration method might result if it were possible to do the flow mapping on the engine rather than on a test stand. Unfortunately, since the reference rate measurement apparatus cannot be used on an engine installation, some alternative reference rate measurement method would have to be employed to accomplish this.

13. LONG TERM CONSIDERATIONS: INJECTOR DECALIBRATION

The problem of nozzle coking, which occurs in all injectors to some degree, is expected to be a possible source of loss of calibration for this sensing mechanism. Long-term wear of
the pintle and nozzle orifice may also contribute to a gradual loss of calibration of the injector. The result is a change in \( f_1 \) over time, which contributes to increased error of the sensing method.

Open nozzles, such as those commonly used in small displacement IDI automotive engines, are usually less prone to nozzle coking than closed, multi-hole nozzles found in larger, lower speed engines. Optimal injector/sensor design to minimize the influence of nozzle deposits on the flow calibration may also reduce this concern to some degree. Adaptive signal processing techniques could be applied to compensate for changes in the sensor calibration, if at least one other indirect indication of fuel delivery is available. One such comparison signal for long-term adaptation is driveshaft torque, sensed directly via magnetostrictive methods [Ribbens81, Fleming82] or indirectly via instantaneous crankshaft velocity as described by Sood [Sood84] and others. An adaptive corrective algorithm is suggested based on periodic comparison of the needle lift derived fuel quantity with a steady-state torque reference point.

The effects of nozzle coking on the long-term accuracy of methods utilizing needle lift to infer fuel delivery were studied by generating parallel bodies of data on the same injector, both clean and heavily coked. A Bosch KCA-series injector subjected to 188,000 km of continuous service was mapped over the previously defined operational range. The injector was then thoroughly cleaned and the mapping process repeated. Average \( f_1 \) functions for the nozzle in clean vs coked condition are overlaid in Figure 27. In clean condition, the static model produced an average absolute error of 6.86\%. This poorer than usual accuracy may be partially attributed to the reference rate apparatus, which alone yielded an average absolute error of 7.02\%. The "clean" \( f_1 \) function was then used to generate fuel delivery estimates from the "coked" injector needle lift data, simulating long-term degradation attributable to coking. An average absolute error of 6.05\% was observed. The small difference (actually a slight improvement for this injector) is indicative of relatively minor calibration change attributable to nozzle deposits.

14. CONCLUSIONS AND AREAS FOR FURTHER STUDY

The results of this work appear to confirm that for two specific types of IDI pintle nozzles, adequate information is contained in the needle lift signal to reasonably estimate the injection rate characteristic and total fuel delivery during off-engine flow tests. The accuracy of the fuel calculation via this method is similar to the accuracy of the rate measurement apparatus used to calibrate the signal processing model. The absolute accuracy of the method degrades when the injectors are used on a running engine, due to apparent differences in the injector needle movement on the engine relative to the test stand, under otherwise identical conditions.
For the off-engine tests of the injectors, a simple static nonlinear model appears adequate to achieve a sensing mechanism of +/- 5% average accuracy. Some further improvement is achievable by inclusion of dynamic effects in the model. However, other factors must be taken into consideration in the injector flow model if accuracy of the fuel delivery estimate is to be maintained when the injectors are installed on the actual engine. Fuel temperature at the injector orifice and cylinder pressure may be two of the most significant factors.

Fuel delivery estimates generated by this method appear to consistently correspond to smoke opacity values, and this relationship remains constant over engine operating conditions and time.

Superior flow model data could be obtained if the injectors could be characterized while installed on the running engine, rather than on a laboratory reference rate measurement apparatus.

Nozzle coking and wear degrade the accuracy only to a minor degree.

Closed-loop fuel control based upon this sensing mechanism appears to be feasible, although improved injector models and further on-engine testing are needed.

15. REFERENCES

[Burman62(2)] Burman, P. G. and DeLuca, F. ibid. p. 120.
[Burman62(3)] Burman, P. G. and DeLuca, F. ibid. p. 123.
[Ives84] Ives, A. P., Frankl, G., David, P. Electronic Fuel Injection Equipment for the Light


[Stumpp83] Stumpp, G. and Kull, H. Strategy for a Fail-Safe Electronic Diesel Control Sys-
tem for Passenger Cars. SAE Paper 830527, 1983.


16. TABLES AND FIGURES
Table 1. Engine tests to determine smoke vs fuel quantity.

<table>
<thead>
<tr>
<th>RPM (engine)</th>
<th>Torque (lb-ft)</th>
<th>Opacity (%)</th>
<th>SenProc Fuel (ul)</th>
<th>SenProc Fuel (kg/hr)</th>
<th>AVL Fuel (kg/hr)</th>
<th>Diff Fuel (%)</th>
<th>Fuel Temp (deg C.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>52.5</td>
<td>1</td>
<td>35.5</td>
<td>3.32</td>
<td>2.45</td>
<td>26</td>
<td>34</td>
</tr>
<tr>
<td>1500</td>
<td>34.1</td>
<td>1</td>
<td>23.4</td>
<td>3.29</td>
<td>2.45</td>
<td>25</td>
<td>37</td>
</tr>
<tr>
<td>2000</td>
<td>50.8</td>
<td>1</td>
<td>28.5</td>
<td>5.34</td>
<td>4.06</td>
<td>24</td>
<td>38</td>
</tr>
<tr>
<td>2500</td>
<td>46.4</td>
<td>1</td>
<td>25.6</td>
<td>5.99</td>
<td>5.07</td>
<td>15</td>
<td>39</td>
</tr>
<tr>
<td>3000</td>
<td>36.8</td>
<td>1</td>
<td>23.2</td>
<td>6.51</td>
<td>5.12</td>
<td>21</td>
<td>40</td>
</tr>
<tr>
<td>3500</td>
<td>28.9</td>
<td>1</td>
<td>21.0</td>
<td>6.88</td>
<td>5.72</td>
<td>17</td>
<td>42</td>
</tr>
<tr>
<td>4000</td>
<td>22.0</td>
<td>1</td>
<td>18.2</td>
<td>7.35</td>
<td>6.41</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>4500</td>
<td>53.4</td>
<td>2</td>
<td>27.5</td>
<td>11.58</td>
<td>10.84</td>
<td>6</td>
<td>47</td>
</tr>
<tr>
<td>5000</td>
<td>58.6</td>
<td>22</td>
<td>35.0</td>
<td>14.74</td>
<td>14.69</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>5500</td>
<td>58.6</td>
<td>20</td>
<td>37.5</td>
<td>15.8</td>
<td>15.80</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>6000</td>
<td>57.8</td>
<td>68</td>
<td>39.9</td>
<td>16.81</td>
<td>17.97</td>
<td>-7</td>
<td>48</td>
</tr>
<tr>
<td>6500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>36.8</td>
<td>3</td>
<td>20.8</td>
<td>9.73</td>
<td>8.51</td>
<td>13</td>
<td>48</td>
</tr>
<tr>
<td>5000</td>
<td>55.1</td>
<td>21</td>
<td>36.9</td>
<td>17.27</td>
<td>15.22</td>
<td>12</td>
<td>48</td>
</tr>
<tr>
<td>5000</td>
<td>52.5</td>
<td>40</td>
<td>40.3</td>
<td>18.86</td>
<td>16.63</td>
<td>12</td>
<td>48</td>
</tr>
<tr>
<td>5000</td>
<td>53.4</td>
<td>76</td>
<td>43.8</td>
<td>20.5</td>
<td>19.84</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td>RPM (engine)</td>
<td>Throttle</td>
<td>Fuel T (deg. C)</td>
<td>Coolant T (deg. C)</td>
<td>Oxygen (%)</td>
<td>Smoke (% Opacity)</td>
<td>Torque (lb-ft)</td>
<td>BHP</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>----------------</td>
<td>-------------------</td>
<td>------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-----</td>
</tr>
<tr>
<td>1000</td>
<td>half</td>
<td>34.1</td>
<td>84.3</td>
<td>6.4</td>
<td>36</td>
<td>63.0</td>
<td>12.0</td>
</tr>
<tr>
<td>1500</td>
<td>half</td>
<td>33.8</td>
<td>90.2</td>
<td>5.1</td>
<td>25</td>
<td>67.4</td>
<td>19.2</td>
</tr>
<tr>
<td>2000</td>
<td>half</td>
<td>33.6</td>
<td>92.2</td>
<td>2.0</td>
<td>20</td>
<td>69.1</td>
<td>26.3</td>
</tr>
<tr>
<td>2500</td>
<td>half</td>
<td>33.6</td>
<td>93.8</td>
<td>2.4</td>
<td>17</td>
<td>68.3</td>
<td>32.5</td>
</tr>
<tr>
<td>3000</td>
<td>half</td>
<td>34.1</td>
<td>93.7</td>
<td>2.5</td>
<td>10</td>
<td>64.8</td>
<td>37.0</td>
</tr>
<tr>
<td>3500</td>
<td>half</td>
<td>34.5</td>
<td>94.8</td>
<td>4.2</td>
<td>3</td>
<td>63.9</td>
<td>42.6</td>
</tr>
<tr>
<td>4000</td>
<td>half</td>
<td>35.5</td>
<td>95.7</td>
<td>-</td>
<td>2</td>
<td>49.9</td>
<td>38.0</td>
</tr>
<tr>
<td>4500</td>
<td>half</td>
<td>36.6</td>
<td>96.0</td>
<td>-</td>
<td>1</td>
<td>43.8</td>
<td>37.5</td>
</tr>
<tr>
<td>1000</td>
<td>full</td>
<td>31.3</td>
<td>87.5</td>
<td>2.6</td>
<td>40</td>
<td>65.6</td>
<td>12.5</td>
</tr>
<tr>
<td>1500</td>
<td>full</td>
<td>31.3</td>
<td>92.2</td>
<td>4.0</td>
<td>35</td>
<td>70.9</td>
<td>20.3</td>
</tr>
<tr>
<td>2000</td>
<td>full</td>
<td>31.5</td>
<td>94.5</td>
<td>0.9</td>
<td>56</td>
<td>70.9</td>
<td>27.0</td>
</tr>
<tr>
<td>2500</td>
<td>full</td>
<td>31.8</td>
<td>94.8</td>
<td>0.9</td>
<td>66</td>
<td>68.3</td>
<td>32.5</td>
</tr>
<tr>
<td>3000</td>
<td>full</td>
<td>32.1</td>
<td>95.7</td>
<td>0.6</td>
<td>67</td>
<td>70.9</td>
<td>40.5</td>
</tr>
<tr>
<td>3500</td>
<td>full</td>
<td>32.6</td>
<td>96.8</td>
<td>0.5</td>
<td>66</td>
<td>68.3</td>
<td>45.5</td>
</tr>
<tr>
<td>4000</td>
<td>full</td>
<td>33.1</td>
<td>98.3</td>
<td>0.4</td>
<td>56</td>
<td>63.9</td>
<td>48.67</td>
</tr>
<tr>
<td>4500</td>
<td>full</td>
<td>34.3</td>
<td>100.8</td>
<td>0.4</td>
<td>68</td>
<td>57.8</td>
<td>49.5</td>
</tr>
</tbody>
</table>
Table 2b. Fuel Flow Measurements from Test Stand ($\rho=0.81$).

<table>
<thead>
<tr>
<th>Cmd Fuel (ul)</th>
<th>Cmd Fuel (kg/hr)</th>
<th>SenProc (ul)</th>
<th>SenProc (kg/hr)</th>
<th>AVL mtr (ul)</th>
<th>AVL mtr (kg/hr)</th>
<th>Buret (ul)</th>
<th>Buret (kg/hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.5</td>
<td>2.09</td>
<td>47.0</td>
<td>4.57</td>
<td>40.5</td>
<td>3.94</td>
<td>40.0</td>
<td>3.89</td>
</tr>
<tr>
<td>19.3</td>
<td>2.81</td>
<td>39.4</td>
<td>5.74</td>
<td>34.9</td>
<td>5.09</td>
<td>33.0</td>
<td>4.81</td>
</tr>
<tr>
<td>18.5</td>
<td>3.60</td>
<td>35.9</td>
<td>6.98</td>
<td>34.2</td>
<td>6.64</td>
<td>32.0</td>
<td>6.22</td>
</tr>
<tr>
<td>17.9</td>
<td>4.35</td>
<td>31.8</td>
<td>7.73</td>
<td>31.6</td>
<td>7.67</td>
<td>29.5</td>
<td>7.17</td>
</tr>
<tr>
<td>17.5</td>
<td>5.10</td>
<td>29.6</td>
<td>8.63</td>
<td>31.0</td>
<td>9.03</td>
<td>29.0</td>
<td>8.46</td>
</tr>
<tr>
<td>16.0</td>
<td>5.44</td>
<td>24.9</td>
<td>8.47</td>
<td>30.8</td>
<td>10.5</td>
<td>26.5</td>
<td>9.02</td>
</tr>
<tr>
<td>14.0</td>
<td>5.44</td>
<td>17.1</td>
<td>6.65</td>
<td>23.0</td>
<td>8.95</td>
<td>19.0</td>
<td>7.39</td>
</tr>
<tr>
<td>12.0</td>
<td>5.25</td>
<td>9.1</td>
<td>3.98</td>
<td>20.4</td>
<td>8.91</td>
<td>16.5</td>
<td>7.22</td>
</tr>
<tr>
<td>22.8</td>
<td>2.22</td>
<td>49.0</td>
<td>4.76</td>
<td>42.5</td>
<td>4.13</td>
<td>41.0</td>
<td>3.99</td>
</tr>
<tr>
<td>21.1</td>
<td>3.08</td>
<td>41.0</td>
<td>5.98</td>
<td>37.6</td>
<td>5.48</td>
<td>35.0</td>
<td>5.10</td>
</tr>
<tr>
<td>22.8</td>
<td>4.43</td>
<td>41.4</td>
<td>8.05</td>
<td>39.9</td>
<td>7.75</td>
<td>42.5</td>
<td>8.26</td>
</tr>
<tr>
<td>23.4</td>
<td>5.69</td>
<td>40.5</td>
<td>9.84</td>
<td>40.5</td>
<td>9.83</td>
<td>38.0</td>
<td>9.23</td>
</tr>
<tr>
<td>25.1</td>
<td>7.32</td>
<td>36.7</td>
<td>10.7</td>
<td>40.2</td>
<td>11.71</td>
<td>37.5</td>
<td>10.94</td>
</tr>
<tr>
<td>24.8</td>
<td>8.44</td>
<td>30.2</td>
<td>10.3</td>
<td>40.8</td>
<td>13.88</td>
<td>38.5</td>
<td>13.10</td>
</tr>
<tr>
<td>24.2</td>
<td>9.41</td>
<td>24.6</td>
<td>9.56</td>
<td>41.1</td>
<td>15.99</td>
<td>36.0</td>
<td>14.0</td>
</tr>
<tr>
<td>25.1</td>
<td>10.98</td>
<td>25.0</td>
<td>10.94</td>
<td>41.9</td>
<td>18.33</td>
<td>37.0</td>
<td>16.18</td>
</tr>
</tbody>
</table>
Table 2c. Fuel Flow Measurements from Engine ($p=0.81$).

<table>
<thead>
<tr>
<th>Cmd Fuel (ul)</th>
<th>Cmd Fuel (kg/hr)</th>
<th>SenProc (ul)</th>
<th>SenProc (kg/hr)</th>
<th>AVL mtr (ul)</th>
<th>AVL mtr (kg/hr)</th>
<th>SenProc Diff (Eng-TS %)</th>
<th>AVL Diff (Eng-TS %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.1</td>
<td>2.15</td>
<td>51.3</td>
<td>4.99</td>
<td>38.7</td>
<td>3.76</td>
<td>8.4</td>
<td>-4.8</td>
</tr>
<tr>
<td>19.4</td>
<td>2.83</td>
<td>45.5</td>
<td>6.63</td>
<td>35.3</td>
<td>5.15</td>
<td>13.4</td>
<td>1.2</td>
</tr>
<tr>
<td>18.6</td>
<td>3.62</td>
<td>41.3</td>
<td>8.03</td>
<td>33.5</td>
<td>6.52</td>
<td>13.1</td>
<td>-1.8</td>
</tr>
<tr>
<td>18.1</td>
<td>4.40</td>
<td>38.7</td>
<td>9.40</td>
<td>32.8</td>
<td>7.97</td>
<td>17.8</td>
<td>3.8</td>
</tr>
<tr>
<td>17.5</td>
<td>5.10</td>
<td>38.2</td>
<td>11.14</td>
<td>32.4</td>
<td>9.44</td>
<td>22.5</td>
<td>4.3</td>
</tr>
<tr>
<td>16.6</td>
<td>5.65</td>
<td>36.4</td>
<td>12.38</td>
<td>30.0</td>
<td>10.22</td>
<td>31.6</td>
<td>-2.7</td>
</tr>
<tr>
<td>14.9</td>
<td>5.79</td>
<td>30.7</td>
<td>11.94</td>
<td>24.6</td>
<td>9.56</td>
<td>44.3</td>
<td>6.4</td>
</tr>
<tr>
<td>12.9</td>
<td>5.64</td>
<td>26.8</td>
<td>11.72</td>
<td>22.5</td>
<td>9.83</td>
<td>66.0</td>
<td>9.4</td>
</tr>
<tr>
<td>23.1</td>
<td>2.25</td>
<td>55.5</td>
<td>5.39</td>
<td>41.3</td>
<td>4.01</td>
<td>11.7</td>
<td>-3.0</td>
</tr>
<tr>
<td>21.1</td>
<td>3.08</td>
<td>51.0</td>
<td>7.44</td>
<td>37.3</td>
<td>5.44</td>
<td>19.6</td>
<td>-0.7</td>
</tr>
<tr>
<td>22.9</td>
<td>4.45</td>
<td>49.3</td>
<td>9.58</td>
<td>40.0</td>
<td>7.77</td>
<td>16.0</td>
<td>0.3</td>
</tr>
<tr>
<td>23.4</td>
<td>5.69</td>
<td>47.2</td>
<td>11.47</td>
<td>39.6</td>
<td>9.63</td>
<td>14.2</td>
<td>-2.1</td>
</tr>
<tr>
<td>24.9</td>
<td>7.26</td>
<td>49.3</td>
<td>14.38</td>
<td>41.7</td>
<td>12.16</td>
<td>25.6</td>
<td>3.7</td>
</tr>
<tr>
<td>24.9</td>
<td>8.47</td>
<td>52.8</td>
<td>17.96</td>
<td>43.1</td>
<td>14.66</td>
<td>42.8</td>
<td>5.3</td>
</tr>
<tr>
<td>24.9</td>
<td>9.49</td>
<td>49.0</td>
<td>19.05</td>
<td>40.9</td>
<td>15.89</td>
<td>49.8</td>
<td>-0.6</td>
</tr>
<tr>
<td>24.9</td>
<td>10.89</td>
<td>48.6</td>
<td>21.26</td>
<td>40.3</td>
<td>17.64</td>
<td>48.6</td>
<td>-3.9</td>
</tr>
</tbody>
</table>
Table 3. Cylinder-to-cylinder fuel distribution.

<table>
<thead>
<tr>
<th>RPM (engine)</th>
<th>Throttle</th>
<th>Cyl 1 (ul)</th>
<th>Cyl 2 (ul)</th>
<th>Cyl 3 (ul)</th>
<th>Cyl 4 (ul)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>half</td>
<td>40.0</td>
<td>37.0</td>
<td>38.0</td>
<td>37.0</td>
</tr>
<tr>
<td>1500</td>
<td>half</td>
<td>33.0</td>
<td>32.0</td>
<td>33.0</td>
<td>32.0</td>
</tr>
<tr>
<td>2000</td>
<td>half</td>
<td>32.0</td>
<td>32.0</td>
<td>32.5</td>
<td>32.0</td>
</tr>
<tr>
<td>2500</td>
<td>half</td>
<td>29.5</td>
<td>29.5</td>
<td>31.5</td>
<td>30.0</td>
</tr>
<tr>
<td>3000</td>
<td>half</td>
<td>29.0</td>
<td>29.0</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>3500</td>
<td>half</td>
<td>26.5</td>
<td>29.0</td>
<td>30.0</td>
<td>29.5</td>
</tr>
<tr>
<td>4000</td>
<td>half</td>
<td>19.0</td>
<td>22.0</td>
<td>22.5</td>
<td>21.5</td>
</tr>
<tr>
<td>4500</td>
<td>half</td>
<td>16.5</td>
<td>20.0</td>
<td>20.5</td>
<td>20.0</td>
</tr>
<tr>
<td>1000</td>
<td>full</td>
<td>41.0</td>
<td>39.0</td>
<td>39.0</td>
<td>39.0</td>
</tr>
<tr>
<td>1500</td>
<td>full</td>
<td>35.0</td>
<td>34.0</td>
<td>35.0</td>
<td>34.0</td>
</tr>
<tr>
<td>2000</td>
<td>full</td>
<td>42.5</td>
<td>42.0</td>
<td>42.5</td>
<td>42.5</td>
</tr>
<tr>
<td>2500</td>
<td>full</td>
<td>38.0</td>
<td>37.5</td>
<td>38.5</td>
<td>38.0</td>
</tr>
<tr>
<td>3000</td>
<td>full</td>
<td>37.5</td>
<td>37.0</td>
<td>38.5</td>
<td>37.5</td>
</tr>
<tr>
<td>3500</td>
<td>full</td>
<td>38.5</td>
<td>38.0</td>
<td>38.5</td>
<td>38.5</td>
</tr>
<tr>
<td>4000</td>
<td>full</td>
<td>36.0</td>
<td>39.5</td>
<td>39.5</td>
<td>39.0</td>
</tr>
<tr>
<td>4500</td>
<td>full</td>
<td>37.0</td>
<td>40.5</td>
<td>40.5</td>
<td>39.5</td>
</tr>
</tbody>
</table>
Figure 1  Cross-sectional View, Optimized Injector/Flow Transducer
Figure 2  Block Diagram of Experimental Apparatus
Figure 3  Closed Loop Adaptive Fuel Control Using Fuel Quantity Feedback
Figure 4a Laboratory Injector Test Apparatus

Figure 4b Bosch Reference Injection Rate Apparatus
**% Difference in Volume Between Numerically Integrated Volume from Raw Rate Reference Data and Measured Burette Volume for Bosch Injector**

<table>
<thead>
<tr>
<th>RPM</th>
<th>22</th>
<th>45</th>
<th>67</th>
<th>90</th>
<th>112</th>
<th>135</th>
<th>157</th>
<th>180</th>
</tr>
</thead>
<tbody>
<tr>
<td>375</td>
<td>3.129</td>
<td>-0.578</td>
<td>1.406</td>
<td>-1.759</td>
<td>-1.765</td>
<td>-3.606</td>
<td>-2.099</td>
<td>1.263</td>
</tr>
<tr>
<td>500</td>
<td>**</td>
<td>**</td>
<td>-1.705</td>
<td>-0.297</td>
<td>-5.326</td>
<td>-6.039</td>
<td>-5.410</td>
<td>-2.505</td>
</tr>
<tr>
<td>625</td>
<td>**</td>
<td>**</td>
<td>-0.882</td>
<td>-1.293</td>
<td>-4.324</td>
<td>-1.137</td>
<td>-2.784</td>
<td>-4.739</td>
</tr>
<tr>
<td>750</td>
<td>**</td>
<td>**</td>
<td>3.363</td>
<td>1.635</td>
<td>-2.991</td>
<td>-5.850</td>
<td>0.359</td>
<td>-2.714</td>
</tr>
<tr>
<td>875</td>
<td>**</td>
<td>**</td>
<td>3.267</td>
<td>-0.765</td>
<td>-4.902</td>
<td>-5.832</td>
<td>-4.233</td>
<td>-2.470</td>
</tr>
<tr>
<td>1000</td>
<td>**</td>
<td>**</td>
<td>3.751</td>
<td>-3.508</td>
<td>-10.767</td>
<td>-9.421</td>
<td>0.848</td>
<td>-6.480</td>
</tr>
<tr>
<td>1125</td>
<td>**</td>
<td>**</td>
<td>-3.510</td>
<td>-0.910</td>
<td>2.134</td>
<td>-10.161</td>
<td>2.757</td>
<td>-4.809</td>
</tr>
<tr>
<td>1250</td>
<td>**</td>
<td>**</td>
<td>-4.699</td>
<td>0.843</td>
<td>-6.554</td>
<td>-5.544</td>
<td>2.274</td>
<td>-5.379</td>
</tr>
<tr>
<td>1375</td>
<td>**</td>
<td>**</td>
<td>-3.582</td>
<td>-11.832</td>
<td>-1.817</td>
<td>-2.043</td>
<td>-3.736</td>
<td>-8.463</td>
</tr>
<tr>
<td>1625</td>
<td>**</td>
<td>**</td>
<td>-0.795</td>
<td>-3.377</td>
<td>-11.372</td>
<td>-1.487</td>
<td>-6.604</td>
<td>-0.194</td>
</tr>
</tbody>
</table>

** Indicates that buret volume was too small to accurately measure

Average magnitude of % difference = 4.816

Figure 5  Percentage Difference Between Volume Calculated by Integration of Reference Rate Data and Actual Burette Volume (RB Injector).
Nonlinear Corrective Function Applied to Reference Rate Data

Figure 6  Nonlinear Corrective Function Applied to Reference Rate Data
Comparison of Corrected and Non-corrected Reference Rates

Figure 7 Comparison of Corrected and Non-corrected Reference Rates
Figure 8  Simplified Flow Model for Injector Nozzle
Figure 9  Nozzle Flow vs Needle Position, Parametric with Pump Speed
Figure 10 $f_1(x)$ Flow Function, RB Injector
Figure 11  $f_1(x)$ Flow Function, DK Injector
RB Injector Response: 1000 rpm, 180 degrees throttle

Figure 12 RB Injector Response: 1000 Pump RPM, Full Throttle
Figure 13 RB Injector Response: 2000 Pump RPM, Full Throttle

Injector Needle Lift and Rate vs Time

RB Injector Response: 2000 rpm, 180 degrees throttle

Static Model Rate (vol. = 24.98 ul)
Dynamic Model Rate (vol. = 25.02 ul)
Reference Rate (vol. = 24.14 ul)

Needle Lift

Measured Burette (vol. = 25.50 ul)
MEASURED BURETTE VOLUME: 31.60 μl

- needle lift
- --- corrected raw rate (vol. = 33.08 μl)
- --- static map rate (vol. = 32.95 μl)
- --- --- dynamic model rate (vol. = 31.76 μl)
- uncorrected raw rate (vol. = 30.42 μl)

Injector Response: dk-1125-180d

Figure 14 DK Injector Response: 1125 Pump RPM, Full Throttle
MEASURED BURETTE VOLUME: 33.90 ul

- Needle lift
- Corrected raw rate (vol. = 32.69 ul)
- Static map rate (vol. = 37.70 ul)
- Dynamic model rate (vol. = 36.36 ul)
- Uncorrected raw rate (vol. = 28.90 ul)

Injector Response: dk-1500-180d

Figure 15 DK Injector Response: 1500 Pump RPM, Full Throttle
### % Difference in Volume Between Volume of Flow Rate Curve Based on Map Function of Needle Lift and Measured Buret Volume for BOSCH Injector

**Throttle (degrees)**

<table>
<thead>
<tr>
<th>RPM</th>
<th>22</th>
<th>45</th>
<th>67</th>
<th>90</th>
<th>112</th>
<th>135</th>
<th>157</th>
<th>180</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>-1.714</td>
<td>-1.138</td>
<td>-4.623</td>
<td>4.767</td>
<td>-1.665</td>
<td>-0.803</td>
<td>7.992</td>
<td>5.465</td>
</tr>
<tr>
<td>250</td>
<td>-5.054</td>
<td>1.104</td>
<td>-2.429</td>
<td>3.657</td>
<td>-0.228</td>
<td>-1.800</td>
<td>1.774</td>
<td>4.645</td>
</tr>
<tr>
<td>375</td>
<td>-3.249</td>
<td>-0.271</td>
<td>-2.521</td>
<td>2.267</td>
<td>-3.101</td>
<td>-3.101</td>
<td>-1.135</td>
<td>1.079</td>
</tr>
<tr>
<td>500</td>
<td>** **</td>
<td>3.187</td>
<td>2.142</td>
<td>-4.249</td>
<td>-4.909</td>
<td>2.749</td>
<td>-0.601</td>
<td></td>
</tr>
<tr>
<td>625</td>
<td>** **</td>
<td>10.684</td>
<td>2.441</td>
<td>-1.826</td>
<td>-3.395</td>
<td>-1.194</td>
<td>-1.913</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>** **</td>
<td>8.099</td>
<td>4.974</td>
<td>-1.879</td>
<td>-4.691</td>
<td>-3.235</td>
<td>-2.727</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>** **</td>
<td>10.496</td>
<td>4.877</td>
<td>-0.599</td>
<td>-4.163</td>
<td>0.634</td>
<td>-2.704</td>
<td></td>
</tr>
<tr>
<td>1125</td>
<td>** **</td>
<td>2.817</td>
<td>12.383</td>
<td>-0.184</td>
<td>-7.298</td>
<td>2.739</td>
<td>-2.245</td>
<td></td>
</tr>
<tr>
<td>1250</td>
<td>** **</td>
<td>3.023</td>
<td>13.470</td>
<td>2.184</td>
<td>-5.064</td>
<td>1.021</td>
<td>-2.242</td>
<td></td>
</tr>
<tr>
<td>1375</td>
<td>** **</td>
<td>5.848</td>
<td>8.568</td>
<td>.380</td>
<td>-7.005</td>
<td>-1.448</td>
<td>-6.482</td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>** **</td>
<td>17.811</td>
<td>10.030</td>
<td>-.722</td>
<td>-5.398</td>
<td>-4.539</td>
<td>-5.625</td>
<td></td>
</tr>
<tr>
<td>1625</td>
<td>** **</td>
<td>18.102</td>
<td>-9.190</td>
<td>.662</td>
<td>-2.357</td>
<td>-7.950</td>
<td>-7.928</td>
<td></td>
</tr>
<tr>
<td>1750</td>
<td>** **</td>
<td>8.724</td>
<td>10.404</td>
<td>.525</td>
<td>-3.323</td>
<td>-5.099</td>
<td>-9.099</td>
<td></td>
</tr>
<tr>
<td>1875</td>
<td>** **</td>
<td>15.362</td>
<td>-5.517</td>
<td>3.562</td>
<td>-5.343</td>
<td>-8.551</td>
<td>-10.735</td>
<td></td>
</tr>
</tbody>
</table>

**Indicates that buret volume was too small to accurately measure**

Average magnitude of % difference = 4.051

---

**Figure 16** Percentage Difference Between Volume Calculated by Static Nonlinear Model and Actual Burette Volume (RB Injector)
Figure 17 Sensor Processor Software Flowchart - Basic Modules
Figure 18 Calculation of Total Fuel Delivery from the Needle Lift Trace Using Static Nonlinear Algorithm
Figure 19 Sensor Processor - System Diagram
Figure 20 Functional Block Diagram - Dual Processor Diesel Engine Control System
Figure 21 Photograph of Prototype Sensor Processor and Interface/Control Computer Assembly
Figure 22 Phantom View of COVEC Injection Pump
Figure 23 Diesel Test Engine on Dynamometer in Test Cell

Figure 24 Sensor Processor, Interface/Control Computer, and Related Electronics Outside Test Cell
Figure 25a Needle lift trace for Injector on Test Stand, 4000 Engine RPM, Full Throttle

Figure 25b Needle Lift Trace for Injector on Engine, Under Conditions Identical to those of Figure 25a
Figure 26 Emissions Control Test: Smoke and Torque, with and without Sensor Processor Control
Figure 27 Degradation of $f_1(x)$ Calibration due to Nozzle Coking, RB (Volkswagen) Injector
17. APPENDIX

1. SAE Paper "An Indirect Sensing Technique for Closed-Loop Diesel Fuel Quantity Con-
trol", presented at 1990 SAE International Congress, Detroit, MI. (Also appears in 1990 SAE Transactions.)

