1991

Adaptive torque control of a diesel engine for transient test cycles

Taner Tuken
Iowa State University

Order Number 9207256

Adaptive torque control of a diesel engine for transient test cycles

Tuken, Taner, Ph.D.

Iowa State University, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Adaptive torque control of a diesel engine for transient test cycles

by

Taner Tuken

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1991

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The purpose of this research is to develop an adaptive torque controller to implement the Environmental Protection Agency (EPA) Federal Transient Test Cycle. The EPA specifies that an engine must be operated over this test cycle while its exhaust emissions are being measured and compared to regulations [49]. The test cycle consists of a precise series of engine speed and torque values at which the engine must be operated according to a time schedule. The cycle contains periods of rapid acceleration and deceleration, idling, and steady cruising to simulate the actual conditions that a diesel engine will encounter in an over-the-road truck. During the 20-minute test, under both cold and hot start operation, the engine and dynamometer pair should be able to follow reference torque and speed trajectories within specified tolerances. The ability to control speed and torque independently is also advantageous for carrying out work on engine mapping where both speed and torque need to be held constant while varying other parameters such as air-to-fuel ratio.

Controllers with fixed parameters (nonadaptive), designed using classical control system design methods, have been shown to be adequate for transient test cycle implementation [1, 2, 3, 4]. Although parameters of the describing equations of the torque control system may vary over the operating range, non-adaptive controllers can cope well with considerable variations in the system dynamics. However, a con-

stant parameter controller designed for a particular system may not give satisfactory performance when alterations are made in system components such as replacing an all-speed governor with a min-max governor, removing the turbocharger, or changing the amount of filtering in the torque and speed measurement lines. Moreover, in test laboratories where different engines with various different components need to be run over the transient test cycle, an adaptive control algorithm can be a requirement. Classical, constant parameter controllers require off-line system identification and controller design for each specific configuration. This can be very time consuming. An adaptive control algorithm would automatically adjust controller parameters based on input-output information. It will be shown in Chapter 6 that when a filter in the torque measurement line was removed in a specific engine and dynamometer setup, the constant parameter controller did not give a satisfactory response – actually led to an unstable closed-loop operation – while the self-tuning adaptive control produced valid transient test cycles under the EPA specifications.

The major objectives of this research are:

1. To investigate the feasibility of adaptive torque control for a diesel engine for implementation of the EPA Federal Transient Test Cycle.

2. To compare the performance of conventional and adaptive torque controllers.

3. To point out the implementation problems of using digital adaptive torque control for a diesel engine.

In this research, three different adaptive control strategies were tested and compared to each other. In the first approach, the complete system identification, parameter estimation, and controller design were carried out on-line during a brief test

period prior to the transient test cycle. A one-shot system identification and parameter estimation was combined with a one-shot controller design algorithm. During the test cycle, the controller parameters were kept constant. Although there was no parameter estimation and adaptation during the 20 minute transient test cycle, this was still considered an adaptive algorithm since no prior knowledge about the system was assumed, and the determination of the system model, time delay, model parameters, and controller and feedforward compensation designs were carried out on-line.

The second approach is similar to the first one, but includes a gain-scheduling control. For different operating ranges, the selected model parameters and corresponding controller parameters were found on-line during a short period prior to the test cycle employing multiple one-shot parameter identification and controller design. During the test cycle, the controller parameters become a function of the operating conditions. Again, as in the first approach, no prior knowledge about the system is needed.

The third method is different from the first two and employs continuous parameter estimation and controller parameter update during the transient test cycle. This is the traditional understanding of an adaptive controller. Prior knowledge about the system, however, was required for this approach. The time-delay, system model, and estimates of the model parameters had to be determined off-line prior to the test cycle. The purpose of this algorithm was to track time-varying parameters of the system due to operating conditions, and hence obtain better tracking of the reference speed and torque trajectories.

This dissertation is organized as follows: Chapter 2 gives a literature review on

engine speed/torque control design for transient test cycles and on adaptive control techniques. Complete system modeling is discussed in Chapter 3. The assumptions and possible simplifications on the overall model are reviewed. The specifications of the system under consideration are also given. Chapter 4 gives the details of the Poisson Moment Functional (PMF) parameter identification algorithm. The PMF method is a continuous-time parameter identification technique, and it has some features that make it superior to discrete-time parameter identification techniques. A detailed comparison of the PMF method to discrete-time methods is given. Application of the PMF method to the adaptive control of the throttle-torque problem is also discussed in this chapter. Chapter 5 explains the controller design in each of the three adaptive control strategies. Due to the non-negligible time delay present in the system, a Smith predictor schema is employed. Inclusion of the Smith predictor in an adaptive control algorithm can provide additional flexibility which enlarges the application areas to time delay systems. Implementation issues, controller software development, design of a feedforward compensator to overcome the load disturbances, and development of a new pole-zero assignment algorithm to ensure stable closed-loop operation are also discussed in Chapter 5. Chapter 6 gives the performance of the controllers in actual transient test cycles. Comparisons are made between the different adaptive control strategies and to fixed parameter controllers. Application of the designed algorithm to other systems is also discussed. Chapter 7 summarizes the conclusions of this study.

2. LITERATURE REVIEW

2.1 Engine Modeling

There are a large number of studies that discuss the dynamic modeling and control of automotive engines [5-15]. Morris et al. [5] reported a mathematical model for the throttle position - engine torque system. An adaptive identification algorithm was employed to find the parameters of the model for their specific spark-ignition engine. However, they did not deal with the problem of engine control, which Powell [6] addressed with his dynamic engine model for control applications. A fuel to air ratio, spark advance, and exhaust gas recirculation control algorithm was developed to allow multivariable engine control based on his dynamic model. The effect of time delay on the overall system response was also included in his study. Tsai and Goyal [7] developed a dynamic turbocharged diesel engine model with simplified linearized equations to describe the governor, the combustion process, and the engine. They also developed a fourth-order linearized model with a time delay to represent the transfer function of the throttle-speed system. Blaney [8] explained the design steps of a digital speed control system. He pointed out that the time varying coefficients of the system were a major factor limiting the use of conventional digital speed control algorithms. He concluded that adaptive control could provide a very practical algorithm for cruise control applications in vehicles. Sobolak [9] summarized

the studies on the Ford vehicle speed control system. He pointed out the high nonlinearity of the system, and concluded that classical feedback control theory has very little value for engine speed control and recommended adaptive algorithms for modeling and control of the system. Kamei et al. [10] and Takahashi et al. [11] explained linear quadratic (LQ) controller design for idle speed control. They also mentioned the varying dynamics of the engine at other than idle speed operation.

Although there are a large number of studies on engine speed control (or cruise control), little work has been reported on the simultaneous torque and speed control of engines for transient test cycles. Due to increasingly stringent emissions legislation and demands for improved fuel economy and driveability, there is a growing need for accurate and repeatable transient control of engines on testbeds. Wellstead and Zanker [16] applied a self-tuning adaptive control concept (the detuned minimum variance algorithm) to a four stroke turbocharged diesel engine coupled to a dynamometer for engine speed control. They considered loads generated by a dynamometer as disturbances in the speed control loop, however, they did not make any attempt to identify the interactive disturbance effects on the system or to compensate for them by a feedforward controller for better performance. They also did not address simultaneous torque and speed control as required for transient test cycles. Koustas and Watson [2] implemented a digital testbed control system consisting of combined PID and time optimal throttle and torque controllers. They obtained good results on simultaneous control of torque and speed. However, since their system used an eddy-current dynamometer, which can not reproduce the motored parts of the cycle, they could not implement the entire transient test, rather only 96 seconds of the cycle was implemented. Noble et al. [3] described the application of generalized

predictive control (GPC) to transient control of speed and torque. They used off-line system identification algorithms coupled with the GPC design algorithm to tune the system for best performance for each installation. They also could not implement the entire transient test cycle due to the use af an eddy-current dynamometer. Tuken et al. [1] described the design steps of a digital torque controller capable of following the EPA transient test cycle. This controller was developed for a turbocharged diesel engine and a direct current dynamometer. A pole assignment algorithm coupled with a Smith Predictor to deal with a time delay and a feedforward compensator was used for better disturbance rejection. Test results showed that the controller consistently satisfied the EPA regulations. Brown and Thompson [4] described a method of controlling both speed and torque independently by designing a pre-compensator that effectively decouples the engine and dynamometer system.

All the above studies showed that for particular engine and dynamometer pairs, off-line system identification methods coupled with conventional control laws can give satisfactory transient performance. Classical non-adaptive closed-loop control systems can cope well with parameter variations, so variation of the open-loop system parameters is not a justification for the use of adaptive control. However, when flexibility for testing different engines is important, then application of adaptive control to simultaneous speed and torque control of engines could provide an important advantage. This is also important in research laboratories where the effect of different engine components and parameters (e.g. fuel to air ratio, governor type, turbocharger) are tested for emissions analyses.

2.2 Adaptive Control

The three basic operations common to most adaptive control systems are the following:

1. it must identify the process,
2. it must compare present system performance with the desired or optimum performance and make a decision to adapt the system so as to tend toward optimum performance, and
3. it must initiate a proper modification so as to drive the system toward the optimum.

Depending on how these functions are brought about, different types of adaptive controllers can be proposed.

There are a large number of references on system identification and adaptive control. Discrete-time system and parameter identification techniques are best explained by Norton [17], Sodestrom and Stoica [18], Ljunj [19], Ljung and Sodestrom [20], Eykhoff [21], Hsia [22], and Goodwin and Payne [23]. Recursive parameter estimation schemes best suited for adaptive control such as recursive least squares, extended least squares, and maximum likelihood identification algorithms and their variations to accommodate stochastic systems are covered in great detail in the above references. Some other excellent references on discrete-time parameter identification are also listed in the bibliography [24-29].

Saha and Rao [30] give a detailed treatment of the Poisson Moment Functional (PMF) approach for identification of continuous dynamic systems. They list immunity to noise, ability to estimate time delay, and applicability to non-linear and

time-varying systems as some of the desirable features of the PMF algorithm. Unbehauen and Rao [31] deal with recent trends in continuous model identification. The PMF method, the orthogonal function method, and the method of linear filters are explained. Recursive estimation of continuous-time model parameters from discrete measurements for adaptive control purposes is also treated. They state that ease of implementation, good noise rejection capability, and computational simplicity are the main advantages of the PMF method over these other continuous-time parameter identification algorithms.

The majority of the reported adaptive control applications employ discrete-time parameter estimation algorithms. This is primarily due to the fact that digital computers are involved in the implementation of the algorithms. However it will be shown here that estimating continuous-time system parameters from discrete sets of measurements and then converting the continuous-time model to its discrete-time counterpart can have significant advantages over estimating discrete-time model parameters directly. It will be shown that this is reasonable since the plants are described by continuous-time equations which are close to the real world, and digital computers operate in a discrete fashion.

An excellent review of the literature on adaptive control up to 1987 is given by Chalam [32]. Although there are vast number of papers on adaptive control, most of the findings are summarized in various fine books [32-46]. There are two principal approaches to adaptive control, mainly model reference adaptive control (MRAC), and the self-tuning regulator (STR). In MRAC the aim is to make the output of an unknown plant approach asymptotically the output of a given reference model. The main advantages of this method are ease of implementation and a high speed of

adaptation since there is no need for identification of the plant dynamic performance. However, it is difficult to analyze the convergence and stability properties of MRAC systems. Moreover MRAC systems are mostly applied to deterministic systems. An excellent discussion of model reference adaptive control as well as parameter and state estimation using model reference adaptive methods is provided in Landau [33].

In contrast to MRAC systems, the estimation of unknown parameters is separated from the design of the controller in STR systems. Applicability to stochastic and time delay systems, and ease of stability and convergence analyses are the main advantages of the STR method when compared to the MRAC method. Harris and Billings [34] have reviewed many aspects of self-tuning adaptive control explored in various papers. Additional self-tuning design methods and implementation issues are discussed by Roffel et al. [35].

In this research the self-tuning approach to adaptive control was selected. Although self-tuning and model reference adaptive control are based on different design approaches, the equivalence between them has been demonstrated in several studies [41, 42]. Among the different controller design approaches in self tuning control, the linear-quadratic-gaussian self-tuning regulator, the minimum variance controller, the pole-zero placement technique, and self-tuning control to give the proportional+integral+derivative (PID) modes are some of the most widely used design methods. Of these, the pole-zero placement technique was selected in this study. This method involves the placement of closed-loop poles and zeros at the prescribed locations which define a required transient response. Due to the nature of the 20-minute EPA transient test cycle, the engine speed and torque should follow a required transient response. The pole-zero placement algorithm is well suited for this applica-

tion. Robustness, applicability to non-minimum phase plants in which time delay or model order is not known, and ability to regulate systems with varying time delays are some of the desirable features of the pole-zero placement technique. Astrom and Wittenmark [38, 43] give a design procedure for the pole-zero placement method. The minimum variance STR, in which the objective is the minimization of the variance of the process output, is not suitable for engine torque control applications since a minimum phase plant assumption is made in the development of this method. The linear-quadratic-gaussian STR demands the most computation, otherwise it carries the same desired features of the pole-zero placement algorithm.

There are no published studies available on adaptive torque control of either gasoline or diesel engines for transient test cycles. Also, there are no studies available on application of continuous-time parameter identification algorithms for use in adaptive control systems. This research hopefully will fill these gaps, and provide further application areas for adaptive control and continuous-time model and parameter estimators.

3. SYSTEM MODELING

The system to be modeled is an engine and dynamometer pair in the Internal Combustion Engine Laboratory of Iowa State University. This engine and dynamometer pair was also the source of all the experimental data discussed in this dissertation. Figure 3.1 shows the experimental apparatus. A John Deere 4276T turbocharged diesel engine was coupled to a General Electric TLC-2544 DC electric dynamometer. Speed control was provided by a General Electric Siltron Dynamometer Controller. A Zenith Corporation Z-386 computer with an Analog Devices RTI-820 data acquisition module was used to supply reference speed and torque trajectories and to accomplish the closed-loop torque control. A PP-125 step-motor driven linear actuator produced by Jasta, Inc. was used to set the governor speed lever position. Engine, dynamometer, and actuator specifications are given in Table 3.1.

Figure 3.2 shows the block diagram representation of the throttle-torque open-loop system. Through RS-232 serial line communication, the computer supplies input commands to the step-motor driven electromechanical actuator which sets the throttle position. Actuator dynamics are also included in the model of the system. A governor-equipped injection pump unit sets the fuel rate that goes to the cylinders; it does so by observing speed control lever position and the speed of the engine. The output of the combustion process is the engine torque. Subtraction of load torque

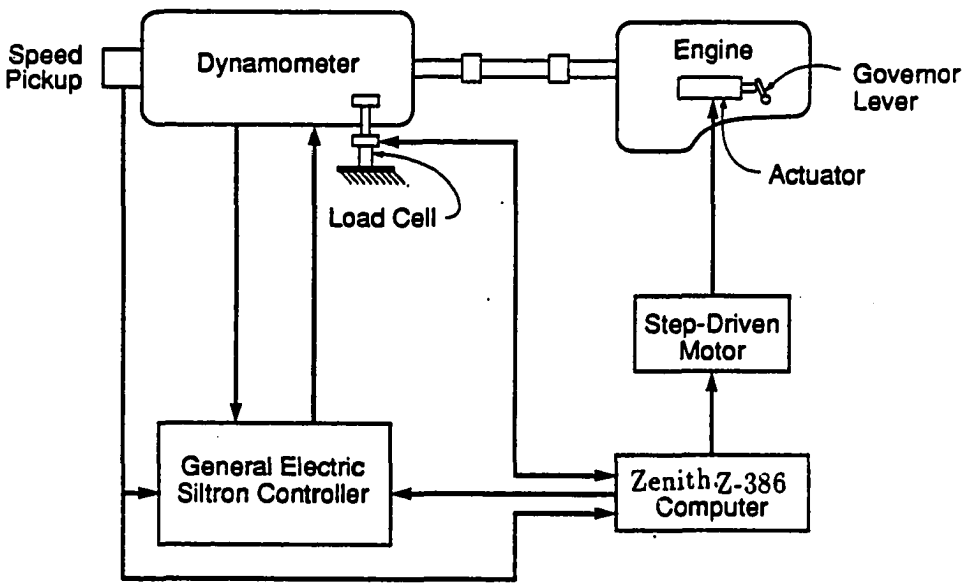


Figure 3.1: Experimental Apparatus for the Torque-Speed Control Loops

Table 3.1: Engine, dynamometer, step-motor and actuator specifications

Engine Specifications

Bore	4.19 *in.* *(106.5 mm)*
Stroke	5.0 *in.* *(127.0 mm)*
Number of cylinders	4
Total displacement	276.0 in^3 *(4525.2 cm^3)*
Compression ratio	16.8:1
Maximum torque	230 $ft-lbs$ @ 1500 *rpm* *(310.5 $N-m$ @ 1500 rpm)*
Maximum power	78 *BHP* *(58.2 kW)*

Dynamometer Specifications

Excitation Voltage	250.0 *V*
Maximum current	410.0 *A*
Maximum absorbed power	150 *BHP* *(111.8 kW)*
Maximum delivered power	120 *BHP* *(89.5 kW)*

Step-Motor and Actuator Specifications

Maximum thrust	125 *lb* *(57.0 kg)*
Maximum linear velocity	15 *in./sec.* *(38.1 cm/sec)*
Stroke	3 *in.* *(7.62 cm)*

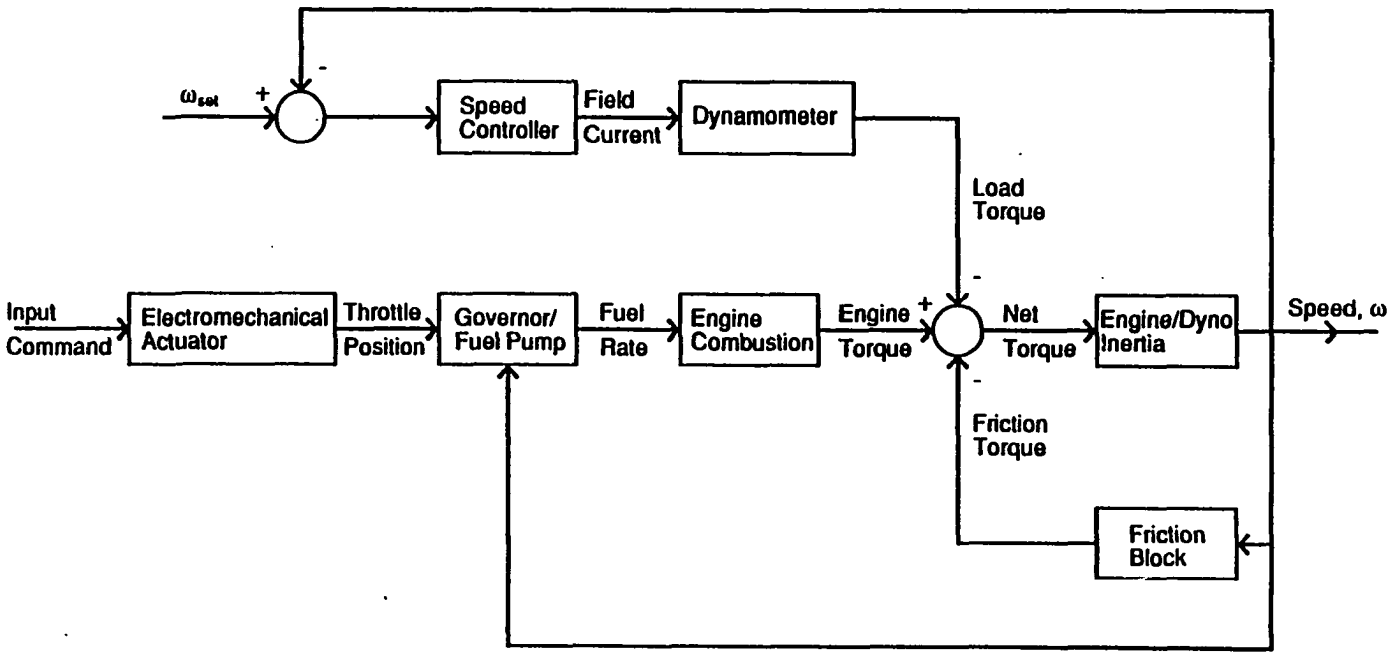


Figure 3.2: Block diagram representation of the system

from engine torque gives the net torque of the system.

To obtain a mathematical model for the throttle-torque system, the model structures of individual blocks in Figure 3.2 were first determined. By using the input and output data obtained from open-loop step response tests, possible simplifications in the overall system model were investigated. This gave rise to a lower order model for the system that was more suitable for identification and adaptive controller design.

3.1 Electromechanical Actuator Dynamics

A permanent magnet step-motor-driven linear actuator was used to set the throttle position.

The step-motor has its own microprocessor to convert input position commands (serial line commands through a RS-232 interface) to voltages for the four phases of the step-motor. The microprocessor and step-motor-driven unit is characterized by a time delay transfer function.

Kuo [47] detailed the dynamic modeling of a permanent magnet step-motor. He developed a third-order non-linear dynamic model to describe the input-output relation of the step-motor. However, after linearization, a linear third-order dynamic model can adequately approximate the step-motor performance.

3.2 Governor Model

An injection-pump-mounted mechanical, hydraulic governor was employed, as is typical of diesel engines for industrial and agricultural use. A schematic representation of the governor unit is shown in Figure 3.3.

In the mechanical part of the governor, the centrifugal force on weights rotating

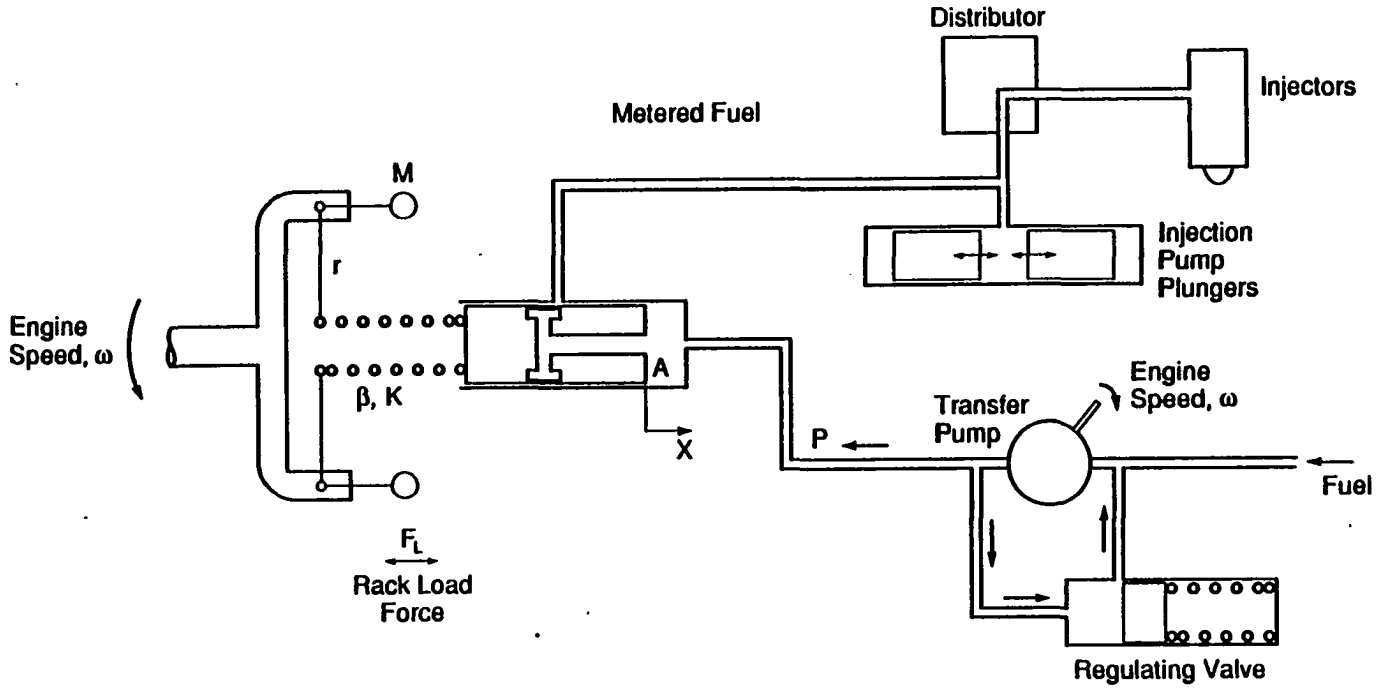


Figure 3.3: Schematic representation of mechanical, hydraulic governor

about a horizontal axis driven by the engine is balanced against a spring. Any change in the speed of rotation shifts the balance position of the mechanism and the movement is transmitted by a linkage to the metering valve of the hydraulic pump unit.

The hydraulic part of the governor includes a transfer pump, metering valve, injection pump plungers, and the distributor. Fuel is fed into the injection pump by the transfer pump. The delivery of the transfer pump is proportional to engine speed, and this is converted into a pressure proportional to engine speed by a regulating valve. This pressure is then applied to a metering valve plunger. Metered fuel is proportional to the output rod position of the mechanical governor. This metered fuel is then fed equally to the engine cylinders by the injection pump plunger and a distributor unit.

The mathematical model for the governor is shown below.

(a) Mechanical part

Summing the forces along the axis of rotation gives the following equation.

$$m\omega^2 r = m_e \frac{d^2x}{dt^2} + \beta \frac{dx}{dt} + Kx + F_l + PA \tag{3.1}$$

$$F_l = K_l x_a \tag{3.2}$$

$$P = K_t \omega \tag{3.3}$$

(b) Hydraulic part

The flow rate of fuel that goes to the engine is directly proportional to the metering valve position as shown in the following equation.

$$q = K_m x \tag{3.4}$$

where the variables in equations 3.1 – 3.4 are defined as follows.

m = Mass of flyweights

ω = Engine speed

r = Radius of the flyweight from the axis of rotation

m_e = Total effective rotating mass

β = Viscous friction coefficient of the moving parts

K = Spring stiffness

F_l = Load force due to throttle rack

P = Output pressure of transfer pump

A = Metering-valve piston area

x_a = Throttle position

x = Metering valve position

q = Fuel rate

K_m = Metering valve constant

K_t = Transfer pump constant

K_l = Constant due to geometry of throttle rack

Thus, a second order non-linear dynamic model represents the relationship between fuel rack position, engine speed and the fuel rate that goes to the engine.

3.3 Engine Combustion System Model

A significant feature of a diesel engine, from a control point of view, is the discontinuous manner in which power is produced by the sequential firing of cylinders.

It means there is a time delay between the action of the governor in demanding a change in fueling rate and the response of the engine to that change. The effective firing delay has been found, in other studies, to be the actual time between consecutive pistons arriving at the injection point plus approximately a quarter of a revolution of the crankshaft [48]. The effective firing delay is therefore approximately:

$$T_f = \frac{60h}{2e\omega} + \frac{60}{4\omega} \tag{3.5}$$

where

h = Number of strokes per cycle

ω = Speed in rev/min

e = Number of cylinders

T_f = Firing delay in seconds

The second term in the firing delay equation comes from the fact that the engine crankshaft does not rotate at a uniform speed but rather experiences a cyclic variation in torque, which gives rise to a cyclic variation in speed.

The transfer function for the engine combustion system can be written in Laplace variables as:

$$\frac{T_E(s)}{q(s)} = K_e \cdot e^{-T_f s} \tag{3.6}$$

where K_e is a constant. In this model, the engine comprises a combustion system which produces a torque, T_E, as a function of the fuel flow rate, q.