2. Research Schedule and Task Delineations.

3. FORTRAN Source Code for Computer Simulation of Injection System Hydraulics


5. Static Nonlinear Map for DK Injector used for Engine Tests.


7. MC68000 Assembly Language Listing of Firmware for Sensor Processor


9. MC68HC11 Assembly Language Listing of Firmware for Interface/Control Computer
1. SAE Paper "An Indirect Sensing Technique for Closed-Loop Diesel Fuel Quantity Control", presented at 1990 SAE International Congress, Detroit, MI. (Also appears in 1990 SAE Transactions.)

SAE Technical Paper Series

900494

An Indirect Sensing Technique for Closed-Loop Diesel Fuel Quantity Control

Carl A. MacCarley, Walter D. Clark and Keay T. Nakae
Electrical and Electronic Engineering Dept.
California State Polytechnic Univ.
San Luis Obispo, CA

Reprinted from SP-805 – Sensors and Actuators 1990

International Congress and Exposition
Detroit, Michigan
February 26 — March 2, 1990
An Indirect Sensing Technique for Closed-Loop Diesel Fuel Quantity Control

Carl A. MacCarley, Walter D. Clark and Keay T. Nakae
Electrical and Electronic Engineering Dept.
California State Polytechnic Univ.
San Luis Obispo, CA

ABSTRACT

Despite significant advances in electronic control technology applied to diesel engines, commercially available injection systems for automotive diesel engines remain limited by the open-loop mapping of the injection pump. An initial calibration is relied upon to translate a fuel delivery command to an actual fuel quantity. In practice, however, these two variables may be substantially different due to the effects of mechanical wear, repair, and the wide range of operating conditions. Possible ramifications of this discrepancy are excessive exhaust smoke due to overfueling, increased fuel consumption, degraded driveability, and poor idle characteristics. The major obstacle to closing the fuel control loop is the lack of a suitable sensor for instantaneous fuel delivery from the injector.

An indirect fuel delivery sensing mechanism based upon the use of the injector needle lift in conjunction with the fuel temperature is evaluated. An estimation of the injection rate characteristic is determined from real-time analysis of the needle lift signal using a high-speed sensor processor. Integration of the rate characteristic and temperature correction yields a total mass delivery estimate for use as a feedback quantity for closed-loop fuel control. Signal processing algorithms are derived from computer modeling of the injector and verified experimentally. Possible long-term decalibration due to nozzle coking is studied. Advantages and limitations of the technique are identified.

BACKGROUND AND PROBLEM DESCRIPTION

Although diesel engines and injection systems represent a mature technology, it is only in recent years that electronic controls have been successfully applied. The majority of diesel applications are still mechanically controlled, with no electronics involved other than the fuel shutoff solenoid valve control.

The potential benefits of microprocessor-based control applied to diesel engines have been well established [Reams82, Kihara83, Martinsons82, Trenne82, Kawai84]. However, many of the improvements made possible by advanced electronic control are dependent upon exact knowledge by the controller of the hydraulic characteristics of the injection system components.

This dependency is particularly important in the case of small-displacement automotive diesel engines, which use low-cost distributor pumps which must accurately meter very small (less than 50 mm³ per injection) fuel quantities at high speeds.

Major incentives for more accurate fuel control have appeared in the form of recent regulatory pressures for "cleaner" diesels along with the demands of the automotive market for driving characteristics more like those of gasoline fueled engines. Gradually increasing concerns about rising gasoline costs may be expected to produce a renewed interest in automotive diesels.

Available and currently envisioned electronically controlled distributor-type fuel injection pumps for automotive diesel engines generally operate with a map-based translation between commanded fuel quantity and actual fuel delivery. A ROM-stored multidimensional map is typically accessed with inputs of commanded fuel volume and pump speed. A third parameter, fuel or pump housing temperature, may be additionally used to modify the table output to yield a corrected fuel control position corresponding to a given commanded fuel mass. The map and correction factor(s) are generated experimentally from pump test data, typically using a reference pump. A fully specified map of adequate resolution and valid correction factors may require the acquisition of a large number of data points on a pump test stand.

There are several limitations of this open loop fuel control method. The generation of individual calibration maps for each pump, and subsequent storage of individual maps in ROM, is impractical in production. At best, a linear correction for pump misalignment is performed during final checkout of individual pumps, using either an external resistor network [Stumpp83] or final PROM programming procedure. However, minor machining differences, even within manufacturing tolerances, can cause noticeable differences between the calibration maps of individual production pumps and the test pump. This difference is compounded by the synergistic relationship between the injectors and the pump. The injectors fitted to a particular production pump may differ slightly in their flow characteristics from those used with the test pump during the master calibration, thus changing the overall calibration. One or more injectors might also be replaced or readjusted at a later date.
Possibly more important is the problem that the injection system often operates under conditions much different than those that existed during the master calibration tests. The calibration also changes over the course of time due to normal wear and corrosion of pump and injector internal components, and the accumulation of carbon deposits in the injector nozzle (nozzle coking).

In actual service, the delivered fuel quantity may differ substantially from the mapped quantity. Possible effects of this discrepancy are excessive exhaust smoke due to overfueling, inaccurate torque limiting, increased fuel consumption, degraded driveability, and unstable or noisy idle characteristics.

If the actual fuel delivery per injection were directly sensible, closed-loop control of the fuel quantity would be possible as a means for improving the fuel control accuracy without the need for further mechanical refinements and tighter tolerances in the pump. The major obstacle to closing the fuel control loop appears to be the lack of a suitable sensor for fuel quantity.

Real-time monitoring of the actual fuel delivery per individual injection stroke is difficult. A number of factors can be cited in relation to this technical obstacle. For a typical small displacement (i.e., under two liter) diesel engine, the fuel delivery volume is very small (in the range of 5 to 50 mm³ per stroke), and the duration very brief (on the order of 1.0 ms). The repetition rate per cylinder is typically from 5 to 50 injections per second, with line pressure fluctuating from the delivery valve opening pressure to the injection peak pressure (as high as 100 MPa) at this repetition rate. The static volume in each fuel injection line and the secondary passages of the pump may exceed the fuel delivery per injection by a large factor. Fluid compressibility, inertial effects, tubing strain, and internal leakage in the pump make the process non-ideal, so that the actual fuel delivery often differs significantly from the metered plunger displacement in the pump. Mass transport in this medium is characterized by propagation of a pressure wave between the pumping chamber and the injector nozzle. Computer simulation of the injection pump hydraulics using finite difference methods is often relied upon to predict the delivered quantity and injection characteristics [Oren83, Kumar83, Sharma83].

While suitable sensors for the rate characteristic have been suggested [Bosch66, Komaroff66, Thoma74] for use in test bench calibration of pumps, a practical sensor suitable for use during actual engine operation as a real-time feedback control device is not, to the best of our knowledge, currently available. Methods utilizing injection pulse duration in conjunction with engine speed to estimate fuel usage have been investigated [Wolff86]. A thermal convection based fuel flow sensor has also been studied [Challen88].

The advantages of closed loop control in general are well established. Efforts to close the control loop on engine torque [Ribbens81, Fleming82, Sood84], combustion luminescence [Bunting84], and cylinder pressure [Challen88] have been important recent contributions to diesel control technology. All of these techniques may be considered as indirect indicators of the injected fuel quantity, which for some performance metrics (i.e., emissions) is the target variable of primary importance.

**EXPERIMENTAL METHOD**

The reported work investigates the feasibility of using information contained in the motion of the injector needle in combination with other sensor signals, to infer the delivered fuel quantity, rate characteristic, and timing of critical portions of the delivery schedule on an individual injection basis. It is hypothesized that the availability of these metrics in real time could facilitate improvements in engine control, with resultant improvements possible in engine emissions, efficiency, driveability and noise.

Needle valve position is often used as an approximate qualitative indicator of the injection rate characteristic [Burman62(1), Hiroyasu80, Obert68(1)]. The beginning and end of needle lift are recognized as the respective start and end of the injection pulse. Start of needle lift is usually used as the injection timing reference event. The use of a needle lift sensor for closed loop injection timing control is common practice [Stumpf83, Ives84, Wolff82]. Pre- and post- injections are also identified by the needle lift. The duration of the injection period, measured using the needle lift signal, has been used as an approximation for the fuel quantity [Stumpf83], and in conjunction with a engine speed has been used to map fuel consumption for a specific engine and pump [Wolff86]. However, the displacement of the needle is, at best, a very nonlinear indicator of actual fuel flow rate through actual engine operation as a real-time feedback control device is not, to the best of our knowledge, currently available. Methods utilizing injection pulse duration in conjunction with engine speed to estimate fuel usage have been investigated [Wolff86]. A thermal convection based fuel flow sensor has also been studied [Challen88].
the injection nozzle. Inertial effects on both the moving parts and the fluid also distort the displacement-flow relationship.

Our preliminary studies indicate that for a restricted class of injectors and subject to certain restrictions applied to the injector design, it appears possible that with appropriate signal analysis, the needle lift trace can be used to characterize the injection rate history, and therefore the total fuel delivery.

The rapid high-speed signal analysis needed to accomplish this indirect sensing task in real-time requires specialized data acquisition and signal analysis hardware. Specific hardware requirements, and the design of a prototype "sensor processor" for this task were described in an earlier report by MacCarley and Meyer [MacCarley87].

Two types of common automotive IDI injectors were instrumented for needle lift and fuel temperature, as illustrated in the cross-sectional diagram of Figure 1. Several possible injector flow models were derived, based on practical as well as accuracy considerations. Experiments were conducted on a pump test stand to calibrate and test the models, directed toward correlating instantaneous nozzle flow with the needle motion and fuel temperature. An apparatus was constructed for reference measurement of the fuel injection rate history, to serve as a tool for model development and evaluation. High-speed data acquisition and signal analysis equipment were used to calibrate and test the static and dynamic flow models for the test injectors over the range of operational conditions. A block diagram of the experimental apparatus is shown in Figure 2. A finite-element hydraulic computer simulation of the pump and injector was also developed to aid in the modeling process of this highly nonlinear system.

The objective was to determine if an inverse injector model could be generally found, which when implemented as a signal processing algorithm could be used to determine some or all of the above-stated metrics with sufficient accuracy and reliability for use as feedback control signals. Engine control algorithms were also studied that could optimally utilize the estimated metrics. A potential closed loop control strategy is suggested in Figure 3, which includes both real-time error reduction, and long term adaptation using nonvolatile random access memory.

The on-engine testing of engine controls based on this method is currently in progress, although results were not yet available at the time of submission of this paper. Of particular interest is the long-term reliability of this sensing method, in view of possible degradation of the injector/sensor due to nozzle coking and internal wear. Cylinder-to-cylinder variations in the injection characteristics and the relative benefit of instrumenting more than one cylinder for fuel flow are also being assessed.

REFERENCE RATE MEASUREMENT APPARATUS

The model development and evaluation process required an accurate reference for the measurement of the injection rate history. A laboratory apparatus was proposed by Wilhelm Bosch [Bosch66] which has been used by other investigators as an accepted standard for injection rate measurement [Kumar83, Kamiyama78]. We fabricated an apparatus conforming to Bosch's specifications for reference rate measurement.

The Bosch apparatus operates on the principle of propagation of a pressure wave through a fluid column. The injector discharges directly into a fluid-filled tube of constant diameter and known length. The passage of the resulting pressure wave past a pressure transducer section of the tube provides a pressure signal that is representative of the instantaneous injection flowrate into the column, delayed slightly by the propagation time from the nozzle to the transducer section. By locating the pressure transducer section close to the nozzle, the delay is negligible. The pressure

![Figure 2 Block Diagram of Experimental Apparatus](image)

![Figure 3 Closed Loop Adaptive Fuel Control Using Fuel Quantity Feedback](image)
wave is attenuated as it is reflected back and forth along the length. Fluid injected into the measurement tube is relieved through a needle valve and pressure relief valve at the end of the tube. The relationship between the pressure of the wave and the injector flowrate is:

\[
\dot{q} = \frac{f_p P}{10 a p} \quad \text{cm}^3 \text{sec}^{-1}
\]

where

- \( f_p \) = flow area of measurement tube (cm²)
- \( a \) = acoustic velocity in diesel fuel (m/sec)
- \( p \) = density of diesel fuel (gm/cm³)
- \( P \) = differential pressure (Pa)

The product \( ap \) is referred to as the acoustic impedance. For diesel fuel at the experimental conditions we used \( a = 1250 \text{ m/sec} \) and \( p = 0.84 \text{ gm/cm}^3 \). The inner flow area \( f_p \) of the measurement tube was 0.317 cm². The pressure transducer must not cause any reflections, so pressure is measured by a strain gauge attached to the outside of a thin-wall section of the measurement tube, referred to as the pressure transducer section. Hoop strain in the tube serves as the mechanism for measurement of the passing wavefront pressure. The quiescent pressure in the measurement tube is maintained by the pressure relief valve at approximately peak cylinder compression pressure to simulate the actual backpressure encountered in an engine installation of the injector.

The length of the measurement tube is critical, since reflected pulses can interfere with the primary wavefront. Interference is selective with pump speed. Bosch prescribed the use of two different length measurement tubes to handle the complete RPM range, 4.67 and 9.34 meters. In order to provide sufficient attenuation of the primary reflex pulse to avoid interference at all speeds, we found it necessary to use a tube 80.16 meters long.

Our apparatus was fitted to accept two types of nozzle holders: Robert Bosch KCA series (RB) and Diesel-Kiki Type 71-1280 (DK). The nozzle holders were fitted internally with Hall-effect needle position sensors manufactured by Wolff Controls. Both nozzle holders work with pintle-type nozzles intended for indirect injection applications. All tests were performed using a Diesel-Kiki (Robert Bosch licensed) NP-VYQ injection pump mounted on a Bacharach Type YYQ pump test stand.

Total fuel delivery per injection may be calculated by integration of reference rate data and actual burette volume.

**STATIC NONLINEAR MODEL**

A simplified injector flow model was previously proposed [MacCarley87] which essentially ignores dynamic effects and assigns a one-to-one mapping between needle position and instantaneous flow rate, modified only by a density factor dependent on fuel temperature. A basic requirement of this model is that the

The product \( ap \) is referred to as the acoustic impedance. For diesel fuel at the experimental conditions we used \( a = 1250 \text{ m/sec} \) and \( p = 0.84 \text{ gm/cm}^3 \). The inner flow area \( f_p \) of the measurement tube was 0.317 cm². The pressure transducer must not cause any reflections, so pressure is measured by a strain gauge attached to the outside of a thin-wall section of the measurement tube, referred to as the pressure transducer section. Hoop strain in the tube serves as the mechanism for measurement of the passing wavefront pressure. The quiescent pressure in the measurement tube is maintained by the pressure relief valve at approximately peak cylinder compression pressure to simulate the actual backpressure encountered in an engine installation of the injector.

The length of the measurement tube is critical, since reflected pulses can interfere with the primary wavefront. Interference is selective with pump speed. Bosch prescribed the use of two different length measurement tubes to handle the complete RPM range, 4.67 and 9.34 meters. In order to provide sufficient attenuation of the primary reflex pulse to avoid interference at all speeds, we found it necessary to use a tube 80.16 meters long.

Our apparatus was fitted to accept two types of nozzle holders: Robert Bosch KCA series (RB) and Diesel-Kiki Type 71-1280 (DK). The nozzle holders were fitted internally with Hall-effect needle position sensors manufactured by Wolff Controls. Both nozzle holders work with pintle-type nozzles intended for indirect injection applications. All tests were performed using a Diesel-Kiki (Robert Bosch licensed) NP-VYQ injection pump mounted on a Bacharach Type YYQ pump test stand.

Total fuel delivery per injection may be calculated by integration of reference rate data and actual burette volume.

**STATIC NONLINEAR MODEL**

A simplified injector flow model was previously proposed [MacCarley87] which essentially ignores dynamic effects and assigns a one-to-one mapping between needle position and instantaneous flow rate, modified only by a density factor dependent on fuel temperature. A basic requirement of this model is that the

The product \( ap \) is referred to as the acoustic impedance. For diesel fuel at the experimental conditions we used \( a = 1250 \text{ m/sec} \) and \( p = 0.84 \text{ gm/cm}^3 \). The inner flow area \( f_p \) of the measurement tube was 0.317 cm². The pressure transducer must not cause any reflections, so pressure is measured by a strain gauge attached to the outside of a thin-wall section of the measurement tube, referred to as the pressure transducer section. Hoop strain in the tube serves as the mechanism for measurement of the passing wavefront pressure. The quiescent pressure in the measurement tube is maintained by the pressure relief valve at approximately peak cylinder compression pressure to simulate the actual backpressure encountered in an engine installation of the injector.

The length of the measurement tube is critical, since reflected pulses can interfere with the primary wavefront. Interference is selective with pump speed. Bosch prescribed the use of two different length measurement tubes to handle the complete RPM range, 4.67 and 9.34 meters. In order to provide sufficient attenuation of the primary reflex pulse to avoid interference at all speeds, we found it necessary to use a tube 80.16 meters long.

Our apparatus was fitted to accept two types of nozzle holders: Robert Bosch KCA series (RB) and Diesel-Kiki Type 71-1280 (DK). The nozzle holders were fitted internally with Hall-effect needle position sensors manufactured by Wolff Controls. Both nozzle holders work with pintle-type nozzles intended for indirect injection applications. All tests were performed using a Diesel-Kiki (Robert Bosch licensed) NP-VYQ injection pump mounted on a Bacharach Type YYQ pump test stand.

Total fuel delivery per injection may be calculated by integration of reference rate data and actual burette volume.

**STATIC NONLINEAR MODEL**

A simplified injector flow model was previously proposed [MacCarley87] which essentially ignores dynamic effects and assigns a one-to-one mapping between needle position and instantaneous flow rate, modified only by a density factor dependent on fuel temperature. A basic requirement of this model is that the

The product \( ap \) is referred to as the acoustic impedance. For diesel fuel at the experimental conditions we used \( a = 1250 \text{ m/sec} \) and \( p = 0.84 \text{ gm/cm}^3 \). The inner flow area \( f_p \) of the measurement tube was 0.317 cm². The pressure transducer must not cause any reflections, so pressure is measured by a strain gauge attached to the outside of a thin-wall section of the measurement tube, referred to as the pressure transducer section. Hoop strain in the tube serves as the mechanism for measurement of the passing wavefront pressure. The quiescent pressure in the measurement tube is maintained by the pressure relief valve at approximately peak cylinder compression pressure to simulate the actual backpressure encountered in an engine installation of the injector.

The length of the measurement tube is critical, since reflected pulses can interfere with the primary wavefront. Interference is selective with pump speed. Bosch prescribed the use of two different length measurement tubes to handle the complete RPM range, 4.67 and 9.34 meters. In order to provide sufficient attenuation of the primary reflex pulse to avoid interference at all speeds, we found it necessary to use a tube 80.16 meters long.

Our apparatus was fitted to accept two types of nozzle holders: Robert Bosch KCA series (RB) and Diesel-Kiki Type 71-1280 (DK). The nozzle holders were fitted internally with Hall-effect needle position sensors manufactured by Wolff Controls. Both nozzle holders work with pintle-type nozzles intended for indirect injection applications. All tests were performed using a Diesel-Kiki (Robert Bosch licensed) NP-VYQ injection pump mounted on a Bacharach Type YYQ pump test stand.

Total fuel delivery per injection may be calculated by integration of reference rate data and actual burette volume.
Ano is a unique function of the needle position \( x \). \( C_d \) is both a function of \( x \) and of the flow velocity, so that normally an iterative solution is necessary for the flow problem. It is reasonable to assume \( p \) to be dependent upon temperature \( T \) only, since fluid compressibility affects it only slightly, even at the high pressures involved [Bosch65]. It is impossible to determine the time history of \( p_2 \) without the use of a cylinder pressure transducer. If a pressure transducer is available to the feedback control system, exact knowledge of \( p_2 \) can be used in this calculation. Lacking such a transducer, a constant average value \( p_2 \) based upon test measurements may be used [Obert68(2)], with some loss of accuracy.

The linear movement of the needle may be modeled by:

\[
p_1 A_1 - (p_1 - p_2) A_2(x) = 10^{-3} M \ddot{x} + 10^3 k_f \dot{x} + 10^4 (k_{spr} x + f_0) \quad (3)
\]

where:
- \( M \) = combined mass of the needle, spindle, and part of the spring mass [gm].
- \( k_f \) = combined linear coefficient of friction [Newtons-sec/mm].
- \( k_{spr} \) = differential linear spring coefficient for small displacements [Newtons/mm].
- \( f_0 \) = nozzle opening force [Newtons].
- \( A_1 \) = upper valve piston area [mm²].
- \( A_2(x) \) = effective lower (counteracting) valve piston area [mm²].
- \( x \) = needle position, \( x > 0 \) [mm].
- \( \dot{x} \) = needle velocity [mm/sec].
- \( \ddot{x} \) = needle acceleration [mm/sec²].

A dynamic model for the nozzle flow must take into account the inertial and frictional effects on the moving parts, as well as the fact that the flowrate is rapidly changing. However, the use of (3) to calculate these effects requires generation of the first and second derivative from samples of the needle position. It was observed experimentally that noise amplification obscured the true dynamics, requiring filtering techniques to yield even an approximation to \( \ddot{x} \) or \( \dot{x} \).

A considerable simplification is possible by neglecting the inertial and frictional effects, so that pressure \( p_1 \) is characterized by \( x \) alone, and none of its derivatives. This is not unreasonable considering the extremely high force to mass ratio acting on the needle assembly. Errors introduced by this assumption tend to be self-canceling when the rate curve is integrated over the entire injection interval, especially if the curve is close to symmetric. The error is also reduced by minimization of the needle assembly mass and frictional contact area.

The final assumption that

\[
A_1 \gg A_2(x) \quad (4)
\]

allows the slight effect of the cylinder pressure to be ignored in the needle position equation. (Assumes fluid pressure acting on upper piston area \( A_1 \) only.) This is actually valid only after the nozzle is flowing, so that some ambiguity occurs during initial and final flow, reducing accuracy particularly at small deliveries.

The preceding assumptions allow \( p_1 \) to be expressed as a unique function of the needle position \( x \):

\[
p_1 = p_1(x) = \frac{10^5}{A_1} k_{spr} x + p_{no}, \quad x > 0, \quad p > p_{no} \quad (5)
\]

where \( p_{no} \) = nozzle opening pressure [kPa].

This permits the discharge coefficient to also be expressed in terms of \( x \) alone:

\[
C_d(x, \dot{m}) = C_d(x) \quad (7)
\]

so that (2) may be simplified to:

\[
\dot{m} = 10^3 A_{no}(x) C_d(x) \left[ 2(p_1(x) - p_2)p(T) \right]^{\frac{3}{4}} \quad (8)
\]

where \( T_0 = \) fuel temperature at which \( f_1 \) is measured.

Total fuel delivery is determined by time integration of \( \dot{m} \) over the duration of the injection period. Since \( T \) is slowly varying, it can be considered constant over the integration period.

\[
m = \frac{1}{4} \int m \, dt = \frac{1}{4} \int f_1(x) f_2(T) \, dt = f_2(T) \frac{1}{4} \int f_1(x) \, dt \quad (9)
\]

In practice \( f_1(x) \) is determined experimentally for a particular injector.

\[
f_1(x) = \dot{m}(x) \bigg|_{T=T_0} \quad (10)
\]

while \( f_2(T) \) is a known function for diesel fuel. This simple flow model is not valid in the case of a chattering or oscillating needle, known to occur under certain conditions in some injectors [Buman62(3)].

The reference rate measurement apparatus was used to generate the nonlinear map of \( f_1 \). The composite \( f_1 \) function is generated by averaging measured flow vs needle position data taken over the speed and throttle angle range of the pump. At each condition, 256 samples of the rate characteristic and needle lift curve are acquired and digitized. Eight throttle positions are tested at each of sixteen pump speeds, from 125 to 2000 RPM. Figure 6 illustrates the variation of the \( f_1 \) function with pump speed for a needle lift curve.
Diesel-Kiki 71-1280 injector. (The governor was operative, so that the fuel sleeve position was also changing with speed in Figure 6.) The rather minor variation in the $f_1$ manifold along the RPM axis is noted.

Figure 7 shows the overall average $f_1$ function for a Robert Bosch KCA30SD27/4 injector. Figure 8 shows the average $f_1$ for the Kiki injector.

This model requires only the nonlinear mapping of needle position into injector flowrate, temperature-density correction, and integration to yield total mass delivery per injection. The effectiveness of this approach is illustrated by Figure 9, which shows a typical needle lift curve (1500 pump RPM, 180 deg. throttle, Kiki injector) with corresponding rate curves derived from (1) the reference rate measurement apparatus, (2) the nonlinear mapping of the simplified static model, and (3) a dynamic model to be described later. Comparison of the rate curve generated by the static model with that obtained from the reference rate apparatus indicates that the flow estimate lags slightly the movement of the needle, underestimating flow while the nozzle is opening, and overestimating while it is closing.

Figure 10 is an error matrix similar to that of Figure 4, comparing the integrated fuel quantity calculated by the static model from the needle lift signal, with the burette volume from the test stand. Over the operational range, an average absolute error of 4.46% (RB) 4.72% (DK) is observed, with a peak error of 18.0% (RB) or 14.6% (DK) at medium-flow conditions in both cases. It may be concluded by comparison of this matrix with that of Figure 4 that the simplified static model provides a fuel quantity estimate of similar accuracy to that obtained from the reference rate apparatus (that was used to generate the $f_1$ function for the static model).
Several closed-form models were studied which incorporate inertial effects acting on the needle and fluid. The best results were obtained with a model developed as a modification of the simple static model.

From (3) and (4), \( p_1 \) may be written as the sum of static and dynamic contributions:

\[
p_1 = p_s(x) + p_d(x, \dot{x}).
\]

From (8),

\[
m = 10^3 A_{\text{no}}(x) C_d(x) \left[ 2 \rho(T_0) \left( p_s(x) + p_d(x, \dot{x}) - \rho_2 \right) \right]^{1/2} f_2(T)
\]

\[
= \left[ \left( 10^3 A_{\text{no}}(x) C_d(x) \right)^2 2 \rho(T_0) \left( p_s(x) - \rho_2 \right) + \left( 10^3 A_{\text{no}}(x) C_d(x) \right)^2 2 \rho(T_0) p_d(x, \dot{x}) \right]^{1/2} f_2(T)
\]

\[
= \left[ f_2(x) + f_2(x, \dot{x}) \right]^{1/2} f_2(T)
\]

where

\[
f_2(x, \dot{x}) = 10^3 A_{\text{no}}(x) C_d(x) \left[ 2 \rho(T_0) \left( p_s(x) - \rho_2 \right) \right]^{1/2} f_2(T)
\]

The product \( A_{\text{no}}(x) C_d(x) \) describes the nozzle as a function of the needle position. It is known to be a monotonic function that saturates at some maximum value of \( x \). Experimental determination of \( A_{\text{no}}(x) C_d(x) \) by continuous flow testing of the injector was unsuccessful due to the high flow at high pressure required for needle lift values above 0.5 mm (10 ml/sec., 20.7 MPa at 0.5 mm lift). An approximate function was used:

\[
A_{\text{no}}(x) C_d(x) = \alpha x \gamma \text{ [mm}^2\text{]}
\]

where \( \alpha \) and \( \gamma \) are model parameters (0 < \( \gamma \) < 1).

With this assumption, it is possible to extend the static model previously derived to include some dynamic (velocity and acceleration) effects.

\[
f_d(x, \dot{x}) = x^\gamma \left[ ax + b \dot{x} \right]^\gamma
\]

where

\[
a = \frac{2 \times 10^3 \alpha^2 \rho(T_0) k_t}{A_1^2} \left[ \frac{mg^2}{mm \cdot sec} \right]
\]

\[
b = \frac{2 \times 10^3 \alpha^2 \rho(T_0) M}{A_1^2} \left[ \frac{mg^2}{mm} \right]
\]

The mass flow relationship is therefore

\[
m = \left[ f_2(x) + x^\gamma \left[ ax + b \dot{x} \right]^\gamma \right]^{1/2} f_2(T)
\]

where \( w = 2\gamma, 0 < w < 2 \).

\( x \) and \( \dot{x} \) are found by numeric differentiation of the needle position. A four-period average derivative is used to suppress the noise amplification effects of the differentiation process. The parameters \( w, a \) and \( b \) were optimally determined by least squares fitting the model to the acquired rate vs needle lift histories over the operational speed and throttle range of the pump. For the DK

** Indicates that burette volume was too small to accurately measure

Average magnitude of \( x \) difference = 4.464

![Figure 9 Rate Characteristics Derived from Needle Lift by Static and Dynamic Models](image)

<table>
<thead>
<tr>
<th>RPM</th>
<th>THROTTLE (degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>250</td>
</tr>
<tr>
<td>45</td>
<td>750</td>
</tr>
<tr>
<td>67</td>
<td>100</td>
</tr>
<tr>
<td>90</td>
<td>1125</td>
</tr>
<tr>
<td>135</td>
<td>1250</td>
</tr>
<tr>
<td>157</td>
<td>1375</td>
</tr>
<tr>
<td>180</td>
<td>1500</td>
</tr>
</tbody>
</table>

** Figure 10 Percentage Difference Between Volume Calculated by Static Nonlinear Model and Actual Burette Volume**
injector, optimum values were found to be \( a = 3.99 \times 10^4 \), \( b = -4.80 \), \( w = 0.366 \).

Figure 9 illustrates the performance of this dynamic model relative to the static model and the reference rate trace. A slightly better correlation with the reference trace is observed. Integration under the curve to calculate quantity yields 30.3 mm\(^3\), which improves slightly upon the static model estimate (31.5 mm\(^3\)) of the measured volume fuel delivery (29.8 mm\(^3\)). For comparison, integration of the reference rate curve yielded a volume of 31.1 mm\(^3\). Over the complete operational range for the DK injector, the dynamic model yielded an average absolute percentage error of 3.7%, an improvement over the 4.7% error of the static model. Peak error was reduced from 14.6% to 10.6%, both occurring at 2000 RPM, 135 deg throttle (measured volume 20.3 mm\(^3\)).

### DEGRADATION DUE TO INJECTOR DECALIBRATION

The problem of nozzle coking, which occurs in all injectors to some degree, is expected to be a possible source of loss of calibration for this sensing mechanism. Long-term wear of the pintle and nozzle orifice may also contribute to a gradual loss of calibration of the injector. The result is a change in \( f_1 \) over time, which contributes to increased error of the sensing method.

Open nozzles, such as those commonly used in small displacement IDI automotive engines, are usually less prone to nozzle coking than closed, multi-hole nozzles found in larger, lower speed engines. Optimal injector/sensor design to minimize the influence of nozzle deposits on the flow calibration may reduce this concern to some degree. Adaptive signal processing techniques may also be applied to compensate for changes in the sensor calibration, if at least one other indirect indication of fuel delivery is available. One such comparison signal for long-term adaptation is driveshaft torque, sensed directly via magnetostrictive methods [Ribbens81, Fleming82] or indirectly via instantaneous crankshaft velocity as described by Sood [Sood84] and others. An adaptive corrective algorithm based on periodic comparison of the needle lift derived fuel quantity with a steady-state torque reference point is currently being studied.

The effects of nozzle coking on the long-term accuracy of methods utilizing needle lift to infer fuel delivery were studied by generating parallel bodies of data on the same injector, both clean and heavily coked. A Bosch KCA-series injector subjected to 188,000 km of continuous service was mapped over the previously defined operational range. The injector was then thoroughly cleaned and the mapping process repeated. Average \( f_1 \) functions for the nozzle in clean vs coked condition are overlaid in Figure 11. In clean condition, the static model produced an average absolute error of 6.86%. This poorer than usual accuracy may be partially attributed to the reference rate apparatus, which alone yielded an average absolute error of 7.02%. The "clean" \( f_1 \) function was then used to generate fuel delivery estimates from the "coked" injector needle lift data, simulating long-term degradation attributable to coking. An average absolute error of 6.05% was observed. The small difference (actually a slight improvement for this injector) is indicative of relatively minor calibration change attributable to nozzle deposits.

![Figure 11 Degradation of \( f_1(x) \) Calibration due to Nozzle Coking, RB (Volkswagen) Injector](image-url)

### CONCLUSIONS

The results of this work appear to confirm that for two specific types of IDI pintle nozzles, adequate information is contained in the needle lift signal to reasonably estimate the injection rate characteristic and total fuel delivery, independent of the pump or engine. The accuracy of the fuel calculation via this method is similar to the accuracy of the rate measurement apparatus used to calibrate the signal processing model.

A simple static nonlinear model appears adequate to achieve a sensing mechanism of ±5% average accuracy. Some further improvement is achievable by inclusion of dynamic effects in the model.

Nozzle coking and wear degrade the accuracy to a minor degree. Compensation for this degradation may be possible via adaptive control methods.

Closed-loop fuel control based upon this sensing mechanism appears to be feasible.
REFERENCES


[Burman62(2)] Burman, P. G. and DeLuca, F. ibid. p. 120.

[Burman62(3)] Burman, P. G. and DeLuca, F. ibid. p. 123.


2. Research Schedule and Task Delineations.

Development of a Sensing and Signal Processing Method to Facilitate Closed-Loop Diesel Fuel Injection Control

LAST MODIFIED:
5/1/89, C. MacCarley, (original draft)
11/1/89, C. MacCarley, (update)
11/5/89, C. MacCarley, (update)
1/21/90, C. MacCarley, (update)

Functional Tasks

1) c) Take injector data on test stand (needle lift vs actual rate and delivery).
2) c) Digitize and analyze data.
3) c) Generate inverse pump models, actual and computer simulation.
4) c) Implement inverse model on sensor processor.
5) c) Construct electronic control system for test engine.
6) c) Evaluate effects of nozzle coking on injector calibration. Study adaptive compensation algorithms for coking effects if indicated.
7) c) Evaluate method on test engine in conjunction with a main electronic control computer.

Specific Tasks ("c" denotes a completed task)

Continuous flow apparatus (JG)
6/19 c a) Locate hydraulic pump.
6/23 c b) Identify parts that need to be ordered.
6/23 c c) Purchase parts.
6/30 c d) Construct and test apparatus.
7/3 c e) Take data: flow vs pressure on Bosch and Kiki injectors.
7/10 c f) Compile and report data.

Reference Rate Measurement Apparatus (JG)
6/16 c a) Complete construction of apparatus.
6/16 c b) Locate and connect strain-gauge amplifier and test with scope.
7/5 c c) Make nozzle adaptor for the Kiki injector.
6/21 c d) Pressure calibrate strain gauge using dead-weight pressure reference. Determine linear calibration constant and range over which it is valid.
6/23 c e) Transfer to pump test stand for operational tests. Verify rate calibration by integrating flowrate curve and comparing with actual fuel delivery per injection from test stand.
6/23 c f) Set up for display simultaneously with needle lift data on oscilloscope.
6/30 c g) Make adaptor and add accumulator to apparatus to solve problem of slow slow decay of reflex pulses. (Solved by using 263 ft. long tube instead.)

Pump Test Stand (WC)
6/12 c a) Finish mounting bracket and coupler.
6/16 c b) Install injection and leak-off lines for Kiki injectors.
6/16 c c) Install injectors: Instrumented Bosch injector for now and instrumented Kiki injector upon arrival.
6/23 c d) Install Bosch rate measurement apparatus on one injector as soon as it is available.
6/16 c e) Exact speed/position indicator: install pickup and reflective strips on pump test stand.
6/16 c f) Set up power supply and adapter box for instrumented injector.
6/19 c g) Take initial data using analog oscilloscope and scope camera. Scope will display:
Rotational angle reference markers (optional)
Needle lift signal
Strain-gauge output voltage from rate measurement rig
Run full pump map, with data on total delivery, rate photos and needle lift at selected points.

Run complete map data on Kiki pump with Bosch nozzle and Kiki nozzle:
At each condition:
- Pump shaft speed: 300, 500, 1000, 1500, 2000, 2500
- Throttle settings: 30, 60, 90, 120, 150, 180 degrees

Record (display, digitize, store and off-line process):
- Needle Lift trace
- Rate trace
- Quantity per injection (from burette)

Test sensor processor/host controller on pump test stand to verify function.
Run comparative tests on coked vs clean injector to determine coking effects on perceived calibration of injector.

Instrumentation and Data Acquisition (WC, KN)

- 6/16 c a) Exact speed/position indicator: install pickup and reflective strips on pump test stand.
- 6/16 c b) Set up power supply and adapter box for instrumented injector.
- 6/30 c c) Set up digital storage scope to acquire data.
- 6/30 c d) Set up interface for IEEE-488 card on PC with the digital oscilloscope.
- 7/7 c e) Write BASIC or C code for PC to process the acquired data and control the digital scope (if possible).
- 12/1 c f) Iteratively modify the code until the results of the program processing of the needle lift and temperature data correspond closely with observed delivery and rate measurement data.
  * Task replaced by static and dynamic model development using the Sun workstation and Cray YMP. Model development still in progress.

- 11/15 c g) Coked injector analysis.
- 1/31 c h) Connect sensor processor/host as soon as it is available.
- 1/31 c i) Write assembly language algorithms for sensor processor and host processor.
- 1/31 c j) Test sensor processor/host controller on pump test stand to verify function.
- 2/28 c k) Design and test adaptive algorithms for compensation of coking and similar decalibration effects on injector.

Injector/pump Computer Simulation (WC)

- 3/15 c a) Just document this: FORTRAN simulation of model of pump.
- 1/31 c b) Refine detailed injector model based on physics of needle dynamics and hydraulic behavior.
- 6/23 c c) Acquire network compatible terminal emulator package for office PC which has Tek 4014 emulation capability.
- 1/31 c d) Add plotting and graphics capabilities via local UNIX utilities.
- 12/1 * e) Validate simulation models by comparison with pump test data on rate and needle lift.
  * f) Convert simulation model into inverse injector model for programming on PC and then sensor processor.
  * Dropped simulation effort -- 11/1/89
- 8/4 c g) Develop user-friendly interface so others can use the simulation with a minimum of in-depth knowledge of the model construction.

Sensor Processor Development (KN, DN, CM)

- 6/23 c a) Select and acquire an appropriate host controller board.
- 1/22 c a1) Info on AMI (KN)
- 7/14 c b) Modify parallel interface on controller to work with the sensor processor. Write code for both boards for communication. Use terminal interface of board for user interface: processor control and data reporting. (KN)
- 1/31 c c) Write engine control code for host control board. (KN)
- 1/22 c c1) Simple governor loop (KN)
- 1/29 c c2) Rev. 1 beta complete (KN)
1/31 c d) Construct necessary control-actuator interface hardware for host controller. (DN)
7/28 c e) Set up sensor processor for test with host controller board.
Use simulated needle lift and timing position signals. (KN)
2/1/90 c f) Debug and refine algorithms and hardware. (KN)
3/1/90 c g) Ruggedize sensor processor and host board in a cabinet for use on injection test stand, and later, a test engine. (KN)
3/1/90 c h) Set up sensor processor, host, terminal, and other instrumentation in pump test room for actual system evaluation. (KN and DN)
1/26 c i) Fix ROM capability of HP64000 (KN)

Test Engine (JG,WC)
8/14 c a) Receive test engine and accessories.
1/31 c b) Set up test engine on dyno in ME engine lab.
2/28 c c) Construct necessary equipment for particulate monitoring as well as other emission species.
1/31 c d) Use Kiki electronic pump with actuators. Build any required actuator interface hardware for controller connection.
2/5 c e) Set up sensor processor/controller unit(s) in engine test cell. Set up instrumentation for engine and terminal interface for sensor processor/controller.
1/31 c f) Design engine test schedule to exercise capabilities of sensor processor and host controller to validate or invalidate the proposed method.
3/1 c g) Emissions and efficiency tests, with and without closed-loop control.
3/1 c h) Refine engine control algorithms on actual engine.
3/15 c i) Evaluate aging and/or coking effects in actual service.
3/15 c j) Study effectiveness of adaptive algorithms for coking compensation.
2/15 c k) Demonstration for GM and Isuzu.
9/31 c l) Completion of engine validation tests.

SAE Paper and Final Report (CM)
10/1 c a) Compile all data available.
10/1 c b) Document tasks performed, results and conclusions.
 c) Answer the questions:
 c) Does the proposed method work?
 c) What restrictions on injector design are necessary?
 c) What processing requirements?
 c) Do long-term degradation effects such as nozzle coking invalidate the usefulness of this method?
 c) Can degradation effects be adaptively compensated?
 c) What potential effects on emissions, efficiency and power output?
 c) Could this method permit automotive diesels to meet California particulate standards cost effectively?
 c) Is this method practical in commercial use?
 c) What extensions of this technique are possible? i.e., use of engine acoustic data for real-time diagnosis and/or engine control; possible integration with drivetrain control for total powertrain control.
 c) Future research directions: what remains to be studied?

11/15 c d) Create graphics for data presentation.
1/31 c dl) QPLOT-based display of rate vs needle lift for each model. (wc)
10/21 c e) Text and bibliographic references.
8/1 c f) Abstract for SAE conference (due October).
10/21 c g) Reduce to final SAE publication version.
2/27 c h) Present SAE paper at conference in Feb.
12/31 c i) Final report for NSF.
3. FORTRAN Source Code for Computer Simulation of Injection System Hydraulics
program diesel
real pnor(1000), plast(1000), qnow(1000), xlast, xold
real at, pr, dpp, a1, k, rho, t, x, gold(1000), delay, c, a2max
real ksp, delt, deltal, time, rpm, qlast(1000), q0now, q0last
real pfeed, ap, vp, l, f1(11), xn, xmax, vpx, q0old
real f1now, xpart, fa, fb, pmax, f, variable, znow, zmax
integer n, tsteps, flag, xa, xb, counter, outflag
real totout, mcheck, pstart, tstart, timcon, ppmpr
integer fueltype, input
real kpsi, kf, mneedle, a2, p2, duration, dps, zlast, zold

call link ('///')
***********************************************************************

* This program is designed to simulate the dynamics of
* fuel injection in a small diesel engine through the method of finite
* differences, wherein the fuel line is divided into n nodes and
* iterations are performed both along the fuel line and
* in time to determine the pressure in, and discharge through,
* each node. The current version of this program
* assumes the use of diesel fuel #1. If diesel fuel #2 is to be used
* substitute 169666 in place of 1531e6 for bulk modulus of elasticity
* and 866.38 in place of 824.86 for density.
* NOTE: SI units are used for all values in the program.
* If units such as cubic meters turn out to be inconvenient as output
* they can be converted as desired upon printout.
* This version of the program considers several parameters
* to be constant which actually vary with temperature and/or pressure:
* bulk modulus of elasticity (k) and density (rho). Variations in the
* dimensions of the fuel line, both along the line itself and with
* respect to pressure and temperature, are also not considered in
* this model.
* The value chosen for rpm below represents the rpm of the engine
* (the fuel pump rpm will be 1/2 of engine rpm).

***************x

DEFINITION OF VARIABLES:

*a1
upper area of needle
*a2
effective nozzle flow area as a function of x
*a2max
maximum nozzle flow area
*ap
area of the fuel pump piston face
*at
cross-sectional area of the fuel line
*c
nozzle discharge coefficient
*counter
used to facilitate output of data
*delay
this is the mechanism for controlling stroke volume
* this is a delay in degrees from the point at
* which the plunger rollers contact the cam for
* max stroke volume (min delay = 0.0 degrees,
* which corresponds to max stroke volume;
* max delay is 6.0 degrees, which would correspond
* to a stroke volume of zero)
*deltal
the length of a node
*deltat
the size of a time step
*dpp
fuel line diameter
*dps
desired time between data points for output
*duration
desired time length of simulation
*f
viscous friction factor of fuel in fuel line
*f1
a mapped function of nozzle discharge rate vs. needle lift
*f1now
the interpolated value of f1 for a given needle lift
*fa
the value of f1 at the mapped value of x < actual x
*fb
the value of f1 at the mapped value of x > actual x
*flag
used to let the program know the status of the check
* flag=0 indicates no fuel pump output yet
* flag=1 indicates fuel pump output has begun
flag=2 indicates that retraction has begun
flag=3 indicates that retraction has ended

*fueltype
indicates the type of diesel fuel (#1 or #2) being used

*kf
bulk modulus of elasticity of fueltype

*kf
combined effective coefficient of friction of needle

*kpspr
spring constant of check valve spring

*kpsr
needle spring constant

*l
length of fuel line

*mccheck
mass of fuel pump outlet check valve

*mneedle
mass of the injector needle

*n
number of nodes in fuel line (integer)

*outflag
this flag is 0 prior to initial needle lift and 1 after

*p1
pressure in the last node in the fuel line

*p2
average engine cylinder pressure

*pfeed
feed pressure in fuel pump

*pmax
pressure in node i during previous time step

*pold
max expected pressure in line

*pno
injector opening pressure

*pnow(i)
presure in node i during current time step

*pmax
current fuel pump discharge pressure during retraction

*pr
residual line pressure

*pstart
fuel pump discharge pressure at start of retraction

*q0last
cumulative discharge from fuel pump as of previous time step

*q0now
cumulative discharge from fuel pump as of current time step

*q0old
cumulative discharge from fuel pump as of two time steps ago

*qmax
value of q0now at the beginning of retraction

*q1ast(i)
cumulative discharge past node i as of previous time step

*q1now(i)
cumulative discharge past node i as of current time step

*q1old(i)
cumulative discharge past node i as of two time steps ago

*rho
density of diesel fuel

*rpm
engine rpm (note: fuel pump rpm is 1/2 engine rpm)

*rvmax
max possible retraction volume based on

*check valve displacement

temperature

*timcon
time constant for fuel pump pressure decay during retraction

*time
Miller?

*totout
total volume of fuel injected into engine

*tstart
time at the start of retraction

*tsteps
number of time steps desired

*vp
fuel pump piston velocity function

*vp
fuel pump piston velocity on current time step

*x
needle lift

*xa
map value of x which is less than x

*xb
map value of x which is greater than x

*xlast
value of needle lift on previous time step

*xold
value of needle lift on second previous time step

*xmax
value of x corresponding to pmax

*xn
intermediate variable used in f1 interpolation

*xpart
fractional part of x used in f1 interpolation

*znow
displacement of check valve as of current time step

*zlast
displacement of check valve as of previous time step

*zold
displacement of check valve as of 2 time steps ago

*zmax
max displacement of check valve at fuel pump outlet

******************************************************************************

 ASSIGN VALUES TO VARIABLES

******************************************************************************

******************************************************************************

 data at,dpp,t, kspr,pno/31.67e-6,.00635,25.,1.2e5,12.01e6/
 data a1, l,time/28.20e-6, .6096,0.0/
 data k/1531.e6/
 data pfeed,ap/6.895e6,63.50e-6/
data (f1(j), j=1,11)/0.0,3.048e-6,3.387e-6,4.064e-6,6.435e-6, 
+8.805e-6,10.16e-6,12.87e-6,25.06e-6,43.35e-6,50.8e-6/ 
rpm=750.0
n=50
delta1=1/real(n)
rho=824.86
counter=0
outflag=0
pr=12.0e6
f=75.0
xn=0.0
vp=0.0
delay=2.5
deltat=1e-6
timcon=0.0025
mcheck=.010
flag=0
q0now=0.0
q0last=0.0
q0old=0.0
fueltype=1
zmax=.0005
tsteps=int((12.0/rpm/deltat)
p2=5.17e6
mneedle=.010
kpspr=50000.0
duration=9.0/rpm
rvmmax=z*at
dps=.0001
kf=0.1
xmax=.0016
xlast=0.0
xold=0.0
x=0.0

*******************************************************************************
*OPEN FILES TO STORE DATA FOR LATER GRAPHING
*******************************************************************************
open (1, file='time', status='new')
open (2, file='cumdschg', status='new')
open (3, file='needlift', status='new')
open (4, file='qdot', status='new')

*******************************************************************************
*BEGINNING OF USER-FRIENDLY INTERFACEECTED MAXIMUM PRESSURE
*******************************************************************************
write(6,'current program parameters are:','
write(6,'1. al:upper needle assy. area = ','al', 'm**2'
write(6,'2. at: fuel line flow area = ','at', 'm**2'
write(6,'3. deltat:time step size = ','deltat', 's'
write(6,'4. dps: time between data points = ','dps', 's'
write(6,'5. durat.:simulation time duration = ','duration', 's'
write(6,'6. fuel: fueltype is diesel fuel #','fueltype
write(6,'7. kpspr: needle spring constant = ','kpspr', 'N/m'
write(6,'8. kpspr: check valve spring const. = ','kpspr', 'N/m'
write(6,'9. l: fuel line length = ','l', 'm'
write(6,'10. mcheck: check valve mass = ','mcheck', 'kg'
write(6,'11. p2: avg. engine cylinder press. = ','p2', 'Pa'
write(6,'12. pno: nozzle opening pressure = ','pno', 'Pa'
write(6,'13. pr: fuel line residual pressure = ','pr', 'Pa'
write(6,'14. delay: stroke vol. factor (0=max,6=min) = ','delay'
write(6,'15. rpm: engine rpm = twice pump rpm = ','rpm', 'rpm'
write(6,'16. t: temperature = ','t', 'degrees C'
write(6,'17. timcon: fuel pump time constant = ','timcon', 's'

write(6,*),'18. zmax : max check valve displacement = ', zmax, ' m'
write(6,*),'19. mneedle: effective mass of needle = ', mneedle, ' kg'
write(6,*),'20. n : number of fuel line nodes = ', n
write(6,*),'21. f : viscous friction factor = ', f
write(6,*),'22. kf : combined needle friction factor = ', kf
write(6,*),'ENTER LINE # OF ITEM TO CHANGE (0=NO CHANGE):'
read(5,*), input
if (input.eq.1) then
write(6,*),'enter needle assy. upper area in square meters: '
write(6,*),'( .000001 <= allowable range =< .0025 )'
read(5,*), variable
if ((variable.lt.0.000001).or.(variable.gt.0.0025)) then
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 11
endif
at=variable
elseif (input.eq.2) then
write(6,*),'enter fuel line flow area in square meters: '
write(6,*),'(note: the fuel line flow area is'
write(6,*),'limited by the values chosen for fuel line'
write(6,*),'length, # of nodes, and time step size.'
write(6,*),'(current values require the fuel line'
write(6,*),'area be greater than '
write(6,*),'real(n)*deltat/100.0/1, )'
read(5,*), variable
if (variable.lt.real(n)*deltat/100.0/1) then
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 12
endif
at=variable
elseif (input.eq.3) then
write(6,*),'enter time step size in seconds: '
write(6,*),'(note: the time step size is'
write(6,*),'limited by the values chosen for fuel line'
write(6,*),'length, # of nodes, and fuel line area.'
write(6,*),'(current values require the time step'
write(6,*),'size be less than '
write(6,*),'100.0*at*1/real(n), )'
read(5,*), variable
if (variable.gt.100.0*at*1/real(n)) then
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 13
endif
deltat=variable
elseif (input.eq.4) then
write(6,*),'enter time between data points in seconds: '
write(6,*),'( ,deltat,' ' <= allowable range )'
read(5,*), variable
if (variable.lt.deltat) then
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 14
endif
dps=variable
elseif (input.eq.5) then
write(6,*),'enter time duration of simulation in seconds: '
write(6,*),'( .001 <= allowable range =< .020 )'
read(5,*), variable
if ((variable.lt.0.001).or.(variable.gt.0.020)) then
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*),'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 15
endif
duration=variable

if (input.eq.6) then
  write(6,'(A)') 'enter diesel fuel type (1 or 2):'
  write(6,'(A)') ' (note: diesel fuel #1 has a density '
  write(6,'(A)') 'of 824.86 kg/m^3 and a bulk modulus '
  write(6,'(A)') 'of elasticity of 1531e6 Pa while '
  write(6,'(A)') 'diesel fuel #2 has a density '
  write(6,'(A)') 'of 866.38 kg/m^3 and a bulk modulus '
  write(6,'(A)') 'of elasticity of 1696e6 Pa')
read(5,*) variable
  if (((int(variable).ne.1).and.(int(variable).ne.2)) then
    write(6,'(A)') 'VARIABLE IS NOT ACCEPTABLE '
    write(6,'(A)') 'VARIABLE IS NOT ACCEPTABLE '
  go to 16
endif
fueltype=variable

elseif (input.eq.7) then
  write(6,'(A)') 'enter needle spring constant in newtons per meter:'
  write(6,'(A)') '( 50000 <= allowable range <= 1000000 )'
read(5,*) variable
  if ((variable.lt.50000).or.(variable.gt.1000000)) then
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
  go to 17
endif
kspr=variable

elseif (input.eq.8) then
  write(6,'(A)') 'enter check vlv. spring const. in newtons per meter:'
  write(6,'(A)') '( 10000 <= allowable range <= 1000000 )'
read(5,*) variable
  if ((variable.lt.10000).or.(variable.gt.1000000)) then
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
  go to 18
endif
kpspr=variable

elseif (input.eq.9) then
  write(6,'(A)') 'enter fuel line length in meters:'
  write(6,'(A)') '(note: the fuel line length is '
  write(6,'(A)') 'limited by the values chosen for time step '
  write(6,'(A)') 'size, # of nodes, and fuel line area.'
  write(6,'(A)') 'current values require the fuel line '
  write(6,'(A)') 'length be greater than '
  write(6,'(A)') 'real(n)*deltat/100.0/at,')'
read(5,*) variable
  if (variable.lt.real(n)*deltat/100.0/at)) then
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
  go to 19
endif
l=variable

elseif (input.eq.10) then
  write(6,'(A)') 'enter check valve mass in kilograms:'
  write(6,'(A)') '(.001 <= allowable range <= .1 )'
read(5,*) variable
  if ((variable.lt.0.001).or.(variable gt.0.1)) then
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
    write(6,'(A)') 'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE '
  go to 20
endif
mcheck=variable

elseif (input.eq.11) then
  write(6,'(A)') 'enter avg. engine cylind. press. in pascals:'
  write(6,'(A)') '( 2.0e6 <= allowable range <= 14.0e6 )'
read(5,*) variable
  if ((variable lt.2.0e6).or.(variable gt.14.0e6)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 21
endif
p2=variable
elseif (input.eq.12) then
  write(6,*)'enter nozzle opening pressure in pascals:'
  write(6,*)'(note: nozzle opening pressure must be greater'
  write(6,*)'than fuel line residual pressure, which has a'
  write(6,*)'current value of ',pr,' )'
read(5,*) variable
if (variable.lt.pr) then
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 22
endif
pno=variable
elseif (input.eq.13) then
  write(6,*)'enter fuel line residual pressure in pascals:,'
  write(6,*)'( 3.0e6 <= allowable range )'
write(6,*)'(note: fuel line residual pressure must also'
  write(6,*)'be less than the nozzle opening pressure,'
  write(6,*)'which has a current value of ',pno,' )'
read(5,*) variable
if (((variable.lt.3.0e6).or.(variable.gt.pno)) then
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 23
endif
pr=variable
elseif (input.eq.14) then
  write(6,*)'enter stroke volume control factor (0.0 to 6.0),'
write(6,*)'where 0 gives max stroke volume and'
write(6,*)'6 gives minimum stroke volume'
read(5,*) variable
if (((variable.le.0.00000).or.(variable.gt.6.0)) then
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 24
endif
delay=variable
elseif (input.eq.15) then
  write(6,*)'enter engine rpm (=twice fuel pump rpm):'
write(6,*)'( 0 < allowable range <= 10000 )'
read(5,*) variable
if (((variable.le.0.0).or.(variable.gt.10000.0)) then
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 25
endif
rpm=variable
elseif (input.eq.16) then
  write(6,*)'enter temperature in degrees centigrade:'
write(6,*)'(-55 <= allowable range <= 200 )'
read(5,*) variable
if (((variable.lt.-55.0).or.(variable.gt.200.0)) then
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
  write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 26
endif
t=variable
elseif (input.eq.17) then
  write(6,*)'enter fuel pump time constant in seconds:'
write(6,*)'( 1e-6 <= allowable range <= 1 )'
read(5,*) variable
if (((variable.le.0.000001).or.(variable.gt.1.0)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 27
endif
timcon=variable
elseif (input.eq.18) then
28 write(6,*)'enter max check valve displacement in meters:'
write(6,*)'( 0 <= allowable range <= .010 )'
read(5,*) variable
if ((variable.lt.0.0).or.(variable.gt.0.01)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 28
endif
zmax=variable
elseif (input.eq.19) then
29 write(6,*)'enter needle mass in kilograms:'
write(6,*)'( .001 <= allowable range <= .1 )'
read(5,*) variable
if ((variable.lt.0.001).or.(variable.gt.0.1)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 29
endif
mneedle=variable
elseif (input.eq.20) then
30 write(6,*)'enter number of fuel line nodes:'
write(6,*)'(note: the number of fuel line nodes is'
write(6,*)'limited by the values chosen for time step'
write(6,*)'size, fuel line area and length, with an'
write(6,*)'absolute maximum of 1000 nodes allowed.'
write(6,*)'(current values require the number of fuel'
if (100.0*at*l/deltat.1000.0) then
write(6,*)'line nodes to be <= ',100.0*at*l/deltat,' )'
else
write(6,*)'line nodes to be <= 1000 )'
endif
if (variable.gt.100.0*at*l/deltat) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 30
endif
n=int(variable)
elseif (input.eq.21) then
31 write(6,*)'enter new viscous friction factor:'
write(6,*)'( 0 <= allowable range <= 1000 )'
read(5,*) variable
if ((variable.lt.0.0).or.(variable.gt.1000.0)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 31
endif
f=variable
elseif (input.eq.22) then
32 write(6,*)'enter new combined needle friction factor:'
write(6,*)'( 0 <= allowable range <= 1000 )'
read(5,*) variable
if ((variable.lt.0.0).or.(variable.gt.1000.0)) then
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
write(6,*)'VARIABLE IS NOT WITHIN ACCEPTABLE RANGE'
go to 32
endif
kf=variable
else
go to 48
tsteps=int(duration/deltat)
deltat=1/real(n)
rvmax=zmax*at
z=zmax
a2max=3.85e-7
pmax=ksp*rmax/2.0/a1+pno
c=50.8e-6/a2max/(2.0*(pmax-p2))*0.5*rho**2.0
zlast=zmax
zold=zmax
if (fueltype.eq.1) then
  k=1531.0e6
  rho=824.86
endif
if (fueltype.eq.2) then
  k=1696.0e6
  rho=866.38
endif