3.4 Engine/Dynamometer Inertia

Engine torque, T_E, acts against the load torque, T_L, and any difference between T_E and T_L accelerates, or retards, the combined inertia $J = J_E + J_D$, where J_E is

the engine inertia including the flywheel and J_D is the dynamometer inertia. The sum of the torques will equal the engine acceleration.

$$T_E - T_L - T_F = J\frac{d\omega}{dt} \tag{3.7}$$

where T_F is the friction torque and can be approximated as a function of speed, e.g. $T_F = B\omega$, and B is the friction torque/speed slope. Then Eq. 3.7 becomes:

$$T_E(t) - T_L(t) = J\frac{d\omega(t)}{dt} + B\omega(t) \tag{3.8}$$

3.5 Model for the GE Speed Controller and Dynamometer

Since no information was available on the GE Siltron Dynamometer Controller, a black-box approach is taken for load torque analysis. So load torque can be written as a function of reference speed and actual speed as seen in Figure 3.2.

$$T_L = f(\omega_{set}, \omega) \tag{3.9}$$

3.6 Model for the Overall System

In the EPA transient test cycle, the reference engine torque trajectory is given and the EPA specifies that this torque is to be measured at the engine's output shaft. This engine torque, however, is not available for direct measurement in the dynamometer facility used for this study, rather load torque is measured with a load cell located on the dynamometer. Two different approaches can be used to overcome this problem:

1. Engine torque can be estimated using an engine and dynamometer inertia model. Since both the engine and dynamometer inertia are known, engine torque can be obtained by measuring load torque and engine speed, and then applying Eq. 3.7. Estimated engine torque can then be used for the identification of the throttle-engine torque model's parameters and as feedback for the engine torque control loop. However, this technique for estimation of engine torque requires differentiation of the speed signal. Due to possible noise in the speed measurement, it is not suitable for differentiation. Even assuming a perfect engine torque estimation, the fifth-order model for the throttle-speed-engine torque system resulting from the analysis given earlier in this chapter may not be suitable for adaptive control purposes. Parameter estimation and adaptive controller design methods call for the simplest possible model for the system. Since most physical systems can be described by third order dynamics, using higher order models can create problems in parameter identification and adaptive control algorithms as will be explained later. A system structure identification, which can be performed either on-line or off-line, through experimental data, can be used in place of the fifth-order model to obtain a lower order model.

2. Using reference speed and reference engine torque trajectories, a reference load torque trajectory can be created by using the known inertia of the system. This converts the engine torque control problem to a load torque control problem. Since the reference speed trajectory contains no noise, differentiation of the speed signals does not pose a problem. Due to the black-box approach followed in modeling the GE speed controller, we no longer know an exact model for

the throttle-speed-load torque system. However, as mentioned earlier, a system structure identification can be carried out through experimentation to find a suitable model for the system for controller design purposes.

The second approach listed above was used in this research. Avoiding differentiation of speed signals and allowing applicability to other systems were the two primary reasons behind the selection of this approach. Since actual engine speed, ω, depends on reference speed, ω_{set}, and throttle valve position, x_a, the load torque given by Eq. 3.9 can be approximated as:

$$T_L \simeq f(\omega_{set}, x_a) \tag{3.10}$$

On-line or off-line process identification techniques can now be employed to find a model that fits the measured input-output data.

4. PARAMETER ESTIMATION

On-line determination of process parameters is a key element in adaptive control. This chapter summarizes and compares some parameter estimation methods described in the literature. Parameter estimation algorithms can be arranged to give either discrete-time or continuous-time model parameters. Estimating continuous-time model parameters is computationally more complicated. Section 4.1 reveals the basic differences of these two cases in the estimation algorithm. The three most commonly used continuous-time model estimation methods are summarized in Section 4.1. Of these methods, the Poisson Moment Functional (PMF) is explained in more detail in Section 4.2. The PMF method was used in this study due to its superior "immunity to noise" feature. Section 4.3 compares the performance of the PMF continuous-time identification technique to discrete-time parameter identification methods. It is shown in Section 4.3 that in the case of noisy measurements the PMF method has superior convergence properties. Section 4.4 deals with the application of the PMF method to the throttle-torque system studied in this research. In Section 4.4, it is shown how the PMF algorithm is developed for each of the adaptive control strategies applied in this research.

4.1 Parameter Estimation in Continuous-Time and Discrete-Time Systems

The process of parameter estimation consists of two stages. These are:

1. The primary stage in which the system of parameter estimation equations is derived from the form of the dynamic model corresponding to the system to be identified.

2. The secondary stage in which the parameters of the model are estimated within the framework of an estimation method.

The secondary stage is independent of the original model form and depends only on the system of parameter estimation equations arising from it. The secondary stage can be applied with little modification for either continuous-time or discrete-time models. Thus, all methods developed for discrete models can be directly implemented in the case of continuous-time models. The major difference between continuous-time and discrete-time parameter identification lies in the primary stage.

The primary stage is trivial in the case of discrete systems since the system of equations can be directly written down from the discrete model of the dynamic system corresponding to the discrete points of available data. For example, consider a plant whose transfer function can be written in the z-domain as:

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^m + b_1 z^{m-1} + ... + b_m}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + ... + a_n} \tag{4.1}$$

where $m \leq n$ and n is the system order. The primary stage for this discrete-time system consists of converting the above transfer function description to a difference

equation representation as shown:

$$\begin{aligned} y(k) &= -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_n y(k-n) \\ &\quad + b_0 u(k+m-n) + b_1 u(k+m-n-1) + \ldots + b_m u(k-n) \end{aligned} \tag{4.2}$$

The primary stage in the continuous-time model needs special consideration. The problem arises from the derivative measurement. Algorithms involving direct generation of the time derivatives of process signals either physically or by computation are good only in deterministic situations or if the noise in the measurements is very low. For simplicity, consider the following single input - singleoutput (SISO) lumped linear continuous system:

$$\sum_{i=0}^{n} a_i \frac{d^{n-i} y(t)}{dt^{n-i}} = \sum_{j=0}^{m} b_j \frac{d^{m-j} u(t)}{dt^{m-j}} \tag{4.3}$$

The above equation can be written as:

$$\sum_{i=0}^{n} S_{y_i} a_i = \sum_{j=0}^{m} S_{u_j} b_j \tag{4.4}$$

where S_{y_i} and S_{u_j} are members of the output and input signal families, and a_i and b_j are the constant unknown parameters of the output and input sections of the process, respectively. Clearly, the primary stage in the continuous-time parameter estimation problem requires the measurement or computation of the S_{y_i} and S_{u_j} terms. In actual practice it is neither possible to directly observe some elements of S_{y_i} and S_{u_j}, those involving derivative operations in particular, nor is it desirable to generate them directly from $y(t)$ and $u(t)$. However, if we perform a suitable linear dynamic operation on both sides of Eq. 4.4, transforming S_{y_i} and S_{u_j} into well-behaved and measurable or computable terms m_{y_i} and m_{u_j}, then we can avoid

the undesirable direct derivative operation on $y(t)$ and $u(t)$. Then Eq. 4.4 would be transformed to:

$$\sum_{i=0}^{n} m_{y_i} a_i = \sum_{j=0}^{m} m_{u_j} b_j \tag{4.5}$$

where $m_{y_i} = LD[S_{y_i}]$ and $m_{u_j} = LD[S_{u_j}]$. LD denotes a linear dynamic operation. This LD operation thus forms the basis of the primary stage. There are several techniques for choosing the LD operation in the primary stage of continuous-time model identification (CMI). These methods can be categorized into three groups:

1. *Modulating functions*: This method involves a linear operation wherein the terms S_{y_i} and S_{u_j} are first multiplied by well-behaved and suitably chosen known functions, and then integrated over the period of available data. The operation LD is determined by off-line computation, so this method is not well suited for real-time on-line applications.

2. *Spectral characterization of signals*: When the spectral coefficients of the process signals are used in the differential equations describing the continuous-time system under consideration, the calculus of these systems is approximated by computationally attractive algebraic expressions. The operation LD can be computed on-line in real-time applications.

3. *Poisson Moment Functional technique*: The off-line computations of the modulating functions method can be avoided by choosing modulating functions that stem from the impulse response functions of linear time invariant dynamic filters. The required values of the definite integrals can then be measured. The convolution integrals are measured as the outputs of the various stages of a set of filter chains. The Poisson Moment Functional (PMF) method employs

such a filter chain. The treatment of signals may then be viewed in terms of distributions or generalized functions.

Although both the second and third methods listed above are suitable for real-time on-line applications, the PMF method has advantages such as computational simplicity, very good noise rejection, and ease of implementation. Therefore, the PMF method is the most commonly used continuous-time model identification method among real-time parameter identification techniques, and it was chosen for the work reported in this dissertation.

4.2 Poisson Moment Functional Approach

The main task in an identification problem is to process the input-output data over a given interval of time. The process signals can be characterized by two different approaches. One method is to treat the process signals as ordinary functions. The other method treats the process signals as distributions or generalized functions. The latter characterization is superior due to its unlimited differentiability. The PMF method stems from this generalized function concept. A signal function $f(t)$, $t \in (0, t_0)$, is treated as a distribution or a generalized function, and expanded about a time instant t_0 in the following exponentially weighted series:

$$f(t) = \sum_{k=0}^{\infty} M_k[f(t)] e^{-\lambda(t-t_0)} \delta^k(t - t_0) \tag{4.6}$$

where $\delta^k(t-t_0)$ is the generalized time derivative of an impulse distribution occurring at $t = t_0$. $M_k[f(t)]$ in Eq. 4.6 can be obtained using the following expression:

$$M_k[f(t)] = f_k^0 = \int_0^{t_0} f(t) P_k(t_0 - t) dt \tag{4.7}$$

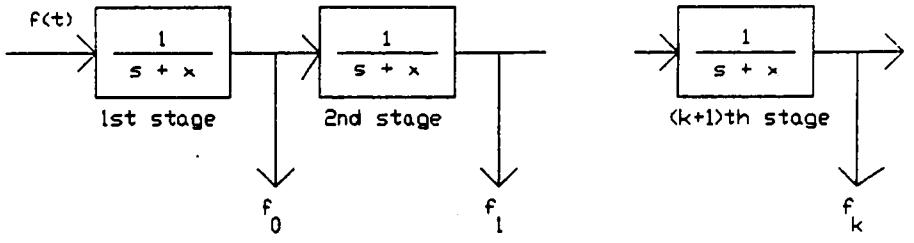


Figure 4.1: A Poisson filter chain

with

$$P_k(t_0) = P_k^0 = \frac{t_0^k}{k!} e^{-\lambda t_0} \tag{4.8}$$

and λ is a positive real number.

P_k^0 is called the k-th order Poisson pulse function at $t = t_0$ and f_k^0 is termed the k-th Poisson Moment Functional (PMF) of f(t) about $t = t_0$. f_k^0 can be viewed as the output due to an input f(t), at $t = t_0$, of the (k+1) stage of a cascaded filter with identical stages, called the Poisson Filter Chain, each element of which has a transfer function $\frac{1}{s+\lambda}$ as indicated in Figure 4.1.

When $\lambda = 1$, these filters are called state variable filters, and $\lambda = 0$ corresponds to a chain of pure integrators. $M_k[\,]$ corresponds to the LD operation of the PMF approach. For zero initial conditions, some frequently used signals and their corresponding $M_k[\,]$ representations are summarized in Table 4.1.

Table 4.1: $M_k[\ ]$ representation of most commonly used signals

$$
\begin{aligned}
M_k[f(t)] &= f_k^0 \\
M_k\left[\frac{df(t)}{dt}\right] &= f_{k-1}^0 - \lambda f_k^0 \\
M_k\left[\frac{d^2f(t)}{dt^2}\right] &= f_{k-2}^0 - 2\lambda f_{k-1}^0 + \lambda^2 f_k^0 \\
M_k\left[\frac{d^3f(t)}{dt^3}\right] &= f_{k-3}^0 - 3\lambda f_{k-2}^0 + 3\lambda^2 f_{k-1}^0 - \lambda^3 f_k^0 \\
M_k\left[\frac{d^4f(t)}{dt^4}\right] &= f_{k-4}^0 - 4\lambda f_{k-3}^0 + 6\lambda^2 f_{k-2}^0 - 4\lambda^3 f_{k-1}^0 + \lambda^4 f_k^0
\end{aligned}
$$

4.3 Advantages of PMF Identification: A Comparison with Discrete Time Identification Algorithms

Estimating the continuous-time model parameters and then transforming to the discrete-time model have some important advantages over estimating discrete-time model parameters directly. The primary advantage comes from "the immunity to noise" property of the PMF method. The following sections will investigate how measurement noise is handled in both continuous-time and discrete-time parameter identification methods using the PMF approach.

4.3.1 Continuous-time parameter identification (PMF approach)

The Poisson Moment Functional (PMF) algorithm is immune to zero mean additive noise. In other words, the PMF approach directly gives the coefficients of the Laplace transformed ("s" domain) transfer function from noisy input and output

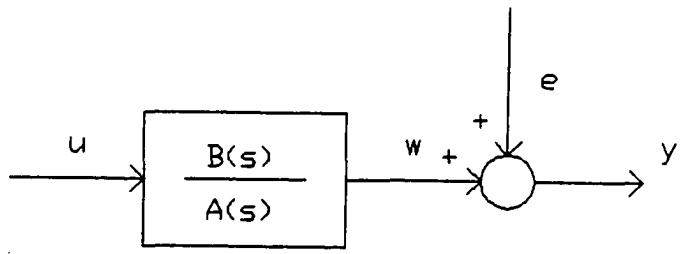


Figure 4.2: A SISO system with measurement noise

measurements as depicted in Figure 4.2. The output, $y(t)$, contains measurement noise, $e(t)$, which can be white (uncorrelated) or colored (correlated) noise. Therefore, the PMF method can be applied to both deterministic and stochastic systems with little or no modifications. Noise to signal ratios more than 25% can be handled very effectively with the PMF method. It will be shown later that even low frequency colored noise will not deteriorate the estimation algorithm's performance significantly. However, this is not true for discrete-time parameter estimation algorithms. Discrete-time estimation algorithms are very sensitive to noise structure, and they may require a detailed noise model to obtain acceptable parameter identification performance.

To show the noise immunity of the PMF method consider the system in Figure 4.2. The output $y(t)$ can be written as:

$$y(t) = w(t) + e(t) \tag{4.9}$$

where $w(t)$ is the output of the deterministic part of the plant. If we take PMF's of this equation, we find:

$$M_k[y] = M_k[w] + M_k[e] \tag{4.10}$$

$$y_k^0 = w_k^0 + e_k^0 \tag{4.11}$$

The amount of filtering in the Poisson Filter Chain depends on the value of λ. Therefore, for sufficiently large t_0 and small λ, e_k^0 approaches zero and consequently its influence on the PMF's of $y(t)$ becomes negligible. So we can write that:

$$y_k^0 \simeq w_k^0 \tag{4.12}$$

By virtue of the low pass filtering nature of the Poisson Filter Chains, zero mean additive noise is removed by successive integration operations. Thus, the PMF method is practically immune to zero mean additive noise.

4.3.2 Discrete-time parameter estimation

In discrete-time parameter estimation algorithms, the measurement noise needs extra consideration. The recursive least squares algorithm gives unbiased estimates if the system description is:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + e(k) \tag{4.13}$$

where q^{-1} is the backward shift operator and e is the white noise (or uncorrelated equation errors) [17]. This identification schema will correspond to the system in Figure 4.3-a.

As seen from Figure 4.3-a there is a strict requirement on the measurement noise in this case. Measurement noise should be the output of a filter with transfer function

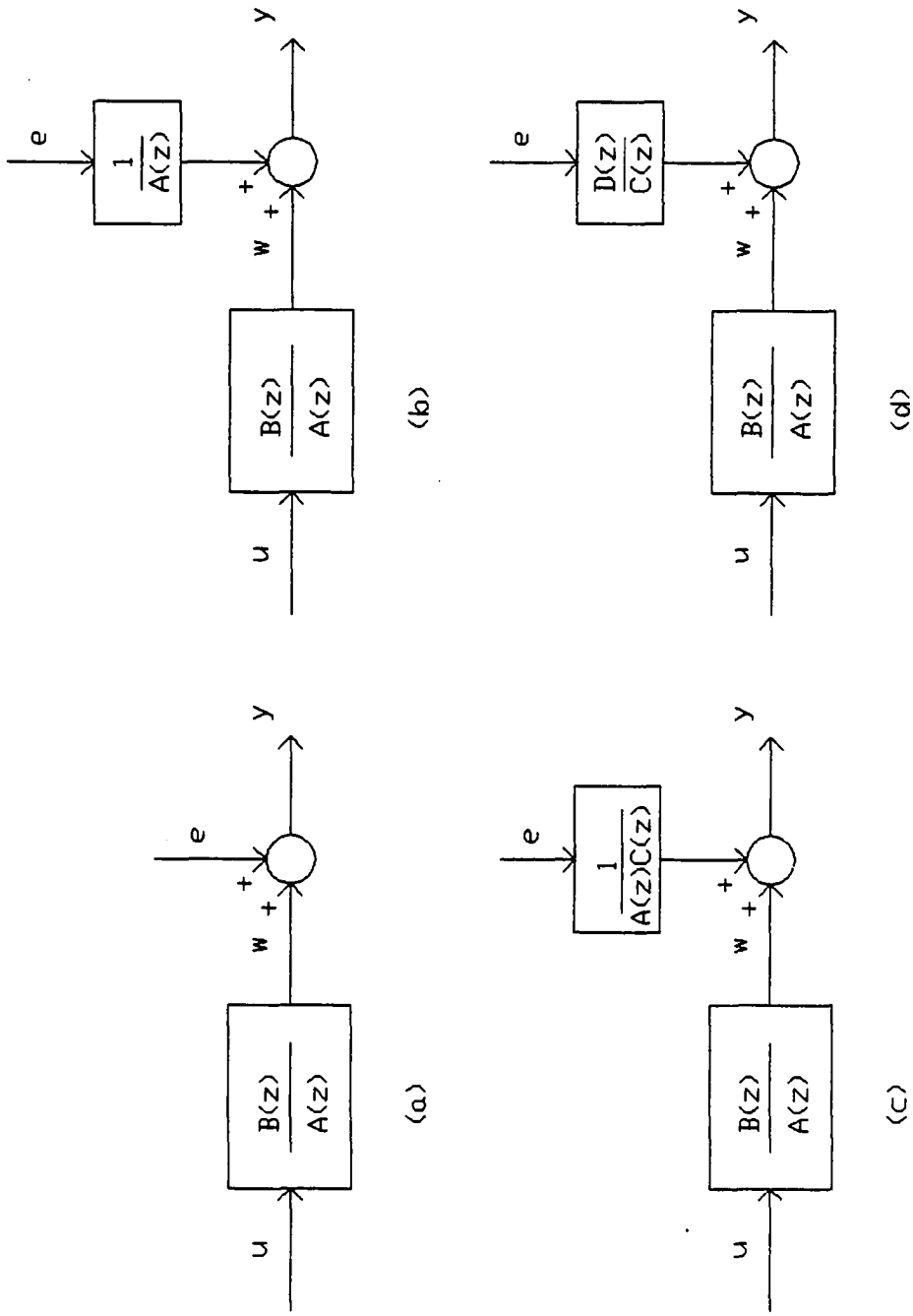


Figure 4.3: Systems with output noise of special types

$1/A(z)$ and with an input of white noise. This is a severe constraint on measurement noise and usually is not realistic in real-time operations. If the noise structure is different from that indicated in Figure 4.3-a, the least-squares algorithm will give biased estimates of system parameters. There are at least three different methods to model noise and construct a corresponding least-squares estimation algorithm in discrete-time systems to overcome the bias problem. These can be summarized as follows.

1. One commonly encountered case in system identification is that the output additive disturbance is a white noise as depicted in Figure 4.3-b. In this case, the system equation becomes:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + A(q^{-1})e(k) \tag{4.14}$$

   Generalized least squares or extended least squares can estimate the parameters of the A and B polynomials without bias. But convergence of the algorithm becomes extremely slow if the noise-to-signal ratio is greater than 5%. One main reason is that the noise terms in the algorithm should be estimated, and any error in this estimation will deteriorate the algorithm's performance.

2. Another way to deal with additive noise is to introduce a whitening filter to convert the correlated residual into a white residual. Consider the model:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + v(k) \tag{4.15}$$

   where v is the correlated residual term. We can assume that the correlated residual v can be described by the following autoregressive model:

$$v(k) + \sum_{i=1}^{p} c_i v(k-i) = e(k) \tag{4.16}$$

in which c_i are constant coefficients, p is the order of the model, and e is white noise. In general c_i and p are unknown a priori. However, a good model can be obtained by preassigning p as 2 or 3, and then estimating c_i using a least-squares estimation algorithm. We can rewrite Eq. 4.16 as:

$$C(q^{-1})v(k) = e(k) \tag{4.17}$$

where the filter $C(q^{-1})$ is called the whitening filter. Combining Eq. 4.15 and Eq. 4.17 the following system-noise model can be derived:

$$C(q^{-1})A(q^{-1})y(k) = C(q^{-1})B(q^{-1})u(k) + e(k) \tag{4.18}$$

A block diagram showing this system is given in Figure 4.3-c. It is clear from Eq. 4.18 that to estimate the system parameters a_i and b_i it is necessary to estimate the residual autoregression coefficients c_i. However, Eq. 4.18 is not linear in the polynomials $A(q^{-1}), B(q^{-1})$, and $C(q^{-1})$. Therefore, these parameters cannot be estimated by linear procedures and a numerical procedure is required. A generalized least-squares solution can be found as explained in Hsia [22]. However, some drawbacks are as follows:

- There are more parameters to estimate due to the C polynomial. This will decrease the convergence rate.
- It is computationally more complicated.
- For high noise-to-signal ratios the convergence rate is very slow.
- For a more correct noise model description, p should be large. Assigning p a small value will decrease the accuracy of the noise model.

3. A more general approach to modeling the noise is to introduce an auto-regressive moving-average (ARMA) model [22] as depicted in Figure 4.3-d. Eq. 4.18 contains the term $A(q^{-1}) \cdot C(q^{-1})$ which in practical situations can be very high order. An ARMA model in this case, however, can be used to lower the order of the transfer function given in Eq. 4.18. With suitable choice of the orders of the D and C polynomials, a generalized least squares algorithm can be derived similar to the previous case. The same four drawbacks given above also apply to this case.

4.3.3 Comparison by an example

To compare the performances of a continuous-time parameter estimation algorithm and a discrete-time parameter estimation algorithm, the system in Figure 4.2 is considered with the following dynamics:

$$G(s) = \frac{B(s)}{A(s)} = \frac{3.5}{s^2 + 3.0s + 3.5} \tag{4.19}$$

and with white noise e. The damping ratio, natural frequency, and gain term of Eq. 4.19 are close to the actual values of the throttle-torque system studied in this dissertation.

The discrete-time representation of the above system with a 50 Hertz sampling frequency is:

$$G(z) = \frac{B(z)}{A(z)} = \frac{10^{-4}(6.861z + 6.725)}{z^2 - 1.9404z + 0.9418} \tag{4.20}$$

Using a random gaussian input sequence uniformly distributed between 0 and 1, the input-output data can be generated using Eq. 4.20 and assuming zero initial conditions. White, zero-mean measurement noise is then added to the output data.

Using these noise-added output and input sequences, the parameters of the model in Eq. 4.19 and Eq. 4.20 can be estimated using continuous-time and discrete-time approaches, respectively. For easy comparison the estimated continuous-time model parameters are converted to discrete-time parameters using a 50 Hertz sampling rate. The comparison is made for 5 different noise-to-signal ratios (NSR). Noise-to-signal ratio (NSR) is defined as the ratio of the variances of the noise and input signals.

1. *Discrete-time parameter identification:* Because of the additive white-noise measurement error terms, Eq. 4.14 describes this system. The model of Eq. 4.14 cannot be converted directly to a regression model because the variables $e(k)$ are not known. By applying the Extended Least Squares (ELS) approach [38], a regression model can be obtained by suitable approximations. The parameter and regressor vectors become:

$$\theta = [-a_1 \quad -a_2 \quad b_1 \quad b_2 \quad c_1 \quad c_2] \tag{4.21}$$

$$\varphi^T(k) = [y(k-1)\ y(k-2)\ u(k-1)\ u(k-2)\ \varepsilon(k-1)\ \varepsilon(k-2)] \tag{4.22}$$

where

$$\varepsilon(k) = y(k) - \varphi^T(k)\theta(k-1) \tag{4.23}$$

The variables $e(k)$ are thus approximated by the prediction errors $\varepsilon(k)$. The model can be approximated by:

$$y(k) = \varphi^T(k)\theta \tag{4.24}$$

The standard least squares algorithm can now be applied to find the parameter vector.

2. *Continuous-time model parameter identification:* The differential-equation representation of the system, shown in Figure 4.2 and Eq. 4.19, is given by:

$$\frac{d^2w}{dt^2} + a_1\frac{dw}{dt} + a_2 = b_1 u \tag{4.25}$$

$$y = w + e \tag{4.26}$$

By taking PMF's of these two equations we get:

$$M_k\left[\frac{d^2w}{dt^2}\right] + a_1 M_k\left[\frac{dw}{dt}\right] + a_2 M_k[w] = b_1 M_k[u] \tag{4.27}$$

$$M_k[y] = M_k[w] + M_k[e] \tag{4.28}$$

Since for sufficiently large time t_0 and small λ, $M_k[e] \rightarrow 0$, we can write that $M_k[y] \simeq M_k[w]$. Now, if we expand Eq. 4.27, and substitute $w_k^0 = y_k^0$, we come up with the following representation:

$$y_{k-2}^0 - 2\lambda y_{k-1}^0 + \lambda^2 y_k^0 + a_1(y_{k-1}^0 - \lambda y_k^0) + a_2 y_k^0 = b_1 u_k^0 \tag{4.29}$$

where y_k^0 and u_k^0 are called the k-th Poisson Moment Functional (PMF) of y(k) and u(k) at $t = t_0$. If we take $\lambda = 1$, $k = 2$, and omit superscripts for simplicity, Eq. 4.29 becomes:

$$y_0 - 2y_1 + y_2 + a_1(y_1 - y_2) + a_2 y_2 = b_1 u_2 \tag{4.30}$$

Figure 4.4 shows how PMF's of the input and output signals are obtained. The parameters a_1, a_2, and b_1 can then be obtained with a classical recursive least-squares algorithm.

The performances of the continuous-time and discrete-time identification algorithms for this example are compared in Table 4.2. As shown in Table 4.2, the PMF

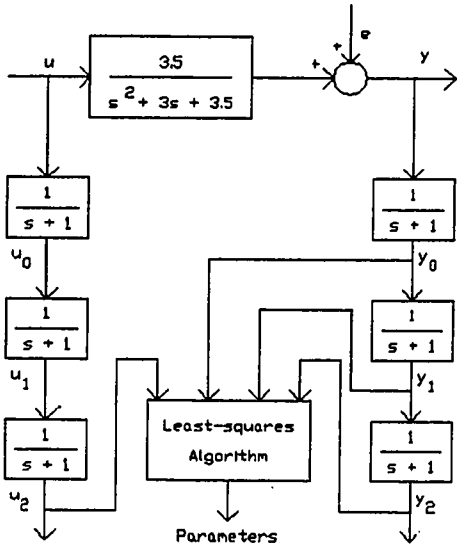


Figure 4.4: PMF's of input and output signals

approach gave very good estimates of system parameters. For the continuous-time case the sum of squares of error was 0.053 even for noise-to-signal ratio (NSR) as high as 80% while the discrete time case was about 0.5 for NSR as low as 5%.

To show the effectiveness of the PMF approach in the case of colored measurement the system in Figure 4.5 was considered. In Figure 4.5, τ represents the time constant of the filter and d represents colored noise. For different filter time constants and different noise-to-signal ratios the PMF algorithm was used to estimate the system parameters. Table 4.3 lists the findings. As seen from Table 4.3 the PMF method gives very good estimates of system parameters even in colored measurement noise cases. Even when the noise-to-signal ratio was 50% and the time constant of the filter was 5 seconds, the estimated denominator parameters were within 3% of

Table 4.2: Comparison of parameter identification algorithms

Method	NSR[a]	a_1	a_2	$b_1 10^{-3}$	$b_2 10^{-3}$	SSE[b]
	0	1.9404	-0.9418	0.6861	0.6725	0.0000
	5	1.9370	-0.9384	0.7053	0.6905	0.0000
CMI[c]	15	1.9348	-0.9362	0.7188	0.7032	0.0084
	30	1.9363	-0.9377	0.6927	0.6781	0.0128
	50	1.9417	-0.9430	0.6271	0.6150	0.0219
	80	1.9498	-0.9508	0.5379	0.5289	0.0530
	0	1.9404	-0.9418	0.6861	0.6725	0.0000
	5	0.4242	0.5388	18.882	17.864	0.5085
DMI[d]	15	0.4025	0.4923	47.351	49.174	3.9225
	30	0.4531	0.4223	59.807	49.293	5.2271
	50	0.4368	0.3933	84.795	68.524	6.3535
	80	0.4018	0.3590	116.82	96.061	8.0575

[a]Noise-to-signal ratio in percentage.

[b]Sum of squares of error between model output and actual output without additive measurement noise.

[c]Continuous-time model identification based on PMF approach. System parameters are obtained via a recursive least-squares algorithm.

[d]Discrete-time model identification based on recursive extended least-squares approach.

their actual values.

4.4 Throttle-Torque System Identification Through PMF Method

As mentioned earlier, three adaptive control algorithms were tested and compared in this research. These algorithms are:

1. One-shot controller design via one-shot parameter estimation.

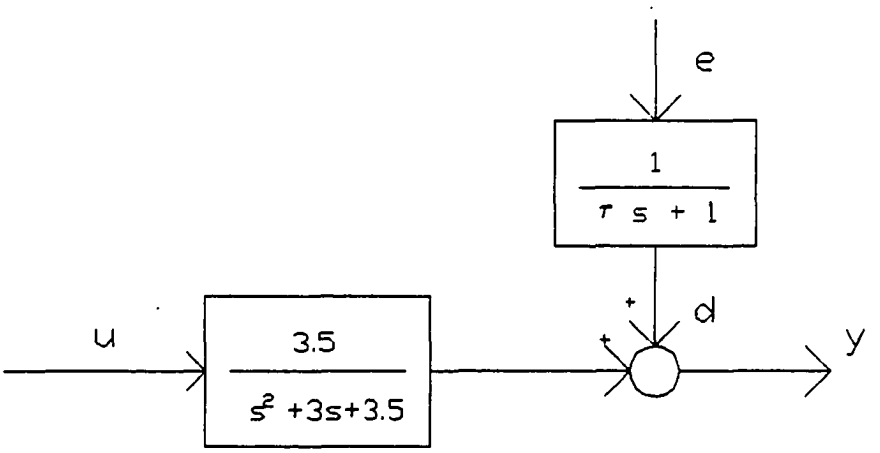


Figure 4.5: Single input single output system with colored measurement noise

2. Gain scheduling control based on multiple one-shot controller designs via multiple one-shot parameter estimations.

3. Continuous adaptation of the controller based on continuous update of parameters.

The first two algorithms assume no prior knowledge of the system dynamics, that is, the continuous-time model order, the time delay, and selected model parameters should all be determined on-line, immediately before starting the transient test cycle. The motivation for these algorithms is to be able to run transient test cycles on different engines without spending any time on system identification and controller design for each engine. In contrast, the third algorithm, which is based on a conventional adaptive control approach, calls for a continuous update of parameters during the test cycle, and requires that model order, time delay, and estimated

Table 4.3: Performance of PMF method in colored noise

NSR[a]	τ[b]	a_1	a_2	$b_1 10^{-3}$	$b_2 10^{-3}$	SSE[c]
0	—	1.9404	-0.9418	0.6861	0.6725	0.0000
10	0.2	1.9417	-0.9431	0.6781	0.6652	0.3216
30	0.2	1.9442	-0.9455	0.6544	0.6423	0.7355
50	0.2	1.9475	-0.9487	0.6221	0.6104	1.4011
10	1.0	1.9437	-0.9450	0.6628	0.6514	1.0712
30	1.0	1.9523	-0.9534	0.5756	0.5662	5.5403
50	1.0	1.9670	-0.9678	0.4147	0.4079	17.912
10	5.0	1.9499	-0.9511	0.5891	0.5797	9.2842
30	5.0	1.9579	-0.9589	0.5018	0.4945	31.598
50	5.0	1.9653	-0.9661	0.4229	0.4163	66.781

[a]Noise-to-signal ratio in percentage.

[b]Filter time constant.

[c]Sum of squares of error between model output and actual output without additive noise.

values of model parameters be obtained off-line prior to the transient test cycle. The motivation for this algorithm is to be able to track time-varying values of system parameters, and therefore provides better tracking of torque and speed trajectories for a specific engine and dynamometer pair. Since it requires prior experiments and controller designs carried out off-line, it may be less advantageous for testing different engine and dynamometer pairs. The following sections detail the application of the PMF technique to all three cases explained above.

4.4.1 One-shot system and parameter identification using the PMF method

The throttle-speed-torque system is a multiple-input single-output (MISO) system with two inputs and one output as depicted in Figure 4.6. As shown in

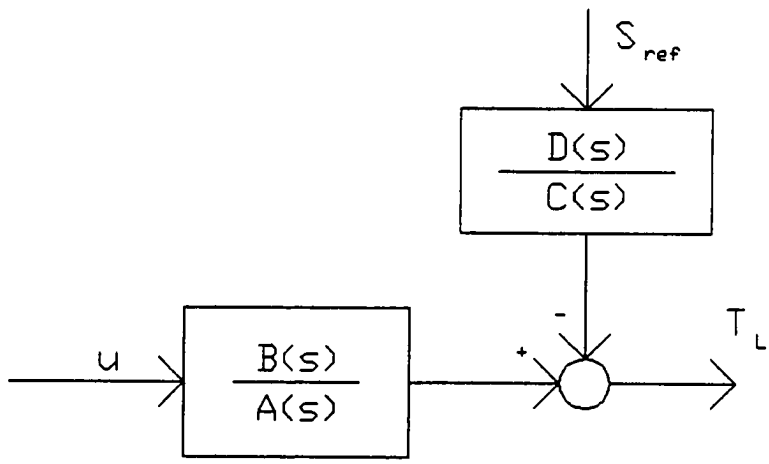


Figure 4.6: Throttle-speed-torque system

Figure 4.6, load torque can be written in the Laplace domain in terms of throttle command, u, and speed reference command, S_{ref}, as:

$$T_L = \frac{B(s)}{A(s)} e^{-T_f s} u + \frac{C(s)}{D(s)} e^{-T_d s} S_{ref} \tag{4.31}$$

Expanding Eq. 4.31 gives:

$$[A(s)D(s)]\,T_L = \left[B(s)D(s)e^{-T_f s}\right] u + \left[C(s)A(s)e^{-T_d s}\right] S_{ref} \tag{4.32}$$

Some of the disadvantages of estimating the parameters of this MISO system are given by Unbehauen and Rao [31] and can be listed as:

1. MISO system identification will increase the system order as shown in Eq. 4.32 and therefore there will be more parameters that need to be estimated. It means more severe persistently exciting conditions on input signals and slower convergence rate of parameters. An input signal is considered to be persistently exciting if it contains a sufficient number of distinct frequencies [32]. Therefore, increasing model order causes more restrictions on input signals.

2. In MISO system identification, we cannot find the parameters of the individual transfer functions $\frac{B(s)}{A(s)}$ and $\frac{C(s)}{D(s)}$. This is due to the fact that the parameters of Eq. 4.32 are not linear on $A(s), B(s), C(s)$, and $D(s)$. Therefore, we need to find the parameters of $\frac{B(s)D(s)}{A(s)D(s)}$ and $\frac{C(s)A(s)}{A(s)D(s)}$. However, in controller design, it is necessary that common poles be canceled out from each transfer function. Thus, extra computation is required to extract $A(s)$, $B(s)$, $C(s)$, and $D(s)$.

3. If the time delays T_f and T_d are different, then estimation of these time delays will increase the computational complexity of the estimation algorithm.

However, for one-shot system identification purposes, the disadvantages listed above can be avoided by considering two separate single-input single-output (SISO) system identification problems as shown in Figure 4.7.

To find the orders and parameters of $B(s)$ and $A(s)$, and the time delay T_f, a speed reference command is kept constant and a step input to the throttle command is given. Load torque signals are sampled at a 50 Hz sampling rate for 5 seconds, and then are stored in the computer's internal memory. Similarly, to find the parameters of $C(s)$ and $D(s)$, and the time delay T_d, the throttle command is kept constant and a step input to the speed reference command is given. Again, load torque signals are

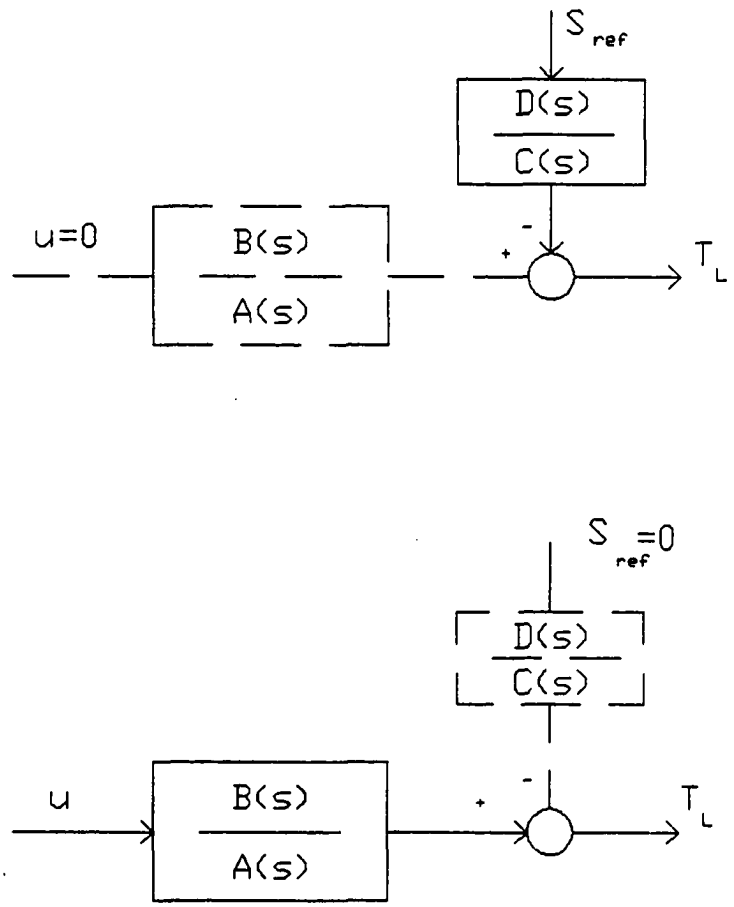


Figure 4.7: Two SISO representation of throttle-speed-torque system for system identification purpose

sampled at a 50 Hz rate for 5 seconds, and the values are stored. Since there is no a priori information assumed about the system, different models can be considered, the parameters of each model determined, and the model that describes the system best is found via an error minimization algorithm. The following models were considered both for throttle-torque and speed-torque systems:

1. First order system
2. Second order system
3. Second order system with first order numerator dynamics
4. Third order system
5. Third order system with first order numerator dynamics
6. Third order system with second order numerator dynamics

Figure 4.8 a-f shows these models in the "s" domain. Derivation of the recursive parameter estimation equations for each case using the PMF approach can be found in Appendix A.

The time delay was estimated through a separate minimization algorithm. For 30 different time delays, evenly distributed between 0 and 0.6 seconds, parameters of each considered system were found. For each time delay considered, the sums of squares of error, $J = \sum_{i=1}^{250}(T_{L,i} - \hat{T}_{L,i})^2$, were calculated. $T_{L,i}$ corresponds to the set of actual observed load torque values and $\hat{T}_{L,i}$ corresponds to the set of estimated load torque values based on the model parameters. The time delay that gives the minimum sum of squares of error is considered to be the best estimate of the actual time delay for that particular model. This is repeated for each of the six cases outlined above. Among these models, the one that gives the minimum sum of

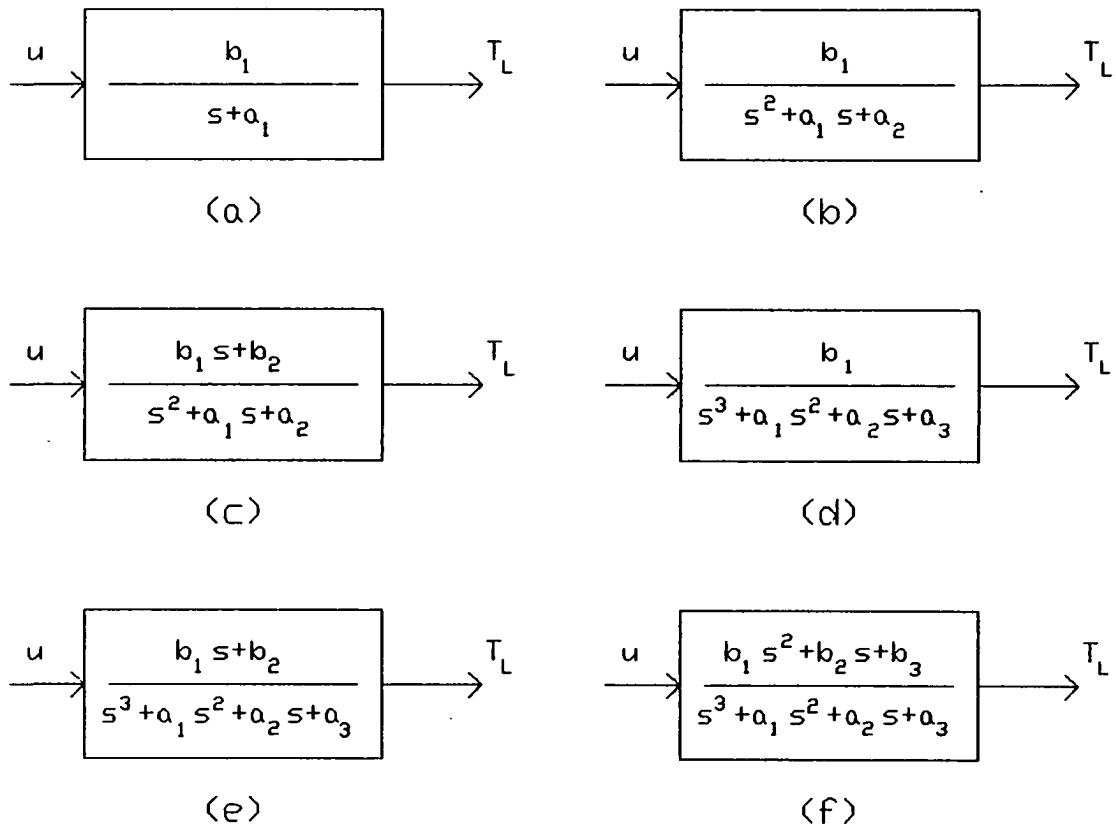


Figure 4.8: Representation of the system model in the "s" domain

squares of the error is selected as the best estimate of the system transfer function and time delay.

4.4.2 PMF system and parameter identification for gain-scheduling control

As seen from Eq. 4.31, the engine speed appears as a quadratic nonlinear input to the describing equation of the governor. Therefore, parameters of the throttle valve-speed-torque system will depend on the operating conditions of the system. A gain-scheduling control was implemented based on the measurements of engine speed. Operating conditions of the engine are divided into 5 regions according to the engine speed: 1200-1400 rpm, 1400-1600 rpm, 1600-1800 rpm, 1800-2000 rpm, and 2000-2200 rpm. For each speed range the parameter and system identification is repeated through multiple one-shot parameter estimation algorithms, in a very similar fashion to the previous section. When implementing the transient test cycle, the controller parameters are changed as a function of the operating conditions based on the known parameters of the system for each speed range.

4.4.3 Continuous update of parameters through the PMF method

If controller parameters are to be changed based on the parameters of the system in a recursive fashion, then system identification must be carried out prior to the transient test cycle. In other words, the system model order, time delay, and an estimate of the system parameters are needed for continuous adaptation of the controller. This is due to the following reasons:

1. For a continuous parameter update of the system, a model for the system should be selected prior to implementing the parameter estimation algorithm.

2. Estimating the time delay in a recursive fashion is very costly from a computational point of view. Therefore, it is usually assumed that an upper bound on the time delay is known.

3. Although recursive parameter estimation algorithms do not theoretically require a prior knowledge of system model parameters, in actual real-time implementation of adaptive control algorithms prior parameter estimation is needed. During the parameter estimation algorithm if the parameters estimate an unstable system, or if they are not within a predefined range, then the system parameters should be switched to their pre-estimated values.

Since parameter updates will be carried out recursively during the transient test cycle, MISO system identification should be possible. If we rewrite the MISO representation of the system as in Eq. 4.32:

$$[A(s)D(s)]\,T_L = \left[B(s)D(s)e^{-T_f s}\right] u + \left[C(s)A(s)e^{-T_d s}\right] S_{ref} \tag{4.33}$$

Eq. 4.33 can be written as:

$$A_c(s)T_L = B_c(s)e^{-T_f s}u + C_c(s)e^{-T_d s}S_{ref} \tag{4.34}$$

where $A_c(s) = A(s)D(s)$, $B_c(s) = B(s)D(s)$ and $C_c(s) = C(s)A(s)$.

Using the technique from the previous section, that is, employing two SISO identification algorithms, $A_c(s)$, $B_c(s)$ and $C_c(s)$ can be found. Applying Eq. 4.34 to the results of the previous section gives a fourth order MISO system. However,

higher order models are not suitable in a recursive parameter estimation algorithm since parameter convergence will be very slow and more severe persistent excitement conditions on the input signals will be required. Therefore, a lower order model is fitted to the experimental data. Using step inputs to the throttle valve and speed reference commands as shown in Figure 4.9, the response of the system is obtained with a 50 Hz sampling rate. Then, using a recursive PMF parameter identification algorithm, as explained in Appendix A, the lowest order of the system that describes the input-output relation with a reasonable accuracy was determined. It was found that, as shown in Eq. 4.35, a second order MISO model can adequately represent the throttle-speed-torque system.

$$G_L = \frac{5.55}{s^2 + 3.76s + 4.34} e^{-0.4s} u + \frac{4.15}{s^2 + 3.76s + 4.34} e^{-0.4s} S_{ref} \qquad (4.35)$$

Figure 4.10 compares the actual load torque output with estimated load torque values when the step inputs shown in Figure 4.9 are applied to the system. As seen from Figure 4.10, the second order model shown in Eq. 4.35 is a good approximation for the overall open-loop throttle-speed-torque system.

It was shown in this chapter that the PMF method was a very powerful tool in estimating the continuous-time model parameters from the discrete set of the input-output measurements, and it showed a significant convergence superiority to the discrete-time parameter identification methods when measurements were corrupted with white (uncorrelated) or colored (correlated) noise. Chaper 6 will show how a pole-zero assignment controller design will be performed based on the process knowledge obtained using the PMF method.

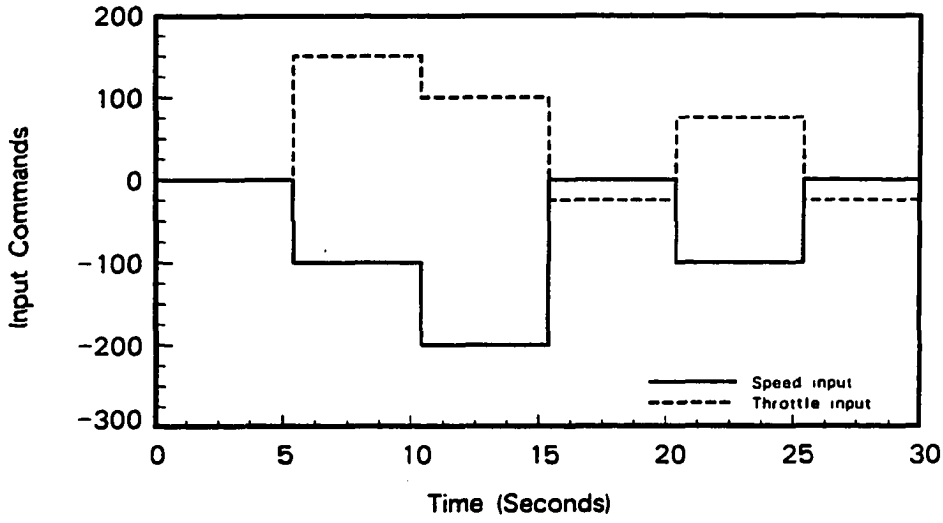


Figure 4.9: Step inputs to throttle valve and speed reference command

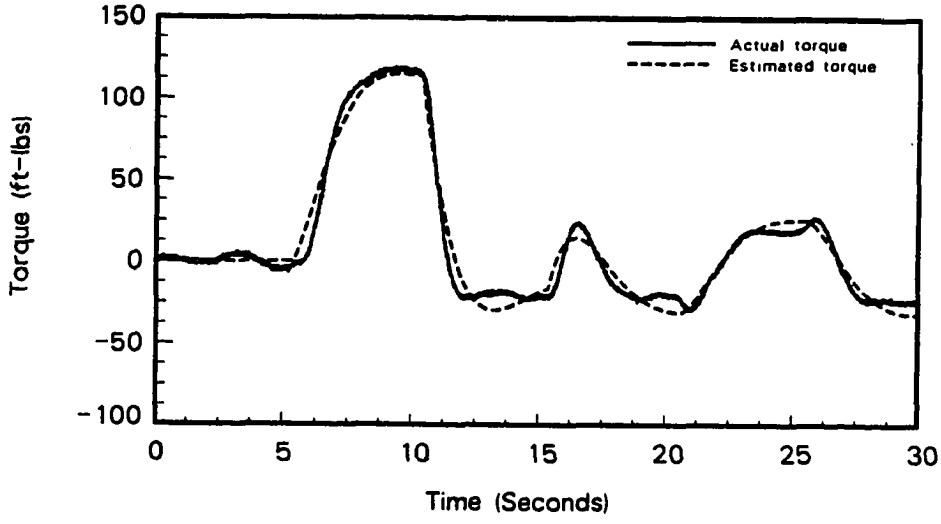


Figure 4.10: Comparison of estimated and actual load torque values

5. CONTROLLER DESIGN

Various controller design methods can be used in self-tuning regulators depending on the specifications of the closed-loop system. Some of the well known design methods are minimum variance control, pole-zero placement, linear quadratic gaussian control, and hybrid control [32]. In the Federal Transient Test procedure, engine speed and torque should follow a trajectory specified by the Environmental Protection Agency (EPA). The EPA provides specifications on the quality of the closed-loop control. Since the closed-loop response specifications will be in the time domain, the pole-zero assignment algorithm is the most suitable controller design method.

For an adaptive control strategy that requires one-shot parameter estimation, a one-shot controller design is performed on-line using the pole-zero assignment algorithm. Since the model order is not known a priori, different pole-zero assignment algorithms are developed to cover every possible case considered in this study. When the throttle-torque process has unstable zeros, that is, zeros outside the unit circle in the discrete domain, then a limiting form of the design algorithm, namely pole-placement, has been employed to prevent canceling unstable zeros. The option of adding an integrator to the controller to have zero steady-state error and to be less sensitive to low frequency modeling errors is used in this research.

For gain-scheduling control, multiple one-shot pole-zero placement methods are

carried out. Depending on the engine speed, the controller parameters are modified during the transient test cycle.

For continuous estimation of system parameters the parameters of the controller are also updated at each sampling point. A pole assignment algorithm with an integrator has been used. The process zeros are not cancelled due to the possibility that the estimation algorithm may sometimes result in unstable process zeros.

Derivation of the pole-zero placement algorithm for the different cases explained above will be given later in this chapter. Section 5.1 gives the Environmental Protection Agency (EPA) specifications for a valid transient test cycle. Section 5.2 summarizes the well-known pole-zero placement controller design method following that of Astrom and Wittenmark [38]. However, during the computer simulations of the closed-loop system performance with the controller designed through this pole-zero assignment algorithm, it was found that closed-loop stability is not guaranteed with the pole-zero assignment algorithm. This was mainly due to the arbitrary selection of the observer polynomial parameters in the pole-zero assignment method. The development of a new modified pole-zero assignment technique to ensure closed-loop stability is explained in Section 5.3. The application of this modified algorithm to the three adaptive approaches to the torque control problem is discussed in Section 5.4. Since there is a time-delay in the throttle-torque system, the Smith Predictor scheme was used to compensate the time-delay effectively. Section 5.5 reviews the Smith predictor design of Marshall [52]. Although the Smith predictor has the advantage of compensating the time-delay, it cannot effectively deal with known disturbances of large amplitudes. Due to the speed-torque interactions, there is a large load disturbance applied to the throttle-torque control problem. Design of a feedforward

controller to compensate for the load disturbances is explained in Section 5.6. Section 5.7 discusses implementation issues of the adaptive torque control approaches such as estimator implementation, sampling rate selection, anti-aliasing filter and antireset-windup implementation. Section 5.8 explains the software development.

5.1 EPA Specifications

The heavy-duty transient engine cycles for gasoline and diesel fueled engines are listed in [49]. These second by second listings simulate typical torque and speed operating conditions of heavy-duty engines. In these listings, both speed and torque are normalized (expressed as a percentage of maximum). Both the speed and torque trajectories should be unnormalized for each specific engine considered. To unnormalize rpm, the following equation is used:

$$Actual\ rpm = \frac{\%rpm\ (Measured\ rated\ rpm - curb\ idle\ rpm)}{100} + curb\ idle\ rpm \tag{5.1}$$

Torque is normalized to the maximum torque that can be produced by the engine at the rpm listed with it. Therefore, to unnormalize the torque values in the cycle, the maximum torque curve for the John Deere engine is used. Figure 5.1 shows the reference torque and speed trajectory in normalized form. Although some torque values are referred to as "closed rack motoring" in the reference torque trajectory, they were set to -10 in Figure 5.1 for clarity. Actual values of these "closed rack motoring" torque values should be calculated for each engine considered following EPA guidelines explained in [49]. One way to calculate those torque values is to find the amount of negative torque required to motor the engine at idle and rated speeds, and then linearly interpolate using these two points.