**INITIALIZE ARRAYS**

**MAIN LOOP: each iteration is a time step**

**find fuel pump piston velocity**

if (((3.0*rpm*time).ge.delay).and.(flag.le.1)) then
  vpx=vp(time,rpm)
endif

**if retraction has ended, then q0now will not change any more**

if (flag.ge.3) then
  go to 60
endif

**if retraction has begun, q0now must be calculated differently**

if (flag.ge.2) then
  go to 160
endif

**find fuel pump discharge pressure**

if (flag.le.1) then
  go to 50
endif
if (vpx.eq.0.0) then
go to 120
endif
flag=1
*
********** if plunger rollers have passed the cam peak then it is time to
********** begin retraction--tstart and pstart must be initialized **
*
50         if (time.ge.(2.0/rpm)) then
            tstart=time
            pstart=plast(1)
            vpx=0.0
            qmax=q0now
            go to 160
endif
*
********** q0now will be calculated here prior to retraction **********
*
            q0now=ap*vpx*deltat+q0last
*
********** if retraction has ended, q0now will no longer change *****
*
60         if (time.ge.3.6667/rpm) then
            flag=3
            endif
**********************************************************************
*NESTED LOOP: here the iterations are in length along fuel line
******************************************t()k**************************
do 70 i=1,n
*
********** special routine for first node of fuel line ***************
*
if (i.eq.1) then
            qnow(i)=(plast(i)-plast(i+1))*at*deltat**2/
            (rho*deltal)+2.0*qlast(i)-gold(i)-f/(2.0*at*
            + dpp)*(qlast(i)-gold(i))*abs(qlast(i)-gold(i))
            pnow(i)=pr+(q0now-qnow(i))*k/(deltal*at)
            if (pnow(i).le.0.0) then
                pnow(i)=0.0
            endif
            go to 70
endif
*
********** special routine for final node of fuel line ***************
*
if (i.eq.n) then
            pnow(i)=pr+(qnow(i-1)-qlast(i))*k/deltal/at
            if (pnow(i).le.0.0) then
                pnow(i)=0.0
            endif
if (xlast.1e.xmax/2.0) then
            a2=a2max/2.0*(4.0*xlast/xmax-4.0*(xlast/xmax)**2)
elseif ((xlast.gt.xmax/2.0).and.(xlast.1e.xmax)) then
            a2=a2max/2.0*(8.0*xlast/xmax-4.0*(xlast/xmax)**2-2.0)
else
            a2=a2max
endif
x=deltat**2/(mneedle+kf*deltat+ksp*deltat**2)*
+(a1*pnow(n)-(pnow(n)-p2)*a2+xlast*(2.0*mneedle
+kf*deltat)/deltat**2-xold*mneedle/deltat**2-
                    pno*al)
            if (x.1e.0.0) then
                x=0.0
            endif
xold=xlast
xlast=x
xn=x/xmax*20.0
if (xn.ge.10.0) then
  qnow(n)=c*a2/rho**0.5*(pnow(n)-p2)**0.5*deltat+
    qlast(n)
go to 70
endif
xa=int(xn+1)
xb=int(xn+2)
fa=f1(xa)
fb=f1(xb)
xpart=(xn+1.0)-xa
flnow=fa+xpart*(fb-fa)
qnow(i)=flnow*deltat+qlast(i)
go to 70
endif

********** routine for all fuel line nodes except first and last *****

  qnow(i)=at*deltat**2*(plast(i)-plast(i+1))/
    (rho*deltal)+2.0*qlast(i)-qold(i)-f/(2.0*at*dpp)*
    (qlast(i)-qold(i))*abs(qlast(i)-qold(i))
pnow(i)=pr+(qnow(i-1)-qnow(i))*k/(deltal*at)
  if (pnow(i).le.0.0) then
    pnow(i)=0.0
  endif

70 continue
totout=qnow(n)

********** set outflag if there has been some nozzle discharge *******

  if ((outflag.eq.0).and.(pnow(n).gt.pno)) then
    outflag=1
  endif
  if ((outflag.eq.l).and.(pnow(n).le.pno)) then
    outflag=0
  endif

********** routine to write to files and give a printout every dps seconds

120 if ((counter.eq.(nint(dps/deltat))).or.(time.eq.0.0)) then
  write (1,200) time
  write (2,200) qnow(n)
  write (3,200) x
  write (6,210) time, q0now,  
    pnow(n), x, (qnow(n)-qlast(n))/deltat, totout
  write (4,200) (qnow(n)-qlast(n))/deltat
  counter=0
endif

********** update the counter and the time ***************************

counter= counter+l
  time=time+deltat

********** update all node variables prior to next time step***********

  do 150 i=1,n
    q0old=q0last
    q0last=q0now
    qold(i)=qlast(i)
    qlast(i)=qnow(i)
    plast(i)=pnow(i)
 150 continue

go to 180
********** routine for calculating q0 now after retraction has begun
********** this routine ends just prior to line 180
*
160    if (flag.ge.2) then
    go to 170
endif
flag=2
170    ppmprt=pfeed+(pstart-pfeed)*exp((-1.0)*(time-tstart)/timcon)
znow=deltat**2/mcheck*((ppmprt-plast(1))*at-kpspr*zlast)
        +2.0*zlast-zold
q0now=q0last+(znow-zlast)*at
zold=zlast
zlast=znow
if (znow.le.0.0) then
    flag=3
endif
    go to 60
180    continue
**********************************************************************
**********************************************************************
*THE MAIN LOOP HAS NOW ENDED*
**********************************************************************
**********************************************************************
*
200    format (e12.6)
210    format (’time=’,f5.4,’ q(0)=’,e7.2,
        +’ p(n)=’,e8.3,’ lift=’,e8.3,
        +’ qdot=’,e8.3,’ fout=’,e8.3)
close(1)
close(2)
close(3)
close(4)
end
**********************************************************************
**********************************************************************
*VELOCITY FUNCTION FOR FUEL PUMP PISTON
*
real function vp(time,rpm)
    real time,rpm
    vp=rpm*(.0003+.0000625*time*rpm*3.0-.0000125*
        +(time*rpm*3.0)**2+.000003125*(time*rpm*3.0)**3)
    return
end

DIESEL INJECTOR MODELING SOFTWARE

Walt Clark

SENIOR PROJECT
ELECTRONIC AND ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo
1990
TITLE: DIESEL INJECTOR MODELING SOFTWARE

AUTHOR: Walt Clark

DATE SUBMITTED: March 9, 1990

[Signatures]
ABSTRACT

This Senior Project was undertaken to support a larger research project directed by Dr. Carl MacCarley. The purpose of the Senior Project was to develop software and associated diesel injector models which could be used to facilitate data collection, reduction, and subsequent model development for diesel injectors. Toward this purpose, six programs were developed, listings of which can be found in appendices A through F. The data collection program obtains digitized data from a HP 54501 100MHz digital oscilloscope and stores the data on a 5 1/4" floppy disk. The static map generation program uses these floppy disk files to create a map which relates needle lift to flow rate. Next, a suitable dynamic model, which takes into account needle velocity and acceleration to improve upon the accuracy of the static map, must be developed. Graphical representation of model-generated flow rate was essential to the model development process, which necessitated the development of a data conversion program (to format the data for graphical display) and a graphical display shell script. Once a suitable model was found, two programs were needed, one to solve for optimal model parameters and the other to quantify the absolute average error of the model in terms of model-generated fuel quantity vs. actual fuel quantity.

The dynamic injector model developed through this process has proved to be highly successful in generating accurate fuel flow rates from needle lift. Absolute average errors in model-generated fuel quantities as compared to measured fuel quantities have been roughly 3%. With this kind of accuracy it becomes feasible to "close the loop" on the diesel fuel injection process with a microprocessor-based control system, using sampled needle lift data from a Hall effect sensor mounted on one injector (of a matched set) of an engine to generate instantaneous fuel flow rate, which can then be integrated to yield the fuel quantity per injection. This accurate measurement of fuel quantity can then be compared to the desired fuel quantity to generate a feedback signal to an electronically controllable fuel pump to correct any overfueling or underfueling conditions. The future success of the implementation of this type of system is critical in reducing the familiar, annoying smoke and noise associated with diesel engines.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>v</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. BACKGROUND</td>
<td>3</td>
</tr>
<tr>
<td>III. REQUIREMENTS</td>
<td>11</td>
</tr>
<tr>
<td>IV. SOFTWARE DEVELOPMENT</td>
<td>13</td>
</tr>
<tr>
<td>V. RESULTS</td>
<td>21</td>
</tr>
<tr>
<td>VI. CONCLUSIONS AND RECOMMENDATIONS</td>
<td>23</td>
</tr>
<tr>
<td>VII. BIBLIOGRAPHY</td>
<td>24</td>
</tr>
</tbody>
</table>

## APPENDICES

A. DATA COLLECTION PROGRAM.................. 25  
B. STATIC MAP GENERATION PROGRAM............ 31  
C. GRAPHICAL DATA CONVERSION PROGRAM......... 36  
D. GRAPHICAL DISPLAY SHELL SCRIPT............ 41  
E. MODEL PARAMETER DETERMINATION PROGRAM..... 43  
F. MODEL ACCURACY DETERMINATION PROGRAM...... 47
# LIST OF TABLES AND FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CONTROL SYSTEM FUNCTIONAL BLOCK DIAGRAM</td>
<td>4</td>
</tr>
<tr>
<td>2. CONTROL SYSTEM BLOCK DIAGRAM</td>
<td>5</td>
</tr>
<tr>
<td>3. SIMPLIFIED INJECTOR FLOW MODEL</td>
<td>7</td>
</tr>
<tr>
<td>4. INSTRUMENTED INJECTOR CROSS-SECTION</td>
<td>8</td>
</tr>
<tr>
<td>5. DATA COLLECTION APPARATUS BLOCK DIAGRAM</td>
<td>9</td>
</tr>
<tr>
<td>6. REFERENCE RATE NONLINEAR CORRECTIVE FUNCTION</td>
<td>15</td>
</tr>
<tr>
<td>7. TYPICAL STATIC MAP</td>
<td>16</td>
</tr>
<tr>
<td>8. TYPICAL GRAPHICAL DISPLAY</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. DYNAMIC MODEL AND CORRECTED RATE ERROR LISTING</td>
<td>20</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

This Senior Project owes its very existence to Dr. Carl MacCarley, who conceived of the possibility of closing the loop on the diesel fuel injection process. His research project, of which this Senior Project is but a small part, has significantly advanced the state-of-the-art in the field of diesel fuel injection, enabling a true control strategy to be applied to diesel fuel injection systems for the first time ever. It has been a true honor and an incredible learning experience to have been able to work on this project. Dr. MacCarley has been intimately involved in the model development, and he has provided key insights and guidance in the software development. Rather than scattering scores of footnotes throughout this paper, suffice it to say that Dr. MacCarley contributed in one way or another to virtually every area that will be touched upon by this report. It should be mentioned, however, that he is the sole author of the graphical display shell script, which is included here for completeness.

The National Science Foundation is to be applauded for their financial support of the overall research project and their recognition of its vital importance. They also provided invaluable CPU time on the Cray X-MP (located near the University of California, San Diego campus).

The Electronic and Electrical Engineering Department deserves great thanks for providing the overall research
project with critically needed laboratory space and for allowing me the use of their Sun 386i workstation.

My appreciation also goes to Keay Nakae, fellow research assistant, who performed at least 80% of the work on the data collection program and who generated Figure 2, the Data Collection Apparatus Block Diagram.
INTRODUCTION

The software and diesel injector model development that comprise this Senior Project are intended for use on a larger research project involving diesel engine fuel injection feedback control systems. Six computer programs are the "product" that was developed to support the larger research project. The six programs, a data collection program, a static map generation program, a graphical data conversion program, a graphical display shell script, a model parameter determination program, and a model accuracy determination program, are included in appendices A through F and discussed in detail in section V, Software Development.

The data collection program is written in BASIC, compiled, and then run on an IBM PC. The graphical display shell script is a UNIX shell script written by Dr. Carl MacCarley and run on a Sun 386i workstation. The remaining programs were all written in Fortan 77 and run on the Sun 386i workstation, with the exception of the model parameter determination program which was run on a Cray X-MP.

It is important to understand the context in which this software is used. To this end, it is necessary to give a description of the goals of the parent research project. If the overall research project is successful, diesel fuel injection will be a closed-loop controllable process for the first time in history. This is critical for several reasons. First, it
is common knowledge that diesel engines tend to produce unwanted smoke and noise. This is due to the fact that only a certain quantity of fuel can be burned in an engine cylinder for a given operating condition. Put in less than this quantity of fuel and engine power is reduced; put in more than this quantity of fuel and the excess is emitted as smoke. The mechanically controlled fuel injection systems of the past, and even modern map-lookup systems, suffer from a fixed, preset fuel delivery schedules which command certain fuel quantities to be injected but have no way to check how much fuel was actually delivered. Thus, even new diesel engines exhibit smoking, and the problem can become progressively worse. By closing the loop on diesel fuel injection, overfueling and underfueling can be essentially eliminated, resulting in smoke free and efficient diesel operation. The software described in this paper performs an essential support task within the framework of this significant attempt to clean up diesel engine performance.
To elaborate on the context of the larger research project, it should be mentioned that the control strategy is to be implemented with a dual microprocessor system [4]. Figure 1, the functional block diagram of this system, shows that the main processor supplies all of the control signals to the engine and the fuel pump. The fuel pump must be electronically controllable, and such a pump was donated to the research project by Isuzu. The second microprocessor is a sensor processor and is tasked by the controller processor with collecting various elements of data as shown on the right hand side of the diagram. The main emphasis of the software discussed in this paper is produce a model of injection rate vs. needle lift so that the sensor processor can then use this model to derive instantaneous injection rate, which can then be integrated to yield fuel volume per injection. This fuel volume can then be compared by the main control processor to the desired fuel volume (i.e. that which provides optimum power but no smoke) to generate the fuel rack position and timing angle control outputs necessary to correct any errors in the actual fuel volume being delivered per injection.

Figure 2 shows a more detailed view of the closed loop fuel delivery system, including injection dynamics, which will be a major factor in the development of the appropriate injector model to be used in the model parameter determination
CONTROL INPUTS

- Throttle Position Sensor
- Keyswitch
- Instrumentation
- Cruise Control
- Crankshaft Position Sensor
- Cyl. #1 TDC Pickup
- Parallel Communication Link

MAIN CONTROL PROCESSOR

- Fuel Rack Position
- Timing Angle
- Rate Angle
- Glowplug On/Off
- EGR Valve Position
- Fuel Shutoff Solenoid
- Injector Needle Lift
- Fuel Temperature
- Injection Line Pressure
- Combustion Luminescence
- Photodetector
- Cylinder Pressure
- Air Pressure
- Engine Coolant Temperature
- Air Temperature
- Knock Detector
- Acoustic Diagnostic Sensor
- Exhaust Smoke Sensor
- Exhaust O₂ Sensor
- Torque Sensor

SENSOR PROCESSOR

CONTROL OUTPUTS

Figure 1. Functional Block Diagram - Dual Processor Diesel Engine Control System
Figure 2. Closed Loop Adaptive Fuel Control Using Sensor Processor Incorporated in Speed/Load Governor Strategy.
Figure 3 illustrates a simplified model of an injector and introduces some of the variables which will be of concern to the software development process. The thrust of the software will be to find the static and dynamic relation between fuel flow rate (m dot) and needle lift (x) such that m dot can be accurately and reliably be predicted based on a knowledge of x alone.

Figure 4 shows a detailed cross-sectional view of a diesel fuel injector. The part that is labeled "valve" in the diagram is most often referred to in diesel parlance as the needle, and it will be referred to as such throughout this paper. The diagram illustrates how the needle lift sensor is implemented. A magnet is attached to the end of the spindle, which travels with the needle. This magnet induces a voltage in the Hall Effect sensor which is proportional to needle lift (x). The Hall Effect voltage is essentially linear with respect to needle lift over the limited range of travel of the needle (typically less than one millimeter).

A block diagram of the data collection apparatus is shown in Figure 5. The pump test stand allowed the testing of injectors under conditions from 0 to 2500 pump rpm (engine rpm is twice pump rpm) and from 0 to 180 degrees throttle (180 degrees throttle is full throttle). The strain gauge and measurement tube comprise the "Bosch Apparatus". This device was proposed by Wilhelm Bosch in 1966 [1] and allows instantaneous fuel injection rate to be determined from instantaneous pressure as measured by the strain gauge. This
Figure 3. Simplified Flow Model of Injector Nozzle
Figure 4. Cross-sectional View, Optimized Injector/Flow Transducer
Figure 5: Data Collection Apparatus Block Diagram
injection rate signal is applied to channel 4 of the digital oscilloscope, and the needle lift signal from the Hall Effect sensor is applied to channel 1. These signals are digitized and downloaded to floppy disk under the control of the IBM PC via the IEEE 488 bus (also referred to as the HPIB bus elsewhere in this paper).
REQUIREMENTS

The software product presented in this report must support the needs of the dual microprocessor system as described in the background section. That is, the programs that will be developed must be able to control the data collection process depicted in Figure 5, must enable the generation of reliable injector model parameters, and must provide an error checking mechanism for model parameters.

The data collection process control and the model parameter error checking can be accomplished with a single program each. The code necessary for the generation of model parameters depends upon the nature of the model chosen. For example, if it were possible to characterize the fuel injection rate of an injector as a function of needle lift by a single equation, then only a single piece of code would be necessary to accomplish this. However, if (as turns out to be the case) the characterization of the injector by a single equation turns out to be unfeasable, then the model may have to include one or more static (i.e. non-dynamic) maps which take the place of non-modelable characteristics of the injector.

The final requirement of the software is that it be able to generate models which can be proven to yield absolute average errors of less than 5% for model-generated (from needle lift data only) injection volumes as compared to the true injection volumes as measured on the pump test stand.
Absolute average errors are calculated by finding the absolute value of the percent volume error at each of approximately 100 different operating conditions and then finding the average of those absolute errors.
SOFTWARE DEVELOPMENT

The first program to be developed was the data collection program, which is included as Appendix A. This program was tasked with controlling the HP 54501 digital oscilloscope during the data collection process to cause the needle lift and injection rate signals to be digitized and then downloaded to floppy disk files for later use. The program was written in BASIC and compiled and run on an IBM PC. The code is fairly straightforward but several points are worth mentioning.

Control of the digital oscilloscope was accomplished via an IEEE 488 bus. Command words were sent to set up the oscilloscope in the proper data collection mode, specifically to digitize 256 points for each signal and to average each of those points 256 times. In other words, the needle lift and rate signals were repeated 256 times (256 injections were observed) and averaged prior to digitization. This provided a measure of noise immunity in the resulting data. Some data processing was also necessary once the data was received at the IBM PC since the digitized data was sent as one long ASCII string, necessitating parsing of the string and conversion to number format. These numbers had to be further processed to return them to floating point voltage values since the returned data was not in terms of voltages but in terms of position on the screen.

Next, a static map generation program was developed. This
Program was tasked with reducing the collected data, which typically amounted to more than 100 data files for a given injector, to a simple look-up table for injection rate in terms of needle lift. This task was complicated by several factors. First, since the needle lift and injection rate sensors were at slightly different physical locations, a time delay was present in the raw data which had to be compensated for. Specifically, the rate signal data point that corresponded to a given needle lift point was actually to be found by looking 60 microseconds later in time (i.e. rate lagged needle lift). The second complicating factor was that the Bosch Apparatus (the injection rate sensor) was not precisely linear. This caused the integration of the injection rate signal itself to have greater than a 6% absolute average error in volume! Obviously, creating a static map that related needle lift to erroneous injection rate data was not very desirable. After painstaking investigation of this problem it was discovered that a corrective function to undo the nonlinearities of the Bosch Apparatus resulted in a reduction of the 6% error cited above to less than 3%. The corrective function is graphically displayed in Figure 6. A typical static map as produced by this program is shown in Figure 7. A program listing is included as Appendix B.

The next two programs, which are included as Appendices C and D, are the graphical data conversion program and the graphical display shell script. These programs work together to allow a graphical display of the raw needle lift and rate data in addition to static map and dynamic model generated
Figure 6: Reference Rate Nonlinear Corrective Function
Figure 7: Typical Static Map
injection rate curves. A typical example of this graphical output is shown in Figure 8. Having the capability of graphing data in this fashion was a critical aid in the model development process.

Appendix E comprises a listing of the model parameter determination program. This program was written on a Sun 386i workstation but had to be run on a CRAY X-MP. The code was sent by ftp to the CRAY down in San Diego and then run via telnet. The reason that the CRAY had to be used was that large arrays of data needed to be processed by an International Math and Statistics Library (IMSL) nonlinear optimization program. The enormous amount of data to be processed would have taken a prohibitively long time on any ordinary computer. In addition, local availability of IMSL routines was virtually nonexistent until just recently.

The model parameter determination program uses the IMSL nonlinear optimization routine BCLS [3]. This routine solves a nonlinear least squares problem using a Levenberg-Marquardt algorithm [5]. The user can define bounds on the variables or leave them unbounded. When reasonable bounds could be estimated, convergence to a solution occurred orders of magnitude faster than with unbounded variables. One nice feature of the BCLS routine was that it computed a finite-difference Jacobian, which turned out to be not only convenient but critical in this case since it was virtually impossible to calculate a closed-form Jacobian. One important item that was critical to obtaining reasonable results (or results at all, for that matter!) was that the parameters
DK injector: 1500 rpm, 180 degrees throttle

needle lift
--- --- --- static map rate
--- --- --- dynamic model rate

Figure 7: Typical Graphical Display
passed to the BCLSF routine had to be of the same general order of magnitude [2]. Since the parameters were actually of quite different orders of magnitude, scaling of the parameters was required prior to calling the BCLSF routine. A typical graphical display showing the model performance is shown in Figure 8.

The final piece of software is the model accuracy determination program, included as Appendix F. This program accepts a list containing the names of all data files associated with a given injector. The needle lift data for each file is then processed through the model and integrated. Then the resultant volume is compared to what was actually measured when the data was originally taken to yield the absolute volume error. Averaging these errors for all of the data files produces the absolute average error. The individual errors as well as the overall errors are then output to a file in tabular form for use in model development and evaluation. A typical error table (actually, a reduced list to enable a one page table) as produced by this program is included as Table 1.
Model: \[ \text{rate} = \sqrt{\text{(static map vol.)}^2 + \text{(needle lift)}^{1.5} \times \left(0.05 \times \text{(needle velocity)} + 5.5 \times 10^{-5} \times \text{(needle acceleration)}\right)} \]

Nonlinear rate correction: \[ \text{new rate} = \text{old rate} + 5.25 \times 10^{-3} \times (\text{old rate})^2 \]

<table>
<thead>
<tr>
<th>filename</th>
<th>actual measured vol. (mm^3)</th>
<th>corrected raw rate vol. (mm^3)</th>
<th>model generated vol. (mm^3)</th>
<th>%error</th>
<th>%error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0375090k.dat</td>
<td>29.50</td>
<td>28.87</td>
<td>28.93</td>
<td>-1.94</td>
<td></td>
</tr>
<tr>
<td>0375135k.dat</td>
<td>43.70</td>
<td>42.48</td>
<td>41.47</td>
<td>-5.10</td>
<td></td>
</tr>
<tr>
<td>0375180k.dat</td>
<td>45.10</td>
<td>44.31</td>
<td>42.42</td>
<td>-5.94</td>
<td></td>
</tr>
<tr>
<td>0500090k.dat</td>
<td>9.60</td>
<td>9.91</td>
<td>9.87</td>
<td>2.78</td>
<td></td>
</tr>
<tr>
<td>0500135k.dat</td>
<td>25.50</td>
<td>25.04</td>
<td>24.10</td>
<td>-5.49</td>
<td></td>
</tr>
<tr>
<td>0500180k.dat</td>
<td>25.60</td>
<td>25.67</td>
<td>24.49</td>
<td>-4.35</td>
<td></td>
</tr>
<tr>
<td>0625090k.dat</td>
<td>8.90</td>
<td>9.24</td>
<td>9.21</td>
<td>3.44</td>
<td></td>
</tr>
<tr>
<td>0625135k.dat</td>
<td>25.20</td>
<td>25.77</td>
<td>23.89</td>
<td>-5.20</td>
<td></td>
</tr>
<tr>
<td>0625180k.dat</td>
<td>25.50</td>
<td>25.30</td>
<td>24.73</td>
<td>-3.01</td>
<td></td>
</tr>
<tr>
<td>0750090k.dat</td>
<td>7.20</td>
<td>7.49</td>
<td>7.53</td>
<td>4.62</td>
<td></td>
</tr>
<tr>
<td>0750135k.dat</td>
<td>23.80</td>
<td>23.30</td>
<td>23.64</td>
<td>-0.66</td>
<td></td>
</tr>
<tr>
<td>0750180k.dat</td>
<td>25.50</td>
<td>24.94</td>
<td>25.20</td>
<td>-1.17</td>
<td></td>
</tr>
<tr>
<td>0875090k.dat</td>
<td>7.20</td>
<td>7.03</td>
<td>7.51</td>
<td>4.30</td>
<td></td>
</tr>
<tr>
<td>0875135k.dat</td>
<td>21.80</td>
<td>21.31</td>
<td>21.60</td>
<td>-0.92</td>
<td></td>
</tr>
<tr>
<td>0875180k.dat</td>
<td>25.90</td>
<td>25.00</td>
<td>25.92</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>1000090k.dat</td>
<td>8.30</td>
<td>8.57</td>
<td>8.46</td>
<td>1.89</td>
<td></td>
</tr>
<tr>
<td>1000135k.dat</td>
<td>17.70</td>
<td>17.50</td>
<td>17.60</td>
<td>-0.59</td>
<td></td>
</tr>
<tr>
<td>1000180k.dat</td>
<td>25.20</td>
<td>24.63</td>
<td>26.01</td>
<td>3.23</td>
<td></td>
</tr>
<tr>
<td>1125090k.dat</td>
<td>8.50</td>
<td>8.63</td>
<td>8.74</td>
<td>2.77</td>
<td></td>
</tr>
<tr>
<td>1125135k.dat</td>
<td>18.50</td>
<td>17.97</td>
<td>18.52</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>1125180k.dat</td>
<td>25.50</td>
<td>25.48</td>
<td>26.31</td>
<td>3.16</td>
<td></td>
</tr>
<tr>
<td>1250090k.dat</td>
<td>8.50</td>
<td>8.82</td>
<td>8.63</td>
<td>1.52</td>
<td></td>
</tr>
<tr>
<td>1250135k.dat</td>
<td>18.60</td>
<td>17.86</td>
<td>18.42</td>
<td>-0.96</td>
<td></td>
</tr>
<tr>
<td>1250180k.dat</td>
<td>25.60</td>
<td>26.46</td>
<td>25.54</td>
<td>-0.24</td>
<td></td>
</tr>
<tr>
<td>1375090k.dat</td>
<td>9.20</td>
<td>9.46</td>
<td>9.10</td>
<td>-1.13</td>
<td></td>
</tr>
<tr>
<td>1375135k.dat</td>
<td>19.70</td>
<td>20.27</td>
<td>20.75</td>
<td>5.35</td>
<td></td>
</tr>
<tr>
<td>1375180k.dat</td>
<td>26.50</td>
<td>26.24</td>
<td>27.97</td>
<td>5.54</td>
<td></td>
</tr>
<tr>
<td>1500090k.dat</td>
<td>8.90</td>
<td>9.29</td>
<td>8.73</td>
<td>-1.95</td>
<td></td>
</tr>
<tr>
<td>1500180k.dat</td>
<td>29.80</td>
<td>31.15</td>
<td>31.10</td>
<td>4.35</td>
<td></td>
</tr>
<tr>
<td>1625090k.dat</td>
<td>8.30</td>
<td>8.55</td>
<td>8.02</td>
<td>-3.38</td>
<td></td>
</tr>
<tr>
<td>1625180k.dat</td>
<td>31.50</td>
<td>32.01</td>
<td>33.24</td>
<td>5.53</td>
<td></td>
</tr>
<tr>
<td>1750090k.dat</td>
<td>7.70</td>
<td>8.05</td>
<td>7.69</td>
<td>-0.16</td>
<td></td>
</tr>
<tr>
<td>1750135k.dat</td>
<td>20.40</td>
<td>20.62</td>
<td>22.24</td>
<td>9.02</td>
<td></td>
</tr>
<tr>
<td>1750180k.dat</td>
<td>32.20</td>
<td>31.32</td>
<td>32.28</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>1875090k.dat</td>
<td>7.70</td>
<td>7.98</td>
<td>7.48</td>
<td>-2.89</td>
<td></td>
</tr>
<tr>
<td>1875135k.dat</td>
<td>20.30</td>
<td>20.60</td>
<td>22.12</td>
<td>8.99</td>
<td></td>
</tr>
<tr>
<td>2000090k.dat</td>
<td>7.30</td>
<td>7.41</td>
<td>7.09</td>
<td>-2.85</td>
<td></td>
</tr>
<tr>
<td>2000180k.dat</td>
<td>32.20</td>
<td>31.96</td>
<td>31.91</td>
<td>-0.91</td>
<td></td>
</tr>
</tbody>
</table>

Nonlinearly corrected raw injection rate average absolute % volume error = 2.72186 %

Dynamic model generated injection rate average absolute % volume error = 3.74261 %

Table 1: Dynamic Model and Corrected Rate Error Listing (partial)
RESULTS

The results obtained by this project not only met but substantially exceeded the required performance as had been specified in the requirements section. It had been required that the final model produced by the software yield absolute average errors of less that 5%. As can be seen at the bottom of Table 1, the final model (shown at the top of Table 1) yielded an absolute average percent volume error of 3.74261%. The units that are used in the model are as follows. Static map volume was in mm^3 (which is equivalent to microliters), needle lift in mm, needle velocity in mm/s, and needle acceleration in mm/s^2. In fact, the static map alone yielded an absolute average error of 4.73084%, which met the original criterion. Several different injectors have been used to test the validity of the model generation process and the results have substantially exceeded the 5% error limit in every case. The typical range of absolute average percent errors for the models for the various injectors was 3.25% to 4.25%.

The nonlinear corrective function for the Bosch Apparatus that proved to yield the best results was new rate = old rate + 5.25e-3 * (old rate)**2, with all rates in units of mm^3. Prior to using the nonlinear corrective function for raw injection rate, integration of raw injection rate data had yielded an absolute average error of 6.25351%. As can be seen in Table 1, use of the nonlinear corrective function improved
the absolute average error for integrated injection corrected
raw injection rate data to 2.72186 %. This type of improvement
attests not only to the validity of the corrective function but
to its critical impact on the entire modeling process.

A final comment should be made regarding the streamlined
nature of the modeling process. Once data has been collected,
a highly accurate model can be generated in less than an hour.
If this process were to be applied to an assembly line
situation in which the injectors were of similar types to begin
with and a reduced set of data was used in the modeling
process, it is reasonable to expect that injectors could be
individually and accurately modeled in less than 30 minutes
apiece. The injectors actually need to be produced in matched
sets, which means that only one in four injectors needs to be
instrumented and modeled.
CONCLUSIONS AND RECOMMENDATIONS

The concept of modeling a diesel injector's injection rate in terms of needle lift alone has proven to be a valid one, and the software product of this project stands as a tool that can be used successfully for this purpose. While the modeling process has proved to exceed original accuracy expectations, there are still several things that can be done to produce even more accurate models in the future. First, a digital oscilloscope with a continuously variable time base would aid in producing data with a better resolution. Second, a strain gauge amplifier could be miniaturized and mounted adjacent to the strain gauge to reduce the noise level on the signal (which currently requires applying a 10 KHz low pass filter to the raw rate data before it is used). Finally, a separate research project could be undertaken to attempt to improve the nonlinear corrective function for raw injection rate. Although this is pure conjecture at this point, it seems as though reducing absolute average errors to below the 3% level is probable if the above recommendations are implemented.
BIBLIOGRAPHY


APPENDIX A: DATA ACQUISITION PROGRAM

************************************************************* DATA ACQUISITION PROGRAM *************************************************************

*** This program can be found under titles datacollection.bas, wethere2.bas, or w2.bas. It is written in BASIC and then compiled and run on an IBM PC.
*** This program accepts test condition information from the user and then sets up the HP 54501 digital oscilloscope (via HPIB interface) appropriately to digitize the needle lift and rate waveforms obtained from the instrumented injector and the rate reference apparatus. This digitized data is then downloaded over the HPIB interface and stored in an appropriately named floppy disk file. Some initial minor signal processing is also performed in this program. Each data file may be given a one-letter identifier. For example, a Bosch injector might be given the letter 'b'. A Bosch data file for 2000 rpm and 180 degrees throttle would then be named '2000180b.dat'.

*** The signal processing accuracy depends on the first ten digitized voltages being essentially flat. The user should ensure this, but if it is overlooked the data will be invalid. The program detects this type of error and does not store the data but instead loops back to the top of the program.

*** Needle lift and flow rate are stored in units of mm and ul/s (not ul/ms!). Flow rate is stored in prescaled form (a holdover from early data collection programs) and must be unscaled prior to use in all data processing programs.

*** VARIABLES:

- CMDS$: command string for HPIB control
- CONST: voltage-to-rate conversion constant in ul/s/volt
- FLOWRATE: flow rate in ul/s (not ul/ms!)
- INJS$: injector-type identification letter
- LIMIT$: maximum allowable averaging
- NAME$: floppy data file name string
- NEDLVOLT: needle lift in mm
- PUMPRPM: nominal pump rpm
- REALRPM: actual pump rpm
- SCALER: rate scale factor (unitless)
- SIZE: number of points to be digitized
- THROTTLE: throttle setting in degrees
- XORG: time origin in seconds
- XINC: time between data points in seconds
- YORG: vertical origin in volts
- YINC: vertical resolution in volts

*************************************************************
*** Interactive interface with the user to input pertinent
*** test condition information, including pump rpm and
*** throttle setting, and to display connection instructions
*** for oscilloscope/sensor interface.
10 PRINT "ENTER NOMINAL PUMP RPM: ";
20 INPUT PUMPRPM
30 PRINT "ENTER THROTTLE POSITION (0-180): ";
40 INPUT THROTTLE
50 PRINT "PUMP SPEED = ";PUMPRPM:"RPM."
60 PRINT "TROTTLE POSITION = ";THROTTLE:"DEGREES."
70 PRINT "ARE THESE CORRECT? (Y/N): ";
80 ANSWER$=INPUT$(1) : PRINT
90 IF ANSWER$ = "N" THEN GOTO 10
100 PRINT "VERIFY THAT CHAN1 IS NEEDLE LIFT & CHAN4 IS PRESSURE (Y/N?)"
110 VERIFY$ = INPUT$(1) : PRINT
120 IF VERIFY$ = "N" OR VERIFY$="n" THEN GOTO 130
125 GOTO 150
130 PRINT "CONNECT NEEDLE LIFT TO CHANNEL 1 AND PRESSURE TO CHANNEL 4"
140 GOTO 100
150 SIZE=256
180 FLAG = 0
190 DIM NUM$(5)

*** Variables to be used in the HPIB interface are initialized
*** here.
200 DEF SEG =&HE000
210 IOOUTPUT%=3 : ENTER%=6 : STATUS%=42
220 ENTERA%=51
240 ADDR%=7 : DATASEG%=&H3000 : LENGTH%=&HFFFF
250 INIT%=0 : MYADDR%=21 : IOPORT%=&H288 :SETTING%=&H0100

*** The HPIB bus is initialized here.
260 CALL ABSOLUTE(IOPORT%,MYADDR%,SETTING%,INIT%)
270 CMD$= ":SYSTEM:HEADER OFF"
280 CALL ABSOLUTE(ADDR%,CMD$,IOOUTPUT%)

*** There is a maximum allowable amount of averaging
*** that can be allowed without overflowing the
*** available oscilloscope memory. This section
*** of the code performs appropriate limiting on the
*** averaging performed to get each digitized data
*** point.
351 PRINT "ENTER ACTUAL PUMP RPM:"
352 INPUT REALRPM
361 IF REALRPM<249 THEN LIMIT%=32 : GOTO 380
362 IF REALRPM<374 THEN LIMIT%=64 : GOTO 380
363 IF REALRPM<624 THEN LIMIT%=128 : GOTO 380
364 LIMIT%=256
380 PRINT "MAXIMUM ALLOWABLE AMOUNT OF AVERAGING IS ";LIMIT%;":"
390 PRINT "ENTER DESIRED AMOUNT OF AVERAGING: ";
400 INPUT AVGNUM%
410 IF AVGNUM% > LIMIT% THEN AVGNUM% = LIMIT%
*** The letter identifying the injector is input
*** here and the filename to be used for the
*** floppy file is formed.
446 PRINT "ENTER A LETTER THAT DESIGNATES INJECTOR:"  
447 INJS=INPUT$(1)  
448 DIM NEDLVOLT(256)  
449 DIM NEDLLIFT(256)  
450 DIM RAWDATA(256)  
451 DIM FLOWRATE(256)  
452 DIM PRESVOLT(256)  
453 PRINT "DESIRED NUMBER OF DATA POINTS IS ";SIZE  
454 PRINT "DESIRED AMOUNT OF AVERAGING OF EACH POINT IS ";AVGNUM%  
455 PRINT "ACTUAL PUMP RPM IS ";REALRPMP  
456 PRINT "INJECTOR TYPE IS ";INJS  
457 PRINT "ARE THESE CORRECT? (Y/N):"; ANSWER$=INPUT$(1); PRINT  
458 IF ANSWER$ = "N" OR ANSWER$="n" THEN GOTO 380  

*** This is the beginning of a data collection loop
*** which is performed twice, once for needle lift and
*** once for reference rate.
558 FOR K = 1 TO 2  
559 DEF SEG = &HE000  
560 CMD$= "ACQUIRE;TYPE AVERAGE;COMPLETE 100;POINTS "  
561 CMD$ = CMD$ + STR$(SIZE)  
562 CMD$ = CMD$ + ";COUNT "  
563 CMD$ = CMD$ + STR$(AVGNUM%)  
564 CALL ABSOLUTE(ADDRT,CMD$, IOOUTPUT)  

*** Command number of data points (256).
570 CALL ABSOLUTE(ADDRT,CMD$, IOOUTPUT)  

*** Clear out memory space in which data will be stored.
575 DEF SEG = &HE000  
576 FOR I = 0 TO 7*SIZE  
577 POKE I,32  
578 NEXT I  
579 DEF SEG = &HE000  
580 IF K=1 THEN CMD$= ";DIGITIZE CHAN1;WAVEFORM;SOURCE CHAN1;FORMAT ASCII;DATA?"  
581 ELSE CMD$=";DIGITIZE CHAN4;WAVEFORM;SOURCE CHAN4;FORMAT ASCII;DATA?"  
582 CALL ABSOLUTE(ADDRT,CMD$, IOOUTPUT) 'SEND COMMAND TO DIGITIZE  
583 CALL ABSOLUTE(ADDRT,DATASEG%,LENGTH%,ENTERA%)  
584 CONDITION%=9 : COUNT%=0  
585 CALL ABSOLUTE(CONDITION%,COUNT%,STATUS%)  
586 DEF SEG = &HE3000  

*** COUNT% is # of bytes of collected data  
587 IF COUNT% < 0 THEN CNT = 65536! + COUNT% ELSE CNT = COUNT%  
588 J=0  
589 FOR I = 1 TO CNT+1  
600 FLAG=0  

*** The data, which has been returned in ASCII format,  
*** must be converted to floating point numbers.  
*** Each item of data is separated by a comma and the last  
*** piece of data is followed by a space. Each number  
*** must be reconstructed byte by byte.
610 IF CHR$(PEEK(I-1)) = "," THEN GOTO 750
28

```plaintext
720 IF CHR$(PEEK(I-1)) = " " THEN GOTO 750
730 NUM$ = NUM$ + CHR$(PEEK(I-1))
740 GOTO 760
750 FLAG=1
760 IF FLAG=0 THEN GOTO 810
770 RAWDATA(J) = VAL(NUM$)
780 NUM$=" ",
790 FLAG=0
800 J = J + 1
810 NEXT I

*** Aquire the waveform preamble, which contains
*** relevant information regarding the digitized
*** signal such as time base and vertical scale
*** information which are needed to convert the
*** integers returned over the HPIB interface to
*** valid times and voltages.
840 DEF SEG = &HE000
850 CMD$ = ":WAVEFORM:PREAMBLE?"
860 CALL ABSOLUTE(ADDR%,CMD$,IOUTPUT$)
870 DAT$ = SPACE$(255)
880 CALL ABSOLUTE(ADDR%,DAT$,ENTER$)
910 DEF SEG = &HE000
920 CMD$ = ":WAVEFORM:YORIGIN?"
930 GOSUB 1110
940 YORG = VAL(X$)
950 CMD$ = ":WAVEFORM:YREFERENCE?"
960 GOSUB 1110
970 YREF = VAL(X$)
980 CMD$ = ":WAVEFORM:YINCREMENT?"
990 GOSUB 1110
1000 YINC = VAL(X$)
1010 CMD$ = ":WAVEFORM:XORIGIN?"
1020 GOSUB 1110
1030 XORG = VAL(X$)
1040 CMD$ = ":WAVEFORM:XREFERENCE?"
1050 GOSUB 1110
1060 XREF = VAL(X$)
1070 CMD$ = ":WAVEFORM:XINCREMENT?"
1080 GOSUB 1110
1090 XINC = VAL(X$)
1100 GOTO 1150

*** Subroutine which aquire the requested information
*** about the signal for use in performing the conversion
*** from integer data to true voltages and times.
1110 CALL ABSOLUTE(ADDR%,CMD$,IOUTPUT$)
1120 X$= SPACE$(255)
1130 CALL ABSOLUTE(ADDR%,X$,ENTER$)
1140 RETURN
1150 FOR I = 0 TO SIZE-1
1160 IF K=1 THEN NEDLVOLT(I)=((RAWDATA(I)-YREF)*YINC)+YORG ELSE PRESVOLT(I)=
        ((RAWDATA(I)-YREF)*YINC)+YORG
1170 NEXT I
1180 NEXT K

*** End of data collection loop.
```
Now waveform baselines will be calculated and the validity of the baseline will be checked. The signal processing accuracy depends on the first ten digitized voltages being essentially flat. The user should ensure this, but if it is overlooked the data will be invalid. The program detects this type of error and does not store the data but instead loops back to the top of the program.

```plaintext
1190 BASE1=0 : BASE4=0
1200 BASELINE1 = 0 : BASELINE4 = 0
1210 FOR I = 1 TO 10
1220 BASE1 = BASE1+NEDLVOLT(I-1) : BASE4 = BASE4+PRESVOLT(I-1)
1230 NEXT I
1240 BASELINE1 = BASE1/10 : BASELINE4 = BASE4/10
1250 RANGE1 = 0 : RANGE4 = 0
1255 FOR I = 1 TO 10
1260 IF (NEDLVOLT(I-1)-BASELINE1)>0.05 THEN RANGE1 = 1
1270 IF (PRESVOLT(I-1)-BASELINE4)>0.05 THEN RANGE4 = 1
1275 IF (NEDLVOLT(I-1)-BASELINE1)<-0.05 THEN RANGE1 = 1
1280 IF (PRESVOLT(I-1)-BASELINE4)<-0.05 THEN RANGE4 = 1
1285 NEXT I
1290 IF RANGE1=1 OR RANGE4=1 THEN PRINT "BASELINE ERROR, REPEAT MEASUREMENT"
1295 IF RANGE1=1 OR RANGE4=1 THEN GOTO 10
```

The baseline is now subtracted from the voltages to generate waveforms referenced to zero.

```plaintext
1300 FOR I = 1 TO SIZE
1305 NEDLVOLT(I-1)=NEDLVOLT(I-1)-BASELINE1
1310 PRESVOLT(I-1)=PRESVOLT(I-1)-BASELINE4
1315 NEXT I
```

The voltage obtained from the strain gauge is now converted to the equivalent flow rate by applying the following (pressure voltage)-to-flow rate conversion:

```plaintext
1320 "* CONST = ($(1000psi/volt)*0.068atm/psi*flow_area*1000((mm/cm)^3))/
1330 \( sound_velocity*fuel_density)\)
1335 CONST = 1000*(22/7)*(.257*2.54)^2/4/.12*(.06805)*1000
1350 FOR I = 1 TO SIZE
1360 FLOWRATE(I-1)=CONST*PRESVOLT(I-1)/10.0
1370 NEXT I
```

Now the area under the flow rate curve is integrated to yield the reference rate volume.

```plaintext
1380 AREA =0
1390 FOR I = 1 TO SIZE
1400 SLICE=(FLOWRATE(I-1)+FLOWRATE(I))/2*XINC
1410 AREA= AREA+SLICE
1420 NEXT I
1426 PRINT "VOLUME UNDER CURVE = ";AREA;" CUBIC mm"
```

The file name (i.e. 2000180b.dat) is generated here.

```plaintext
1510 IF LEN(STR$(PUMPRP))<5 THEN B$="0"
```
1520 IF LEN(STR$(PUMPRPM))<4 THEN B$="00"
1530 IF LEN(STR$(PUMPRPM))<3 THEN B$="000"
1540 IF LEN(STR$(PUMPRPM))<5 THEN NAME$=B$
1550 A1$ = MID$(STR$(PUMPRPM),2,4)
1560 IF NAME$=B$ THEN NAME$=NAME$+A1$ ELSE NAME$=A1$
1570 IF LEN(STR$(THROTTLE))<4 THEN B$="0"
1580 IF LEN(STR$(THROTTLE))<3 THEN B$="00"
1590 IF LEN(STR$(THROTTLE))<4 THEN NAME$ = NAME$ + B$
1600 NAME$=MID$(STR$(THROTTLE),2,3)
1610 NAME$=NAME$ + A1$
1615 NAME$=NAME$+MID$(INJS,1,1)
1620 A1$ = ".DAT"
1630 NAME$=NAME$+A1$

*** The scale correction factor is generated and used to scale data here. Also, the measured fuel quantity is entered.

1640 PRINT "ENTER THE VOLUME OF FUEL COLLECTED IN CUBIC mm"
1650 INPUT MEASVOL
1651 PRINT "ENTER THE FUEL TEMPERATURE IN DEGREES FARENHEIT:"
1652 INPUT TEMP
1660 SCALER = MEASVOL/AREA
1661 PRINT "CORRECTION FACTOR FOR THIS RUN = ";SCALER
1662 PRINT "IF DATA IS BAD TYPE IN 'RESTART' TO BEGIN AGAIN."
1663 PRINT "IF MEASURED VOLUME WAS INCORRECT, TYPE 'MEASURE'."
1664 PRINT "IF EVERYTHING IS OK THEN HIT RETURN."
1665 INPUT ERROR$
1666 IF ERROR$="RESTART" OR ERROR$="restart" THEN GOTO 10
1667 IF ERROR$="MEASURE" OR ERROR$="measure" THEN GOTO 1640
1671 FOR I = 1 TO SIZE
1672 FLOWRATE(I-1)=FLOWRATE(I-1)*SCALER
1673 NEXT I

*** The data is now output to the floppy data file.

1680 OPEN NAME$ FOR OUTPUT AS #1
1690 PRINT #1,PUMPRPM
1691 PRINT #1,THROTTLE
1692 PRINT #1,XORG
1693 PRINT #1,XINC
1694 PRINT #1,SCALER
1695 PRINT #1,MEASVOL
1696 PRINT #1,SIZE
1700 FOR I = 1 TO SIZE
1720 PRINT #1,NEDLvolt(I-1)
1730 NEXT I
1740 PRINT:PRINT
1750 FOR I = 1 TO SIZE
1760 PRINT #1,FLOWRATE,(I-1)
1770 NEXT I
1773 PRINT #1,REALRPM
1774 PRINT #1,TEMP
1775 PRINT NAME$,REALRPM,"MEASVOL= ";MEASVOL,AREA,SCALER
1776 PRINT "THIS IS NEW TRANSDUCER DATA"
1780 RESTORE
1790 CLOSE #1
1800 END
APPENDIX B: STATIC MAP GENERATION PROGRAM

program finalmaker

*** This program is designed to generate a simple look-up table for
*** injection rate as a function of needle lift for a given diesel
*** injector. The user is prompted to input several items that
*** describe the data files to be used in the mapping process as
*** well as the nonlinear corrective function to be used on the raw
*** injection rate data. The program requires as inputs the name
*** of a list which contains the names of all of the data files to
*** be used and lets the user choose from among several options
*** that can be used in the map making process. A file called
*** scaling.list must also be available which contains four scale
*** factors that are used to extrapolate the map values beyond the
*** actual data that was collected. The user inputs a map name and
*** the generated map is stored in a file of that name for later
*** use. Injection rates are converted to mm^3/ms for processing
*** through the nonlinear corrective function but are stored in the
*** map in standard SI units of m^3/s. Needle lifts are in units
*** of meters throughout.

*** VARIABLE DECLARATIONS
real tempmap(256), x(256), scaler, dx, xmax, dt, cstart
real rdummy
real tshift, dj, qdot(256), volmeas
real dmap, slope1, slope2, slope3, slope4, a, b, c, d, q, r, p
integer oldstyle
integer imax, maptop, itshift, numtimes(256)
character*20 datafile, filelist, mapname
character*80 dummy
character*1 answer

*** open the scaling.list file and input the scale factors
open (8, file='scaling.list', status='unknown')
read (8, *) a, b, c, d

*** variable initializations
oldstyle=0
dx=3.90625e-6
tshift=60.0e-6

*** prompt user to input information about the data files
write (6,*) 'Are the first 7 items of data on separate'
write (6,*) 'lines (was data collected after 12/1/89)? (y/n)'
read (5, '(a1)') answer
if (answer.eq.'n') oldstyle=1
*** nonlinear correction parameter entered here
write (6,*) 'Correction polynomial is of the form:'
write (6,*) 'new rate=old rate+r*(old rate)**2'
write (6,*) 'Enter coefficient (r) for qdot:'
write (6,*) '( a value of 5.25e-3 is recommended )'
read (5,*) r

*** The user can choose a delay to use to compensate for the fact
*** that injection rate lags needle lift by approximately 60
*** microseconds due to physical sensor placement.
8 write (6,*) 'Do you want to try a different delay than'
write (6,*) '60.0e-6 seconds between x and qdot? (y/n)'
read (5,*) answer
if (answer.eq.'y') then
  write (6,*) 'Enter new delay:'
  read (5,*) tshift
endif

*** the user inputs the file list name and the desired map name
write (6,*) 'Name of file containing names of all data files?'
read (5,*) filelist
open (9, file=filelist, status='unknown')
write (6,*) 'Desired name of output map?'
read (5,*) mapname
open (2, file=mapname, status='unknown')

*** initialize the temporary map array to all zeros
do 10 i=1,256
  tempmap(i)=0.0
  numtimes(i)=0
10 continue

*** This is the start of the MAIN LOOP, which opens all of the data
*** in succession and processes the data to make the static map.
do 500 k=1,200
  read (9,'(a20)',end=501) datafile
  open (1, file=datafile, status='unknown')

*** if the data has the first 7 items on one line, the line must be
*** parsed to separate the data items.
if (oldstyle.eq.1) then
  ktotal=k
  if (ktotal .eq. 0) print*, "BARF"
  read (1,'(a80)') dummy
  ii=1
  ki=1
155 continue
if (dummy(ii:ii) .ne. ' ') then
  ji=ii
  ii=ii+1
200 if (dummy(ii:ii) .eq. ' ') then
    read(dummy(ji:ii),'(f20.0)') temp
    if (ki .eq. 4) dt=temp
    if (ki .eq. 5) scaler=temp
    if (ki .eq. 6) volmeas=temp
    ki=ki+1
ii=ii+1
    goto 155
else
    goto 200
endif
endif
if (ii .lt. 80) then
    ii=ii+1
    goto 155
endif

*** if the first 7 data items were on separate lines, the data items are read in here
else
    read (1,*) rdummy
    read (1,*) rdummy
    read (1,*) rdummy
    read (1,*) dt
    read (1,*) scaler
    read (1,*) volmeas
    read (1,*) rdummy
endif
itshift=int(tshift/dt)

*** Needle lift is read in here.
do 20 i=1,256
    read (1,*) x(i)
    x(i)=x(i)/1000.
20 continue

*** a digital 10 kHz filter will be applied to the needle lift data
y1=0.0
y2=0.0
do 21 i=1,256
    y=y1*(2.5079079039e-5*dt+5.06605918211e-10)
    y=y-y2*(2.53302959106e-10)+x(i)*dt**2
    y=y/(dt**2+2.5079079039e-5*dt+2.53302959106e-10)
    y2=y1
    y1=y
    x(i)=y
21 continue

*** The raw injection rate data is read in here. Units are converted to mm^3/ms for processing through the nonlinear corrective function and then returned to m^3/s for further processing.
do 30 i=1,256
    read (1,*) qdot(i)
    q=qdot(i)/scaler/1000.0
    if (q.ge.cstart) then
        q=q+r*q**2
    endif
    qdot(i)=q/1.0e6
30 continue

*** The rising edge of the needle lift trace is scanned here and
appropriate map locations are loaded.

i=1

40 do 45 j=1,256
dj=real(j)*dx

if (x(i) .lt. dj.and.x(i+1).ge.dj) then
  tempmap(j)=tempmap(j)+qdot(i+itshift)+
  (dj-x(i))/(x(i+1)-x(i))*(qdot(i+1+itshift))
  -qdot(i+itshift))
  xmax=x(i+1)
  imax=i+1
  numtimes(j)=numtimes(j)+1
  go to 45
endif

if (i+2.ge.256-itshift.or.i+2.ge.256) go to 45
i=i+1

45 continue

maptop=int(xmax/dx)
i=imax

The falling edge of the needle lift trace is scanned here and
appropriate map locations are loaded.

do 55 j=maptop,1,-1
dj=real(j)*dx

if (x(i) .ge. dj.and.x(i+1).lt.dj) then
  tempmap(j)=tempmap(j)+qdot(i+itshift)+
  (dj-x(i))/(x(i+1)-x(i))*(qdot(i+1+itshift))
  -qdot(i+itshift))
  numtimes(j)=numtimes(j)+1
  go to 55
endif

if (i+2.ge.256-itshift.or.i+2.ge.256) go to 55
i=i+1

55 continue

500 continue

This is the end of the MAIN LOOP.

any negative map values are zeroed out

do 510 i=1,256
  if (tempmap(i) .le. 0.0) tempmap(i)=0.0
  if (tempmap(i) .le. 1.0e-20.and.i.ge.20) then
    maptop=i-1
  endif
  go to 515
endif

510 continue

The map values are calculated here by dividing the contents of
each temporary map location by the number of elements of data
loaded into that location.

do 520 j=1,maptop
  if (numtimes(j).ne.0) tempmap(j)=tempmap(j)/numtimes(j)

520 continue
the map is extrapolated here
slopel = (tempmap(maptop-8) - tempmap(maptop-18))/10.0
slope2 = (tempmap(maptop-6) - tempmap(maptop-16))/10.0
slope3 = (tempmap(maptop-4) - tempmap(maptop-14))/10.0
slope4 = (tempmap(maptop) - tempmap(maptop-6))/6.0

dmap = (a*slopel + b*slope2 + c*slope3 + d*slope4)/(a+b+c+d)
do 550  j=maptop,256
   tempmap(j) = tempmap(j-1) + dmap
continue
do 600  j=1,256
write (2,* ) tempmap(j)
continue
write (2,*) maptop
write (2,*) tshift
write (2,*) r
write (2,* ) 'The four lines immediately above contain:'
write (2,* ) 'last nonextrapolated map point'
write (2,* ) 'delay used from lift to rate data (seconds)'
write (2,* ) 'correction polynomial coefficient r'
write (2,* ) 'correction polynomial=q+r*q^2'
end
APPENDIX C: GRAPHICAL DATA CONVERSION PROGRAM

program datacon

*** This program takes raw data for injection rate and needle lift
*** and processes the data through the static map and then the
*** static map plus model. This processed data is then loaded into
*** appropriately formatted files that can be graphically plotted.
*** Two arguments are required on the command line and a file
*** called dclist must be available which is read for the
*** model parameters and other data variables. An example of this
*** data file is included at the end of this program listing as
*** comments. The two required arguments on the command line are
*** the name of the file to be processed and the header of the
*** graphics input files. A sample command line would be
*** datacon 2000180k.dat dk-2000-180. Units are mm^3/ms for
*** injection rate and mm for needle lift.