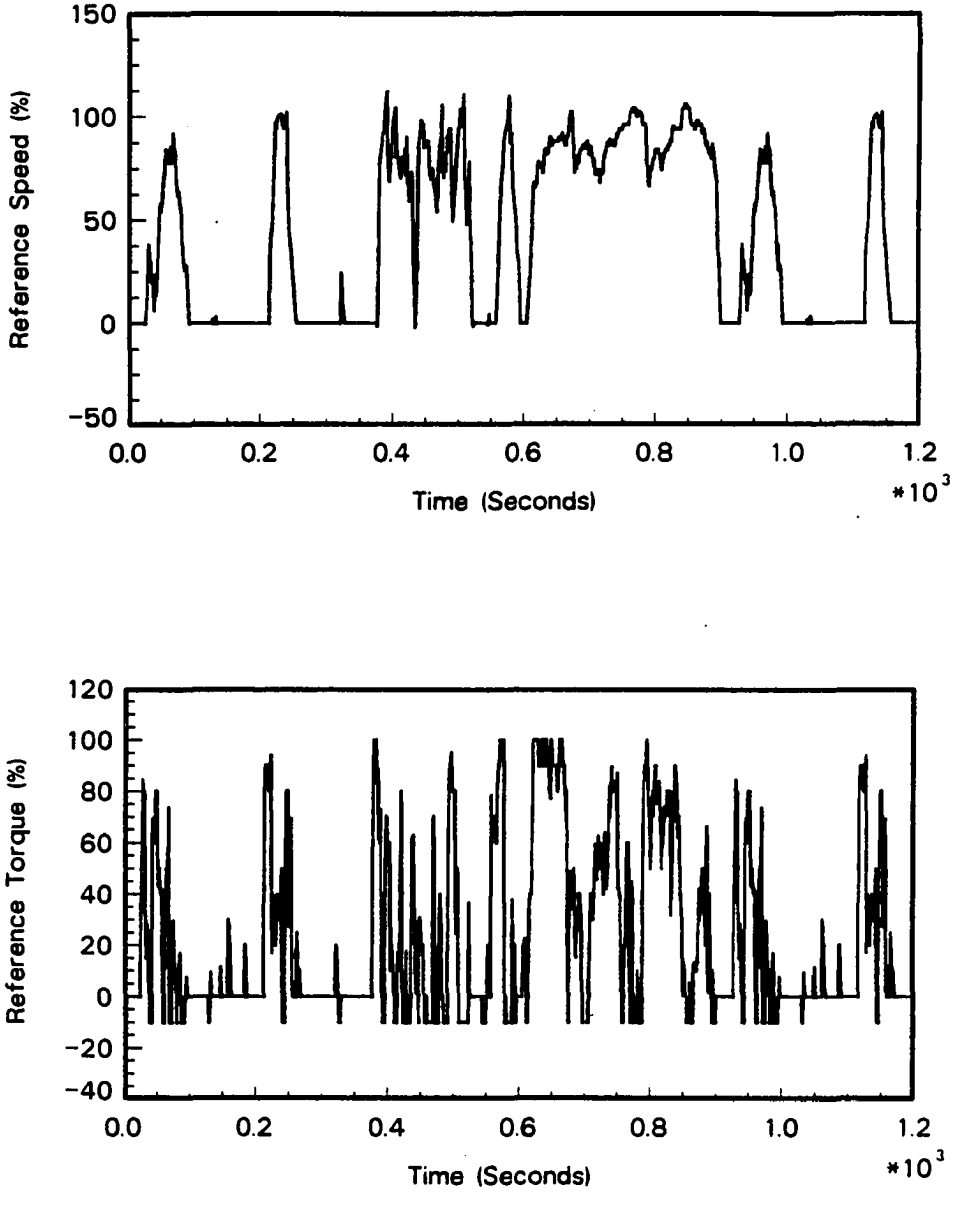


Figure 5.1: Reference speed and torque trajectories for heavy-duty diesel engines

To have a valid transient test cycle, the following specifications must be met by the system:

- The coefficients of the linear regression equation between the actual and reference values of torque, speed, and brake horsepower must lie within the specified limits given in Table 5.1. The method of least squares is used to find the best fit equation having the form

$$y = mx + b \tag{5.2}$$

where

y = The feedback value of speed, torque, or brake horsepower
m = Slope of the regression line
x = The reference value of speed, torque, or brake horsepower
b = The y intercept of the regression line

- The standard error of estimate SE of y on x and the coefficient of determination (r^2) must also be within the limits given in Table 5.1. The values of SE and r^2 are determined from the following equations [55]:

$$r^2 = \frac{\left[\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})\right]^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2} \tag{5.3}$$

$$SE = \sqrt{\frac{\sum_{i=1}^{n}[y_i - (mx_i + b)]^2}{n-1}} \tag{5.4}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$n = 1199 \cdot (Number\ of\ observations)$$

For a test to be considered valid, the criteria in Table 5.1 must be met for both cold and hot cycles individually. The cold cycle procedure specified for the federal test procedure transient test is to cold-soak the engine by shutting it off for a minimum of 12 hours or until the oil temperature reaches 75^{o}F. The hot cycle follows the cold cycle after a hot-soak period of 20 minutes.

Table 5.1: Regression line tolerances

	Speed	Torque	BHP[a]
SE_{max}	100 rpm	28.0[b] ft-lb	6.4[c] BHP
m	0.97-1.03	0.83-1.03 hot 0.77-1.03 cold	0.89-1.03 hot 0.87-1.03 cold
r^2_{min}	0.97	0.88 hot 0.85 cold	0.91
b	±50 rpm	±15 ft-lb	±5 BHP

[a] Brake horsepower.

[b] 13 percent of power map maximum engine torque.

[c] 8 percent of power map maximum BHP.

- The total work done during the cycle is calculated by integrating the power based on the actual speed and torque. This actual work is used for comparison to the reference work. The reference work for the John Deere engine tested in this research was 5.46 brake horsepower-hour. According to the EPA, the actual brake work for each cycle (cold and hot start) must be between +5 percent and −15 percent of the reference brake work in order for a test to be valid.

5.2 Controller Design: Pole-Zero Assignment Algorithm

In this section, the well-known pole-zero assignment algorithm will be reviewed. Some deficiencies of this method and the development of a new pole-zero assignment algorithm will be explained in Section 5.3. Controller design using the pole assignment algorithm is explored in many references. In this research, the notation follows that of Astrom and Wittenmark [43].

Consider the closed-loop torque control system shown schematically in Figure 5.2. T_r represents the reference torque trajectory, T_a is the actual torque output, and u is the control signal. The process model is specified by $\frac{B(z)}{A(z)}$. Since a Smith predictor will be used in the final design, the time delay is excluded from the process model for controller design purposes. The use of the Smith predictor to overcome the time delay problem, and feedforward compensator design for the reference speed-torque system will be explained later in this chapter. The desired closed-loop system response is specified by $\frac{B_m(z)}{A_m(z)}$. The control signal, u, is calculated by $R(z)u = T(z)T_r - S(z)T_a$ as seen in Figure 5.2.

The pole-zero placement problem then becomes the determination of polynomials $R(z), S(z)$, and $T(z)$ such that the closed-loop system transfer function will be equal

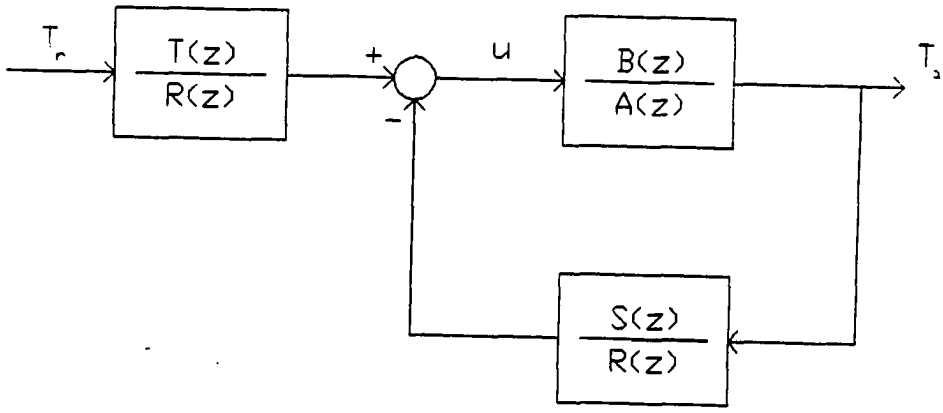


Figure 5.2: Block diagram of the pole assignment control law

to the desired closed-loop transfer function, $\frac{B_m(z)}{A_m(z)}$. The solution is given by the following seven step process.

1. Specify the desired closed-loop response. In other words, specify the $A_m(z)$ and $B_m(z)$ polynomials from the design specifications. The orders of the $A_m(z)$ and $B_m(z)$ polynomials should meet the following inequality:

$$degA_m(z) - degB_m(z) \geq degA(z) - degB(z) \tag{5.5}$$

   Setting $degA_m(z) = degA(z)$ and $degB_m(z) = degB(z)$ will simplify the controller design.

2. Factor $B(z)$ as $B(z) = B^-(z)B^+(z)$, where $B^-(z)$ has all its zeros outside the unit circle in the "z" domain and $B^+(z)$ has all its zeros inside the unit circle. Fix the highest power of $B^+(z)$ to unity. In other words $B^+(z)$ should

be a monic polynomial. If cancellation of process zeros are not desired, then set $B^+(z) = 1$, and $B^-(z) = B(z)$.

3. Factor $B_m(z)$ as $B_m(z) = B^-(z)B_m^*(z)$. (Closed loop zeros should include all open-loop zeros that are not cancelled)

4. Introducing an integrator to the controller will improve the closed-loop system's response in the following ways:

   (a) It ensures that the closed-loop system has a high feedback gain at low frequencies. It means that the closed-loop system will be insensitive to low-frequency modeling errors and low-frequency disturbances.

   (b) It forces the steady state error to go to zero in case of modeling errors and system disturbances.

   Integral action can be achieved by requiring that $(z-1)$ is a factor of $R(z)$:

$$R(z) = R_1(z) \cdot (z-1) \tag{5.6}$$

5. Find the degree of the observer polynomial from the following inequality:

$$degA_o(z) \geq 2degA(z) - degA_m(z) - degB^+(z) \tag{5.7}$$

   Select an observer polynomial $A_o(z)$. A dead-beat observer can be chosen for simplicity. In principle, observer polynomial dynamics should be faster than system dynamics.

6. Find $R_1(z)$ and $S(z)$ from the following Diophantine equation:

$$A(z)(z-1)R_1(z) + B^-(z)S(z) = A_o(z)A_m(z) \tag{5.8}$$

Choose a solution such that $degS(z) \leq degA(z)+1$ and $degR_1(z) = degA_o(z)+ degA_m(z) - degA(z)$.

7. Calculate $R(z)$ and $T(z)$ from following equations:

$$R(z) = B^+(z)(z-1)R_1(z) \tag{5.9}$$

$$T(z) = B_m^*(z)A_o(z) \tag{5.10}$$

5.3 A Modified Pole-Zero Assignment Algorithm

In the pole-zero assignment algorithm of Astrom and Wittenmark, as explained in Section 5.2, the observer polynomial $A_o(z)$ was selected somewhat arbitrarily. The only condition imposed on the observer polynomial was to restrict all the zeros of the observer polynomial to lie inside the unit circle in the z-plane. However, the selection of the observer polynomial plays an important role in the adaptive control applications as listed below:

1. As mentioned in Section 5.2, the observer polynomial dynamics should be faster than system dynamics to provide a better disturbance rejection capability.

2. Although the observer polynomial does not have any effect on the overall closed-loop transfer function, it effects the closed-loop stability in the real-time implementation. An arbitrarily selected observer polynomial could result in the design of an unstable controller. In other words, there is always a possibility that the zeros of the $R(z)$ polynomial can lie outside the unit circle in the z-plane. This means that although the overall closed-loop system looks stable from the theoretical analyses, it cannot be implemented due to unstable controller design.

3. The observer polynomial also effects the input signals applied to the system. When observer polynomial zeros are close to the origin in the z-plane, the resulting control law can create large control input signals which may not be realizable in the real-time implementation. On the other hand, setting the zero locations close to the unit circle will result in high sensitivity to any disturbance that enters the system.

To overcome these limitations of the pole-zero assignment algorithm, some modifications can be made. The proposed algorithm becomes:

1. Follow steps 1 through 4 from Section 5.2.

2. Find the degree of the observer polynomial from Eq. 5.7.

3. Initially set all the zeros of the observer polynomial to the origin in the z-plane. This produces a dead-beat observer which has very good noise rejection capability.

$$A_O(z) = (z - a_O)^l \tag{5.11}$$

where $a_O = 0$, and l is the degree of the observer polynomial.

4. Solve for $R_1(z)$ and $S(z)$ from the Diophantine equation given in Eq. 5.8.

5. Calculate the zeros of $R_1(z)$. If any of these zeros lie outside the unit circle, it means that the pole-assignment controller with selected observer polynomial results in an unstable controller which makes it impossible to implement in practice. If this is the case, then set $a_O = a_O + \Delta a_O$, where Δa_O is a predefined increment, and repeat steps 3 to 5 until all the zeros of $R_1(z)$ lie inside the unit circle.

6. Calculate $R(z)$ and $T(z)$ from Eq. 5.9 and Eq. 5.10, respectively.

7. If the amplitude of the input signal is a concern, simulate the closed-loop system response to a step input. Calculate the maximum required control signal from the simulations. If it is too high to apply to the real system, then set $a_O = a_O + \Delta a_O$, where Δa_O is same as before, and repeat steps 3 to 6.

8. Repeat steps 3 to 6 at each sampling point.

As mentioned earlier, three adaptive control strategies were developed in this research. Two of these algorithms assume no information about the system. Therefore, model order of the plant transfer function is unknown a priori. Application of the pole-zero placement algorithm to the three algorithms will be explained in the following sections.

5.3.1 Adaptive control with one-shot controller design

Although six different models were considered in the continuous time domain for system identification purposes, only three different cases exist in the discrete-time. Controller design is carried out after the "s" domain transfer function description of the system is converted to a "z" domain transfer function. So controller designs corresponding to all 3 cases need to be considered. The discrete-time representation of the process is:

- if the system is first order:

$$\frac{B(z)}{A(z)} = \frac{b_1}{z + a_1} \tag{5.12}$$

- if the system is second order:

$$\frac{B(z)}{A(z)} = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2} \tag{5.13}$$

- if the system is third order:

$$\frac{B(z)}{A(z)} = \frac{b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3} \tag{5.14}$$

Application of the modified pole-assignment algorithm to the throttle-torque control system is shown by the following steps:

1. *Specification of the closed-loop response:* To avoid complexity, the order of the closed-loop reference model was selected to be equal to the order of the open-loop process model. Therefore, depending on the open-loop system order, the following desired closed-loop transfer functions were selected:

   (a) if the system is first order:

$$\frac{B_m(z)}{A_m(z)} = \frac{1-a}{z-a} \tag{5.15}$$

   (b) if the system is second order:

   - when process zeros are cancelled

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^2 z}{(z-a)^2} \tag{5.16}$$

   - when process zeros are not cancelled

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^2}{b_1 + b_2} \cdot \frac{b_1 z + b_2}{(z-a)^2} \tag{5.17}$$

   (c) if the system is third order:

- when process zeros are cancelled

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^3 z^2}{(z-a)^3} \tag{5.18}$$

- when process zeros are not cancelled

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^3}{b_1+b_2+b_3} \cdot \frac{b_1 z^2 + b_2 z + b_3}{(z-a)^3} \tag{5.19}$$

where $a = e^{-T/\tau}$, T is the sampling period, and τ is the desired time constant of the closed-loop system. The gain term ensures that the closed-loop system will have zero steady-state error. The Environmental Protection Agency (EPA) specifications suggest that the closed-loop system should reach its steady-state value, to a given reference step input, in one-second or less since torque and speed trajectories are given by seconds. Therefore, the desired time constant, τ, is set to 0.25 seconds, since a closed-loop system will reach 98% of its steady-state value to a given step-input in four time constant.

2. *Factorization of $B(z)$ as $B(z) = B^-(z)B^+(z)$:* Depending on whether the process zeros are cancelled or not, $B^-(z)$ and $B^+(z)$ have following values:

   - $B^+(z) = 1$ and $B^-(z) = B(z)$ if process zeros are not cancelled, and
   - $B^+(z) = B(z)/b_1$ and $B^-(z) = b_1$ if process zeros are cancelled. Note that the division by b_1 is required to keep $B^+(z)$ a monic polynomial.

3. *Factorization of $B_m(z)$ as $B_m(z) = B^-(z)B_m^*(z)$:* Using the information from the first two steps $B_m^*(z)$ can be found as: $B_m^*(z) = B_m(z)/B^-(z)$.

4. *Introducing an integrator to the controller:* When an integrator is included as a part of the controller, then the controller polynomial, $R(z)$, becomes:

$$R(z) = (z-1)R_1(z)B^+(z) \tag{5.20}$$

where the degree of the $R_1(z)$ polynomial is the same as the order of the open-loop transfer function. Therefore, the $R_1(z)$ polynomial is selected as follows:

- if process zeros are cancelled: $R_1(z) = 1$, and
- if process zeros are not cancelled:
  (a) $R_1(z) = z + r_1$ if the degree of $A(z)$ is two, and
  (b) $R(z) = z^2 + r_1 z + r_2$ if the degree of $A(z)$ is three.

5. *Determination of the observer polynomial:* The degree of the observer polynomial is determined from:

$$degA_O(z) = 2degA(z) - degA_m(z) - degB^+(z) \tag{5.21}$$

The above formula will give us the following values for the degree of the observer polynomial:

- $degA_O(z) = 1$ if the process zeros are cancelled, and
- $degA_O(z) = degA(z)$ if the process zeros are not cancelled.

The observer polynomial was selected so that all of its zeros lie on the real axis of the z-plane. Furthermore, the zero locations were restricted to the range of 0 to 1 in the z-plane to preserve stability and to prevent oscillatory output

behavior. Initially all zeros were located at the origin. In general, the observer polynomial was selected as:

$$A_O(z) = (z - a_O)^l \tag{5.22}$$

where l is the degree of the observer polynomial and a_O is the zero location, with the initial value of $a_O = 0$.

6. *Determination of the controller polynomials:* The controller polynomials, $R_1(z)$ and $S(z)$, were determined from the Diophantine equation given in Eq. 5.8. The degree of the $S(z)$ polynomial in that equation was selected to be equal to the degree of the $A(z)$ polynomial to preserve a causal control law. The parameters of $R_1(z)$ and $S(z)$ were found by the Gauss elimination method.

7. *Stability considerations:* The roots of $R_1(z)$ were checked. If any root was found outside the unit circle in the z-plane, then observer zero locations were reset as $a_O = a_O + \Delta a_O$, where Δa_O was set to 0.01. Steps 5, 6, and 7 were repeated until all the roots of $R_1(z)$ were inside the unit circle.

5.3.2 Adaptive control with continuous update of controller parameters

As explained in Chapter 4, a second order model was selected to best describe the process under consideration. Therefore, all the controller design equations remain the same as the previous section's second order case. The difference from the one-shot pole placement algorithm will be the calculation of controller parameters at each sampling instant based on the estimated parameters of the system.

In adaptive control systems where continuous update of controller parameters is employed, a pre-designed safe-control algorithm is needed in practice. This is

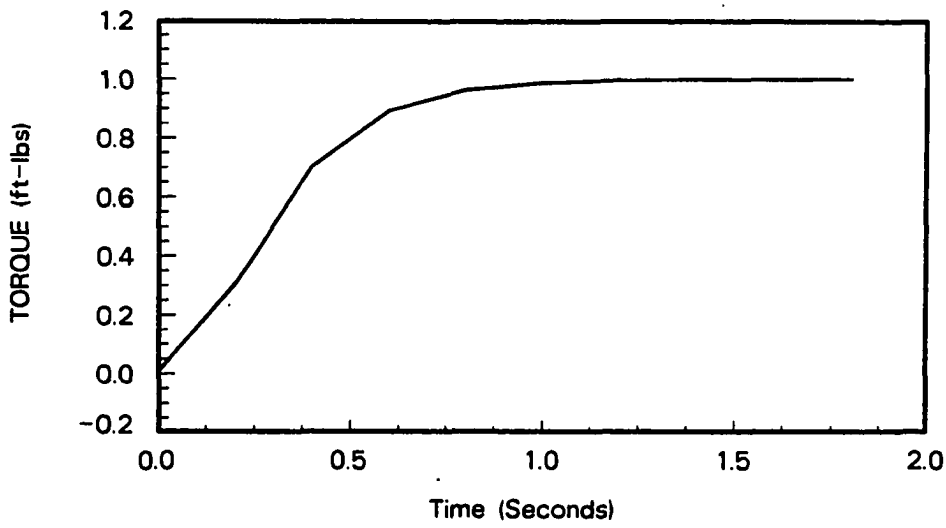


Figure 5.3: Closed loop system simulation when safe controller is employed

due to the fact that when the parameter estimation algorithm occasionally comes up with an unstable process estimate then the resulting pole-placement algorithm will actually lead to an unstable closed-loop operation. Therefore, the following controller is selected as the safe controller using the process description of Eq. 4.35 and a sampling frequency of 5 Hertz.

$$R(z) = z^2 - 0.822z - 0.178 \tag{5.23}$$

$$S(z) = 4.271z^2 - 5.506z + 1.823 \tag{5.24}$$

$$T(z) = 1.940z^2 - 1.744z + 0.392 \tag{5.25}$$

These controller parameters will be used on-line whenever the estimated process parameters are not within the range of reasonable values, as will be explained later in this chapter. Figure 5.3 shows the simulated response of the closed-loop system with the above controller to a step input.

5.4 Smith Predictor

One common characteristic of many process control problems is that the system to be controlled contains a significant time delay. There are three basic design approaches for systems that contain time delays: conventional design methods, optimal control design, and the Smith predictor [50]. Although conventional design methods have the advantage of being less complicated, the closed-loop system will not perform as well as when the time delay is zero. The reason for this is that the delay introduces additional phase shift in the loop and thus tends to destabilize the closed-loop system. To counteract this, the gain of the controller must be reduced below the value which would be used if the delay was zero. Hence the system's response will be slower to input commands.

Application of optimal control approaches to time delay systems is explained in detail in [51]. When optimal design is applied to a system with delay in the control, the basic approach is to convert the problem to a nondelay problem and then use the standard techniques for such problems to obtain a solution. It turns out that the solution requires the prediction of the system state at T_d time units into the future where T_d is the time delay. In this respect, there exists a similarity between the optimal design configuration and the Smith predictor configuration. One disadvantage of the optimal design method to the Smith predictor is that a state estimator of some form must be implemented. In contrast, the Smith predictor requires only output feedback.

Marshall [52] gives a detailed treatment of the Smith predictor design approach. The Smith predictor configuration is used in this research to enlarge the application of adaptive control systems to time delay processes. A block diagram representation

of the Smith predictor is shown in Figure 5.4. A load disturbance is not included

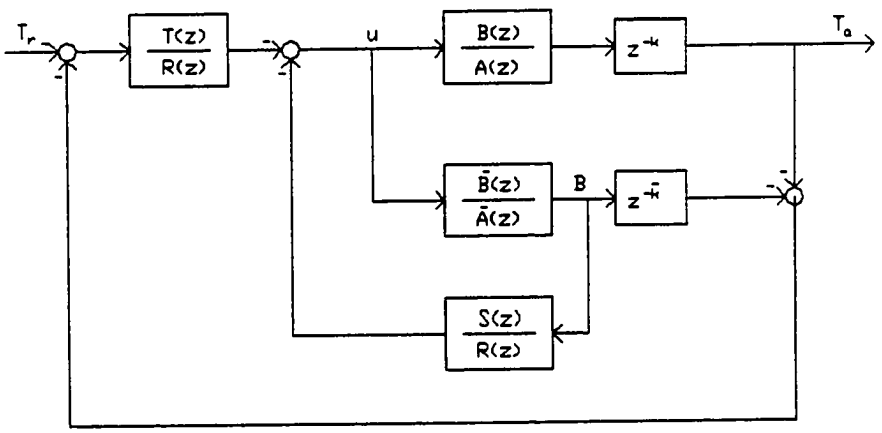


Figure 5.4: Smith predictor for throttle-torque system

in this representation for simplicity. In this figure $T(z)$, $R(z)$, and $S(z)$ represent the delay-free controller, $B(z)/A(z)$ is the delay free part of the process transfer function, $k * T$ is the time delay of the system where T is sampling period, and $\hat{B}(z)/\hat{A}(z)$ and $\hat{k} * T$ are the estimates of the process transfer function and time delay, respectively.

The control signal, u, is delayed by $k * T$ seconds before it effects the output. During this initial period of time the controller cannot influence the output. However, the signal shown on Figure 5.4 at point B is a predicted version of the output signal and can be used as a feedback signal. Thus, the controller design can be carried out using classical methods for delay-free systems. Therefore, the specifications of system performance can be given in familiar delay-free terms.

The second outer loop in Figure 5.4 is included because of the possible mismatch between actual and estimated system parameters. Removal of the outer loop gives

open-loop control, which makes no use of actual output information, and can lead to inferior performance in practice.

5.5 Feedforward Compensation

Any variation of the speed reference command causes a load disturbance to the throttle-torque system. Since the load disturbance does not appear explicitly in Smith's method, a feedforward compensator, $C_d(z)$, is added to the Smith predictor as shown in Figure 5.5. $D(z)/C(z)$ in Figure 5.5 represents the reference speed-torque transfer function. As feedforward does not influence the stability of a control loop, the feedforward control system can be added after the design of the system controller. For an ideal feedforward control $C_d(z)$ is calculated by:

$$C_d(z) = \frac{\hat{G}_d(z)z^{-\hat{k}_d}}{\hat{G}_p(z)z^{-\hat{k}}} \tag{5.26}$$

where $G_p(z) = B(z)/A(z)$, $G_d(z) = D(z)/C(z)$, and $(\hat{})$ represents the estimated quantities of actual variables.

If the above feedforward control can be realized and if it is stable, then the influence of the disturbance, S_{ref}, on the output torque, T_a, is completely eliminated. To obtain a stable feedforward control the zeros of $G_p(z)$ should all be inside the unit circle. Depending on the zero locations of $\hat{G}_p(z)$ and the employed adaptive control strategy, different feedforward controller design strategies are followed to cover every possible case that can be encountered in transient test cycle implementation. These will be explained in following sections.

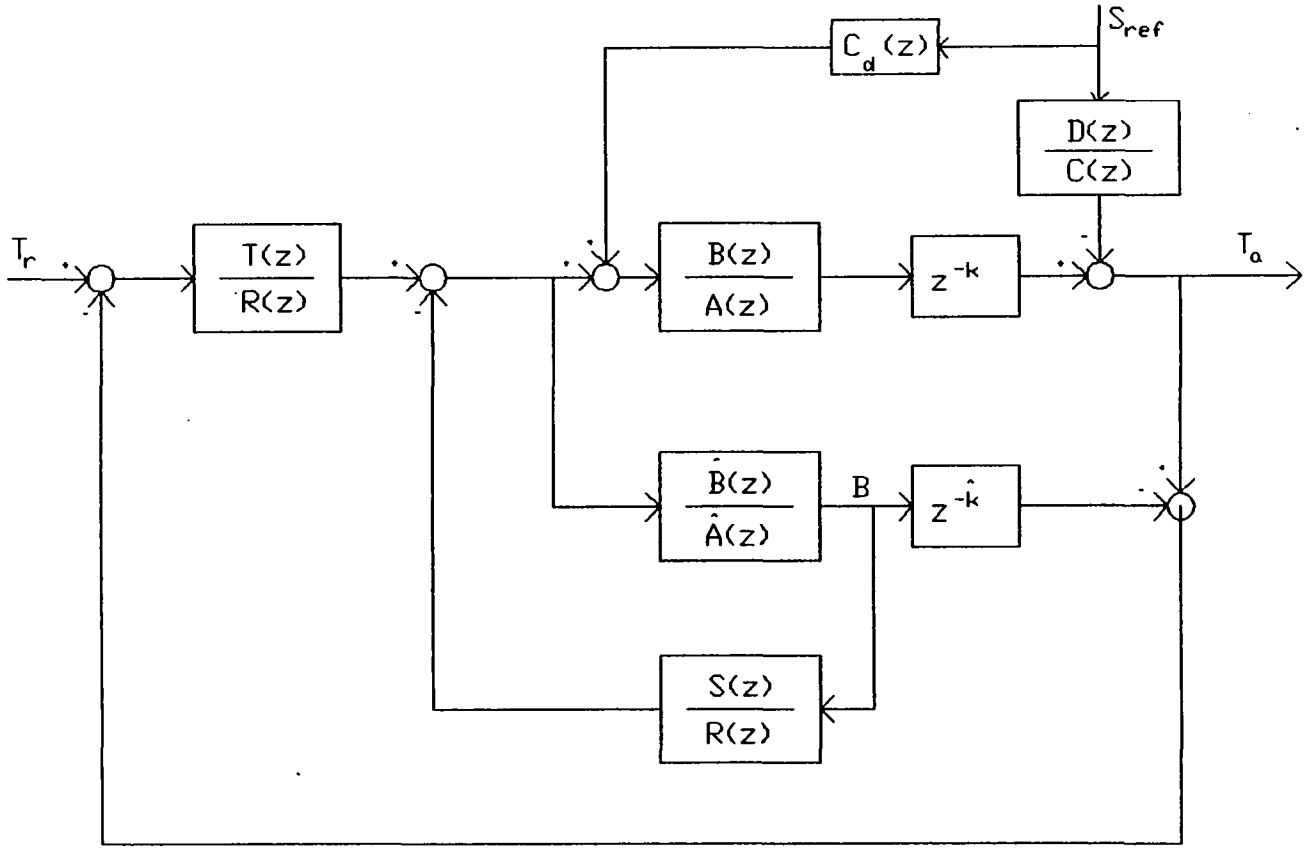


Figure 5.5: Smith predictor with feedforward compensator

5.5.1 One-shot feedforward controller design

As discussed in Chapter 4, six different "s" domain models were considered for both reference speed-torque and throttle-torque systems. The "s" domain transfer functions that were obtained are discretized in software to form "z" domain transfer functions using a sampling period of T seconds. Based on these "z" domain transfer functions, a feedforward controller is designed using following criteria:

1. If $G_p(z)$ does not have any zero inside the unit circle, and if the order of $G_d(z)$ is greater than or equal to the order of $G_d(z)$, then a perfect cancellation feedback is obtained by selecting a feedforward compensator of the form:

$$C_d(z) = \frac{\hat{G}_d(z)}{\hat{G}_p(z)} \tag{5.27}$$

2. If $G_p(z)$ has a zero which is located inside the unit circle then a cancellation feedback is not possible. In this case, the discretization of $\hat{G}_d(s)/\hat{G}_p(s)$ is taken as an approximation to the feedforward control design problem. Since the zeros of $\hat{G}_p(s)$, if any, lie on the right half plane of the "s" domain, the discretization of $\hat{G}_d(s)/\hat{G}_p(s)$ always results in the poles of the feedforward compensator being inside the unit circle, allowing a stable compensator. Therefore, in these cases the feedforward compensator is designed as:

$$C_d(z) = \Xi\left[\frac{\hat{G}_d(s)}{\hat{G}_p(s)}\right] \tag{5.28}$$

   where Ξ represents z-transform operator with zero-order-hold.