*** Variable declarations
real map(256), x(256), scaler, dx, dt, a, b, w, f1, mdot
real xdot(256), xddot(256), qdotraw(256), qdotnew(256)
real dj, qdot(256), volmeas, r, delay, s
real rho, f1po(256), y, y1, y2
integer i, k, j, chipflag, newflag
character*22 mapname, headername, arg, string
character*30 mapratefile, modratefile, nlifile
character*30 refratefile, timefile
character*80 dummy

*** variable initialization
dx=3.90625e-3
rho=.8249
newflag=0
r=8.0e-3
s=8.0e-5

*** get the first command line argument and open the named file
call getarg(1, arg)
open (1, file=arg, status='unknown')
open (3, file='dclist', status='unknown')

*** get the second command line argument and form the required
*** output filenames from the header
call getarg(2, arg)
headername=arg
k = index(headername, ' ')
mapratefile=headername(1:k-1)//'.maprate'
modratefile=headername(1:k-1)//'.modrate'
The map is loaded here (map units are m^3/s for rate) *****
Note: multiplication by rho (density in kg/m^3) and by 1.e6 produces mdot, which is mass flow rate in ug/s. Later division by rho will give flow rate in ul/ms.

read (3,*) mapname
open (2, file=mapname, status='unknown')
do 10 i=1,256
read (2,*) map(i)
map(i)=map(i)*rho*1.e6
10 continue

The first seven lines of data are loaded here. The user must let the program know if the data file is old format (first seven items of data on one line) or new format (one item of data per line).

read (3,*) string
if (string.eq.'newdata') newflag=1
if (newflag.eq.0) then
    read (1,'(a80)') dummy
    ii=1
    ki=1
155 continue
    if (dummy(ii:ii) .ne. ') then
        ji=ii
        200 ii=ii+1
        if (dummy(ii:ii) .eq. '') then
            read(dummy(ji:ii),'(f20.0)') temp
            if (ki .eq. 4) dt=temp
            if (ki .eq. 5) scaler=temp
            if (ki .eq. 6) volmeas=temp
            ki=ki+1
            ii=ii+1
            goto 155
        endif
        goto 200
    endif
    endif
    if (ii .lt. 80) then
        ii=ii+1
        goto 155
    endif
else
    read (1,*) temp
read (1,*) temp
read (1,*) temp
read (1,*) dt
read (1,*) scaler
read (1,*) volmeas
read (1,*) temp
endif

***** Raw data for needle lift in mm is read in here *****
do 20 i=1,256
   read (1,*), x(i)
20 continue

***** The user must let the program know if the data *****
***** was from the transducer-on-a-chip or from the *****
***** strain gauge. If the data was from the strain *****
***** gauge, then a digital 10kHz filter is applied *****
***** to the data.
read (3,*) string
if (string.eq.'filter') then
   chipflag=0
elseif (string.eq.'nofilter') then
   chipflag=1
endif
if (chipflag.eq.0) then
   y1=0.0
   y2=0.0
   do 21 i=1,256
      y=y1*(2.25079079039e-5*dt+5.06605918211e-10)
      y=y-y2*(2.53302959106e-10)+x(i)*dt**2
      y2=y1
      y1=y
      x(i)=abs(y)
   21 continue
endif

***** Needle velocity and acceleration are calculated here *****
xdot(1)=0.0
xdot(2)=0.0
xdot(255)=0.0
xdot(256)=0.0
do 50 i=3,254
   xdot(i)=(x(i+4)-x(i-2))/4.0/dt
50 continue
xdot(1)=0.0
xdot(2)=0.0
xdot(3)=0.0
xdot(4)=0.0
xdot(253)=0.0
xdot(254)=0.0
xdot(255)=0.0
xdot(256)=0.0
do 70 i=5,252
   xddot(i)=(x(i+4)-x(i)*2.0+x(i-4))/16.0/dt**2
70 continue
The fl function (mass flow rate from needle lift) is calculated here.

** MODELS MODEL EQUATION

This section of the program is where the model is implemented.**

NOTE that this is inside a do loop (i = 1 to 256).

Models should be designed to calculate mass flow rate, andConversion to volume flow rate occurs on last line of this section.

** UNITS AND DESCRIPTIONS:

* a [mg^2/(mm*s)]: model parameter, best value is .3998e-1
* b [mg^2/mm]: model parameter, best value is -.48042e-5
* fl [mg/s]: mass flow rate from map
* mdot [mg/s]: mass flow rate produced by the model
* qdot [ul/ms]: volume flow rate produced by the model
* rho [kg/m^3]: diesel fuel density
* w [unitless]: model parameter, best value is .36627
* x [mm]: needle lift
* xdot [mm/s]: needle velocity
* xddot [mm/s^2]: needle acceleration

Model parameters for new models can be entered below (they should be entered on the first loop iteration, when i = 1), and they can be declared at the top of the program, where they can be cross checked for variable duplication.

if (i.eq.1) then
    write (6,*) 'Model is mdot=sqrt(f1^2+x*w*(a*xdot+b*xddot))'
    read (3,*) a
    read (3,*) b
    read (3,*) w
endif

mdot=f1^2+x(i)*w*(a*xdot(i)+b*xddot(i))
if (mdot.lt.0.0) then
    mdot=-1.0*sqrt(-1.0*mdot)
else
    mdot=sqrt(mdot)
endif
qdot(i)=mdot/rho

The raw flow rate data is read in from the data file here.**

At the initial time of data collection, the flow rate data was prescaled. Therefore it must be unscaled now. In addition, volume flow rate was stored in ul/s so we must divide by 1000 to get units of ul/ms.
read (1,*) qdotraw(i)
qdotraw(i)=qdotraw(i)/scaler/1000.0
if (i.eq.1) then
  read (3,*) r
  read (3,*) s
endif
qdotnew(i)=qdotraw(i)+r*qdotraw(i)**2+s*qdotraw(i)**3
flpo(i)=f1/rho

continue
close (unit=1)

****** The data is now output to the various files for use with ******
****** dplot. Time will be in milliseconds, needle lift in mm, ******
****** and rates in ul/ms. ********
read (3,*) delay
idelay=int(delay/dt)
do 35 i=25,225
  time=real((i-25)*(1.0e6*dt))/1000.0
  write (20,*) time
  write (21,*) x(i)
  if (i+idelay.gt.256) then
    write (22,*) 0.0
  else
    write (22,*) qdotnew(i+idelay)
  endif
  write (23,*) flpo(i)
  write (24,*) qdot(i)
format (6x, f6.2, 4x, f7.3, 5x, f6.4, 
+3x, f6.2, 7x, f6.2, 9x, f6.2)
35 continue
end

*** This is the dclist file that must be available for the
*** datacon program (minus asterisks)

*** bestkmap  this is the static map to be used
*** olddata   (newdata/olddata) indicates data format
*** filter   (filter/nofilter) want to 10kHz filter needle lift?
*** .05      model parameter a in abw model
*** 0.000055  model parameter b in abw model
*** 1.50     model parameter w in abw model
*** 5.25e-3  coefficient of qdotold**2 in corrective function
*** 0        coefficient of qdotold**3 in corrective function
*** 160e-6   delay to apply to raw rate to align it in time

*** This list is used as input to the datacon program,
*** which converts raw data files to files that can be used by dplot.
*** The command line format to run datacon is "datacon 2000180k.dat
*** dk-2000-180", where the 1st argument is the datafile to be
*** opened, and the 2nd argument is the header name for the files
*** to be used with dplot.
APPENDIX D: GRAPHICAL DISPLAY SHELL SCRIPT

This program accepts files in the format provided by the datacon program and produces a laser printed graphical display of the data. The command line can accept 3 arguments, two of which are optional. A typical command line would be "dplot dk-2000-180 3 4" where dk-2000-180 is the header for the files produced by datacon, 3 is the maximum time in milliseconds, and 4 implies a vertical maximum of .4 mm for needle lift and 40 mm^3/ms for injection rate.

#!/bin/csh

# This version for laser printer output.
set laserflag = y
if -e ${1}.time then
else
  echo "data files not found"
  goto OUT
endif

# Input Parameters
# Laser print or tektronix screen plot?
  if ($laserflag :: y) then
    set size = 0.85
  else
    set size = 1.0
  endif
# The title of the plot:
  set title="Injector Response: $1"
# Number of points plotted
  set pts=201
# Time axis max value
  if ( $2 == ' ' ) then
    set time = 4.0
  else
    set time = $2
  endif
# Vertical axis scaling
  if ( $3 == ' ' ) then
    set vaxis = 8
  else
    set vaxis = $3
  endif
if -e gl rm gl

qplot x=$1.time,a y=$1.nlift,a titanium="Time (ms)"
    ylabel="Needle Lift (mm) or Rate (0.1 L/s)"
    title="$title" bl="Injector Needle Lift and Rate vs Time"
    xmin=0.0 xmax=$time
    ymin=0.0 ymax="0.${vaxis}" dig=3 scfac=$size xp=0 yp=0
    xlen=10.0 ylen=8.0 xtic=1.25 ytic=1.0
-q>$1

qplot x=$1.time,a y=$1.refrate,a xmin=0.0 xmax=$time
    ymin=0.0 ymax="$vaxis" dig=3 scfac=$size xp=0 yp=0
    xlen=10.0 ylen=8.0 xtic=1.25 ytic=1.0
-P -d dash=0.01 gap=0.05 -a -b >> gl

qplot x=$1.time,a y=$1.maprate,a xmin=0.0 xmax=$time
    ymin=0.0 ymax="$vaxis" dig=3 scfac=$size xp=0 yp=0
    xlen=10.0 ylen=8.0 xtic=1.25 ytic=1.0
-P -d dash=0.05 gap=0.05 -a -b >> gl

qplot x=$1.time,a y=$1.modrate,a xmin=0.0 xmax=$time
    ymin=0.0 ymax="$vaxis" dig=3 scfac=$size xp=0 yp=0
    xlen=10.0 ylen=8.0 xtic=1.25 ytic=1.0
-P -d dash=0.2 gap=0.05 -a -b >> gl

if ($laserflag == y) then
    lpr -g gl
else
    if -e plotready rm plotready
    plot -T4014 gl > plotready
cat plotready
    rm plotready
    set dumb=($<)
    set dumb=($<)
endif

OUT:
echo " "
}
This program is designed to be used on the CRAY X-MP in San Diego. It uses an IMSL nonlinear least squares optimization routine to solve for model parameters $a$, $b$, and $w$ in the diesel fuel injector model $mdot=(f1^2+x^w(a*xdot+b*xddot))^{0.5}$. Special note: All variables are in SI units, needle lift in meters and injection rate in $m^3/s$. This means that parameters $a$ and $b$ will be 1e6 times larger than in other programs!

variable and common declarations
character*20 datafile
integer i,k
integer iparam(6),m,n,ldfjac
real x(256),y,y1,y2,dt(20),scaler
real xguess(3),xscale(3),fscal(20),s(3)
real rparam(7),fvec(20),fjac(20,3),xlb(3),xub(3)
real ascale,bscale,map(256),ax(5120)
real dx,rho, delay, volmeas(20)
common ascale,bscale,map,ax,
+dx,rho, delay, volmeas, dt
call link ('//')

variable initialization
rho=.8249
dx=3.90625e-3
delay=60.e-6
ntemp=0

open the map file and read in the map values
open (11, file='fkmap', status='unknown')
do 20 i=1,256
read (11,*) map(i)
map(i)=map(i)*rho*1.0e9
continue
close (unit=11)

read in the data from the data files
open (9, file='kiki180', status='unknown')
do 500 k=1,200
read (9,'(a20)', end=501) datafile
open (1, file=datafile, status='unknown')
read (1,*) y
read (1,*) y
read (1, *) y
read (1, *) dt(k)
read (1, *) scaler
read (1, *) volmeas(k)
read (1, *) y

do 10 i = 1, 256
    read (1, *) x(i)
10 continue

v1 = 0.0
v2 = 0.0

* ¡k* apply a 10 kHz filter to needle lift

do 21 i = 1, 256
    y = y1 * (2.25079079039e-5 * dt(k) + 5.06605918211e-10)
    y = y - y2 * (2.5330199106e-10) + x(i) * dt(k) ** 2
    y = y / (dt(k) ** 2 + 2.25079079039e-5 * dt(k) + 2.5330199106e-10)
    y1 = y2
    y2 = y
    x(i) = y
21 continue

close (unit=1)

*** load needle lift into a huge array

do 71 i = 1, 256
    ntemp = ntemp + 1
    ax(ntemp) = x(i)
71 continue

500 continue

501 close (unit=9)

*** enter scale factors here to keep a and b on the same order of magnitude

write (6, *) 'Enter scale factor for a (nom. 1.0e?):'
read (5, *) ascale
write (6, *) 'Enter scale factor for b (nom. 1.0e2):'
read (5, *) bscale

*** initialize parameters for BCLSF subroutine

ldfjac = 20
m = 20
n = 3

do 1 i = 1, 3
    xscale(i) = 1.0
1 continue

do 2 i = 1, 20
    fscale(i) = 1.0
2 continue

iparam(1) = 0

*** input initial guesses and bounds for parameters

write (6, *) 'mdot=(f1^2+x^w*(a*xdot+b*xddot))^0.5',
write (6, *) 'enter initial value for a:'
read (5, *) xguess(1)
write (6, *) 'enter initial value for b:'
read (5, *) xguess(2)
write (6,*) 'enter initial value for w:'
read (5,*) xguess(3)
write (6,*) 'bounds?: enter 1 if all positive, 0 otherwise'
read (5,*) ibtype
if (ibtype.eq.0) then
  write (6,*) 'enter lower and upper bounds on a:'
  read (5,*) xlb(l_), xub(l_)
  write (6,*) 'enter lower and upper bounds on b:'
  read (5,*) xlb(2), xub(2)
  write (6,*) 'enter lower and upper bounds on w:'
  read (5,*) xlb(3), xub(3)
endif

*** call the BCLSF subroutine
call bclsf(fcn,m,n,coeff,xguess,ibtype,xlb,xub,xscale,
          +fscale,iparam,mparam,s,fvec,fjac,ldfjac)

*** Print out the results and store them in a file.
format(3(3x,e12.5))
s(1)=s(1)*ascale
s(2)=s(2)*bscale
write (6,*),' a b w'
write (6,3) s(1),s(2),s(3)
open (unit=2,file='abw',status='unknown')
write (2,*),s(1),s(2),s(3)
end

*** This subroutine provides information to the BCLSF subroutine to enable it to calculate the finite-
*** difference Jacobian.
subroutine fcn(m,n,coeff,f)
  real ascale,bscale,map(256),ax(5120),x(256)
  real dx,rho,de1ay,dt(20),volmeas(20),mdot
  integer istep
  real a,b,coeff(3),f(20),xdot(256),xddot(256)
  common ascale,bscale,map,ax,
  +dx,rho,de1ay,volmeas,dt
  a=coeff(1)*ascale
  b=coeff(2)*bscale
  w=coeff(3)
  write (6,*)'1',a,b,w
  do 10 i=1,20
    f(i)=0.0
  continue
  xdot(1)=0.0
  xdot(2)=0.0
  xdot(255)=0.0
  xdot(256)=0.0
  xddot(1)=0.0
  xddot(2)=0.0
  xddot(3)=0.0
  xddot(4)=0.0
  xddot(253)=0.0
  xddot(254)=0.0
  xddot(255)=0.0
xddot(256)=0.0
do 50 istep=1,20
   is=(istep-1)*256
   do 25 i=1,256
      x(i)=ax(is+i)
      continue
   enddo
   do 30 i=3,254
      xdot(i)=(x(i+2)-x(i-2))/4.0/dt(istep)
      continue
   enddo
end

xxddot(i)=(x(i+4)-x(i)+2.0+x(i-4))/
+16.0/dt(istep)**2
continue
write (6,*) '2',
do 45 i=1,256
   x(i)=abs(x(i))
   j=int(x(i)/dx)
   dj=real(j)*dx
   if (j.eq.0) then
      fl=x(i)/dx*map(1)
   else
      fl=map(j)+(x(i)-dj)/dx*(map(j+1)-map(j))
   endif
   mdot=fl**2+x(i)**w*(a*xdot(i)+b*xxddot(i))
   if (mdot.le.0.0) mdot=1.0e-40
   mdot=sqrt(mdot)
write (6,*) '3',
f(istep)=f(istep)+mdot*dt(istep)/rho
continue
f(istep)=volmeas(istep)-f(istep)
continue
return
end
APPENDIX F: MODEL ACCURACY DETERMINATION PROGRAM

program final_error

*** This program produces an error table and calculates the absolute average error for a given set of model parameters. The user is prompted for various information. The units used internally by the program are standard SI units, but in terms of model parameters and nonlinear correction factor the units of needle lift can be considered mm and the units of injection rate mm^3/ms. Internal conversions take care of this.

*** variable declarations
character*20 datafile, filelist, mapname
character*80 dummy
character*1 answer
real map (256), x(256), scaler, dx, dt,a,b,w,f1,mdot
real xdot(256),xdot(256),qdotraw(256),r,qdotnew(256)
real dj, qdot(256), volmeas, volcalc,accerr,aberr
real volbosch,berr,cerr,tberr,tcerr,count,rho
integer i,k,j,chipflag,newflag

*** variable initialization
dx=3.90625e-3
rho=.8249
r=5.25e-6
open (7,file='printout',status='unknown')

*** user is prompted for various information
write (6,*) 'Name of file containing names of all data files?'
read (5,*) filelist
open (9,file=filelist,status='unknown')
write (6,*) 'What is the map name?'
read (5,*) mapname
open (2,file=mapname,status='unknown')

*** map values are read in here
do 10 i=1,256
   read (2,*) map(i)
   map(i)=map(i)*rho*1.e9
continue
count=0.0
 tberr=0.0
tcerr=0.0
aberr=0.0
acerr=0.0
do 500 k=1,200
read (9,'(a20)',end=501) datafile
open (1, file=datafile, status='unknown')
ktotal=k

***** The first seven lines of data are loaded here. The *****
***** user must let the program know if the data file is *****
***** old format (first seven items of data on one line) *****
***** or new format (one item of data per line).  *****

if (k.eq.1) then
   write (6,*) 'Is the data new format (separate lines for',
     +' first 7 items)? (y/n)
   read (5,*) answer
   if (answer.eq.'n') then
      newflag=0
   elseif (answer.eq.'y') then
      newflag=1
   else
      write (6,*) 'Answer must be y or n.'
      goto 11
   endif
endif
if (newflag.eq.0) then
   read (1, '(a80)'). dummy
   ii=1
   ki=1
   continue
   if (dummy(ii:ii) .ne. ', ') then
      ji=ii
      ii=ii+1
      if (dummy(ii:ii) .eq. ', ') then
         read(dummy(ji:ii),'(f20.0)') temp
         if (ki .eq. 4) dt=temp
         if (ki .eq. 5) scaler=temp
         if (ki .eq. 6) volmeas=temp
         ki=ki+1
         ii=ii+1
         goto 155
      else
         goto 200
      endif
   endif
   if (ii .lt. 80) then
      ii=ii+1
   goto 155
   endif
else
   read (1,*) temp
   read (1,*) temp
   read (1,*) temp
   read (1,*) dt
   read (1,*) scaler
   read (1,*) volmeas
   read (1,*) temp
endif

do 20 i=1,256
   read (1,*) x(i)
 20 continue

***** The user must let the program know if the data *****
***** was from the transducer-on-a-chip or from the *****
***** strain gauge. If the data was from the strain *****
***** gauge, then a digital 10kHz filter is applied *****
***** to the data.

if (k.eq.1) then
   write (6,*) 'Was data from Chip transducer or strain'
   +,' Gauge? (c/g)'
   read (5,*) answer
   if (answer.eq.'c') then
      chipflag=1
   elseif (answer.eq.'g') then
      chipflag=0
   else
      write (6,*) 'Answer must be c or g.'
      goto 19
   endif
endif
if (chipflag.eq.0) then
   y1=0.0
   y2=0.0
   do 21 i=1,256
      y=y1*(2.25079079039e-5*dt+5.06605918211e-10)
      y=y2*(2.53302959106e-10)+x(i)*dt**2
      y=y/(dt**2+2.25079079039e-5*dt+2.53302959106e-10)
      y2=y1
      y1=y
      x(i)=abs(y)
      21 continue
endif
   xdot(1)=0.0
   xdot(2)=0.0
   xdot(255)=0.0
   xdot(256)=0.0
   do 50 i=3,254
      xdot(i)=(x(i+2)-x(i-2))/4.0/dt
      50 continue
   xddot(1)=0.0
   xddot(2)=0.0
   xddot(3)=0.0
   xddot(4)=0.0
   xddot(253)=0.0
   xddot(254)=0.0
   xddot(255)=0.0
   xddot(256)=0.0
do 70 i=5,252
    xdot(i)=(x(i+4)-x(i)*2.0+x(i-4))/16.0/dt**2
70 continue

do 30 i=1,256
if (x(i).le.0.0) x(i)=0.0
j=int(x(i)/dx)
dj=real(j)*dx
if (j.eq.0) then
    f1=x(i)/dx*map(1)
else
    f1=map(j)+(x(i)-dj)/dx*(map(j+1)-map(j))
endif

*********** MODEL EQUATION ***********

*** This section of the program is where the model is implemented.***
***** NOTE that this is inside a do loop (i = 1 to 256).*****
***** Models should be designed to calculate mass flow rate, and *****
***** Conversion to volume flow rate occurs on last line of this *****
***** section.*****

***** UNITS AND DESCRIPTIONS: ******
* a [mg^2/(mm*s)]: model parameter, best value is .3998e-1
* b [mg^2/mm]: model parameter, best value is -.4804e-5
* f1 [mg/s]: mass flow rate from map
* mdot [mg/s]: mass flow rate produced by the model
* qdot [ul/ms]: volume flow rate produced by the model
* rho [kg/m^3]: diesel fuel density
* w [unitless]: model parameter, best value is .36627
* x [mm]: needle lift
* xdot [mm/s]: needle velocity
* xddot [mm/s^2]: needle acceleration

***** Model parameters for new models can be entered below (they should*****
****** be entered on the first loop iteration, when i = 1), and they*****
****** can be declared at the top of the program, where they can be*****
****** cross checked for variable duplication.*****

if (k.eq.1.and.i.eq.1) then
    write (6,*)'Model is mdot=sqrt(f1^2+x^w*(a*xdot+b*xddot))'
    write (6,*)'Enter value for a:'
    read (5,*) a
    a=a*1.0e6
    write (6,*)'Enter value for b:'
    read (5,*) b
    b=b*1.0e6
    write (6,*)'Enter value for w:'
    read (5,*) w
endif
mdot=f1**2+x(i)**w*(a*xdot(i)+b*xddot(i))
if (mdot.lt.0.0) then
    mdot=0.0
else
\[
\text{mdot} = \sqrt{\text{mdot}}
\]

endi

\[
\text{qdot}(i) = \text{mdot}/\rho
\]

read (1,*) qdotraw(i)
qdotraw(i) = qdotraw(i)/scaler
if (k.eq.1.and.i.eq.1) then
  s=0.0
  p=0.0
  write (6,*) 'There is a nonlinear correction (the so-called',
  '+' Amazing Walt Function')
  write (6,*) 'of the form new=old+r*old^2',
  '+' which is applied to the'
  write (6,*) 'raw rate data. The default values are r=5.25e-6,'
  '+' and phase in at 0 ul/ms:'
  write (6,*) 'are these values OK? (y/n)',
  read (5,*) answer
  if (answer.eq.'n',) then
    write (6,*) 'Enter new value for r:',
    read (5,*) r
    r=r/1000.0
    write (6,*) 'Enter new value for p:',
    read (5,*) p
    p=p*1000.0
  endif
endif
if (qdotraw(i).gt.p) then
  qdotnew(i)=qdotraw(i)+r*qdotraw(i)**2
else
  qdotnew(i)=qdotraw(i)
endif
if (k.eq.1.and.i.eq.1) then
  write (7,*)
  write (7,*) 'Map used is ',mapname,' Data from ',
  +filelist
  write (7,*)
  write (7,9) a/1.0e6,b/1.0e6,w
  write (7,*)
  write (7,9) 'Nonlinear correction to rate is: qdotnew'
  write (7,9) '=qdotold+',r*1000.0,'*qdotold**2'
  write (7,*)
  write (7,*) 'filename buret (mm^3) ',
  '+int. rate/%error calculated/%error'
  write (7,*)
endif
if (i.eq.1) then
  volcalc=qdot(i)*dt/2.0
  volbosch=qdotnew(i)*dt/2.0
else
  volcalc=volcalc+(qdot(i)+qdot(i-1))/2.0*dt
  volbosch=volbosch+(qdotnew(i)+qdotnew(i-1))/2.0*dt
endif
continue
format (6x,'a= ',e12.5,9x,'b= ',e12.5,9x,'w= ',e12.5)
close (unit=1)
cerr = 100.0 * (volcalc - volmeas) / volmeas
berr = 100.0 * (volbosch - volmeas) / volmeas
write (7,51) datafile, volmeas, volbosch, berr, volcalc, cerr
format (6x, a16, 4x, f6.2, 3x, 2(2x, f6.2), 2x, 2(2x, f6.2))
tberr = tberr + abs(berr)
tcerr = tcerr + abs(cerr)
aberr = aberr + berr
acerr = acerr + cerr
count = count + 1.0
continue
500 tberr = tberr / count
tcerr = tcerr / count
aberr = aberr / count
acerr = acerr / count
write (7, *)
write (7, *) ' Average absolute integrated ref. rate %',
+ ' error = ', tberr
write (7, *)
write (7, *) ' Average absolute calc. (via map) vol. %',
+ ' error = ', tcerr
write (7, *)
write (7, *) ' Average (signed) integrated ref. rate %',
+ ' error = ', aberr
write (7, *)
write (7, *) ' Average (signed) calc. (via map) vol. %',
+ ' error = ', acerr
close (unit=7)
end
5. Static Nonlinear Map for DK Injector used for Engine Tests.