3. If the order of $\hat{G}_p(z)$ is higher than the order of $\hat{G}_d(z)$, then cancellation feedback is not realizable. In these cases the order of the $\hat{G}_d(s)$ can be increased

with extra dynamics which are much faster than the dynamics of $G_d(s)$. Thus the model still keeps its accuracy to describe the system, yet now it is suitable for feedforward compensator design. If the increased order model of $\hat{G}_d(s)$ is shown as $\hat{G}_d^*(s)$, then the feedforward compensator is designed either as:

(a)

$$C_d(z) = \frac{\hat{G}_d^*(z)}{\hat{G}_p(z)} \tag{5.29}$$

if $\hat{G}_p(z)$ has all of its zeros inside the unit circle, or,

(b)

$$C_d(z) = \Xi\left[\frac{\hat{G}_d^*(s)}{\hat{G}_p(s)}\right] \tag{5.30}$$

if $\hat{G}_p(z)$ has any of its zeros outside the unit circle.

Implementing a continuous-time parameter estimation algorithm thus provides a significant advantage in designing a feedforward compensator for the system. It provides the design of a stable feedforward compensator even if the process has unstable zeros in the "z" domain or if the original feedforward compensator is not realizable.

Discretization of the continuous-time transfer functions has been carried out in software through the following methods:

- For systems up to third order models, exact discretization is carried out (zero-order-hold is included) using z-transform tables loaded into software.
- For systems higher than third order models, the Tustin approximation is used. In the Tustin approximation, the z-domain transfer function (with zero-order-hold sampling) is obtained by $s = \frac{2}{T}\frac{z-1}{z+1}$ substitution in the s-domain transfer functions.

5.5.2 Adaptive feedforward compensator with continuous update of process parameters

When updating the parameters of the process at each sampling instant, the parameters of the feedforward compensator as well as the parameters of the pole-placement controller are updated at each sampling point based on the process parameters. The zero locations of $\hat{G}_p(z)$ are checked at each update. If the zeros of $\hat{G}_p(z)$ are not outside the unit circle, then a perfect cancellation feedforward controller of $C_d(z) = \hat{G}_d(z)/\hat{G}_p(z)$ is applied. If any zeros of $\hat{G}_p(z)$ lies inside the unit circle, then the feedforward compensator parameters are switched to a safe controller whose parameters are found by using $\hat{G}_p(z)$ and $\hat{G}_d(z)$ from Eq. 4.35. So the "safe feedforward compensator" becomes:

$$[C_d(z)]_{safe} = \frac{0.06464z + 0.0503}{0.0879z + 0.0684} \tag{5.31}$$

5.6 Implementation Issues

Although the use of adaptive control is very appealing in many situations, there are several key issues that need to be taken care of in practice. In the idealized environment of simulations it is easy to get different types of adaptive algorithms to perform exceptionally well. However, in practice, the situation can be quite the opposite. The reason for this is that in practical situations there are all kinds of violations of the conditions of the theory. The self-tuning controller must be able to handle nonlinearities, unmodeled dynamics, and unmodeled disturbances over a wide range of operating conditions. Some of the different aspects of these implementation issues are given in this section.

- **Estimator implementation:** As mentioned in Chapter 4, a continuous-time parameter estimation algorithm was used for this study. Robustness and immunity to noise were two of the desired features of this algorithm. In one-shot parameter estimation the large amplitude input signals are used to improve the numerical conditioning. To increase the numerical accuracy of the estimator it is usually desirable in practice to have the input and output signals in the same amplitude range. Since the input and output signal ratios in the throttle-speed-torque control system are close to one, normalization is not required.

  Although an on-line batch least-squares method can be applicable for one-shot parameter estimation, a recursive least squares algorithm is used to show the applicability of the PMF method to adaptive control systems. The recursive least square estimator is described by the following equations [35]:

$$\begin{aligned} \theta(k) &= \theta(k-1) + P(k)\varphi(k)\varepsilon(k) \\ P(k) &= \left[P(k-1) - \frac{P(k-1)\varphi(k-1)\varphi^T(k-1)P(k-1)}{\lambda + \varphi^T(k-1)P(k-1)\varphi(k-1)} \right] /\lambda \end{aligned} \tag{5.32}$$

  where θ is a vector consisting of the parameters to be estimated, φ is a vector of delayed inputs and outputs, and ε is the prediction error. P is proportional to the covariance matrix of the estimation error and λ is an exponential forgetting factor to allow tracking of time varying parameters. However, updating of the P covariance matrix is not well conditioned from a numerical point of view. In our research, the system identification and parameter estimation was first carried out using the above update formula for P. Although calculations were carried out in double precision, convergence of the parameters was not found to be satisfactory. To overcome the problem, the $U - D$ factorization by Bierman

and Thornton [53] was used. Simulation results employing $U - D$ algorithm for the P matrix update showed satisfactory convergence results. The data for this simulation was obtained from the transient tests performed with the engine and dynamometer pair used in this study.

In one-shot parameter estimation algorithms, the forgetting factor λ is set equal to one. This means that all the data in the step-response test is weighted equally in the parameter estimation algorithm. However, during the continuous update of parameters, a variable forgetting factor due to Ydstie et al. [54] was employed. The recursive equations to update the forgetting factor, λ, are given by:

$$\lambda(k) = \frac{1}{2}\left[n(k) + \sqrt{(n(k))^2 + 4w(k)}\right] \tag{5.33}$$

where

$$n(k) = 1 - w(k) - \frac{(\varepsilon(k))^2}{\sigma_O} \tag{5.34}$$

$$w(k) = \varphi^T(k-1)P(k-1)\varphi(k-1) \tag{5.35}$$

and where σ_O is the tuning parameter, in the order of 0.1 to 10 [35]. A large value of σ_O gives small adaptation. A small value of σ_O , on the other hand, results in fast adaptation but at the cost of larger parameter uncertainity. In the experimentation in this research, σ_O was set to 0.5. The forgetting factor calculated by the above equation will be close to unity if the process is not excited or if the parameter vector θ is close to its correct value. Hence, the problems associated with the continuous increase of the parameters of the P matrix, also called "estimator windup", can be effectively dealt with. Although

the use of the variable forgetting factor helps to avoid the estimator windup, it does not guarantee that P stays bounded. Therefore, as a precaution, updating of the parameters and the covariance matrix are stopped whenever the process excitation is low or the estimation error is sufficiently small.

In continuous adaptation of the process parameters, a range of acceptable parameters was given to the algorithm. This range in our experimentation was between the values of 2 and 7 for all parameters. Although the actual variations the system parameters were less than the range given above, as explained in Section 6.2, it was felt that providing a wider range could prevent the estimation algorithm to from frequently switching to the values given in Eq. 4.35.

In continuous adaptation of process parameters, a problem particular to our system was also observed. Input commands (serial line throttle valve commands) less than or equal to 5 were sometimes not transferred to the throttle valve actuator due to an error in the RS-232 data communication line. It caused severe problems in the parameter estimation algorithm since no actual input is applied to the system although the computer assumes input signals are applied. To work around this problem, the input commands of the actuator were set to zero whenever the calculated absolute value of the throttle valve input command is less than 5. However, controlling torque values close to their desired trajectories becomes very difficult since input commands with small magnitudes are not allowed in the closed-loop system. In adaptive control with one-shot estimation and controller design, these problems do not exist due to the nature of the problem.

- **Sampling rate selection:** The choice of sampling period is discussed in great detail by Astrom and Wittenmark [43]. The sampling rate for the estimation algorithm used in this study was set to 50 Hertz. Unlike discrete-time parameter estimation, using a high sampling rate will improve the performance of the continuous time parameter estimation algorithm. This can be noted as another advantage of continuous-time estimation. It is suggested in Astrom and Wittenmark that the sampling rate for a closed-loop control system should be chosen to provide two to three samples during the rise time. In Section 5.3 it was mentioned that the desired closed-loop system should have a time constant of 0.25 second. This yields a rise time of about 0.6 second. Based on this rise time a sampling rate around 5 Hertz seems reasonable for our closed-loop system. Another consideration for the selection of sampling period is the amount of time delay in the system. In order to get full use of the Smith predictor, the ratio of the sampling time to time delay should be an integer number. For continuous update of parameters, the time delay was found to be 0.4 seconds from previous experiments, and therefore a fixed sampling period of 0.2 seconds was selected. However, for the one shot estimator and controller design algorithm, the time delay estimation was carried out on-line. Therefore, the sampling period was also calculated on line, and allowed to lie in the region of 0.15 to 0.30 seconds, depending on the value of the estimated time delay. This was another flexibility that the one-shot adaptive control algorithm provides. A limitation for the maximum sampling rate came from the actuator hardware. It was not possible to send commands to the actuator microprocessor at a rate higher than 7 Hertz. However, this did not cause a severe limitation in the

sampling rate selection.

- **Anti-aliasing filter:** To avoid aliasing problems it is necessary to use an analog prefilter to eliminate disturbances with frequencies higher than the Nyquist frequency associated with the sampling rate. High frequency signals may otherwise be mis-interpreted as low frequency signals and may introduce disturbances in the closed loop system. The bandwidth, ω_B, of the prefilter is suggested to be inversely proportional to the sampling period. A rule of thumb is given as $\omega_B T \cong 0.5 - 1$ [38]. There are four user selectable analog prefilters on the analog torque sensor of the engine and dynamometer system used for this study. The bandwidths of these filters are found as 1.45, 6.5, 200, and 2000 rad/sec, respectively. Selecting the filter with bandwidth of 1.45 rad/sec is sufficient in our application considering a sampling period of 0.15–0.30 seconds.

- **Antireset windup:** Since the pole assignment algorithm includes an integrator, reset windup can occur if the output saturates and the controller continues to integrate the error. Therefore, an antireset windup algorithm is employed in the software. It is done by stopping the updating of the integrator when the controller output is limited.

5.7 Software Description

Three separate Fortran programs were written to cover the self-tuning control with one-shot estimation and controller design, the self-tuning gain-scheduling control with multiple one-shot estimation and controller designs, and the adaptive regulator with continuous update of process and controller parameters. To improve the

readability of the software, the programs are composed of subroutines where each subroutine performs a specific task. The program listings for the self-tuning algorithm with one-shot estimation and controller design, and for adaptive regulator are given in Appendix C. Organization of the software is shown in Figure 5.6. Each subroutine is described briefly below.

- **Main program:** Communicates with subroutines, reads torque and speed trajectories from an external data file, determines the sampling period based on the estimated time delay, performs some safety check-ups, performs closed-loop torque control and provides reference speed trajectories for the General Electric speed controller, and stores the torque and speed readings during the transient test cycle and writes them to an external data file.

- **Subroutine Start:** Starts-up the engine and dynamometer. Brings engine speed to idle speed (1200 rpm), and slowly moves the throttle valve until a reading of 0 ft-lbs is achieved.

- **Subroutine Stop:** Shuts-down the engine and dynamometer pair. First brings the throttle valve to closed-rack position, then brings the engine to a full stop (0 rpm).

- **Subroutine Ident:** Performs two open-loop step response tests for the identification of reference speed-torque and throttle-torque systems. Takes data at 50 Hertz sampling rate for 5 seconds during open-loop tests, and stores them to be used later in the parameter estimation algorithm.

- **Subroutine Parest:** Performs system and parameter identification for both speed-torque and throttle-torque systems. Selects six different models and tries

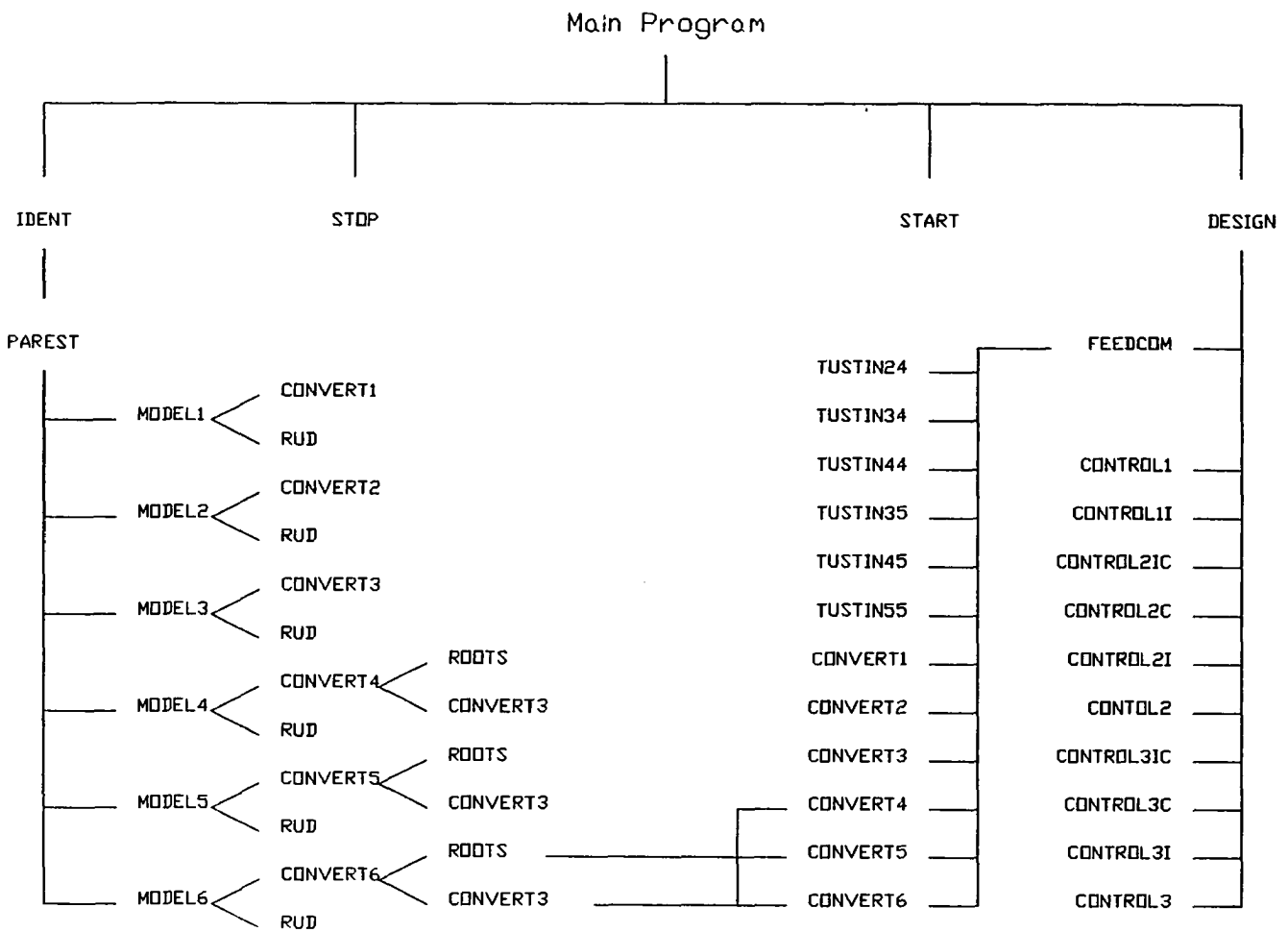


Figure 5.6: Organization of the software for self-tuning control algorithm with one-shot estimator and controller design

to find the best model that gives minimum sums of squares of error. Time delay estimation is also included as a part of the identification problem.

- **Subroutine Model1–Model6:** Using a recursive least squares approach coupled with the PMF method, these subroutines tried to find the parameters of the assumed models and time delays. Return sums of squares of errors for each model. Considered models are:
  - First order system (Model1)
  - Second order system (Model2)
  - Second order system with first order numerator dynamics (Model3)
  - Third order system (Model4)
  - Third order system with first order numerator dynamics (Model5)
  - Third order system with second order numerator dynamics (Model6)
- **Subroutine Convert1–Convert6:** This set of subroutines discretizes the "s" domain transfer functions to obtain the "z" domain description of the system. Discrete-time transfer functions are needed in the pole-assignment controller and feedforward compensator design.
- **Subroutine Tustin24-34-44-35-45-55:** Discretize the "s" domain transfer functions to obtain the "z" domain description of the system when the order of the system is greater than three. The first number is the numerator order and the second number is the denominator order of the "s" domain transfer function. It uses the Tustin approximation to discretize the continuous-time system.

- **Subroutine Design:** This subroutine communicates with the user. It gives information about the parameter estimation results, and asks the user which controller strategy should be applied. The user specifies the desired closed-loop response, decides whether an integrator will be included as part of the controller, and decides whether zero cancellation is desired or not.

- **Subroutine Feedcom:** This subroutine designs the feedforward compensator. It first tries to develop a perfect cancellation compensator. If it is unstable or unrealizable then it uses different approximations as explained in Section 5.5.

- **Subroutine Control1-1i-2ic-2i-2c-2-3ic-3i-3c-3:** This set of subroutines performs the pole assignment controller design. It returns $S(z)$, $T(z)$, and $R(z)$ to the main program. The numeric number shows the order of the selected discrete-time throttle-torque system. Existence of 'i' indicates that integral action is included as a part of the design. Existence of 'c' indicates that open-loop zero cancellation is desired.

- **Subroutine Roots:** This subroutine finds the roots of a third order polynomial. It is used in subroutines Convert3, Convert4 and Convert5, and in monitoring the convergence of the eigenvalues.

- **Subroutine Rud:** This subroutine performs recursive least-squares identification. It updates the system parameters and the covariance matrix at each sampling point. It employs $U - D$ factorization for the covariance matrix update.

- **Subroutine Forget:** This subroutine calculates the forgetting factor recursively.

6. RESULTS AND DISCUSSION

The three proposed adaptive control algorithms were implemented on a Zenith-386 microcomputer. Ten transient test cycles were run with each algorithm to validate the statistical properties of the results. These results are presented and discussed in this chapter. Section 6.1 to 6.3 present the results for the adaptive control strategies considered in this research. Section 6.4 compares the adaptive control approaches and comments on their advantages and disadvantages. Section 6.5 compares the conventional constant-parameter non-adaptive control approach to the adaptive approaches studied in this research. Section 6.6 discusses the application of the proposed algorithms to other systems.

6.1 Self-Tuning Control with One-Shot Parameter Estimation and Controller Design

As explained in the previous sections, in this method the complete system identification, including time-delay estimation, and controller and feedforward compensator designs were carried out on-line prior to the transient test cycle.

For the John Deere diesel engine and General Electric DC Dynamometer used in this study, step change input signals of 1/8 inch movement of throttle rack and 100 rpm change of reference speed command, were applied when the system was running

under idle conditions of zero torque and 1200 rpm. Table 6.1 gives the estimated parameters of the throttle-torque system for the six models considered, as discussed in Section 4.4, along with the estimated time delay T_f for a particular run. In these models, the parameters a_1, a_2, and a_3 represent the denominator coefficients, and the parameters b_1, b_2, and b_3 represent the numerator coefficients of the open-loop throttle-torque transfer function in the "s" domain. As can be seen from Table 6.1, case 5, which is a third order model with first order numerator dynamics, gives the minimum sum of squares. However, cases 2 and 3, second order models with and without numerator dynamics, also describe the throttle-torque model well since their sum of squares of error values are very comparable to case five. Table 6.2 shows the zero and pole locations of the throttle-torque open-loop transfer function that corresponds to the models of Table 6.1. Table 6.2 suggests that one complex conjugate pole is dominant in the open-loop system response in all models, except the first order model, providing a damping ratio of approximately 0.9 and a natural frequency of approximately 2.1 rad/sec. Figure 6.1 compares the actual load torque output with the estimated third order model output for a step input of 1/8 inch movement of throttle rack while the speed reference command is kept at 1200 rpm. It can be seen in Figure 6.1 that the model closely estimates the actual input-output behavior.

Table 6.3 gives the estimated parameters of the speed-torque system for each model considered along with the estimated time delay T_d for a particular test. Case 5, which is a third order model with first order numerator dynamics, again gives the minimum sum of squares among all 6 cases considered. Again cases 2, 3, and 6 are also acceptable candidates for the speed-torque process description. Figure 6.2 compares

Table 6.1: Estimated parameters of throttle-load torque system for 6 different models

Case	n[a]	T_f	a_1	a_2	a_3	b_1	b_2	b_3	SSE[b]
1	1	0.60	1.29	—	—	1.81	—	—	2502
2	2	0.42	3.86	4.69	—	6.29	—	—	355
3	2	0.42	3.68	4.45	—	0.06	5.94	—	351
4	3	0.38	20.22	71.87	85.45	112.04	—	—	1267
5[c]	3	0.38	4.16	8.24	5.18	4.73	7.00	—	336
6	3	0.52	5.76	16.94	18.07	1.05	5.07	23.96	682

[a]Model order.
[b]Sum of squares of errors between actual output and model output.
[c]Selected model.

Table 6.2: Estimated pole-zero locations of throttle-load torque system for 6 different models

Case	Pole locations	Zero locations
1	-1.29	—
2	$-1.93 \pm 0.98j$	—
3	$-1.84 \pm 1.03j$	-99
4	-16.0, $-2.07 \pm 1.01j$	—
5	-1.03, $-1.59 \pm 1.56j$	-1.47
6	-1.86, $-1.95 \pm 2.42j$	$-2.41 \pm 4.25j$

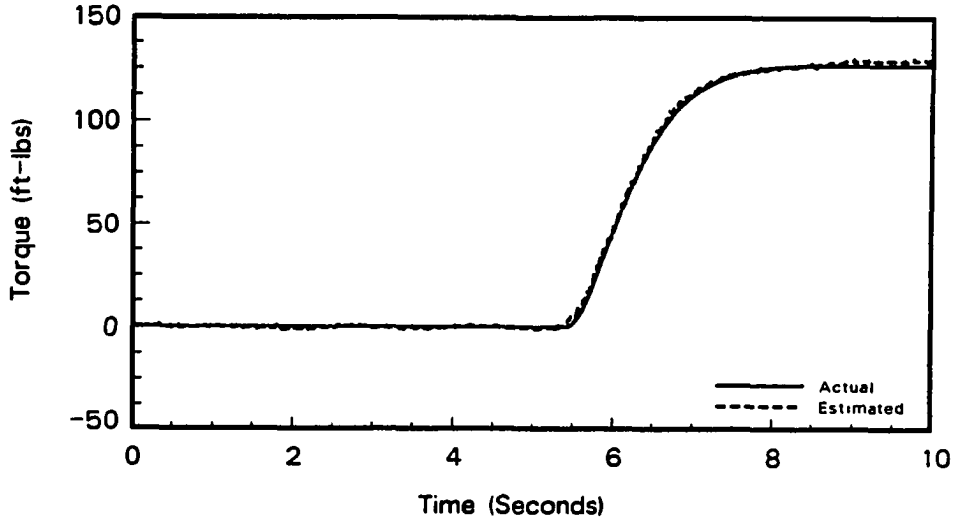


Figure 6.1: Comparison of actual and predicted responses due to a step input of the throttle valve position

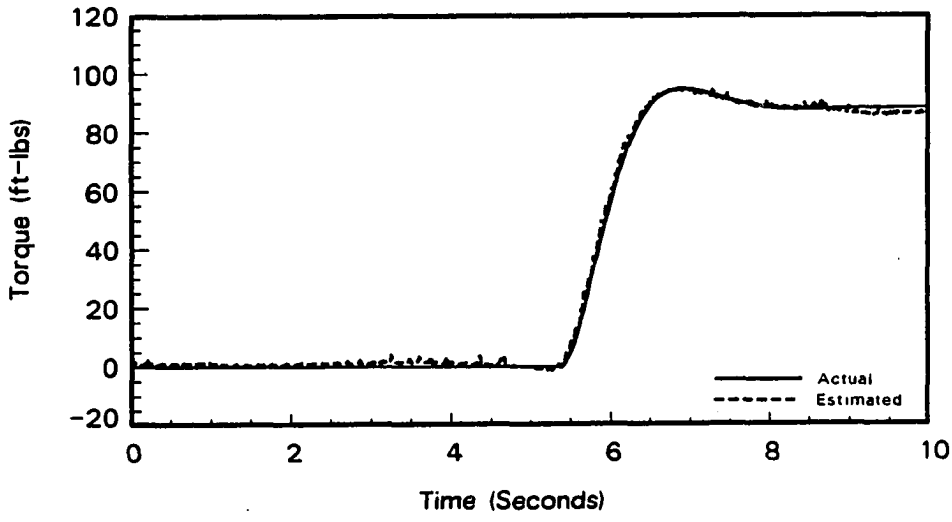


Figure 6.2: Comparison of actual and predicted responses due to a step input to the speed reference command

the actual load torque output with the estimated third order model output for a step input of 100 rpm change in the speed reference command while the throttle rack position is kept constant. As seen in Figure 6.2, the third order model very closely approximates the actual open-loop speed-torque system. Table 6.4 shows the zero and pole locations of the speed-torque open-loop transfer function that corresponds to the models of Table 6.3. Table 6.4 suggests that one complex conjugate pole is dominant in the open-loop system response in all models, except the first order model, providing a damping ratio of approximately 0.7, and a natural frequency of approximately 2.7 rad/sec. Figure 6.2 compares the actual load torque output with the estimated load torque output for a step change of 100 rpm speed reference command while throttle rack position is kept constant. Both Figures 6.1 and 6.2 show that the continuous-time PMF parameter identification method yields system parameters which accurately describe the input-output relation, despite the noise in the torque measurements. In both throttle-torque and speed-torque models a first order system approximation was not found suitable due to the very large sum of squares between actual system output and model output.

This particular control technique offers the option of either cancelling the process zeros or not cancelling the zeros during the controller design. Ten EPA transient cycle tests were conducted for each case so that the performance could be compared for each option. All the transient test cycles, when open-loop process zeros were included in the desired closed-loop transfer function, were found to be valid according to the Environmental Protection Agency (EPA) specifications. However, only six of the ten transient test cycles were found to be valid under the EPA regulations when zero cancellation was employed in the controller design. Table 6.5 and Table 6.6

Table 6.3: Estimated parameters of reference speed-load torque system for 6 different models

Case	n[a]	T_d	a_1	a_2	a_3	b_1	b_2	b_3	SSE[b]
1	1	0.60	2.51	—	—	2.22	—	—	3112
2	2	0.42	3.55	7.26	—	6.21	—	—	224
3	2	0.44	3.59	7.37	—	0.10	6.31	—	225
4	3	0.34	13.32	41.54	75.32	63.42	—	—	548
5[c]	3	0.40	3.49	8.18	3.23	5.32	2.89	—	149
6	3	0.44	4.01	10.03	7.66	0.26	5.29	6.63	154

[a]Model order.
[b]Sum of squares of errors between actual output and model output.
[c]Selected model.

Table 6.4: Estimated pole-zero locations of speed-load torque system for six different models

Case	Pole locations	Zero locations
1	-2.51	—
2	$-1.77 \pm 2.03j$	—
3	$-1.79 \pm 2.04j$	-63.1
4	-9.98, $-1.71 \pm 2.16j$	—
5	-0.48, $-1.50 \pm 2.11j$	-0.54
6	-1.13, $-1.44 \pm 2.16j$	-1.29, -19.05

give average values of the regression line parameters for each of the two cases listed above. In addition to the mean values, the standard deviations of the parameters are also listed in these tables. Standard error of estimate (SE), slope (m), coefficient of determination (r^2), intercept (b), and their standard deviations ($\sigma_{SE}, \sigma_m, \sigma_{r2}$, and σ_b) are given in these tables for the speed, torque, and brake horsepower (BHP) regression line analyses.

It is desired that the slope (m) and coefficient of determination (r^2) be as close as possible to one, and the standard error of estimate (SE) and intercept (b) should be as close as possible to zero for a perfect transient test cycle. By direct comparison of Table 6.5 and Table 6.6, it can be said that better results were obtained when the process zeros were not cancelled. Approximately 30-35% improvements on the slope and coefficient of determination values of the torque and brake horsepower regression line were obtained when the process zeros were cancelled. This is due to the fact that canceling underdamped process zeros caused ringing in the input signal and also required input signals with higher amplitudes. Therefore, any mismatch between actual system parameters and estimated model parameters gave rise to a less-stable system. For example, in a sample transient test cycle, the following throttle-torque plant description was obtained:

$$\frac{B(z)}{A(z)} = \frac{0.081z + 0.064}{z^2 - 1.389z + 0.502} \tag{6.1}$$

The throttle-torque open-loop system shown above contains a zero located at $z = -0.79$, which can be considered as an underdamped zero. In the zero cancellation case, the desired closed-loop transfer function , as suggested in Section 5.5, becomes:

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^2 z^2}{(z-a)^2} \tag{6.2}$$

Table 6.5: Regression line values for one-shot self-tuning control without zero cancellation

	Speed	Torque	BHP
SE,σ_{SE}	35.6 rpm, 0.26 rpm	18.2 ft-lb, 1.01 ft-lb	6.0 BHP, 0.42 BHP
m, σ_m	0.996, 0.0002	0.960, 0.0086	0.950, 0.0090
r^2, $\sigma_r 2$	0.991, 0.0001	0.923, 0.0092	0.930, 0.0100
b, σ_b	3.20 rpm, 0.38 rpm	-1.66 ft-lb, 0.28 ft-lb	-0.34 BHP, 0.08 BHP

Table 6.6: Regression line values for one-shot self-tuning control with zero cancellation

	Speed	Torque	BHP
SE,σ_{SE}	35.6 rpm, 0.27 rpm	22.3 ft-lb, 6.06 ft-lb	7.4 BHP, 2.03 BHP
m, σ_m	0.996, 0.0001	0.943, 0.0210	0.940, 0.0184
r^2, $\sigma_r 2$	0.991, 0.0001	0.880, 0.0627	0.893, 0.0560
b, σ_b	4.61 rpm, 0.43 rpm	-2.47 ft-lb, 1.06 ft-lb	-0.67 BHP, 0.37 BHP

where $a = e^{-T/\tau}$, $\tau = 0.25$ second, and $T = 0.2$ second. The following controller polynomials were obtained following the steps of the pole-zero assignment algorithm given in Section 5.3:

$$R_c(z) = z^2 - 0.205z - 0.795 \tag{6.3}$$

$$S_c(z) = 10.52z^2 - 13.64z + 4.55 \tag{6.4}$$

$$T_c(z) = 2.93z^2 - 1.51z \tag{6.5}$$

where 'c' subscript refers to the zero cancellation design. The above controller was implemented and the resulting transient test cycle did not satisfy the EPA regression line specifications. Figure 6.3a shows the computer simulation of the response of the closed-loop system to a reference input with a magnitude of one when the open-loop process zeros were cancelled in the controller design. Figure 6.3b shows the controller output signals (input signals to the throttle actuator) in this simulation.

Second, the controller design was carried out when zero cancellation was avoided. When process zeros are not cancelled, the desired closed-loop transfer function becomes:

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^2}{B(1)} \frac{B(z)}{(z-a)^2} \tag{6.6}$$

where $B(z)$ and a have the same values as the zero cancellation case. Following the pole-zero assignment algorithm the controller polynomials were found to be:

$$R(z) = z^2 - 0.934z - 0.066 \tag{6.7}$$

$$S(z) = 3.348z^2 - 4.558z + 1.597 \tag{6.8}$$

$$T(z) = 1.635z^2 - 1.678z + 0.430 \tag{6.9}$$

The above controller was implemented and the resulting transient test cycle satisfied

the EPA specifications. Figure 6.4a shows the computer simulation of the response of the closed-loop system to a reference input with a magnitude of one when the process zeros were not cancelled in the controller design. Figure 6.4b shows the controller output signals (input signals to the throttle actuator) in this simulation.

When the coefficients of $S(z)$ and $T(z)$ are compared we can see that the zero cancellation case produces controller polynomials with larger coefficients. Comparison of Figure 6.3 and Figure 6.4 revealed that cancellation of the underdamped zero from the desired closed-loop transfer function caused ringing and larger amplitudes in the input signal. From the reasons discussed above, the zero cancellation in the pole-assignment algorithm was avoided in the rest of the tests with other control approaches.

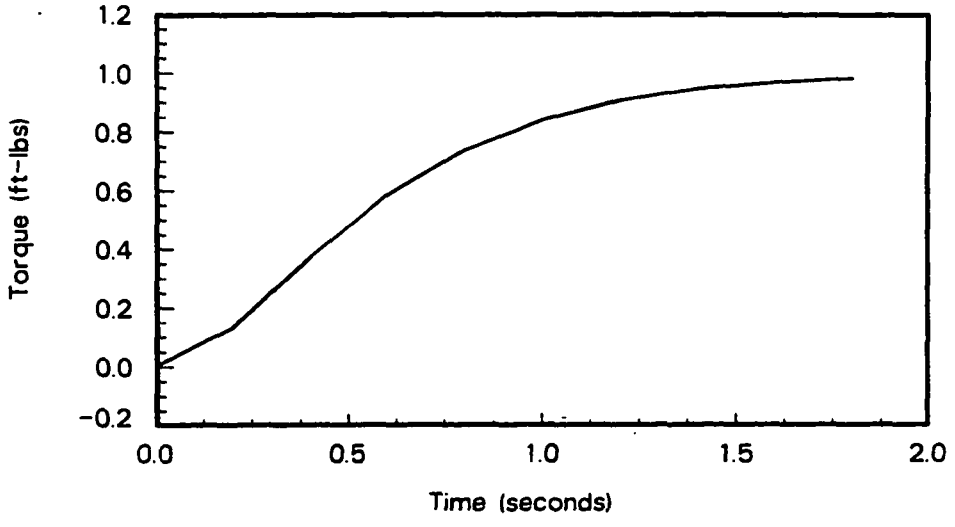
The observer polynomial zero locations were determined using the new pole-zero assignment algorithm described in Section 5.3. It was found that the observer zeros were located near to the closed-loop poles and varied from $a_O = 0.3$ to $a_O = 0.5$. To show the importance of this modified pole-assignment algorithm the following throttle-torque plant description, obtained on-line in a transient test cycle by the PMF algorithm, is provided as an example.

$$\frac{B(z)}{A(z)} = \frac{0.075z^2 + 0.00236z - 0.049}{z^3 - 2.245z^2 + 1.717z - 0.450} \tag{6.10}$$

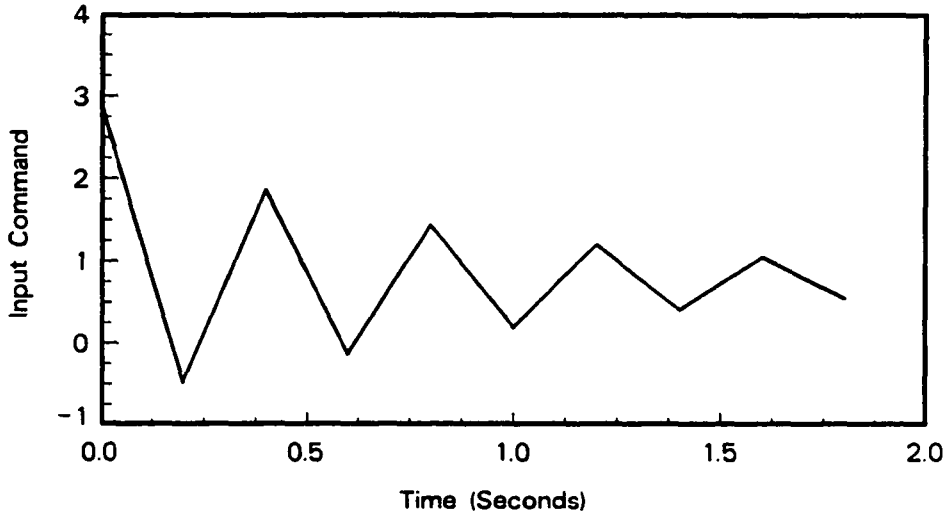
The following desired closed-loop transfer function was selected:

$$\frac{B_m(z)}{A_m(z)} = \frac{(1-a)^3 z^3}{(z-a)^3} \tag{6.11}$$

where $a = e^{-T/\tau}$, $T = 0.2$ second, and $\tau = 0.25$ second. First, all observer poles were intentionally placed at the origin, as suggested in most of the adaptive control literature, to give an observer polynomial of $A_O(z) = z^3$. Following the pole-assignment

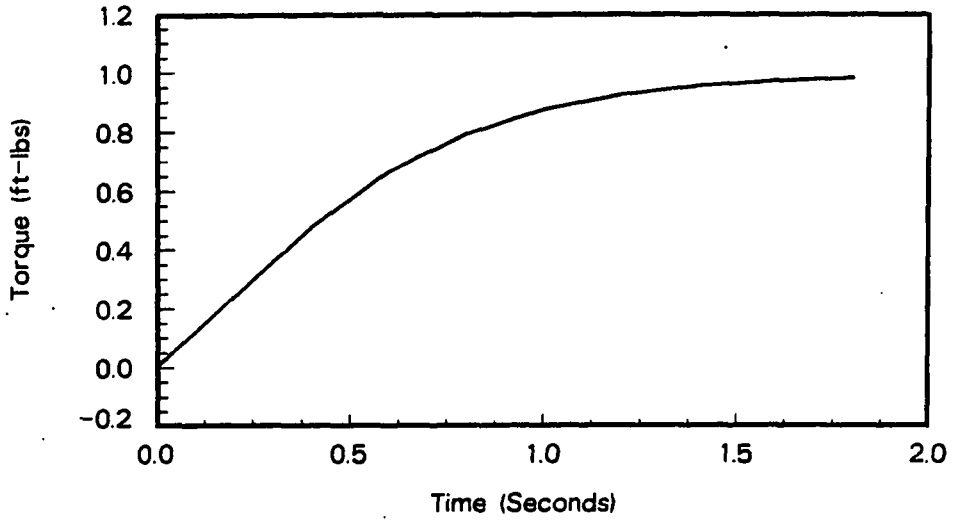


a

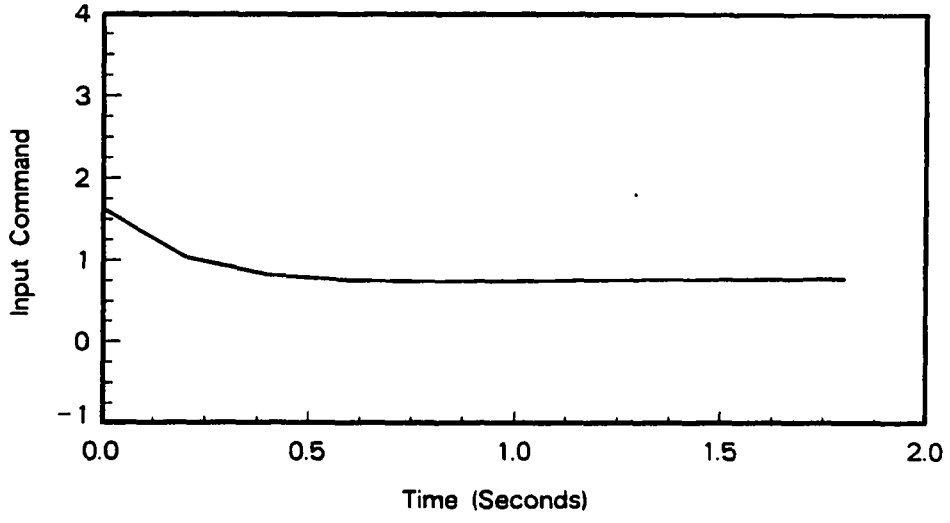


b

Figure 6.3: Step response of the closed-loop torque control system: process zeros were cancelled



a



b

Figure 6.4: Step response of the closed-loop torque control system: process zeros were not cancelled

controller design steps of Section 5.3, the parameters of the controller were found to be:

$$R(z) = z^3 + 14.25z^2 - 3.13z - 12.12 \quad (6.12)$$

$$S(z) = -180.55z^3 + 46.16z^2 - 387.59z + 110.66 \quad (6.13)$$

$$T(z) = 4.106z^3 \quad (6.14)$$

The $R(z)$ polynomial shown above contains a root outside the unit circle at $z = -14.4$. Although the controller polynomials shown above produced the stable closed-loop transfer function, shown in Eq. 6.11, when the controller was implemented on the digital computer, the closed-loop system went to unstable operation within seconds of the start of the transient test cycle.

Second, the modified pole-assignment algorithm was applied. In this case, the modified algorithm set all observer zeros to $a_O = 0.42$ and came up with stable controller polynomials of:

$$R^*(z) = z^3 + 0.248z^2 - 0.256z - 0.991 \quad (6.15)$$

$$S^*(z) = -10.69z^3 + 31.73z^2 - 29.08z + 8.85 \quad (6.16)$$

$$T^*(z) = 4.106z^3 - 5.174z^2 + 2.173z - 0.304 \quad (6.17)$$

where * denotes the modified controller polynomials. Now $R^*(z)$ contains all of its roots inside the unit circle. The above controller also produced the same desired closed-loop transfer function. Implementation of this controller resulted in a stable closed-loop operation and satisfied all of the EPA specifications.

All the experiments with the one-shot parameter identification algorithm showed that the parameter convergence was fast. The parameters converged in less than 250

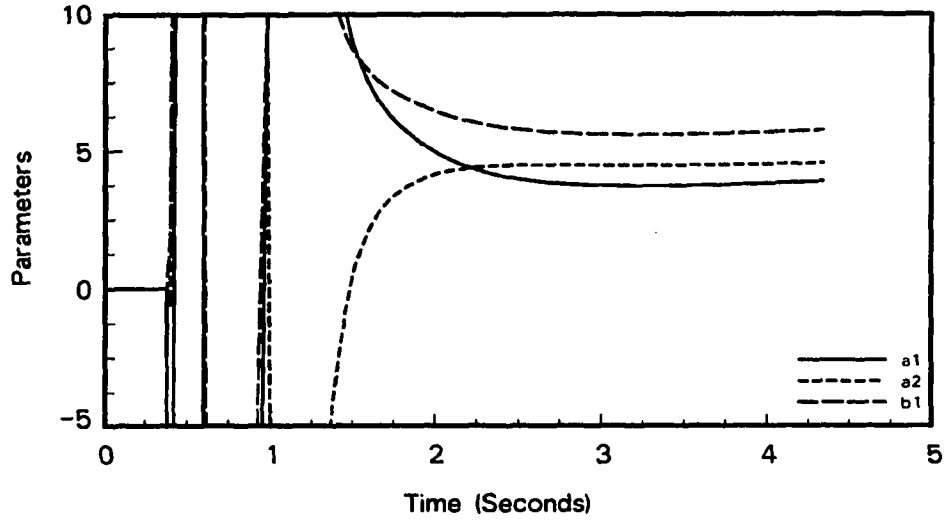


Figure 6.5: Convergence of the second-order model parameters with one-shot self-tuning control

samples. Figure 6.5 shows the convergence of the parameters when a second order model was found to be the best fit. In the first one-second of the estimation the parameters were far away from their converged values, and for clarity, the whole scale for the parameters were not given in Figure 6.5. However, at three seconds, the parameters converged to their final values. The forgetting factor in the recursive least-squares estimation algorithm was set to 1.0 in the parameter convergence plot shown in Figure 6.5.

Of the ten transient cycle tests, a second-order model was selected in eight of them, and a third order model was selected in the remaining two. However, the difference between the sum of square of errors of the second-order and third-order models were not significant, as seen from Table 6.1 and Table 6.3, therefore suggesting

that models higher than a third-order model was not needed. Variation of the model parameters from one test to another was also very small, showing the repeatability of the tests. The estimated time delay in each run was practically constant ranging from 0.36 to 0.44 seconds. The parameters of the selected model in each test is given in Appendix B.

In all ten transient tests the zeros of the $B(z)$ polynomial were inside the unit circle, and therefore a perfect cancellation feedforward compensator was applied in all ten cases. The zero locations of $B(z)$ varied between 0.3 and 0.8 in the z-plane. A significant portion of the software developed in this study was devoted to the development of a feedforward compensator when the zeros of the $B(z)$ polynomial were outside the unit-circle in the z-plain. Therefore, a reduction in the software may be possible for our particular engine and dynamometer pair. However, in applying the algorithm to different systems, that reduction in software may not be suitable due to the possibility that the process zeros could lie outside the unit circle in the z-plane.

To see the importance of the integrator in the controller, the pole-zero assignment algorithm was also carried out without including an integrator in the $R(z)$ polynomial. The controller polynomials based on the process description of Eq. 6.10 became:

$$R(z) = z - 0.07 \tag{6.18}$$

$$S(z) = -0.988z + 1.56 \tag{6.19}$$

$$T(z) = 2.93z - 1.51 \tag{6.20}$$

$R(z)$ in the above equation does not include a zero at $z = 1$, and therefore no integral action is incorporated to the controller. This controller was implemented in

the engine and dynamometer system and the resulting transient test cycle did not satisfy the EPA specifications. An additional four tests, without incorporating an integrator into the controller, were performed and again none of them satisfied the EPA regulations. Therefore, it was concluded that an integrator was required in the controller for successful transient test cycling. The possible reasons why the cases without an integrator failed to give satisfactory results can be listed as:

1. The integrator in the controller greatly reduced the effect of the low-frequency torque measurement noise in the closed-loop torque control system.

2. If there were any unmodeled disturbances acting on the system, and if known load disturbance was not suitably compensated with a feedforward compensator, then the integral action tried to remove the effects of these disturbances from the closed-loop response.

3. Without an integrator, it was not possible to reach zero steady-state error with the pole-assignment algorithm unless the model was perfect.

4. Although the Smith predictor has the advantage of compensating the time-delay effectively, it has a disadvantage that the sensitivity of the closed-loop system to any mismatch between the actual system and the model parameters is higher than a classical feedback control system without a Smith predictor. Therefore, inclusion of the integrator in this case reduced the sensitivity of the overall system to errors in the parameter estimation.

As a conclusion of this section it can be said that the self-tuning control with one-shot estimation and controller design gave satisfactory results all the time when the process zeros were not cancelled and an integrator was included in the controller.

6.2 Self-Tuning Gain-Scheduling Control with Multiple One-Shot Estimation and Controller Design

In gain-scheduling control the parameter identification and controller design were carried out for five different speed ranges as explained in Section 4.4. Different operating conditions were defined based on the speed variations, not the torque variations, since the engine speed was the major nonlinearity as seen in Eq. 3.1. Ten transient test cycles were run. All of these tests satisfied the Environmental Protection Agency (EPA) regression line specifications. In nine of the tests a second order model was found to be the best fit, and in the remaining test a third order model was selected. Table 6.7 gives the average values of the regression line parameters obtained from these ten tests.

Table 6.7: Regression line values for gain-scheduling self-tuning control

	Speed	Torque	BHP
SE,σ_{SE}	35.5 rpm, 0.29 rpm	18.5 ft-lb, 1.22 ft-lb	5.7 BHP, 0.38 BHP
m, σ_m	0.996, 0.0003	0.970, 0.0146	0.977, 0.0115
r^2, $\sigma_{r}2$	0.991, 0.0001	0.922, 0.0113	0.940, 0.0084
b, σ_b	2.6 rpm, 0.58 rpm	-2.7 ft-lb, 0.38 ft-lb	-0.81 BHP, 0.08 BHP

Table 6.8 and Table 6.9 show how parameters of the throttle-torque and speed-torque system are changed based on the different speed regions for one particular test. The time delays of the throttle-torque and speed-torque system were increased with increasing speed range. This was as expected since the describing function of the engine combustion system, as shown in Eq. 3.6, included a variable time delay

as a function of the engine speed. A close to linear change of the denominator parameters (a_1 and a_2) over the operating range, as seen in Tables 6.2 and 6.4, is a good justification for the implementation of a gain-scheduling control for the transient test cycle. Table 6.10 and Table 6.11 indicate the variations of the steady-state gain ($K = b_2/a_2$), the damping ratio ($\xi = a_1/2\sqrt{a_2}$), and the natural frequency ($\omega_n = \sqrt{a_2}$) over the engine speed range for the systems of throttle-torque, and speed-torque, respectively. The steady state gain and natural frequency in both systems increased with increasing engine speed. The damping ratio decreased with increasing speed in the throttle-torque system while it increased with increasing speed in the speed-torque system. Figure 6.6 and Figure 6.7 shows the comparison of the step responses of the open-loop systems for two extreme operating conditions for both throttle-torque and speed-torque systems. As seen in Figures 6.6 and 6.7 there is a considerable difference in the responses of both the throttle-torque and speed-torque systems in two extreme speed ranges. Due to the black-box approach taken in the modeling of the GE dynamometer, a complete physical explanation of the parameter variations given in Tables 6.8–6.11 could not be obtained. However, the following can be said about the variations of the system parameters:

- In Eq. 3.1, the engine speed appears as quadratic in the input side. Therefore increasing engine speed could result in the increasing gain of the speed-torque system.

- In the throttle-torque system, the same movement of the governor speed lever position produces a higher fuel rate to the cylinders in the high speed range. Therefore, the steady state gain is expected to increase with increasing speed.

Table 6.8: Variation of the throttle-torque system parameters over different speed ranges

Speed range	T_f	a_1	a_2	b_1	b_2	SSE[a]
1200-1400 rpm	0.42	3.68	4.28	0.09	5.28	344
1400-1600 rpm	0.46	3.65	4.25	0.19	5.38	357
1600-1800 rpm	0.48	3.62	4.21	0.16	5.50	351
1800-2000 rpm	0.48	3.58	4.18	0.07	5.37	351
2000-2200 rpm	0.52	3.32	3.99	0.14	5.36	335

[a]Sum of squares of error between actual output and model output.

Table 6.9: Variation of the reference speed-torque system parameters over different speed ranges

Speed range	T_d	a_1	a_2	b_1	b_2	SSE[a]
1200-1400 rpm	0.38	3.25	7.07	0.36	6.01	321
1400-1600 rpm	0.42	3.50	6.89	0.12	5.35	169
1600-1800 rpm	0.44	3.52	6.86	0.04	6.16	164
1800-2000 rpm	0.46	3.93	6.25	0.21	6.94	310
2000-2200 rpm	0.46	3.93	5.99	0.08	7.19	368

[a]Sum of squares of errors between actual output and model output.

Table 6.10: Variation of the steady state gain K, damping ratio ξ, and natural frequency ω_n over different speed ranges for throttle-torque system

Speed range	K	ξ	ω_n
1200-1400 rpm	1.23	0.890	2.069
1400-1600 rpm	1.26	0.885	2.061
1600-1800 rpm	1.30	0.882	2.051
1800-2000 rpm	1.28	0.875	2.044
2000-2200 rpm	1.34	0.830	1.997

Table 6.11: Variation of the steady state gain K, damping ratio ξ, and natural frequency ω_n over different speed ranges for speed-torque system

Speed range	K	ξ	ω_n
1200-1400 rpm	0.85	0.611	2.659
1400-1600 rpm	0.87	0.666	2.624
1600-1800 rpm	0.89	0.672	2.619
1800-2000 rpm	1.11	0.786	2.501
2000-2200 rpm	1.20	0.803	1.442

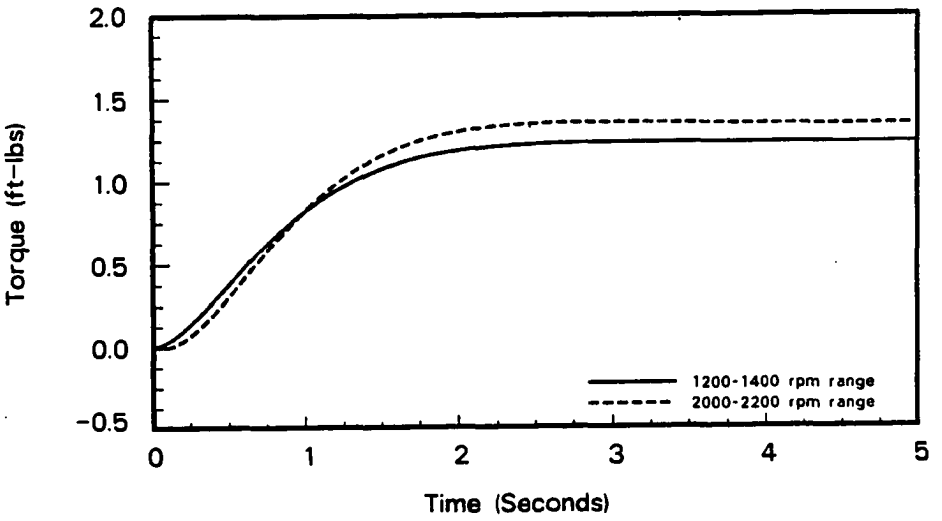


Figure 6.6: Comparison of the step response of the throttle-torque system for two extreme operating conditions

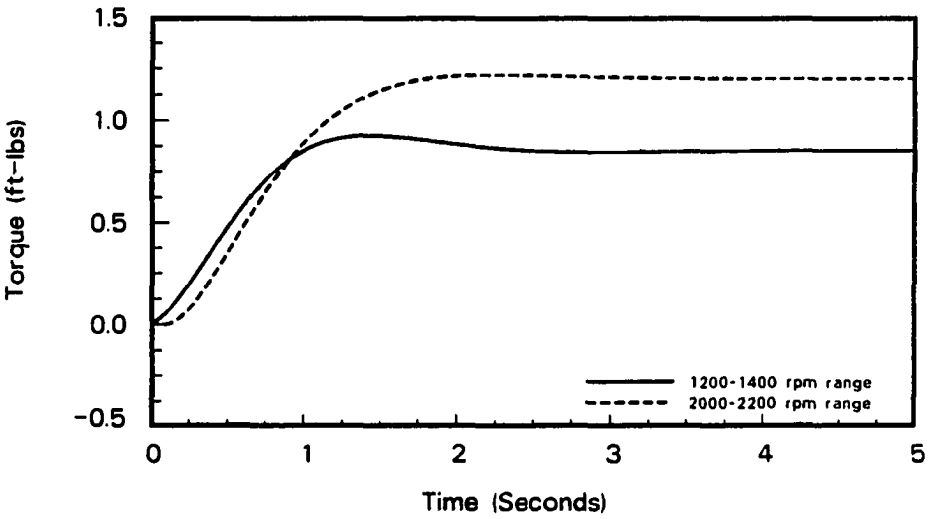


Figure 6.7: Comparison of the step response of the speed-torque system for two extreme operating conditions

- The parameters of the describing equation of governor shown in Eq. 3.1 were assumed to be constant. However, parameters like the viscous friction coefficient and total mass referred to the rotation axis in Eq. 3.1 are actually variable with engine speed. Those variations and the control law in the GE dynamometer controller could result in the variations of the throttle-torque and speed-torque systems.