<table>
<thead>
<tr>
<th>point</th>
<th>lift (mm)</th>
<th>rate (ul/ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
<td>0.000</td>
</tr>
<tr>
<td>1</td>
<td>0.0039</td>
<td>0.872</td>
</tr>
<tr>
<td>2</td>
<td>0.0078</td>
<td>1.646</td>
</tr>
<tr>
<td>3</td>
<td>0.0117</td>
<td>2.372</td>
</tr>
<tr>
<td>4</td>
<td>0.0156</td>
<td>2.939</td>
</tr>
<tr>
<td>5</td>
<td>0.0195</td>
<td>3.363</td>
</tr>
<tr>
<td>6</td>
<td>0.0234</td>
<td>3.669</td>
</tr>
<tr>
<td>7</td>
<td>0.0273</td>
<td>3.912</td>
</tr>
<tr>
<td>8</td>
<td>0.0312</td>
<td>4.096</td>
</tr>
<tr>
<td>9</td>
<td>0.0352</td>
<td>4.245</td>
</tr>
<tr>
<td>10</td>
<td>0.0391</td>
<td>4.366</td>
</tr>
<tr>
<td>11</td>
<td>0.0430</td>
<td>4.475</td>
</tr>
<tr>
<td>12</td>
<td>0.0469</td>
<td>4.575</td>
</tr>
<tr>
<td>13</td>
<td>0.0508</td>
<td>4.660</td>
</tr>
<tr>
<td>14</td>
<td>0.0547</td>
<td>4.736</td>
</tr>
<tr>
<td>15</td>
<td>0.0586</td>
<td>4.816</td>
</tr>
<tr>
<td>16</td>
<td>0.0625</td>
<td>4.885</td>
</tr>
<tr>
<td>17</td>
<td>0.0664</td>
<td>4.964</td>
</tr>
<tr>
<td>18</td>
<td>0.0703</td>
<td>5.036</td>
</tr>
<tr>
<td>19</td>
<td>0.0742</td>
<td>5.099</td>
</tr>
<tr>
<td>20</td>
<td>0.0781</td>
<td>5.151</td>
</tr>
<tr>
<td>21</td>
<td>0.0820</td>
<td>5.214</td>
</tr>
<tr>
<td>22</td>
<td>0.0859</td>
<td>5.285</td>
</tr>
<tr>
<td>23</td>
<td>0.0898</td>
<td>5.351</td>
</tr>
<tr>
<td>24</td>
<td>0.0938</td>
<td>5.418</td>
</tr>
<tr>
<td>25</td>
<td>0.0977</td>
<td>5.487</td>
</tr>
<tr>
<td>26</td>
<td>0.1016</td>
<td>5.550</td>
</tr>
<tr>
<td>27</td>
<td>0.1055</td>
<td>5.609</td>
</tr>
<tr>
<td>28</td>
<td>0.1094</td>
<td>5.673</td>
</tr>
<tr>
<td>29</td>
<td>0.1133</td>
<td>5.745</td>
</tr>
<tr>
<td>30</td>
<td>0.1172</td>
<td>5.824</td>
</tr>
<tr>
<td>31</td>
<td>0.1211</td>
<td>5.899</td>
</tr>
<tr>
<td>32</td>
<td>0.1250</td>
<td>5.975</td>
</tr>
<tr>
<td>33</td>
<td>0.1289</td>
<td>6.057</td>
</tr>
<tr>
<td>34</td>
<td>0.1328</td>
<td>6.144</td>
</tr>
<tr>
<td>35</td>
<td>0.1367</td>
<td>6.236</td>
</tr>
<tr>
<td>36</td>
<td>0.1406</td>
<td>6.331</td>
</tr>
<tr>
<td>37</td>
<td>0.1445</td>
<td>6.424</td>
</tr>
<tr>
<td>38</td>
<td>0.1484</td>
<td>6.507</td>
</tr>
<tr>
<td>39</td>
<td>0.1523</td>
<td>6.589</td>
</tr>
<tr>
<td>40</td>
<td>0.1562</td>
<td>6.668</td>
</tr>
<tr>
<td>41</td>
<td>0.1602</td>
<td>6.760</td>
</tr>
<tr>
<td>42</td>
<td>0.1641</td>
<td>6.856</td>
</tr>
<tr>
<td>43</td>
<td>0.1680</td>
<td>6.958</td>
</tr>
<tr>
<td>44</td>
<td>0.1719</td>
<td>7.060</td>
</tr>
<tr>
<td>45</td>
<td>0.1758</td>
<td>7.170</td>
</tr>
<tr>
<td>46</td>
<td>0.1797</td>
<td>7.270</td>
</tr>
<tr>
<td>47</td>
<td>0.1836</td>
<td>7.369</td>
</tr>
<tr>
<td>48</td>
<td>0.1875</td>
<td>7.486</td>
</tr>
<tr>
<td>49</td>
<td>0.1914</td>
<td>7.595</td>
</tr>
<tr>
<td>50</td>
<td>0.1953</td>
<td>7.718</td>
</tr>
<tr>
<td>51</td>
<td>0.1992</td>
<td>7.821</td>
</tr>
<tr>
<td>52</td>
<td>0.2031</td>
<td>7.931</td>
</tr>
<tr>
<td>53</td>
<td>0.2070</td>
<td>8.052</td>
</tr>
<tr>
<td>54</td>
<td>0.2109</td>
<td>8.180</td>
</tr>
<tr>
<td>55</td>
<td>0.2148</td>
<td>8.294</td>
</tr>
<tr>
<td>56</td>
<td>0.2188</td>
<td>8.411</td>
</tr>
<tr>
<td>57</td>
<td>0.2227</td>
<td>8.523</td>
</tr>
<tr>
<td>58</td>
<td>0.2266</td>
<td>8.644</td>
</tr>
<tr>
<td>59</td>
<td>0.2305</td>
<td>8.767</td>
</tr>
<tr>
<td></td>
<td>0.2344</td>
<td>8.902</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>60</td>
<td>0.2383</td>
<td>9.032</td>
</tr>
<tr>
<td>61</td>
<td>0.2422</td>
<td>9.154</td>
</tr>
<tr>
<td>62</td>
<td>0.2461</td>
<td>9.283</td>
</tr>
<tr>
<td>63</td>
<td>0.2500</td>
<td>9.395</td>
</tr>
<tr>
<td>64</td>
<td>0.2539</td>
<td>9.515</td>
</tr>
<tr>
<td>65</td>
<td>0.2578</td>
<td>9.641</td>
</tr>
<tr>
<td>66</td>
<td>0.2617</td>
<td>9.772</td>
</tr>
<tr>
<td>67</td>
<td>0.2656</td>
<td>9.892</td>
</tr>
<tr>
<td>68</td>
<td>0.2695</td>
<td>10.011</td>
</tr>
<tr>
<td>69</td>
<td>0.2734</td>
<td>10.149</td>
</tr>
<tr>
<td>70</td>
<td>0.2773</td>
<td>10.277</td>
</tr>
<tr>
<td>71</td>
<td>0.2812</td>
<td>10.409</td>
</tr>
<tr>
<td>72</td>
<td>0.2852</td>
<td>10.548</td>
</tr>
<tr>
<td>73</td>
<td>0.2891</td>
<td>10.690</td>
</tr>
<tr>
<td>74</td>
<td>0.2930</td>
<td>10.896</td>
</tr>
<tr>
<td>75</td>
<td>0.2969</td>
<td>11.061</td>
</tr>
<tr>
<td>76</td>
<td>0.3008</td>
<td>11.285</td>
</tr>
<tr>
<td>77</td>
<td>0.3047</td>
<td>11.464</td>
</tr>
<tr>
<td>78</td>
<td>0.3086</td>
<td>11.654</td>
</tr>
<tr>
<td>79</td>
<td>0.3125</td>
<td>11.829</td>
</tr>
<tr>
<td>80</td>
<td>0.3164</td>
<td>12.062</td>
</tr>
<tr>
<td>81</td>
<td>0.3203</td>
<td>12.327</td>
</tr>
<tr>
<td>82</td>
<td>0.3242</td>
<td>12.623</td>
</tr>
<tr>
<td>83</td>
<td>0.3281</td>
<td>12.943</td>
</tr>
<tr>
<td>84</td>
<td>0.3320</td>
<td>13.285</td>
</tr>
<tr>
<td>85</td>
<td>0.3359</td>
<td>13.665</td>
</tr>
<tr>
<td>86</td>
<td>0.3398</td>
<td>14.071</td>
</tr>
<tr>
<td>87</td>
<td>0.3438</td>
<td>14.601</td>
</tr>
<tr>
<td>88</td>
<td>0.3477</td>
<td>15.198</td>
</tr>
<tr>
<td>89</td>
<td>0.3516</td>
<td>15.798</td>
</tr>
<tr>
<td>90</td>
<td>0.3555</td>
<td>16.517</td>
</tr>
<tr>
<td>91</td>
<td>0.3594</td>
<td>17.333</td>
</tr>
<tr>
<td>92</td>
<td>0.3633</td>
<td>18.119</td>
</tr>
<tr>
<td>93</td>
<td>0.3672</td>
<td>18.927</td>
</tr>
<tr>
<td>94</td>
<td>0.3711</td>
<td>19.946</td>
</tr>
<tr>
<td>95</td>
<td>0.3750</td>
<td>20.980</td>
</tr>
<tr>
<td>96</td>
<td>0.3789</td>
<td>21.998</td>
</tr>
<tr>
<td>97</td>
<td>0.3828</td>
<td>23.176</td>
</tr>
<tr>
<td>98</td>
<td>0.3867</td>
<td>24.416</td>
</tr>
<tr>
<td>99</td>
<td>0.4006</td>
<td>25.807</td>
</tr>
<tr>
<td>100</td>
<td>0.3945</td>
<td>27.221</td>
</tr>
<tr>
<td>101</td>
<td>0.3984</td>
<td>28.550</td>
</tr>
<tr>
<td>102</td>
<td>0.4023</td>
<td>29.556</td>
</tr>
<tr>
<td>103</td>
<td>0.4063</td>
<td>30.499</td>
</tr>
<tr>
<td>104</td>
<td>0.4102</td>
<td>31.499</td>
</tr>
<tr>
<td>105</td>
<td>0.4141</td>
<td>32.542</td>
</tr>
<tr>
<td>106</td>
<td>0.4180</td>
<td>33.391</td>
</tr>
<tr>
<td>107</td>
<td>0.4219</td>
<td>34.208</td>
</tr>
<tr>
<td>108</td>
<td>0.4258</td>
<td>35.291</td>
</tr>
<tr>
<td>109</td>
<td>0.4297</td>
<td>36.128</td>
</tr>
<tr>
<td>110</td>
<td>0.4336</td>
<td>37.122</td>
</tr>
<tr>
<td>111</td>
<td>0.4375</td>
<td>38.067</td>
</tr>
<tr>
<td>112</td>
<td>0.4414</td>
<td>39.519</td>
</tr>
<tr>
<td>113</td>
<td>0.4453</td>
<td>41.037</td>
</tr>
<tr>
<td>114</td>
<td>0.4492</td>
<td>42.515</td>
</tr>
<tr>
<td>115</td>
<td>0.4531</td>
<td>44.013</td>
</tr>
<tr>
<td>116</td>
<td>0.4570</td>
<td>45.539</td>
</tr>
<tr>
<td>117</td>
<td>0.4609</td>
<td>47.029</td>
</tr>
<tr>
<td>118</td>
<td>0.4648</td>
<td>48.599</td>
</tr>
<tr>
<td>119</td>
<td>0.4688</td>
<td>50.060</td>
</tr>
<tr>
<td>120</td>
<td>0.4727</td>
<td>51.525</td>
</tr>
<tr>
<td>121</td>
<td>0.4766</td>
<td>52.241</td>
</tr>
<tr>
<td>122</td>
<td>0.4805</td>
<td>53.208</td>
</tr>
<tr>
<td>123</td>
<td>0.4844</td>
<td>54.417</td>
</tr>
<tr>
<td>124</td>
<td>0.4883</td>
<td>55.565</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>-------</td>
</tr>
<tr>
<td>126</td>
<td>0.4922</td>
<td>56.518</td>
</tr>
<tr>
<td>127</td>
<td>0.4961</td>
<td>57.469</td>
</tr>
<tr>
<td>128</td>
<td>0.5000</td>
<td>58.404</td>
</tr>
<tr>
<td>129</td>
<td>0.5039</td>
<td>59.518</td>
</tr>
<tr>
<td>130</td>
<td>0.5078</td>
<td>60.391</td>
</tr>
<tr>
<td>131</td>
<td>0.5117</td>
<td>61.301</td>
</tr>
<tr>
<td>132</td>
<td>0.5156</td>
<td>62.480</td>
</tr>
<tr>
<td>133</td>
<td>0.5195</td>
<td>63.440</td>
</tr>
<tr>
<td>134</td>
<td>0.5234</td>
<td>64.799</td>
</tr>
<tr>
<td>135</td>
<td>0.5273</td>
<td>66.593</td>
</tr>
<tr>
<td>136</td>
<td>0.5312</td>
<td>67.560</td>
</tr>
<tr>
<td>137</td>
<td>0.5352</td>
<td>69.577</td>
</tr>
<tr>
<td>138</td>
<td>0.5391</td>
<td>70.222</td>
</tr>
<tr>
<td>139</td>
<td>0.5430</td>
<td>70.841</td>
</tr>
<tr>
<td>140</td>
<td>0.5469</td>
<td>71.416</td>
</tr>
<tr>
<td>141</td>
<td>0.5508</td>
<td>72.059</td>
</tr>
<tr>
<td>142</td>
<td>0.5547</td>
<td>72.675</td>
</tr>
<tr>
<td>143</td>
<td>0.5586</td>
<td>73.194</td>
</tr>
<tr>
<td>144</td>
<td>0.5625</td>
<td>73.708</td>
</tr>
<tr>
<td>145</td>
<td>0.5664</td>
<td>74.167</td>
</tr>
<tr>
<td>146</td>
<td>0.5703</td>
<td>75.064</td>
</tr>
<tr>
<td>147</td>
<td>0.5742</td>
<td>75.474</td>
</tr>
<tr>
<td>148</td>
<td>0.5781</td>
<td>75.933</td>
</tr>
<tr>
<td>149</td>
<td>0.5820</td>
<td>76.566</td>
</tr>
<tr>
<td>150</td>
<td>0.5859</td>
<td>77.755</td>
</tr>
<tr>
<td>151</td>
<td>0.5898</td>
<td>77.916</td>
</tr>
<tr>
<td>152</td>
<td>0.5938</td>
<td>78.661</td>
</tr>
<tr>
<td>153</td>
<td>0.5977</td>
<td>79.407</td>
</tr>
<tr>
<td>154</td>
<td>0.6016</td>
<td>80.153</td>
</tr>
<tr>
<td>155</td>
<td>0.6055</td>
<td>80.899</td>
</tr>
<tr>
<td>156</td>
<td>0.6094</td>
<td>81.645</td>
</tr>
<tr>
<td>157</td>
<td>0.6133</td>
<td>82.390</td>
</tr>
<tr>
<td>158</td>
<td>0.6172</td>
<td>83.136</td>
</tr>
<tr>
<td>159</td>
<td>0.6211</td>
<td>83.882</td>
</tr>
<tr>
<td>160</td>
<td>0.6250</td>
<td>84.628</td>
</tr>
<tr>
<td>161</td>
<td>0.6289</td>
<td>85.373</td>
</tr>
<tr>
<td>162</td>
<td>0.6328</td>
<td>86.119</td>
</tr>
<tr>
<td>163</td>
<td>0.6367</td>
<td>86.865</td>
</tr>
<tr>
<td>164</td>
<td>0.6406</td>
<td>87.611</td>
</tr>
<tr>
<td>165</td>
<td>0.6445</td>
<td>88.357</td>
</tr>
<tr>
<td>166</td>
<td>0.6484</td>
<td>89.103</td>
</tr>
<tr>
<td>167</td>
<td>0.6523</td>
<td>89.848</td>
</tr>
<tr>
<td>168</td>
<td>0.6563</td>
<td>90.594</td>
</tr>
<tr>
<td>169</td>
<td>0.6602</td>
<td>91.340</td>
</tr>
<tr>
<td>170</td>
<td>0.6641</td>
<td>92.086</td>
</tr>
<tr>
<td>171</td>
<td>0.6680</td>
<td>92.831</td>
</tr>
<tr>
<td>172</td>
<td>0.6719</td>
<td>93.577</td>
</tr>
<tr>
<td>173</td>
<td>0.6758</td>
<td>94.323</td>
</tr>
<tr>
<td>174</td>
<td>0.6797</td>
<td>95.069</td>
</tr>
<tr>
<td>175</td>
<td>0.6836</td>
<td>95.815</td>
</tr>
<tr>
<td>176</td>
<td>0.6875</td>
<td>96.560</td>
</tr>
<tr>
<td>177</td>
<td>0.6914</td>
<td>97.306</td>
</tr>
<tr>
<td>178</td>
<td>0.6953</td>
<td>98.052</td>
</tr>
<tr>
<td>179</td>
<td>0.6992</td>
<td>98.798</td>
</tr>
<tr>
<td>180</td>
<td>0.7031</td>
<td>99.544</td>
</tr>
<tr>
<td>181</td>
<td>0.7070</td>
<td>100.289</td>
</tr>
<tr>
<td>182</td>
<td>0.7109</td>
<td>101.035</td>
</tr>
<tr>
<td>183</td>
<td>0.7148</td>
<td>101.781</td>
</tr>
<tr>
<td>184</td>
<td>0.7188</td>
<td>102.527</td>
</tr>
<tr>
<td>185</td>
<td>0.7227</td>
<td>103.273</td>
</tr>
<tr>
<td>186</td>
<td>0.7266</td>
<td>104.018</td>
</tr>
<tr>
<td>187</td>
<td>0.7305</td>
<td>104.764</td>
</tr>
<tr>
<td>188</td>
<td>0.7344</td>
<td>105.510</td>
</tr>
<tr>
<td>189</td>
<td>0.7383</td>
<td>106.256</td>
</tr>
<tr>
<td>190</td>
<td>0.7422</td>
<td>107.002</td>
</tr>
<tr>
<td>191</td>
<td>0.7461</td>
<td>107.747</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>192</td>
<td>0.7500</td>
<td>108.493</td>
</tr>
<tr>
<td>193</td>
<td>0.7539</td>
<td>109.239</td>
</tr>
<tr>
<td>194</td>
<td>0.7578</td>
<td>109.985</td>
</tr>
<tr>
<td>195</td>
<td>0.7617</td>
<td>110.730</td>
</tr>
<tr>
<td>196</td>
<td>0.7656</td>
<td>111.476</td>
</tr>
<tr>
<td>197</td>
<td>0.7695</td>
<td>112.222</td>
</tr>
<tr>
<td>198</td>
<td>0.7734</td>
<td>112.968</td>
</tr>
<tr>
<td>199</td>
<td>0.7773</td>
<td>113.714</td>
</tr>
<tr>
<td>200</td>
<td>0.7813</td>
<td>114.459</td>
</tr>
<tr>
<td>201</td>
<td>0.7852</td>
<td>115.205</td>
</tr>
<tr>
<td>202</td>
<td>0.7891</td>
<td>115.951</td>
</tr>
<tr>
<td>203</td>
<td>0.7930</td>
<td>116.697</td>
</tr>
<tr>
<td>204</td>
<td>0.7969</td>
<td>117.443</td>
</tr>
<tr>
<td>205</td>
<td>0.8008</td>
<td>118.188</td>
</tr>
<tr>
<td>206</td>
<td>0.8047</td>
<td>118.934</td>
</tr>
<tr>
<td>207</td>
<td>0.8086</td>
<td>119.680</td>
</tr>
<tr>
<td>208</td>
<td>0.8125</td>
<td>120.426</td>
</tr>
<tr>
<td>209</td>
<td>0.8164</td>
<td>121.172</td>
</tr>
<tr>
<td>210</td>
<td>0.8203</td>
<td>121.917</td>
</tr>
<tr>
<td>211</td>
<td>0.8242</td>
<td>122.663</td>
</tr>
<tr>
<td>212</td>
<td>0.8281</td>
<td>123.409</td>
</tr>
<tr>
<td>213</td>
<td>0.8320</td>
<td>124.155</td>
</tr>
<tr>
<td>214</td>
<td>0.8359</td>
<td>124.901</td>
</tr>
<tr>
<td>215</td>
<td>0.8398</td>
<td>125.646</td>
</tr>
<tr>
<td>216</td>
<td>0.8438</td>
<td>126.392</td>
</tr>
<tr>
<td>217</td>
<td>0.8477</td>
<td>127.138</td>
</tr>
<tr>
<td>218</td>
<td>0.8516</td>
<td>127.884</td>
</tr>
<tr>
<td>219</td>
<td>0.8555</td>
<td>128.630</td>
</tr>
<tr>
<td>220</td>
<td>0.8594</td>
<td>129.375</td>
</tr>
<tr>
<td>221</td>
<td>0.8633</td>
<td>130.121</td>
</tr>
<tr>
<td>222</td>
<td>0.8672</td>
<td>130.867</td>
</tr>
<tr>
<td>223</td>
<td>0.8711</td>
<td>131.613</td>
</tr>
<tr>
<td>224</td>
<td>0.8750</td>
<td>132.359</td>
</tr>
<tr>
<td>225</td>
<td>0.8789</td>
<td>133.104</td>
</tr>
<tr>
<td>226</td>
<td>0.8828</td>
<td>133.850</td>
</tr>
<tr>
<td>227</td>
<td>0.8867</td>
<td>134.596</td>
</tr>
<tr>
<td>228</td>
<td>0.8906</td>
<td>135.342</td>
</tr>
<tr>
<td>229</td>
<td>0.8945</td>
<td>136.088</td>
</tr>
<tr>
<td>230</td>
<td>0.8984</td>
<td>136.833</td>
</tr>
<tr>
<td>231</td>
<td>0.9023</td>
<td>137.579</td>
</tr>
<tr>
<td>232</td>
<td>0.9063</td>
<td>138.325</td>
</tr>
<tr>
<td>233</td>
<td>0.9102</td>
<td>139.071</td>
</tr>
<tr>
<td>234</td>
<td>0.9141</td>
<td>139.817</td>
</tr>
<tr>
<td>235</td>
<td>0.9180</td>
<td>140.562</td>
</tr>
<tr>
<td>236</td>
<td>0.9219</td>
<td>141.308</td>
</tr>
<tr>
<td>237</td>
<td>0.9258</td>
<td>142.054</td>
</tr>
<tr>
<td>238</td>
<td>0.9297</td>
<td>142.800</td>
</tr>
<tr>
<td>239</td>
<td>0.9336</td>
<td>143.546</td>
</tr>
<tr>
<td>240</td>
<td>0.9375</td>
<td>144.291</td>
</tr>
<tr>
<td>241</td>
<td>0.9414</td>
<td>145.037</td>
</tr>
</tbody>
</table>

\[ a = 0.62500E-01 \quad b = 0.48828E-03 \quad w = 0.30000E+01 \]

Nonlinear correction to rate is: \( qdotnew = 1.03000*qdotold + 3.70000E-03*qdotold^2 \)

<table>
<thead>
<tr>
<th>filename</th>
<th>buret (mm^3)</th>
<th>int. rate/%error</th>
<th>calculated/%error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0250022d.dat</td>
<td>44.30</td>
<td>44.96</td>
<td>1.50</td>
</tr>
<tr>
<td>0250045d.dat</td>
<td>44.40</td>
<td>43.72</td>
<td>-1.54</td>
</tr>
<tr>
<td>0250067d.dat</td>
<td>43.70</td>
<td>42.68</td>
<td>-2.33</td>
</tr>
<tr>
<td>0250090d.dat</td>
<td>43.60</td>
<td>42.48</td>
<td>-2.57</td>
</tr>
<tr>
<td>0250112d.dat</td>
<td>43.50</td>
<td>42.53</td>
<td>-2.23</td>
</tr>
<tr>
<td>0250135d.dat</td>
<td>43.50</td>
<td>43.99</td>
<td>1.12</td>
</tr>
<tr>
<td>0250157d.dat</td>
<td>43.50</td>
<td>43.08</td>
<td>-0.97</td>
</tr>
<tr>
<td>0250169d.dat</td>
<td>43.50</td>
<td>42.45</td>
<td>-2.41</td>
</tr>
<tr>
<td>0250180d.dat</td>
<td>43.90</td>
<td>43.84</td>
<td>-0.14</td>
</tr>
<tr>
<td>0375022d.dat</td>
<td>41.10</td>
<td>41.68</td>
<td>1.42</td>
</tr>
<tr>
<td>0375045d.dat</td>
<td>41.50</td>
<td>41.92</td>
<td>1.00</td>
</tr>
<tr>
<td>0375067d.dat</td>
<td>41.50</td>
<td>40.97</td>
<td>-1.28</td>
</tr>
<tr>
<td>0375090d.dat</td>
<td>41.40</td>
<td>41.24</td>
<td>-0.38</td>
</tr>
<tr>
<td>0375112d.dat</td>
<td>41.30</td>
<td>41.46</td>
<td>0.38</td>
</tr>
<tr>
<td>0375135d.dat</td>
<td>41.00</td>
<td>41.16</td>
<td>0.40</td>
</tr>
<tr>
<td>0375157d.dat</td>
<td>41.10</td>
<td>41.80</td>
<td>1.71</td>
</tr>
<tr>
<td>0375169d.dat</td>
<td>41.40</td>
<td>41.52</td>
<td>0.30</td>
</tr>
<tr>
<td>0375180d.dat</td>
<td>41.10</td>
<td>43.69</td>
<td>6.30</td>
</tr>
<tr>
<td>0500045d.dat</td>
<td>14.10</td>
<td>14.82</td>
<td>5.11</td>
</tr>
<tr>
<td>0500067d.dat</td>
<td>22.10</td>
<td>22.57</td>
<td>2.12</td>
</tr>
<tr>
<td>0500090d.dat</td>
<td>29.00</td>
<td>28.67</td>
<td>-1.12</td>
</tr>
<tr>
<td>0500112d.dat</td>
<td>31.80</td>
<td>31.18</td>
<td>-1.94</td>
</tr>
<tr>
<td>0500135d.dat</td>
<td>32.00</td>
<td>30.73</td>
<td>-3.97</td>
</tr>
<tr>
<td>0500157d.dat</td>
<td>31.60</td>
<td>31.09</td>
<td>-1.61</td>
</tr>
<tr>
<td>0500169d.dat</td>
<td>31.60</td>
<td>31.21</td>
<td>-1.22</td>
</tr>
<tr>
<td>0500180d.dat</td>
<td>32.80</td>
<td>31.31</td>
<td>-4.54</td>
</tr>
<tr>
<td>0625045d.dat</td>
<td>13.50</td>
<td>14.56</td>
<td>7.82</td>
</tr>
<tr>
<td>0625067d.dat</td>
<td>21.30</td>
<td>22.25</td>
<td>4.46</td>
</tr>
<tr>
<td>0625090d.dat</td>
<td>27.80</td>
<td>30.18</td>
<td>8.56</td>
</tr>
<tr>
<td>0625112d.dat</td>
<td>30.30</td>
<td>30.60</td>
<td>1.00</td>
</tr>
<tr>
<td>0625135d.dat</td>
<td>30.50</td>
<td>32.42</td>
<td>6.28</td>
</tr>
<tr>
<td>0625157d.dat</td>
<td>30.50</td>
<td>32.62</td>
<td>6.96</td>
</tr>
<tr>
<td>0625169d.dat</td>
<td>30.60</td>
<td>31.38</td>
<td>2.54</td>
</tr>
<tr>
<td>0625180d.dat</td>
<td>30.90</td>
<td>31.22</td>
<td>1.04</td>
</tr>
<tr>
<td>0750045d.dat</td>
<td>13.50</td>
<td>14.27</td>
<td>5.73</td>
</tr>
<tr>
<td>0750067d.dat</td>
<td>20.50</td>
<td>21.39</td>
<td>4.33</td>
</tr>
<tr>
<td>0750090d.dat</td>
<td>27.30</td>
<td>27.76</td>
<td>1.67</td>
</tr>
<tr>
<td>0750112d.dat</td>
<td>28.50</td>
<td>29.41</td>
<td>3.21</td>
</tr>
<tr>
<td>0750135d.dat</td>
<td>28.40</td>
<td>28.87</td>
<td>1.66</td>
</tr>
<tr>
<td>0750157d.dat</td>
<td>28.50</td>
<td>29.41</td>
<td>3.21</td>
</tr>
<tr>
<td>0750169d.dat</td>
<td>28.50</td>
<td>29.07</td>
<td>2.00</td>
</tr>
<tr>
<td>0750180d.dat</td>
<td>28.50</td>
<td>29.53</td>
<td>3.61</td>
</tr>
<tr>
<td>0875045d.dat</td>
<td>12.80</td>
<td>13.09</td>
<td>2.27</td>
</tr>
<tr>
<td>0875067d.dat</td>
<td>19.50</td>
<td>20.33</td>
<td>4.24</td>
</tr>
<tr>
<td>0875090d.dat</td>
<td>27.10</td>
<td>27.40</td>
<td>1.10</td>
</tr>
<tr>
<td>0875112d.dat</td>
<td>30.50</td>
<td>30.13</td>
<td>-1.20</td>
</tr>
<tr>
<td>0875135d.dat</td>
<td>30.30</td>
<td>29.70</td>
<td>-1.98</td>
</tr>
<tr>
<td>0875157d.dat</td>
<td>30.50</td>
<td>29.80</td>
<td>-2.28</td>
</tr>
<tr>
<td>0875169d.dat</td>
<td>30.30</td>
<td>29.92</td>
<td>-1.27</td>
</tr>
<tr>
<td>0875180d.dat</td>
<td>30.00</td>
<td>29.84</td>
<td>-0.52</td>
</tr>
<tr>
<td>1000045d.dat</td>
<td>13.30</td>
<td>13.58</td>
<td>2.09</td>
</tr>
<tr>
<td>1000067d.dat</td>
<td>17.50</td>
<td>17.13</td>
<td>-2.14</td>
</tr>
<tr>
<td>1000090d.dat</td>
<td>25.60</td>
<td>24.39</td>
<td>-4.75</td>
</tr>
<tr>
<td>1000112d.dat</td>
<td>32.30</td>
<td>32.24</td>
<td>-0.18</td>
</tr>
<tr>
<td>1000135d.dat</td>
<td>31.80</td>
<td>32.03</td>
<td>0.71</td>
</tr>
<tr>
<td>1000157d.dat</td>
<td>31.80</td>
<td>32.38</td>
<td>1.81</td>
</tr>
<tr>
<td>1000169d.dat</td>
<td>32.50</td>
<td>32.35</td>
<td>-0.46</td>
</tr>
<tr>
<td>File Name</td>
<td>Volume 1</td>
<td>Volume 2</td>
<td>Volume 3</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>31.70</td>
<td>32.65</td>
<td>3.00</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>13.40</td>
<td>13.35</td>
<td>-0.37</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>17.50</td>
<td>17.62</td>
<td>0.71</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>24.80</td>
<td>24.65</td>
<td>-0.61</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>32.00</td>
<td>32.41</td>
<td>1.28</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>31.70</td>
<td>32.47</td>
<td>2.44</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>31.80</td>
<td>32.36</td>
<td>1.76</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>32.10</td>
<td>32.54</td>
<td>1.36</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>31.60</td>
<td>33.08</td>
<td>4.70</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>13.00</td>
<td>13.68</td>
<td>5.25</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>18.20</td>
<td>17.69</td>
<td>-2.79</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>25.60</td>
<td>26.39</td>
<td>3.09</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>32.80</td>
<td>34.33</td>
<td>4.68</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>32.50</td>
<td>32.88</td>
<td>1.18</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>32.60</td>
<td>34.15</td>
<td>4.75</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>32.60</td>
<td>32.83</td>
<td>0.72</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>32.60</td>
<td>34.23</td>
<td>5.01</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>10.50</td>
<td>10.67</td>
<td>1.63</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>17.60</td>
<td>17.73</td>
<td>0.73</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>27.70</td>
<td>27.95</td>
<td>0.88</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>33.30</td>
<td>33.50</td>
<td>0.61</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>33.80</td>
<td>32.45</td>
<td>-3.99</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>33.50</td>
<td>32.19</td>
<td>-3.92</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>33.60</td>
<td>34.51</td>
<td>2.71</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>35.50</td>
<td>34.01</td>
<td>-4.21</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>35.50</td>
<td>35.15</td>
<td>-0.98</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>15.50</td>
<td>15.33</td>
<td>-1.09</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>27.00</td>
<td>27.40</td>
<td>1.48</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>36.60</td>
<td>36.19</td>
<td>-1.12</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>37.50</td>
<td>36.90</td>
<td>-1.61</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>37.50</td>
<td>37.31</td>
<td>-0.51</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>37.20</td>
<td>37.30</td>
<td>0.28</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>37.30</td>
<td>38.12</td>
<td>2.20</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>11.80</td>
<td>11.57</td>
<td>-1.91</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>23.00</td>
<td>22.89</td>
<td>-0.47</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>35.60</td>
<td>36.80</td>
<td>3.37</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>38.70</td>
<td>40.09</td>
<td>3.87</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>38.70</td>
<td>39.80</td>
<td>2.85</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>38.70</td>
<td>39.93</td>
<td>3.18</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>38.80</td>
<td>39.11</td>
<td>0.79</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>8.50</td>
<td>8.69</td>
<td>2.25</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>21.50</td>
<td>21.36</td>
<td>-0.64</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>35.10</td>
<td>35.94</td>
<td>2.40</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>38.60</td>
<td>39.61</td>
<td>2.61</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>38.50</td>
<td>39.79</td>
<td>3.35</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>38.50</td>
<td>39.52</td>
<td>2.92</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>39.30</td>
<td>39.87</td>
<td>1.45</td>
</tr>
<tr>
<td>Average (signed) dynamic model</td>
<td>40.50</td>
<td>40.60</td>
<td>0.24</td>
</tr>
<tr>
<td>Average absolute reference rate</td>
<td>40.50</td>
<td>40.47</td>
<td>-0.06</td>
</tr>
<tr>
<td>Average (signed) reference rate</td>
<td>40.60</td>
<td>39.34</td>
<td>-3.09</td>
</tr>
<tr>
<td>Average absolute dynamic model</td>
<td>40.40</td>
<td>40.92</td>
<td>1.28</td>
</tr>
</tbody>
</table>