It will be shown later in this chapter that a constant-parameter controller can cope well with these variations in system dynamics, and therefore could provide valid transient test cycles without difficulty. However a gain-scheduling or a full adaptive regulator may provide better results than a constant parameter non-adaptive controller since the controller parameters will be adjusted based on the estimate of the process parameters. The EPA, on the other hand, tightens its emissions regulations periodically, and with future regulations it could be possible that a constant parameter non-adaptive controller may not provide sufficiently consistent results to reliably measure the engine emissions. If that becomes the case, then the application of the gain-scheduling or full adaptive regulator would be an alternative to non-adaptive torque control.

All the observations explained in Section 6.1 were also valid for the gain-scheduling control. As in Section 6.1 the gain-scheduling control gave best results when an integrator was included as a part of the controller and when the open-loop process zeros were included in the desired closed-loop transfer function. However, to avoid abrupt parameter changes from one operating condition to another, a straight line-interpolation was applied. It was observed that the regression line parameters were closer to the EPA specified values when straight line-interpolation was applied.

When the gain-scheduling control was employed, All the transient tests satisfied the EPA specifications. Since the gain-scheduling self-tuning control updates the controller parameters based on the variations in system dynamics, it is expected that it should give regression line parameters closer to the EPA specifications than the one-shot self-tuning control. Comparisons of these two approaches will be presented later in this chapter.

6.3 Adaptive Regulator with Continuous Update of Process and Controller Parameters

As explained in Section 4.5, a second order model with a time delay of 0.40 seconds was selected for the adaptive controller with continuous update for both the throttle-torque and speed-torque systems. Ten transient test cycles were performed, but only six of the transient test cycles satisfied the EPA specifications. Table 6.12 shows the average values of the regression line parameters obtained from these ten test cycles. Figure 6.3 shows how parameters of the system varied during the whole transient test cycle. In the implementation of the adaptive regulator, whenever the obtained system parameters were not within the prespecified range given in Section 5.6, then the process and controller parameters were set to their estimated values as given by Eq. 4.35.

Both a constant forgetting factor with values of 0.975, 0.99, and 0.999 and a variable forgetting factor due to Ydstin et al. [54], as discussed in Section 5.6, were used in the parameter estimation algorithm. It was found that the variable forgetting factor of Ydstin et al. [54] gave better parameter convergence, and it was used in the results presented in Table 6.12.

In Table 6.12 the standard error of the brake horsepower regression line (6.51 BHP)is above the limit specified by the EPA (6.40 BHP). However, this is due to the inclusion of the four tests that did not satisfy the EPA specifications. The reasons why the adaptive regulator did not lead to valid test cycles all the time and why the regression line parameters were not better than other adaptive control strategies tested in this research will be given in the next section.

Table 6.12: Regression line values for adaptive regulator

	Speed	Torque	BHP
SE,σ_{SE}	35.6 rpm, 0.04 rpm	20.4 ft-lb, 1.97 ft-lb	6.5 BHP, 0.70 BHP
m, σ_m	0.996, 0.0002	0.982, 0.0028	0.977, 0.0033
r^2, $\sigma_{r}2$	0.991, 0.0001	0.908, 0.0165	0.924, 0.0158
b, σ_b	0.3 rpm, 0.45 rpm	-2.8 ft-lb, 0.38 ft-lb	-0.78 BHP, 0.10 BHP

6.4 Comparison of Adaptive Control Strategies

Intuitively it would be expected that the self-tuning gain scheduling control should give better results than the self-tuning one-shot control, and similarly, the adaptive regulator should give better results than the self-tuning gain-scheduling control. Therefore, these comparisons will be shown separately. The regression line parameters obtained in each test with all three adaptive control approaches are given in Appendix B.

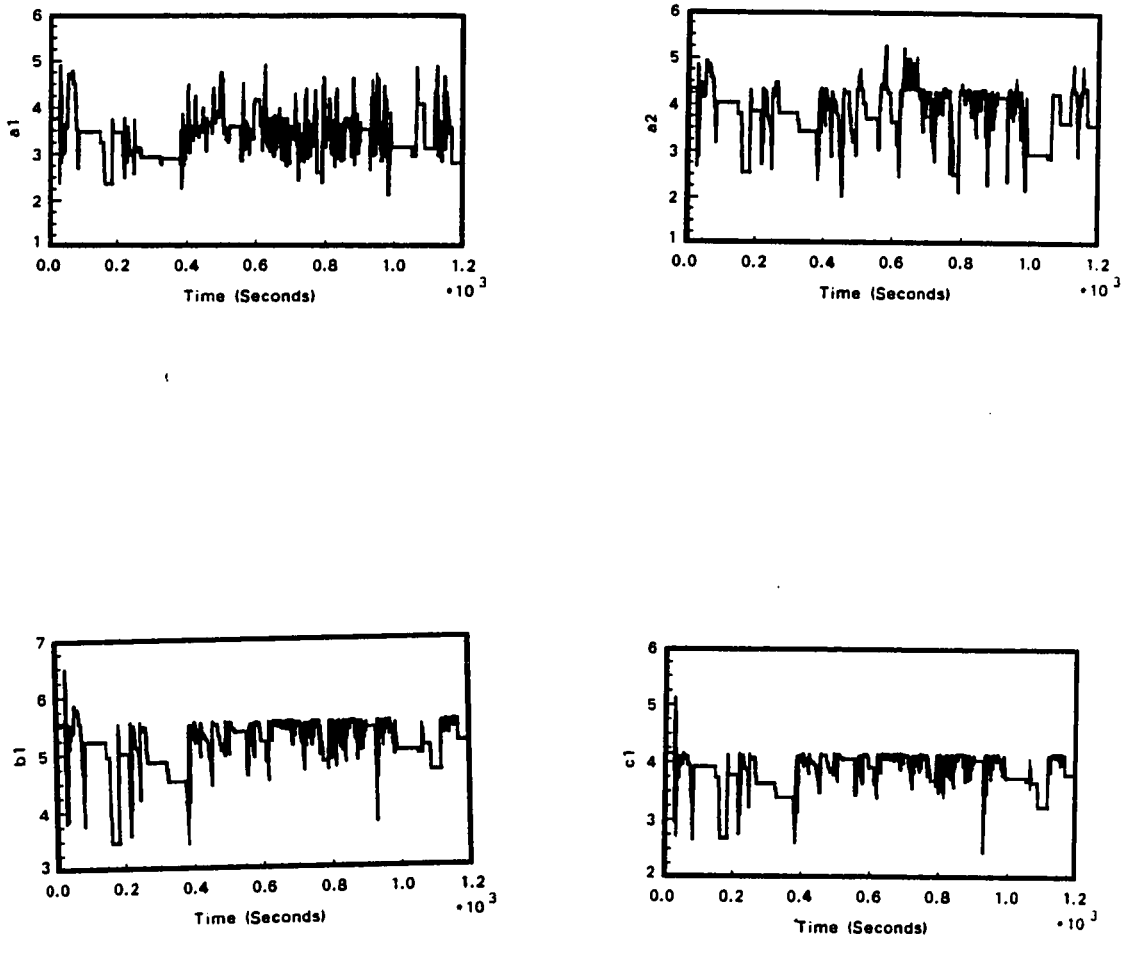


Figure 6.8: Convergence of parameters during the whole transient test cycle

6.4.1 Gain-scheduling self-tuning versus one-shot self-tuning

As discussed in Sections 6.1 and 6.2, both algorithms gave satisfactory results and they satisfied all of the EPA specifications in every transient test cycle. To be able to see whether gain-scheduling is better or not, some statistical measures were computed. The regression line parameters in both methods were compared with each other. Since the sample size was less than thirty, Student's t-statistic was applied to see if there was any significant difference between means [55]. In Student's t-test, the mean values of two groups are compared using the following formula:

$$t = \frac{|\bar{x}_1 - \bar{x}_2|}{\sigma_d} \tag{6.21}$$

where $\bar{x}_1$ and $\bar{x}_2$ show the mean values of the two groups to be compared. If calculated standard deviations of each group are σ_1 and σ_2 and the number of samples observed in each group are n_1 and n_2, then σ_d in the equation shown above can be calculated as:

$$\sigma_d = \sqrt{\frac{\sigma_1^2(n_1 - 1) + \sigma_2^2(n_2 - 1)}{n_1 + n_2 - 2} \cdot \frac{n_1 + n_2}{n_1 n_2}} \tag{6.22}$$

If this calculated t is greater than the tabulated value at the specified level of significance, then it is concluded that the difference in means is statistically significant. Tabulated values of t for various levels of significance can be found in [55].

In our study, the comparisons were made at the ten percent significance level. The results of the comparisons employing the Student's t-statistic showed that there was no significant difference between the means of the speed regression line values. This is as expected due to the fact that the speed control was governed by the GE Dynamometer Controller. In addition, the mean of the slope (m) of the torque regression line was found to be significantly closer to 1.0 in the gain-scheduling control.

This indicates better torque control with gain-scheduling. This is also as expected since the controller parameters were continuously changed based on the operating conditions while the one-shot self-tuning control assumes the same process and controller parameters for all operating conditions. The mean of the slope (m) of the brake horsepower regression line was significantly closer to 1.0, and the standard error of estimate (SE) value of the brake horsepower regression line was also significantly lower in the gain-scheduling control. This also indicates better torque control with gain-scheduling, and again this is what we expect from the gain-scheduling control. The improvements on the parameters of the torque and brake horsepower regression line were up to 35% with the gain-scheduling control. The means of all the other regression line parameters were not found to be different from each other at the the ten percent significance level. This could be due to the small sample size.

From the comparisons above it can be concluded that the gain scheduling self-tuning control gives closer values to the EPA regression line specifications than the one-shot self-tuning control.

6.4.2 Adaptive regulator versus gain-scheduling control

From the direct comparison of the results presented in Sections 6.2 and 6.3, it is not clear that the adaptive regulator gave better results than the gain-scheduling self-tuning control. In fact, in four of the ten transient test cycles, the adaptive regulator did not even lead to valid tests under the EPA regulations. Therefore this case represents a contradiction to the theory.

There may be several factors that lead to the deteriorated performance with the adaptive regulator. In the gain-scheduling control the process parameters were

determined from the well-designed on-line open-loop tests. Therefore, the parameter convergence problems were minimized. However, in the adaptive regulator, the process parameters were determined in closed-loop. Hence, there was no guarantee on the stability and convergence of the estimated parameters.

Another point to be considered is that the parameters of the process transfer function vary continuously based on the operating conditions. During the transient test cycle, the time to go from one extreme operating condition to another extreme case is sometimes as low as two seconds. Therefore, it requires a very fast tracking of system parameters. Recursive parameter estimation algorithms work well if there is no noise in the system or the system parameters vary slowly. However, as stated in Chalam [32], recursively estimating rapidly varying parameters in a noisy environment is very difficult and subject to stability and convergence problems.

Adaptive regulators also require minimum order of system description for best parameter convergence. The majority of the successful adaptive regulators reported in the literature employ a first order system as the process model. However, in our throttle-torque system, a first order model cannot adequately represent the input-output behavior, and therefore a minimum of a second order model is needed. Thus, convergence problems become more serious with second order models.

Adaptive regulators also require that the time delay in the system is known and constant over the operating range for better parameter estimation and better closed-loop control. However, in the throttle-torque system studied in this research, the actual time delay varied based on the operating conditions although it was assumed constant in the estimation algorithm. This alone can cause convergence problems in the parameter estimation algorithm.

In addition to the concerns with the adaptive regulator listed above, there exists another problem particular to the torque control setup used in this study. As mentioned in Section 5.5, controller output signals of less than five units were not applied to the system due to some serial-line communication problems. This also deteriorated the closed-loop system performance.

As a summary of this section, it can be concluded that the gain-scheduling self-tuning control gave the best results among all three adaptive control approaches studied in this research. One-shot self-tuning control also gave good results, and it had the advantage of being computationally less complex than a gain-scheduling self-tuning control. Application of the full adaptive regulators, however, may not be recommended for the torque control of diesel engines operating over transient test cycles.

6.5 Comparisons with Constant Parameter Non-Adaptive Controllers

Tuken et al. [1] described the design steps of a conventional digital torque control system capable of following the EPA transient test cycle developed for the same engine and dynamometer pair used in this study. Controller and feedforward compensator designs and Smith predictor were carried out with methods similar to the ones explained in this dissertation. Table 6.13 shows the average regression line parameters of 24 transient test cycles reported in their study.

To compare the regression line parameters in Table 6.13, obtained with conventional non-adaptive control, with the regression line parameters in Table 6.7, obtained through the gain-scheduling control, the same statistical measures discussed earlier were used. Student's t-statistics, as explained in Section 6.4, were calculated to see if

Table 6.13: Regression line values for non-adaptive control

	Speed	Torque	BHP
SE,σ_{SE}	10.2 rpm, 0.25 rpm	17.5 ft-lb, 1.06 ft-lb	5.3 BHP, 0.30 BHP
m, σm	0.990, 0.0001	0.940, 0.0210	0.950, 0.0184
r^2, $\sigma_{r}2$	0.998, 0.0001	0.940, 0.0227	0.940, 0.0160
b, σ_b	14.9 rpm, 1.43 rpm	-0.33 ft-lb, 0.16 ft-lb	-0.2 BHP, 0.07 BHP

any significant difference exists between the means of each parameter. The following results were obtained:

1. The means of the slope (m) of the torque and brake horsepower regression line were significantly higher (up to 30%) in the gain-scheduling control. It means that better tracking of torque and brake horsepower reference trajectory was obtained when gain-scheduling control was employed.

2. Only the standard error of estimate (SE) of the brake horsepower regression line was lower in the non-adaptive control case.

3. All other parameters were not found to be statistically different from each other at the 10 percent significance level when t-test was employed.

The above comparison indicates that the gain-scheduling control gives results as good as or better than the constant parameter controller.

6.6 Application of the Proposed Self-Tuning Control to Other Systems

Of course, the small improvements in the regression line parameters are not a justification for employing a self-tuning controller. The main justification for the application of self-tuning control was the capability of controlling different engines, or the same engine with different components, without any prior design and testing. All the transient tests described in this dissertation were performed on the same engine and dynamometer pair described in Section 3.1. Although the best way to test the versatility of the self-tuning algorithms in this research would be to test the algorithm on different engine and dynamometer pairs, it was not possible due to the expense involved in obtaining different engine and dynamometer pairs. Instead, to give an example of why the self-tuning controller may be needed, the following two studies were performed:

1. To test the self-tuning algorithm when changes in system components are made, the transient tests were performed on the engine after the turbocharger had been removed. The self-tuning control employing one-shot system identification was employed. The resulting transient test was found to be valid under the EPA regulations. However, the parameters of the open-loop throttle-torque and speed-torque system did not vary significantly when the turbocharger was removed. Using the same constant parameter non-adaptive controller the transient test cycle was performed again, and the resulting test also satisfied the EPA regulations. Therefore, it was felt that this could not be considered a significant change in system components and dynamics since the engine in the consideration was very lightly turbocharged. However, it was still an example

of obtaining valid transient test cycles with self-tuning control.

2. As explained in Section 5.5, an analog prefilter with a bandwidth of 1.45 rad/sec was used both in this study and in the design of the conventional constant-parameter controller explained in Tuken et al. [1]. In the EPA transient test cycle, electronically filtering torque and speed measurements is allowed and the amount of filtering is left to the user. Therefore, to provide a significant change in overall system components and dynamics, the filter bandwidth was switched to 6.5 rad/sec.

   Figure 6.9 and Figure 6.10 show the effect of prefiltering on the open-loop response of the throttle-torque and the speed-torque systems, respectively. The describing transfer function of the throttle-torque system was found, employing the on-line recursive PMF identification algorithm, as:

$$\frac{B(s)}{A(s)} = \frac{8.80s + 55.30}{s^2 + 9.10s + 47.25} \tag{6.23}$$

   This process description contains two poles located at $-4.55 \pm 5.15j$ and a zero located at -6.28. Comparison of these values with the ones shown in Table 6.1 and Table 6.2 show that a very significant change occurred in the system dynamics. The damping ratio and natural frequency of the throttle-torque plant are 0.66 and 6.85 rad/sec when an analog prefilter with a bandwidth of 1.45 rad/sec is used, and 0.89 and 2.07 rad/sec when an analog prefilter with a bandwidth of 6.5 rad/sec, respectively.

   Similarly, the describing transfer function of the speed-torque open-loop system was found as:

$$\frac{D(s)}{Cs} = \frac{39.94s + 50.10}{s^2 + 14.52 + 61.10} \tag{6.24}$$

The speed-torque process described above contains two poles located at $-7.26 \pm 2.90j$ and a zero located at -1.25. Again there is a significant change in speed-torque system dynamics when compared to the models in Table 6.3 and Table 6.4. The damping ratio and natural frequency of the speed-torque plant are 0.93 and 7.82 rad/sec when an analog prefilter with a bandwidth of 1.45 rad/sec is used, and 0.61 and 2.66 rad/sec when an analog prefilter with a bandwidth of 6.5 rad/sec, respectively.

Transient test cycles were performed using both the constant parameter non-adaptive controller, designed when the filter bandwidth was set to 1.45 rad/sec, and the one-shot self-tuning control, which uses no a priori information about the engine and dynamometer system. It was found that the conventional non-adaptive design produced an unstable closed-loop operation while the self-tuning control was successfully implemented and the test results satisfied the EPA specifications. Therefore, the results of these tests proved that the designed self-tuning control can be successfully applied to systems with dramatically different dynamic characteristics.

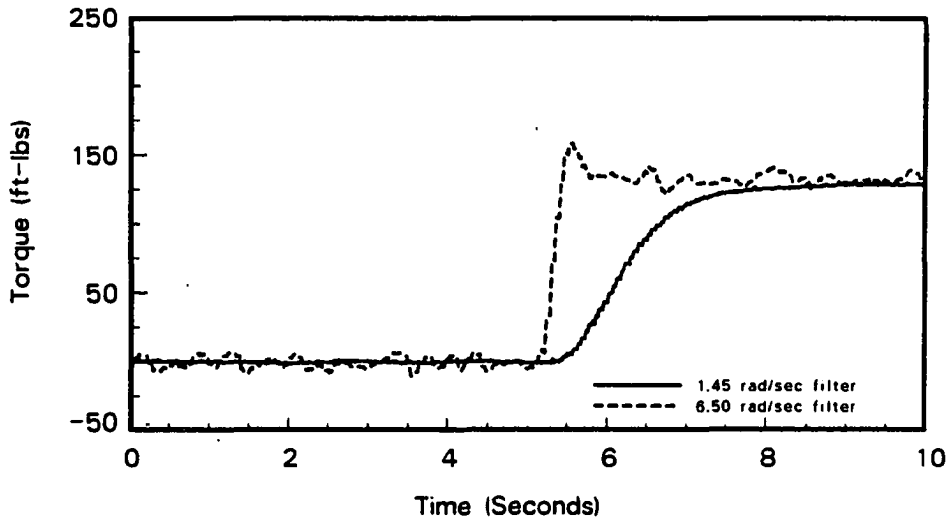


Figure 6.9: The effect of prefiltering on throttle-torque open-loop system

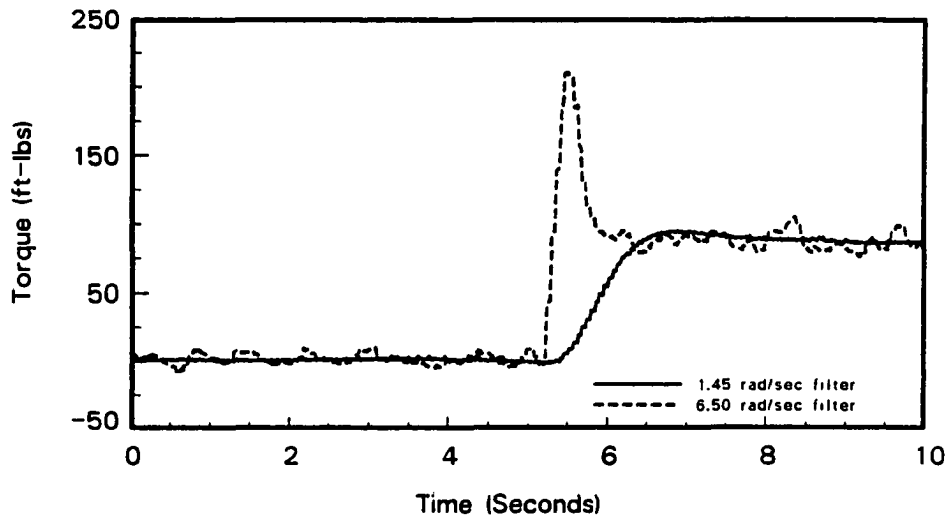


Figure 6.10: The effect of prefiltering on speed-torque open-loop system

7. CONCLUSION

The objectives of this project were to investigate the feasibility of using adaptive torque control to implement the EPA Federal Transient Test Cycle, to compare the performance of conventional and adaptive torque controllers, and to point out the implementation problems of using digital adaptive torque control for a diesel engine. The purpose of this chapter is to summarize the experimental results as they relate to the objectives and then to make recommendations for future research.

7.1 Summary

The original intention of this project was to design an adaptive torque control algorithm which could prevent an off-line system identification and controller design for each specific engine and dynamometer pair. A one-shot self-tuning control algorithm was developed for this purpose. Complete system identification and controller design were carried out on-line prior to the transient test cycle in this method. A new pole-zero assignment algorithm was developed to prevent the overall closed-loop controller from unstable operation due to the practical implementation problems. A feedforward controller and Smith pre-

dictor were used to compensate for the effects of load disturbances and time delay, respectively. Due to its superior noise rejection capability, a continuous-time parameter identification algorithm, namely the Poisson Moment functional (PMF) method, was employed. Six different models were considered in the algorithm as the best possible candidates to describe the input-output behavior of the open-loop throttle-speed-torque system. Test results showed that either a second order or a third order model could closely approximate the actual system. Ten transient test cycles were performed with this method and all the tests satisfied the EPA specifications. The most significant feature of this algorithm was shown to be the capability of controlling systems with dramatically different dynamic characteristics without requiring any prior knowledge about the system. Therefore, a significant time savings could be possible with this algorithm in engine test laboratories where different engine and dynamometer pairs need to be run over transient test cycles.

Since linear models were used to approximate a nonlinear system, the parameters of the selected model using the one-shot self-tuning algorithm varied with operating conditions. A gain-scheduling self-tuning control algorithm was developed to preserve the advantages of the one-shot self-tuning control and to compensate for the variations of the system parameters. In this method the selected model parameters and corresponding controller parameters were found on-line prior to the test cycle for each operating range employing multiple one-shot identification and controller design. Again, ten transient test cycles were implemented using the gain-scheduling self-tuning control algorithm. All of these ten tests satisfied the EPA requirements. When compared to the basic

one-shot self-tuning control, the gain-scheduling self-tuning control provided improvements up to 35% on the parameters of the torque and brake horsepower regression line. The gain-scheduling self-tuning control also did not require any prior knowledge about the system dynamics, yet it provided better reference torque and brake horsepower trajectory tracking.

Although the self-tuning gain-scheduling controller considered the variations in system dynamics, it did not update the controller and process parameters at each sampling instant. To provide better torque control, a full adaptive regulator was developed to continuously update the controller parameters based on the varying process parameters. However, it required that the time delay, model order, and estimate of the model parameters be known prior to the test cycle. Therefore, the motivation for the design of this algorithm was to provide better torque control for a particular engine and dynamometer pair. Due to the required prior knowledge about system dynamics, this algorithm was not ideal for testing different engine and dynamometer pairs. Ten transient test cycles were performed with this method. Only six of these ten tests satisfied the EPA requirements. Even in these six tests, the average values of the regression-line parameters were not better than either the gain-scheduling self-tuning controller or the one-shot self-tuning controller. The fast variation of system dynamics, the minimum order model requirement, constant time delay assumption, and the observed problems with serial line communication during the implementation of the test cycle were the main reasons for deteriorated performance with the full adaptive regulator.

Comparisons with constant parameter non-adaptive controllers showed that the

gain-scheduling control provided improvements of up to 30% on the mean values of slope of the torque and brake horsepower regression line parameters. It was also shown in this research that the self-tuning regulator could be applied to systems with dramatically different dynamic characteristics while the constant parameter non-adaptive controller, designed for a particular engine and dynamometer pair, could not control other engine and dynamometer pairs if there was a significant difference in the system dynamics.

7.2 Recommendations for Future Work

Although successful results were obtained with the self-tuning controller in the implementation of the EPA transient test cycles, there are several issues need to be addressed for better torque control and for closer tracking of the EPA regression line parameters. These can be listed as follows:

(a) The self-tuning controllers designed in this dissertation did not consider the variation in the time delay over the entire operating range. This was due to the complexity of the on-line estimation of the time delay and the difficulty of practical implementation of a variable sampling rate as a function of the time-delay. However, better torque control and better tracking of the EPA regression line parameters could be obtained if the variation of the time delay were incorporated into the controller design algorithm.

(b) In this research the adaptive regulator with continuous update of the process and controller parameters did not lead to consistently valid transient

test cycles. The main reason was that the adaptive control required a minimum degree of system order due to the convergence properties of the estimation algorithm. The selection of a first order model was not possible in the engine and dynamometer pair considered in this study, and therefore a second order model was considered. However, a full adaptive regulator could provide satisfactory results if an engine and dynamometer pair were described by a first order model. Wellstead and Zanker [16] investigated the feasibility of the self-tuning controller to the speed control of a diesel engine whose transfer function was described by a first order model. They obtained satisfactory transient response with their self-tuning controller, but they did not deal with the torque control problem for the EPA transient test cycles. Therefore, before reaching a conclusion that the full adaptive regulator is not applicable to the transient test cycle implementation, some further study is required with an engine and dynamometer pair whose describing transfer function is first order.

BIBLIOGRAPHY

[1] Tuken, T., Fullmer, R. R., and Van Gerpen, J. *Modeling, Identification, and Torque Control of a Diesel Engine for Transient Test Cycles*, Society of Automotive Engineers, Paper No. 900235, 1990.

[2] Koustas, J., and Watson, N. *A Transient Diesel Test Bed with Direct Digital Control*, Society of Automotive Engineers, Paper No. 840347, 1984.

[3] Noble, A. D., Beaumont, A. J., and Mercer, A. S. *Predictive Control Applied to Transient Engine Testbeds*, Society of Automotive Engineers, Paper No. 880487, 1988.

[4] Brown, D. G., and Thompson, S. *A Novel Approach to Engine Torque Speed Control*, Society of Automotive Engineers, Paper No. 831302, 1983.

[5] Morris, R. L., Hopkins, H. G., and Borcherts, R. H. *An Identification Approach to Throttle-Torque Modeling*, Society of Automotive Engineers, Paper No. 810448, 1981.

[6] Powell, B. K. *A Dynamic Model for Automotive Engine Control Analysis*, Proceedings of the 18th IEEE Conference on Decision and Control, Fort Lauderdale, Florida, December 1979, pp. 120-126.

[7] Tsai, S., and Goyal, M. R. *Dynamic Turbocharged Diesel Engine Model for Control Analysis and Design*, Society of Automotive Engineers, Paper No. 860455, 1986.

[8] Blaney, P. G. *Improvements to Cruise Control Utilizing Microprocessor Technology*, Society of Automotive Engineers, Paper No. 830662, 1983.

[9] Sobolak, S. J. *Simulation of the Ford Vehicle Speed Control System*, Society of Automotive Engineers, Paper No. 820777, 1982.

[10] Kamei, E., Namba, H., Osaki, K., and Ohba, M. *Application of Reduced Order Model to Automotive Engine Control System*, Transactions of the ASME, 232, September 1987, pp. 232-237.

[11] Takahashi, T, Ueno, T., Yamamoto, A., and Sanbuichi, H. *A Simple Engine Model for Idle Speed Control*, Society of Automotive Engineers, Paper No. 850291, 1985.

[12] Watson, N. *Dynamic Turbocharged Diesel Engine Simulator for Electronic Control System Development*, Journal of Dynamic Systems, Measurement, and Control, 106, March 1984.