Average absolute reference rate \% volume error = 2.297%  
Average (signed) reference rate \% volume error = 0.891%  
Average absolute dynamic model \% volume error = 3.895%  
Average (signed) dynamic model \% volume error = -1.564%
7. MC68000 Assembly Language Listing of Firmware for Sensor Processor

*  
* SENSOR PROCESSOR OPERATIONAL CODE  
*  
* FILE NAME: SENPROC15  
*  
* REVISION: 4.0 1/02/90  
*  
* AUTHOR: C. MACCARLEY & S. D'ANGELO  
*  
* ***************************************************************************  
*  
* ADDRESS EQUATES  
*  
* PI/T Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>EQU</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGR</td>
<td>EQU</td>
<td>9001H</td>
</tr>
<tr>
<td>PSRR</td>
<td>EQU</td>
<td>9003H</td>
</tr>
<tr>
<td>PADDR</td>
<td>EQU</td>
<td>9005H</td>
</tr>
<tr>
<td>PBDDR</td>
<td>EQU</td>
<td>9007H</td>
</tr>
<tr>
<td>PCDDR</td>
<td>EQU</td>
<td>9009H</td>
</tr>
<tr>
<td>PIVR</td>
<td>EQU</td>
<td>900BH</td>
</tr>
<tr>
<td>PACR</td>
<td>EQU</td>
<td>900DH</td>
</tr>
<tr>
<td>PBCR</td>
<td>EQU</td>
<td>900FH</td>
</tr>
<tr>
<td>PADR</td>
<td>EQU</td>
<td>9011H</td>
</tr>
<tr>
<td>PBDR</td>
<td>EQU</td>
<td>9013H</td>
</tr>
<tr>
<td>PAAR</td>
<td>EQU</td>
<td>9015H</td>
</tr>
<tr>
<td>PBAR</td>
<td>EQU</td>
<td>9017H</td>
</tr>
<tr>
<td>PCDR</td>
<td>EQU</td>
<td>9019H</td>
</tr>
<tr>
<td>PSR</td>
<td>EQU</td>
<td>901BH</td>
</tr>
<tr>
<td>TCR</td>
<td>EQU</td>
<td>9021H</td>
</tr>
<tr>
<td>TIVR</td>
<td>EQU</td>
<td>9023H</td>
</tr>
<tr>
<td>CPRH</td>
<td>EQU</td>
<td>9027H</td>
</tr>
<tr>
<td>CPRM</td>
<td>EQU</td>
<td>9029H</td>
</tr>
<tr>
<td>CPRL</td>
<td>EQU</td>
<td>902BH</td>
</tr>
<tr>
<td>TSR</td>
<td>EQU</td>
<td>9035H</td>
</tr>
</tbody>
</table>

* ADDRESSES OF INTERRUPT SERVICE ROUTINES

<table>
<thead>
<tr>
<th>Port</th>
<th>EQU</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVEC</td>
<td>EQU</td>
<td>64</td>
</tr>
<tr>
<td>TVEC</td>
<td>EQU</td>
<td>65</td>
</tr>
</tbody>
</table>

* PERIPHERAL ADDRESSES

<table>
<thead>
<tr>
<th>Port</th>
<th>EQU</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>EQU</td>
<td>008000H</td>
</tr>
</tbody>
</table>

* VARIABLES (RAM BASE IS AT 004000H)

<table>
<thead>
<tr>
<th>Port</th>
<th>EQU</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESTVAR</td>
<td>EQU</td>
<td>004000H</td>
</tr>
<tr>
<td>FTEMP</td>
<td>EQU</td>
<td>004004H</td>
</tr>
<tr>
<td>FMASS</td>
<td>EQU</td>
<td>004006H</td>
</tr>
<tr>
<td>PORTCOUNT</td>
<td>EQU</td>
<td>004008H</td>
</tr>
<tr>
<td>RPMCOUNT</td>
<td>EQU</td>
<td>00400AH</td>
</tr>
<tr>
<td>RPM</td>
<td>EQU</td>
<td>00400CH</td>
</tr>
<tr>
<td>NMAPIN</td>
<td>EQU</td>
<td>00400EH</td>
</tr>
<tr>
<td>LOWRPM</td>
<td>EQU</td>
<td>004020H</td>
</tr>
<tr>
<td>LASTSCALE</td>
<td>EQU</td>
<td>004022H</td>
</tr>
<tr>
<td>LASTTFTEMP</td>
<td>EQU</td>
<td>004024H</td>
</tr>
<tr>
<td>LASTNLIFT</td>
<td>EQU</td>
<td>004026H</td>
</tr>
<tr>
<td>LASTFMAS</td>
<td>EQU</td>
<td>004028H</td>
</tr>
<tr>
<td>DATABASE</td>
<td>EQU</td>
<td>004030H</td>
</tr>
<tr>
<td>DATATOP</td>
<td>EQU</td>
<td>004FD0H</td>
</tr>
</tbody>
</table>

* CONSTANTS
* WINDOW EQU 00065B9AH ; (WINDOW WIDTH ENG DEG) / [(TICS*8) - Revised Window
WINDOW EQU 000697D2H ;
TICS EQU 02H ; LOAD VALUE FOR CPRH
TEMPFACT EQU 0001H ; TEMP SCALE FACTOR
INPUTSCALE EQU 0FF0H ; INPUT SCALE FACTOR (4 BITS) (12 BIT)
STX EQU 0FCH ; START TRANSMISSION CODE
ETX EQU 0FEH ; END TRANSMISSION CODE

* INITIALIZATIONS

ORG 000400H ; START OF INITIALIZATION CODE
MOVE.B #0FH, PGCR ; SET UP P/I/T IN MODE 0, SUBMODE 1X
MOVE.B #18H, PSRR ; INTERRUPT CONFIG SET
MOVE.B #0FFH, PADDR ; PORT A IS OUTPUT ONLY FOR DISPLAY
MOVE.B #0FFH, PBDDR ; PORT B IS COMMUNICATIONS, 2-WAY
MOVE.B #02H, PADDR ; MAKE PC0 INPUT AND PC1 OUTPUT
MOVE.B #0FH, PADDR ; INTERRUPT VECTOR FOR COMMUNICATIONS
MOVE.B #0FH, PBDDR ; PORT A CONTROL TO MODE 0 SUBMODE
MOVE.B #00H, PACR ; PORT B CONTROL TO MODE 0 SUBMODE
***
MOVE.B #00H, PAT ; PERIODIC INTERRUPT GENERATOR MODE
MOVE.B #00H, PBAT ; TIMER INTERRUPT VECTOR
MOVE.B #0FFH, TSRC ; CLEAR 2DS BIT OF TIMER
MOVE.B #00H, CPRH ; INITIALIZE COUNTER PRELOAD
MOVE.B #00H, CPRL ; REGISTER (9USEC PER TIC) TO
MOVE.B #TICS, CPRH ; 000002H => 24 USEC SAMPLE PERIOD
MOVE.W #2000H, SR ; ENABLE INTERRUPTS
MOVE.W #0000H, ADC ; INITIAL CHANNEL SETTING OF MUX
ANDI.B #0FDH, PCDR ; SET PC1=0 => NO DATA AVAILABLE YET
MOVE.W #4000H, LASTSCALE ; INIT SCALE FILTER
MOVE.W #0FFH, LASTTEMP ; INIT FTEMP FILTER
MOVE.W #0000H, LASTNLIFT ; INIT NLIFT FILTER
MOVE.W #0000H, LASTFMASS ; INIT FMASS FILTER
JMP START ; START PROGRAM

* MAIN ROUTINE

ORG 000500H ; START MAIN ROUTINE

* MEMORY TEST DIAGNOSES

START MOVE.L #004000H, A4 ; BASE OF RAM
MOVE.W #0000H, D1 ; CLEAR ERROR COUNTER

SOFINE MOVE.W A4, D4 ; WRITE TO MEMORY LOCATION
MOVE.W D4, [A4] ; GET THAT BYTE BACK FROM MEMORY
MOVE.W [A4]+, D6 ; CHECK IT OUT
CMP.W D4, D6
BEQ NOPROB ; OK....
* START OF MAIN LOOP

GO

MOVE.W  #0000H,ADC
        #0000H,RPMCOUNT
        ADC,[A1]+  ;LOAD INDEX REG WITH BASE OF DATA F:
        #DATABASE,A1
        #0,PCDR
        BTST.B  #0,PCDR  ;TEST PC0 TO SEE IF TIME TO TAKE DA'NT
        BEQ    EDGEUP  ;TEST AGAIN TO MAKE DAMN SURE THIS IS AN UP *E!
        BTST.B  #0,PCDR
        BNE    EDGEUP
        BTST.B  #0,PCDR
        BNE    EDGEUP
        MOVE.B  #0A1H,TCR  ;START PRIODIC INTRP GENERATOR
        #0FDH,PCDR  ;CLEAR PCI => DATA NOT AVAILABLE YET
        MOVE.L  ADC,[A1]+  ;LOAD INDEX REG WITH BASE OF DATA F:
        MOVE.W  #DATABASE,A1
        #0,PCDR
        BTST.B  #0,PCDR  ;TEST PC0 FOR END OF DATA COL SIGNA-
        BEQ    TAKEWAIT  ;LOOP IF PC0 STILL 1, ELSE CONTINUE
        BEQ    TAKEWAIT
        BRA    DONEDATA

TAKEWAIT

CMPA.L  #DATATOP,A1
        BHI    OUTARAM  ;IF MEMORY FULL?
        MOVE.B  #11H,PADR
        BTST.B  #0,PCDR
        BNE    TAKEWAIT
        BRA    DONEDATA

OUTARAM

MOVE.B  #01H,LOWRPM  ;SET LOW RPM FLAG
        #0A0H,TCR  ;DISABLE THE PERIODIC GENERATOR

* NEEDLE LIFT DATA ACQUIRED, NOW TAKE FUEL TEMPERATURE, STORE IN D6

ZOOM

MOVE.B  #33H,PADR  ;CHANGE MUX CHANNEL TO 1 (FUEL TEMP
        #0011H,ADC  ;KILL SOME TIME
        NOP  ;READ FUEL TEMP INTO D6
        MOVE.W  ADC,D6
        MOVE.W  LASTFTEMP,D5  ;DIGITAL FILTER FTEMP
        MULU  #00F0H,D5
        LSR.L  #8,D5
        ADD.W  D6,D5
        MOVE.W  D5,LASTFTEMP
        LSR.W  #4,D5

* PROCESS TEMP VOLTAGE THROUGH FUEL TEMP MAP

MOVE.W  D5,D6  ;MAKE A COPY OF THE VALUE
        LSR.W  #8,D5
        LSL.W  #1,D5
        LEA.L  FTEMPMAP,A3  ;AND RIGHT SHIFT TO
        ADDA.W  D5,A3  ;MULT BY 2 FOR WORD INDEXINGER
        MOVE.W  [A3]+,D5  ;LOAD THE TABLE ADDRESS
        MOVE.W  D5,D2  ;AND INDEX INTO IT
        MOVE.W  [A3],D4  ;READ THE CURRENT AND
        MOVEM.W  D5,D2  ;KEEP A COPY OF BASE
        SUB.W  D4,D5  ;FOLLOWING TABLE ENTRIES
        MOVE.W  D5,D7  ;CALCULATE THE DIFFERENCE
        ANDI.W  #00FFH,D7  ;A POS VALUE
        MULU.W  #D5,D7  ;NOW ISOLATE THE NUMBER OF STEPS
        ADDA.W  D5,A3  ;IN THE INTERPOLATION
        MULU.W  #D5,D7  ;MULTIPLY DIFFERENCE TIMES THE LOW
LSR.L #8,D7 ; AND RIGHT SHIFT TO NORMALIZE
SUB.W D7,D2 ; ADD RESULT TO THE BASE VALUE
MOVE.W D2,FTEMP ; STORE IN FUEL TEMP VAR

* RPM CALCULATION - RPMCOUNT IS NUMBER OF TIMER INTERRUPT INTERVALS

***
MOVE.W RPMCOUNT,D1 ; FOR DEBUGGING:
***
JSR HEXCON
***
LSR.W #8,D1 ; DISPLAY RPM X 100
***
MOVE.B D1,PAADR ; DISPLAY RPMCOUNT LOW BYTE

CMPW.W #0000H,RPMCOUNT ; CALCULATE ENGINE RPM FROM RPM COUN'
BNE CONTINUE
MOVE.W #0000EH,RPM ; AVOID ZERODIVIDE - MAX RPM=FFFF
JMP PROCESS
CONTINUE
MOVE.L #WINDOW,D0
DIVU RPMCOUNT,D0 ; RPM = (16 * 44,643) / COUNT FOR A
MOVE.W D0,RPM ; STORE IN RPM VARIABLE

MOVE.W RPM,D1 ; GET READY TO DISPLAY RPM
JSR HEXCON ; IN DECIMAL
LSR.W #8,D1 ; DISPLAY RPM X 100
MOVE.B D1,PAADR ; ON BOARD DISPLAY

* BEGIN PROCESSING DATA

PROCESS MOVE.L #00000000H,D3 ; CLEAR INTEGRATION SUM

* DETERMINE DC OFFSET OF NEEDLE LIFT SIGNAL

LEA DATABASE,A2 ; ADDRESS OF FIRST DATA POINT
MOVE.W [A2]+,D6

AVGLOOP
ADD.W [A2]+,D6
CMPA #DATABASE+20H,A2
BNE AVGLOOP
LSR.W #4,D6 ; DIVIDE BY 16

* DC OFFSET IN D6

***
MOVE.W #0000H,D6 ; *** TEST *** ZERO DC OFFSET
***
MOVE.W D6,FTEMP ; *** TEST ONLY ***

* READ NLIFT DATA FROM DATA FIELD

LEA DATABASE,A1 ; BASE OF DATA FIELD
MOVE.W RPMCOUNT,D1 ; # OF PTS GOES IN D1
ADDI.W #0001H,D1 ; PLUS 1 FOR END POINT
LSL.W #1,D1 ; TIMES 2 FOR WORD ACCESS
ADDA.W D1,A1 ; ADDR OF LAST PT IN FIELD

MOREPOINTS MOVE.W-[A1],D4 ; GET DATA POINT FROM DATA FIELD

* 1ST ORDER IIR FILTER FOR NLIFT SIGNAL

***
JMP NOPFILTER ; *** TEST: TURN OFF FILTER

MOVE.W LASTNLIFT,D5 ; SINGLE POLE, a = 0.75
MULU #0C0000H,D5
MULU #040000H,D4
ADD.L D5,D4
LSL.L #8,D4
LSR.L #8,D4
MOVE.W D4,LASTNLIFT

NOPFILTER SUB.W D6,D4 ; SUBTRACT DC OFFSET
BCC NOTMREG ; CLAMP RESULT AT ZERO
NOTNEG
ANDI.L #0000H,D4 ; AND CLEAR UPPER WORD
***
MOVE.W D4,RP ; *** TEST ONLY *** MAP INDEX VALUE

* MULTIPLY BY INPUT SCALE FACTOR (NOM = 1)

MULU.W #INPUTSCALE,D4 ; SCALE FACTOR
LSR.L #8,D4 ; SHIFT TO NORMALIZE (1.000)
LSR.L #4,D4 ;

* PROCESS THROUGH F1(x) MAP

MOVE.W #0055H,D5 ; DIVIDE THE VALUE BY
DIVU D5,D4 ; 55 HEX = 85 DEC
MOVE.L D4,D5 ; THIS ISOLATES 48 QUANTA
ANDI.L #0000FFFFH,D5 ; D5 HOLDS QUOTIENT (0 - 48 DEC)

***
MOVE.W D5,ITEMP ; *** TEST ONLY ***
LSR.L #8,D4
LSR.L #8,D4 ; D4 HOLDS REMAINDER (0 - 84 DEC)
CMPI.W #0030H,D5 ; SEE IF D5 IS GREATER THAN 48 DEC M;
BLS OKAYVALUE ; SKIP IF LESS THAN MAX VALUE
MOVE.W #0030H,D5 ; LOAD WITH 48 DEC OTHERWISE

OKAYVALUE
LEA MAPTABLE,A2 ; LOAD LOOKUP TABLE POINTER INTO A2
ADDA.L D5,A2 ; AND INDEX INTO IT
ADDA.L D5,A2 ; BY WORDS (Add twice)
MOVE.W [A2],D5 ; READ IN VALUE
MOVE.W [A2],D7 ; AND THE ONE FOLLOWING IT
SUB.W D5,D7 ; PLACE THE DIFFERENCE IN D7
LSL.W #8,D4 ; LEFT SHIFT THE REMAINDER INTO
MOVE.W #0055H,D2 ; TOP HALF OF WORD AND DIVIDE
DIVU D2,D4 ; BY 85 DEC
ANDI.L #0000FFFFH,D4 ; CLEAR OUT REMAINDER
MULU D4,D7 ; MULTIPLY THIS BY THE DIFFERENCE
LSR.L #8,D7 ; AND RIGHT SHIFT TO NORMALIZE
ADD.W D5,D7 ; INDEXED VALUE + INTERPOLATED VALUE

* OUTPUT OF F1(x) MAP (RATE) IN D7

ADDA.L D7,D3 ; ADD RESULT TO THE SUM
CMPI.A.L #DATABASE,A1 ; DONE WITH ALL THE POINTS?
BNE MOREPOINTS ; LOOP IF MORE POINTS REMAIN
LSR.L #8,D3 ; SCALE, INTEGRATION SUM IN D3
LSR.L #1,D3 ; (S/B #1) SHOULD FIT IN 16 BITS NOW

* RAW SUM IN D3, TEMP CORRECT D3 -> D7 (FMASH)

JSR TEMPCOR ; TEMP CORRECTION: FVOLV TO FMASS: D

* FMASS IN D7, AS A LONG WORD. NOW SCALE FOR OUTPUT IN BCD.

MULU.W #0625H,D7 ; MUL BY 15.25878*10**-6
LSR.L #7,D7
MULU.W #64H,D7 ; MUL BY 100 AND
LSR.L #8,D7 ; DIVIDE BY 256 TO GET RESULT IN 100

* APPLY FINAL CORRECTION FACTOR FROM MUX CH 2

MOVE.W #0002H,ADC ; SET THE ADC MUX TO THE TRIMPOT
NOP
MOVE.W ADC,D4 ; READ IN THE TRIMPOT VALUE
MOVE.W LASTSCALE,D5 ;DIGITAL FILTER FOR SCALE FACTOR
MULU   #00F0H,D5
LSR.L  #8,D5
ADD.W  D4,D5
MOVE.W D5,LASTSCALE
LSR.W  #4,D5

***
MOVE.W D5,FTEMP ;TEST ONLY *************
MULU.W D5,D7 ;MULTIPLY FMASS BY IT
LSR.L  #8,D7 ;XX.XXXXXXXXXX IS FORMAT
LSR.L  #2,D7 ;SHIFT DOWN 10 BITS
ENTIRE SUM SHOULD BE IN LOW WORD

* FMASS IS IN D7, FILTER FMASS WITH PREV FMASS RESULTS TO TAKE OUT NOISE

***
JMP NOSMOOTH ;*** TEST: NO SMOOTHING FILTER
MOVE.W LASTFMASS,D5
MULU   #0000H,D5
MULU   #4000H,D7
ADD.L  D5,D7
LSR.L  #0,D7
LSR.L  #8,D7
MOVE.W D7,LASTFMASS

NOSMOOTH
MOVE.W D7,FMASS ;IF NO FILTER USED

* DONE PROCESSING. DATA AVAILABLE.

DONEPROC ORI.B #02H,PCDR ;SET PC1=1 SIGNALLING DATA AVAILABLE

* COMMUNICATIONS

***
MOVE.W #1000H,FTEMP ;TEST ONLY
***
MOVE.W #1000H,FMASS ;TEST ONLY
***
MOVE.W #1000H,RPM ;TEST ONLY

MOVE.B #STX,D0
JSR OUTSUB ;SEND STX START CODE TO HOST

MOVE.W FTEMP,D1
JSR HEXCON ;BCD
MOVE.L D1,D0 ;BEFORE SHIPPING
MOVE.W #8901H,D0
JSR OUTSUB ;SEND LOWER BYTE OF FTEMP
ASR.L  #8,D0
JSR OUTSUB ;SHIFT UPPER BYTE DOWN

MOVE.W FMASS,D1
JSR HEXCON
MOVE.L D1,D0
JSR OUTSUB
ASR.L  #8,D0
JSR OUTSUB

MOVE.W RPM,D1
JSR HEXCON
MOVE.W D1,D0
JSR OUTSUB
ASR.W  #8,D0
JSR OUTSUB

MOVE.B #ETX,D0 ;SEND ETX END CODE
*** SUBROUTINES AND INTERRUPT SERVICE ROUTINES

* TRANSMIT A BYTE TO THE HOST

OUTSUB
MOVE.B
#00H, PORTCOUNT
; CLEAR THE PORT ACCESS COUNT REG

TRYAGAIN
BST
#6, PSR
; TEST H3 LEVEL. IF H3=0 THEN
BEQ
CLEARPORT
; WAIT A WHILE AND RETRY...

DOITANYWAY
MOVE.B
D0, PBDR
; THEN CLEAR THE PORT AND RETURN.
MOVE BYTE FROM D0 TO PORT B.

ANDI.B
#08H, PBCR
; PULSE H4 BY SETTING BIT 3 OF PBCR.

RTS
; THEN RESET IMMEDIATE. THIS EDGE IS LA

CLEARPORT
ADDI.B
#01H, PORTCOUNT
; INCR THE PORT ACCESS COUNT

CMPI.B
#0FFH, PORTCOUNT
; IS LESS THAN THE 255'TH TRY?

BCS
TRYAGAIN
; IF SO, CHK THE PORT AGAIN

JMP
DOITANYWAY
; ELSE SEND THE BYTE ANYWAY

OUTAHERE
RTS
; RETURN, READY TO SEND ANOTHER BYTE

* TIMER INTERRUPT SERVICE ROUTINE. SAMPLES CH0 (NEEDLE LIFT) PERIODICALLY

TSERV
MOVE.W
ADC, [A1]+
; READ A/D NEEDLE LIFT

ADDI.W
#01H, RPMCOUNT
; ACCUMULATE RPM COUNT

MOV.W
RPMCOUNT, D4

MOV.B
D4, PADDR

MOV.B
#22H, PADDR

MOV.B
#0FH, TSR
; RESET ZDS BIT TO CLEAR INTR REQ

* TEMPERATURE CORRECTION SUBROUTINE. GETS INTEGRATED FUEL VOLUME FROM D3
AND TEMPERATURE FROM D6 TO CONVERT VOLUME FUEL DELIVERY TO MASS,
WHICH IS RETURNED IN D7.

TEMPCOR
MOVE.L
D3, D7
; STUB FOR NOW. JUST USE FUEL VOL A:

LSL.W
#4, D6
; LEFT SHIFT TEMP PRIOR TO DIVISION

DIVU.W
#TEMPFACT, D6
; SCALE THE FUEL TEMP

LSR.W
#4, D6
; RIGHT SHIFT BACK

MULT.W
D6, D7
; TEMP CORRECT

RTS

* HEX TO DECIMAL CONVERSION (D1 IN, D1 OUT)

HEXCON
CMPI.W
#270FH, D1
; CHECK IF TOO BIG

BHI
TOOHIGH

MOVE.W
#0000, D3
; CLEAR RESULT

MOV.W
D1, D2
; D2 IS WORKING REG

AND1.W
#000FH, D2
; MASK OUT LOW NIBBLE

MOV.W
D2, D3
; PUT NIBBLE IN RESULT FOR NOW

SUB.W
#000AH, D2
; IF NIBBLE < 0A HEX

BCS
N1B2
; THEN SKIP TO NEXT NIBBLE

MOV.W
D2, D3
; ELSE PUT POSITIVE DIFFERENCE IN D3

ADDI.W
#0010H, D3
; THEN ADD TO RESULT
NIB2
MOVE.W D1,D2 ;PUT ORG NUMBER IN WORKING REG AGAIN
ANDI.W #00H,D2 ;MASK OUT NIBBLE 2
ROR.W #4,D2 ;SHIFT INTO LOW NIBBLE

LOOP1
CMPI.W #0000H,D2
BEQ NIB3
MOVE.B #16H,D4
MOVE.B #04H,CCR
ABCD D4,D3
MOVE.B #00H,D4
ROR.L #8,D3
ABCD D4,D3
ROL.L #8,D3
SUBQ.W #01H,D2
BRA LOOP1

NIB3
MOVE.W D1,D2
ANDI.W #0F00H,D2
ROR.W #8,D2

LOOP2
CMPI.W #0000H,D2
BEQ NIB4
MOVE.B #56H,D4
MOVE.B #04H,CCR
ABCD D4,D3
MOVE.B #02H,D4
ROR.L #8,D3
ABCD D4,D3
ROL.L #8,D3
SUBQ.W #01H,D2
BRA LOOP2

NIB4
MOVE.W D1,D2
ANDI.W #0F000H,D2
ROR.W #8,D2
ROR.W #4,D2

LOOP3
CMPI.W #0000H,D2
BEQ DONECVT
MOVE.B #96H,D4
MOVE.B #04H,CCR
ABCD D4,D3
MOVE.B #40H,D4
ROR.L #8,D3
ABCD D4,D3
ROL.L #8,D3
SUBQ.W #01H,D2
BRA LOOP3

DONECVT
MOVE.W D3,D1
RTS

TOOHIGH
MOVE.W #0AAA0H,D1
RTS

******************************************************************************************

* PORT INTERRUPT SERVICE ROUTINE (NOT USED; THIS IS AN EMERGENCY TRAP)

PSERV
MOVE.B #24H,PAJR ;WRITE TO DISPLAY AND STOP IF PORT
RTE

******************************************************************************************

* BUSS ERROR SERVICE ROUTINE

BUSSERR
MOVE.B #99H,PAJR ;JUST RETURN
* f1(x) TABLE.  QQZZ: TABLE ENTRIES CALCULATED FROM:
* QQ = HEX( XX ), ZZ = HEX(.YY * 4096 ) WHERE XX.YY IS DECIMAL RATE IN UL/MS
* EACH MAP INCR IS 0.0156 MM NEEDLE LIFT ( .75 / 48 ) INDEXED 0 TO 4080.
* .75 MM (.75 VOLTS) AT NLIFT SENSOR ANALOG SCALED TO (4080/4096)*10.0 VOLTS AT A/D

MAPTABLE

| DC.W | 0000H, 02F0H, 0419H, 0493H | ; NLIFT (mm) | 0.0000 |
| DC.W | 04E3H, 0527H, 0562H, 05AC7H | ; 0.625 |
| DC.W | 05FAH, 0655H, 06A8H, 070F7H | ; 1.250 |
| DC.W | 077CH, 07E4H, 0869H, 08E7H | ; 1.875 |
| DC.W | 0965H, 09E4H, 0A69H, 0B10H | ; 2.500 |
| DC.W | 0BD4H, 0CF1H, 0E9AH, 1155H | ; 3.125 |
| DC.W | 14FBH, 19CFH, 1E80H, 2235H | ; 3.750 |
| DC.W | 2611H, 2C35H, 32F6H, 366BH | ; 4.375 |
| DC.W | 3A67H, 3E7BH, 438FH, 476AH | ; 5.000 |
| DC.W | 49B5H, 1BEFH, 4EA9H, 515AH | ; 5.625 |
| DC.W | 54A1H, 579CH, 5A98H, 5D94H | ; 6.250 |
| DC.W | 608FH, 638BH, 6687H, 6983H | ; 6.875 |
| DC.W | 6C7EH, 6F7AH | ; 7.500 |

MAPTABLE

| DC.W | 0000H, 02F0H, 03FFH, 0431H | ; 0.000 |
| DC.W | 0458H, 046CH, 0485H, 0498H | ; 0.625 |
| DC.W | 04F7H, 0546H, 058FH, 05DEH | ; 1.250 |
| DC.W | 063AH, 069FH, 0713H, 078CH | ; 1.875 |
| DC.W | 0872H, 0882H, 08E2H, 0921H | ; 2.500 |
| DC.W | 09A7H, 0A57H, 0B38H, 0DA7H | ; 3.125 |
| DC.W | 157CH, 1634H, 1A94H, 1E55H | ; 3.750 |
| DC.W | 1F81H, 25FBH, 2BF7H, 2FC4H | ; 4.375 |
| DC.W | 32E8H, 3E38H, 3E98H, 4090H | ; 5.000 |
| DC.W | 41EFH, 4333H, 4556H, 47F0H | ; 5.625 |
| DC.W | 4A9BH, 4D38H, 4FD6H, 5275H | ; 6.250 |
| DC.W | 5514H, 5780H, 5A4FH, 5CEDH | ; 6.875 |
| DC.W | 5F8AH, 6236H | ; 7.500 |

* FUEL TEMPERATURE MAP TABLE

FTEMPMAP

| DC.W | 05DCH, 0408H, 0384H, 0302H |
| DC.W | 026CH, 0208H, 01C2H, 018BH |
| DC.W | 0159H, 012CH, 00FAH, 00CAH |
| DC.W | 0091H, 0066H, 004BH, 000FH |
| DC.W | 0000H |