[13] Winterborne, D. E., Thiruarooran, C., and Wellstead, P. E. *A Wholly Dynamic Model of a Turbocharged Diesel Engine for Transfer Function Evaluation*, Society of Automotive Engineers, Paper No. 770124, 1977.

[14] Morris, R. L., Warlick, M. V., and Borcherts, R. H. *Engine Idle Dynamics and Control: A 5.8L Application*, Society of Automotive Engineers, Paper No. 820778, 1982.

[15] Dobner, D. J. *A Mathematical Engine Model for Development of Dynamic Engine Control*, Society of Automotive Engineers, Paper No. 800054, 1980.

[16] Wellstead, P. E., and Zanker, P. M. *Application of Self-Tuning to Engine Control*, in *Self-Tuning and Adaptive Control: Theory and Applications*, Eds., C. J. Harris, and S. A. Billings, Peter Peregrinus Ltd., London, England, 1981, pp. 282-295.

[17] Norton, J. P. *An Introduction to Identification*, Academic Press, London, England, 1986.

[18] Sodestrom, T., and Stoica, P. *System Identification*, Prentice-Hall International, Hemel Hempstead, U.K., 1989.

[19] Ljung, L. *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[20] Ljung, L., and Sodestrom, T. *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Massachusetts, 1983.

[21] Eykhoff, P. (Ed.) *Trends and Progress in System Identification*, Pergamon Press, Oxford, England, 1981.

[22] Hsia, T. C. *System Identification: Least Squares Methods*, Lexington Books, Lexington, Massachusetts, 1977.

[23] Goodwin, G. C., and Payne, R. L. *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, New York, 1977.

[24] Kagiwada, H. H. *System Identification: Methods and Applications*, Addison Wesley Publishing Company, Reading, Massachusetts, 1974.

[25] Mendel, J. M. *Discrete Techniques of Parameter Estimation*, Society of Automotive Engineers, Paper No. 90900, 1990.

[26] Goodwin, G. C., and Middletown, R. H. *Digital Control and Estimation: A Unified Approach*, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[27] Eykhoff, P. (Ed.) *Identification of Industrial Processes: The Application of Computers in Research and Production Control*, North Holland Publishing Company, New York, 1980.

[28] Mendel, J. M. *Lessons in Digital Estimation Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[29] Eykhoff, P. *System Identification: Parameter and State Estimation*, Wiley, New York, 1974.

[30] Saha, D. C., and Rao, G. P. *Identification of Continuous Dynamical Systems: The Poisson Moment Functional (PMF) Approach*, Springer-Verlag, New York, 1983.

[31] Unbehauen, H., and Rao, G. P. *Identification of Continuous Systems*, Elsevier Science Publishing Company, Inc., New York, 1987.

[32] Chalam, V. V. *Adaptive Control Systems: Techniques and Applications*, Marcel Dekker, Inc., New York, 1987.

[33] Landau, Y. D. *Adaptive Control: The Model Reference Approach*, Marcel Dekker, Inc., New York, 1979.

[34] Harris, C. J., and Billings, S. A. (Eds.) *Self-Tuning and Adaptive Control: Theory and Applications*, Peter Peregrinus Ltd., New York, 1981.

[35] Roffel, B., Vermeer, P. J., and Chin, P. A. *Simulation and Implementation of Self-Tuning Controllers*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[36] Gawthrop, P. J. *Continuous Time Self-Tuning Control I*, Research Studies Press Ltd., Letchworth, U.K., 1987.

[37] Goodwin, G. C., and Sin, K. S. *Adaptive Filtering Prediction and Control*, Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[38] Astrom, K. J., and Wittenmark, B. *Adaptive Control*, Addison-Wesley Publishing Company, New York, 1989.

[39] Unbehauen, H. (Ed.) *Methods and Applications in Adaptive Control*, Springer-Verlag, Berlin, 1980.

[40] Narendra, K. S., and Monopoli, R. V. (Eds.) *Applications of Adaptive Control*, Academic Press, New York, 1980.

[41] Egardt, B. *Unification of some Discrete-Time Adaptive Control Schemes*, IEEE Transactions in Automatic Control, AC-25, August 1980, pp. 693-697.

[42] Landau, I. D. *Combining Model Reference Adaptive Control Systems and Stochastic Self-Tuning Regulators*, Automatica, 18, January 1982, pp. 77-84.

[43] Astrom, K. J., and Wittenmark, B. *Computer Controlled Systems: Theory and Design*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

[44] Egardt, B. *Stability of Adaptive Controllers*, Springer-Verlag, Berlin, 1979.

[45] Ioannou, P. A., and Kokotovic, P. V. *Adaptive Systems with Reduced Models*, Springer-Verlag, Berlin, 1983.

[46] Canudas, D. W., and Carlas, A. *Adaptive Control for Partially Known Systems: Theory and Applications*, Elsevier Publishing Company, Inc., New York, 1988.

[47] Kuo, B. C. *Mathematical Modeling of Permanent-Magnet Step Motors*, in *Step Motors and Control Systems*, Ed., Benjamin C. Kuo, SRL Publishing Company, Champaign, Illinois, 1979, pp. 87-113.

[48] Gant, G. C. *The Governing of Diesel Engines*, in *Principles and Performance in Diesel Engineering*, Eds. Sam Haddad and Neil Watson, John Wiley & Sons, Ltd., New York, 1984, pp. 77-121.

[49] *Code of Federal Regulations*, Protection of Environment, 40, CFR 86-1341-84, U.S. Government Printing Office, Washington, 1988.

[50] Donaghue, J. F. *Review of Control Design Approaches for Transport Delay Processes*, ISA Transactions, 16, April 1990, pp. 27-34.

[51] Malek-Zavarei, M., and Jamshidi, M. *Time Delay Systems: Analysis, Optimization and Applications*, North-Holland, New York, 1987.

[52] Marshall, J. E. *Control of Time-Delay Systems*, Peter Peregrinus Ltd., New York, 1979.

[53] Bierman, G. J., and Thornton, C. L. *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.

[54] Ydstie, B. E., Kershenbaum, L. S., and Sargent, R. W. H. *Theory and Applications of an Extended Horizon Self-Tuning Controller*,AIChE Journal, 31, November 1985, pp. 1771-1780.

[55] Kennedy, B. J., and Neville, A. M. *Basic Statistical Methods for Engineers and Scientists*, Harper–Row, New York, 1976.

ACKNOWLEDGEMENTS

My sincerest thanks go to Dr. Jon Van Gerpen for his guidance and encouragement throughout this research. I consider myself very lucky for having met Jon Van Gerpen and having worked with him.

Special thanks are extended to Dr. Rees Fullmer for his advice and guidance during the initial phase of this project.

I would like to thank my office mate Qiqing Jiang for lots of good company and conversation.

I am thankful to my mother-in-law, Kaviye Ozbay, for taking care of my kids and family. Without her help my graduation would be delayed for more than one year.

I am most grateful to my wife, Fethiye, my daughter, Erin, and my son, Alper, for their support, faith, and patience during the course of this research.

APPENDIX A. DERIVATION OF THE RECURSIVE PARAMETER ESTIMATION EQUATIONS

The derivation of the recursive parameter estimation equations is very similar in all the adaptive control strategies used in this research. Adaptive control with one-shot identification uses a single-input single-output model and parameter identification whereas the continuous adaptation of process and controller parameters requires a multiple-input single-output parameter identification. The following sections explain how to obtain continuous time parameters of a system from discrete measurements of data when the Poisson Moment Functional (PMF) identification algorithm is employed.

A.1 SISO System Parameter Estimation

The open-loop throttle-torque and speed-torque models are described in Section 5.1 as shown in Figure 4.8. The following sections describe the development of recursive parameter estimation equations for each case considered. The first equation in each section gives the differential equation formulation of the system. The PMF's of the first equation yields the second equation. The third equation is obtained by expanding the second equation using the identities given

in Table 4.1. The fourth equation in each section is obtained by arranging the third equation for parameter estimation purposes.

* **First order system:**

$$\frac{dT_L}{dt} + a_1 T_L = b_1 u \tag{A.1}$$

$$M_k\left[\frac{dT_L}{dt}\right] + a_1 M_k\left[T_L\right] = b_1 M_k\left[u\right] \tag{A.2}$$

$$T_{L_0} - T_{L_1} + a_1 T_{L_1} = b_1 u_1 \tag{A.3}$$

$$T_{L_0} - T_{L_1} = \left[-T_{L_1}\right] a_1 + \left[u_1\right] b_1 \tag{A.4}$$

* **Second order system with no numerator dynamics:**

$$\frac{d^2T_L}{dt^2} + a_1\frac{dT_L}{dt} + a_2 T_L = b_1 u \tag{A.5}$$

$$M_k\left[\frac{d^2T_L}{dt^2}\right] + a_1 M_k\left[\frac{dT_L}{dt}\right] + a_2 M_k\left[T_L\right] = b_1 M_k\left[u\right] \tag{A.6}$$

$$T_{L_0} - 2T_{L_1} + T_{L_2} + a_1\left[T_{L_1} - T_{L_2}\right] + a_2\left[T_{L_2}\right] = b_1\left[u_2\right] \tag{A.7}$$

$$T_{L_0} - 2T_{L_1} + T_{L_2} = a_1\left[T_{L_2} - T_{L_1}\right] + a_2\left[-T_{L_2}\right] + b_1\left[u_2\right] \tag{A.8}$$

* **Second order system with first order numerator dynamics:**

$$\frac{d^2T_L}{dt^2} + a_1\frac{dT_L}{dt} + a_2 T_L = b_1\frac{du}{dt} + b_2 u \tag{A.9}$$

$$M_k\left[\frac{d^2T_L}{dt^2}\right] + a_1 M_k\left[\frac{dT_L}{dt}\right] + a_2 M_k\left[T_L\right] = b_1 M_k\left[\frac{du}{dt}\right] + b_2 M_k\left[u\right] \tag{A.10}$$

$$T_{L_0} - 2T_{L_1} + T_{L_2} + a_1\left[T_{L_1} - T_{L_2}\right] + a_2\left[T_{L_2}\right] = b_1\left[u_1 - u_2\right] + b_2 u_2 \tag{A.11}$$

$$T_{L_0} - 2T_{L_1} + T_{L_2} = a_1\left[T_{L_2} - T_{L_1}\right] + a_2\left[-T_{L_2}\right] + b_1\left[u_1 - u_2\right] + b_2\left[u_2\right] \tag{A.12}$$

* **Third order system with no numerator dynamics:**

$$\frac{d^3T_L}{dt^3} + a_1\frac{d^2T_L}{dt^2} + a_2\frac{dT_L}{dt} + a_3T_L = b_1u \tag{A.13}$$

$$M_k\left[\frac{d^3T_L}{dt^3}\right] + a_1M_k\left[\frac{d^2T_L}{dt^2}\right] + a_2M_k\left[\frac{dT_L}{dt}\right] + a_3M_k\left[T_L\right] = b_1M_k\left[u\right] \tag{A.14}$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} + a_1\left[T_{L_1} - 2T_{L_2} + T_{L_3}\right] + a_2\left[T_{L_2} - T_{L_3}\right] + a_3T_{L_3} = b_1u_3 \tag{A.15}$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} = a_1\left[-T_{L_1} + 2T_{L_2} - T_{L_3}\right] + a_2\left[T_{L_3} - T_{L_2}\right] + a_3\left[-T_{L_3}\right] + b_1\left[u_3\right] \tag{A.16}$$

* **Third order system with first order numerator dynamics**

$$\frac{d^3T_L}{dt^3} + a_1\frac{d^2T_L}{dt^2} + a_2\frac{dT_L}{dt} + a_3T_L = b_1\left[\frac{du}{dt}\right] + b_2u \tag{A.17}$$

$$M_k\left[\frac{d^3T_L}{dt^3}\right] + a_1M_k\left[\frac{d^2T_L}{dt^2}\right] + a_2M_k\left[\frac{dT_L}{dt}\right] + a_3M_k\left[T_L\right] = b_1M_k\left[\frac{du}{dt}\right] + b_2M_k\left[u\right] \tag{A.18}$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} + a_1\left[T_{L_1} - 2T_{L_2} + T_{L_3}\right] + a_2\left[T_{L_2} - T_{L_3}\right] + a_3T_{L_3} = b_1\left[u_2 - u_3\right] + b_2u_3 \tag{A.19}$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} = a_1\left[-T_{L_1} + 2T_{L_2} - T_{L_3}\right] + \quad \text{(A.20)}$$
$$a_2\left[T_{L_3} - T_{L_2}\right] + a_3\left[-T_{L_3}\right] + b_1\left[u_2 - u_3\right] + b_2\left[u_3\right]$$

Third order system with second order numerator dynamics

$$\frac{d^3T_L}{dt^3} + a_1\frac{d^2T_L}{dt^2} + a_2\frac{dT_L}{dt} + a_3T_L = b_1\left[\frac{d^2u}{dt^2}\right] + b_2\left[\frac{du}{dt}\right] + b_3 \quad \text{(A.21)}$$

$$M_k\left[\frac{d^3T_L}{dt^3}\right] + a_1M_k\left[\frac{d^2T_L}{dt^2}\right] + a_2M_k\left[\frac{dT_L}{dt}\right] + a_3M_k\left[T_L\right] = \quad \text{(A.22)}$$
$$b_1M_k\left[\frac{d^2u}{dt^2}\right] + b_2M_k\left[\frac{du}{dt}\right] + b_3M_k\left[u\right]$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} + a_1\left[T_{L_1} - 2T_{L_2} + T_{L_3}\right] + \quad \text{(A.23)}$$
$$a_2\left[T_{L_2} - T_{L_3}\right] + a_3T_{L_3} = b_1\left[u_1 - 2u_2 + u_3\right] + b_2\left[u_2 - u_3\right] + b_3u_3$$

$$T_{L_0} - 3T_{L_1} + 3T_{L_2} + T_{L_3} = a_1\left[-T_{L_1} + 2T_{L_2} - T_{L_3}\right] + \quad \text{(A.24)}$$
$$a_2\left[T_{L_3} - T_{L_2}\right] + a_3\left[-T_{L_3}\right] + b_1\left[u_1 - 2u_2 + u_3\right] +$$
$$b_2\left[u_2 - u_3\right] + b_3\left[u_3\right]$$

A.2 MISO System Parameter Estimation

The multiple-input single-output (MISO) representation of the system is shown in Figure 4.6. The differential equation that describes the second order system is given by:

$$\frac{d^2T_L}{dt^2} + a_1\frac{dT_L}{dt} + a_2T_L = b_1u + c_1S_{ref} \quad \text{(A.25)}$$

Taking the Poisson Moment Functional's of this equation, we find:

$$M_k\left[\frac{d^2T_L}{dt^2}\right] + a_1 M_k\left[\frac{dT_L}{dt}\right] + a_2 M_k\left[T_L\right] = b_1 M_k\left[u\right] + +c_1 M_k\left[S_{ref}\right] \tag{A.26}$$

Substituting Eq. 4.9 into Eq. A.26 results in:

$$T_{L_0} - 2T_{L_1} + T_{L_2} + a_1\left[T_{L_1} - T_{L_2}\right] + a_2 T_{L_2} = b_1\left[u_2\right] + c_1 S_{ref_2} \tag{A.27}$$

Finally arranging Eq. A.27 for parameter estimation purpose, we come up with:

$$T_{L_0} - 2T_{L_1} + T_{L_2} = +a_1\left[T_{L_2} - T_{L_1}\right] + a_2\left[-T_{L_2}\right] + b_1\left[u_2\right] + c_1\left[S_{ref_2}\right] \tag{A.28}$$

APPENDIX B. TABULATED DATA

This Appendix contains tabular listings of the experimental data reported in this thesis. Table B.1 and Table B.2 show the variations of the throttle-torque and speed-torque model parameters in the ten transient test cycles when the one-shot self-tuning control was employed. Tables B.3–B.6 show the torque and brake horsepower regression line parameters in the ten transient test cycles for all three adaptive control strategies reported in this research.

Table B.1: Estimated parameters of throttle-load torque system for 10 transient test cycles

Case	n[a]	T_f	a_1	a_2	a_3	b_1	b_2	b_3	SSE[b]
2	2	0.38	3.58	4.06	—	5.18	—	—	357
2	2	0.42	3.88	4.54	—	5.74	—	—	393
3	2	0.44	3.63	4.30	—	0.15	5.56	—	357
3	2	0.44	3.80	4.36	—	0.25	5.64	—	330
3	2	0.44	3.41	4.00	—	0.43	5.14	—	316
3	2	0.42	3.59	4.15	—	0.15	5.35	—	317
3	2	0.38	3.39	3.78	—	0.16	4.87	—	384
3	2	0.44	3.74	4.42	—	0.20	5.66	—	341
5	3	0.38	4.16	8.24	5.18	4.73	7.00	—	336
5	3	0.44	3.98	7.19	3.92	4.55	5.21	—	267

[a]Model order.

[b]Sum of squares of errors between actual output and model output.

Table B.2: Estimated parameters of speed-load torque system for 10 transient test cycles

Case	n[a]	T_f	a_1	a_2	a_3	b_1	b_2	b_3	SSE[b]
2	2	0.40	3.17	7.07	—	5.84	—	—	170
2	2	0.38	3.12	6.54	—	5.47	—	—	196
2	2	0.40	3.40	7.03	—	6.20	—	—	270
3	2	0.40	3.24	6.56	—	0.05	5.53	—	185
3	2	0.44	3.12	6.28	—	0.68	5.08	—	194
3	2	0.44	2.68	5.86	—	0.52	4.68	—	203
3	2	0.44	3.49	7.74	—	0.08	6.33	—	253
3	2	0.44	3.12	6.85	—	0.41	5.47	—	175
5	3	0.40	3.49	8.18	3.23	5.32	2.89	—	149
5	3	0.42	3.49	8.15	5.56	4.45	4.87	—	288

[a]Model order.

[b]Sum of squares of errors between actual output and model output.

Table B.3: Variation of the slope in three adaptive control strategies

Torque Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
0.9589	0.9891	0.9831
0.9746	0.9685	0.9795
0.9682	0.9739	0.9835
0.9513	0.9563	0.9819
0.9473	0.9608	0.9789
0.9607	0.9432	0.9808
0.9653	0.9874	0.9851
0.9535	0.9848	0.9779
0.9762	0.9836	0.9801
0.9608	0.9742	0.9864

Brake Horsepower Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
0.9502	0.9948	0.9791
0.9654	0.9732	0.9713
0.9598	0.9405	0.9793
0.9425	0.9674	0.9788
0.9363	0.9700	0.9746
0.9493	0.9544	0.9755
0.9522	0.9854	0.9810
0.9547	0.9889	0.9804
0.9430	0.9857	0.9741
0.9643	0.9869	0.9728

Table B.4: Variation of the standard error of equation in three adaptive control strategies

Torque Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
17.624	21.043	19.348
17.181	17.143	23.762
16.911	18.605	22.119
19.139	20.011	19.706
19.999	19.631	19.181
18.183	20.070	20.211
18.275	18.942	18.599
17.192	16.855	18.346
18.565	18.400	22.902
15.821	17.385	21.697

Brake Horsepower Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
5.908	6.385	6.216
5.583	5.207	7.869
5.542	5.754	7.081
6.395	6.163	6.329
6.343	6.161	6.026
6.120	6.190	6.317
6.101	5.861	5.850
5.745	5.231	5.785
6.273	5.762	7.108
5.202	5.493	6.950

Table B.5: Variation of the coefficient of determination in three adaptive control strategies

Torque Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
0.9274	0.9051	0.9176
0.9328	0.9323	0.8800
0.9339	0.9220	0.8951
0.9142	0.9079	0.9183
0.9064	0.9118	0.9104
0.9233	0.9050	0.9238
0.9226	0.9194	0.9146
0.9315	0.9367	0.9256
0.9192	0.9251	0.8872
0.9426	0.9343	0.8980

Brake Horsepower Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
0.9337	0.9298	0.9312
0.9422	0.9501	0.8925
0.9424	0.9405	0.9125
0.9221	0.9307	0.9288
0.9108	0.9311	0.9345
0.9292	0.9284	0.9265
0.9299	0.9391	0.9387
0.9377	0.9512	0.9413
0.9249	0.9416	0.9110
0.9493	0.9462	0.9144

Table B.6: Variation of the intercept in three adaptive control strategies

Torque Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
-1.455	-3.844	-2.746
-2.125	-2.412	-3.756
-2.018	-2.506	-2.799
-1.684	-2.216	-2.528
-1.260	-2.157	-2.454
-1.509	-1.980	-2.747
-1.725	-2.854	-3.118
-1.766	-2.877	-2.770
-1.428	-3.159	-2.886
-1.959	-2.667	-2.797

Brake Horsepower Regression Line

One-Shot	Gain-Scheduling	Fully Adaptive
-0.309	-1.097	-0.752
-0.477	-0.699	-1.022
-0.458	-0.749	-0.777
-0.339	-0.721	-0.723
-0.223	-0.679	-0.661
-0.409	-0.779	-0.779
-0.291	-0.888	-0.812
-0.361	-0.791	-0.754
-0.362	-0.900	-0.841
-0.298	-0.668	-0.743

APPENDIX C. COMPUTER PROGRAMS

The software developed for the implementation of the Environmental Protection Agency (EPA) transient test cycles is given in this chapter for the cases of the self-tuning control with one-shot estimation and the adaptive regulator. The program listing for the self-tuning control with gain-scheduling is very similar to the self-tuning control with one-shot estimation, and therefore it is not included.

C.1 Program Listing for Self-Tuning Control with One-Shot Parameter Estimation and Controller Design

```
PROGRAM ONESHOT
INTEGER*2 ERSTAT;LCHAN,BOARD,COUNT,PCHAN,LCHAN1
INTEGER*2 CHA(0:3),PVOLT(3),ERCODE
INTEGER*1 CARR
CHARACTER*9 FILEN,STEP,STEPP,STEPS,STO
real*4 thetap(6),thetad(6)
DIMENSION A(15),B(15),Y(15),T(15),CON(15),C(15)
DIMENSION Y1(15),DE(15),CON1(15),DIS(15),DS(0:1200)
DIMENSION SPEED(1201),SPREF(1200),TORREF(1200),TR(1200)
DIMENSION TORQUE(1200),TIME(1200),ER1(15),ER2(15),ER3(15)
DIMENSION ER4(15)
DATA CARR /1*13/
CHA(0)=3
CHA(1)=0
```

```
      CHA(2)=1
      CHA(3)=2
      LCHAN=1
      LCHAN1=2
      BOARD=1
      COUNT=3
      PCHAN=0
      ERSTAT=0
      CALL INIT(ERSTAT)
      IF(ERSTAT.NE.0) GO TO 666
      CALL AIG820 (LCHAN,BOARD,CHA,COUNT,ERSTAT)
      CALL AOT820 (LCHAN1,BOARD,PCHAN,ERSTAT)
      STEP='    +    '
      STEPP='    -    '
      STO='        @'
100   WRITE(*,*) '*************************************************'
      WRITE(*,*) '*                                               *'
      WRITE(*,*) '*   SPEED AND TORQUE CONTROL OF A DIESEL ENGINE  *'
      WRITE(*,*) '*                                               *'
      WRITE(*,*) '*   SELECT ONE OF THE ITEMS BELOW:              *'
      WRITE(*,*) '*          1)START-UP THE ENGINE/DYNO           *'
      WRITE(*,*) '*            (SPEED ~ 1200 RPM, TORQUE ~ 70 lbft2)*'
      WRITE(*,*) '*          2)TRANSIENT ENGINE CYCLE             *'
      WRITE(*,*) '*          3)SHUT-DOWN THE ENGINE/DYNO          *'
      WRITE(*,*) '*          4)EXIT                               *'
      WRITE(*,*) '*                                               *'
      WRITE(*,*) '*************************************************'
      READ(*,*) ISELE
      IF(ISELE.EQ.1) CALL START(ERCODE)
      IF(ISELE.EQ.2) GO TO 50
      IF(ISELE.EQ.3) CALL STOP
      IF(ISELE.EQ.4) GO TO 6
      IF(ERCODE.EQ.1.0) GO TO 6
      GO TO 100
50    OPEN(UNIT=4,FILE='INPUTDOC',STATUS='OLD')
      DO 999 J=1,1199
999   READ(4,*) SECT,SPREF(J),TORREF(J)
      CLOSE(UNIT=4)
      DO 998 J=1,1199
      SPREF(J)=9.0*SPREF(J)+1200.0
```

```
      IF(SPREF(J).LT.1400.0) TORREF(J)=TORREF(J)*2.15
      IF(SPREF(J).LT.1400.0) GO TO 997
      IF(SPREF(J).LT.1700.0) TORREF(J)=TORREF(J)*2.05
      IF(SPREF(J).LT.1700.0) GO TO 997
      IF(SPREF(J).LT.1900.0) TORREF(J)=TORREF(J)*2.0
      IF(SPREF(J).LT.1900.0) GO TO 997
      IF(SPREF(J).LT.2100.0) TORREF(J)=TORREF(J)*1.9
      IF(SPREF(J).LT.2100.0) GO TO 997
      IF(SPREF(J).LT.2150.0) TORREF(J)=TORREF(J)*1.8
      IF(SPREF(J).LT.2150.0) GO TO 997
      IF(SPREF(J).LT.2175.0) TORREF(J)=TORREF(J)*1.7
      IF(SPREF(J).LT.2175.0) GO TO 997
      IF(SPREF(J).LT.2190.0) TORREF(J)=TORREF(J)*1.55
      IF(SPREF(J).LT.2190.0) GO TO 997
      IF(SPREF(J).LT.2200.0) TORREF(J)=TORREF(J)*1.40
      IF(SPREF(J).LT.2200.0) GO TO 997
      TORREF(J)=TORREF(J)*1.30
997   IF(TORREF(J).LT.0.0) TORREF(J)=-20.0
998   CONTINUE
      SPREF(1200)=SPREF(1199)
      TORREF(1200)=TORREF(1199)
      TR(1)=TORREF(1)
      TR(1200)=TORREF(1200)
      DO 994 J=2,1199
      TR(J)=TORREF(J)
      IF(SPREF(J+1).EQ.SPREF(J)) GO TO 994
      IF(TORREF(J).LT.0.0) GO TO 994
      TORREF(J)=TORREF(J)-0.2748081*(SPREF(J+1)-SPREF(J-1))
994   CONTINUE
      DO 996 J=1,1200
996   DS(J)=SPREF(J)-SPREF(1)
      ds(0)=ds(1)
      CALL IDENT(THETAP,THETAD,NNUM,NDEN,NDNUM,NDDEN,KP,KD)
      IF(KP.GE.1) KDEL=0
      IF(KP.GE.6) KDEL=1
      IF(KP.GE.14) KDEL=2
      IF(KP.GE.26) KDEL=3
      IF(KP.GE.1) FSAMPLE=5.0
      IF(KP.GE.6) FSAMPLE=7.0
      IF(KP.GE.8) FSAMPLE=6.0
```

```
      IF(KP.GE.10) FSAMPLE=5.0
      IF(KP.GE.12) FSAMPLE=4.0
      IF(KP.GE.14) FSAMPLE=7.0
      IF(KP.GE.16) FSAMPLE=6.0
      IF(KP.GE.19) FSAMPLE=5.0
      IF(KP.GE.23) FSAMPLE=4.0
      IF(KP.GE.26) FSAMPLE=6.0
      IF(KP.GE.29) FSAMPLE=5.0
      H=1.0/FSAMPLE
      CALL DESIGN(THETAP,THETAD,NNUM,NDEN,NDNUM,NDDEN,H,B1,B2,B3,
     @A1,A2,A3,BD1,BD2,BD3,AD1,AD2,AD3,NCN,NCD,CB1,CB2,CB3,CB4,CB5,
     @CB6,CA1,CA2,CA3,CA4,CA5,R1,R2,R3,S0,S1,S2,S3,T0,T1,T2,T3)
      write(*,*) 'DO YOU WANT ERROR FEEDBACK ONLY: (T=S)'
      WRITE(*,*) '    1=YES  2=NO'
      READ(*,*) IEF
      write(12,*) 'DO YOU WANT ERROR FEEDBACK ONLY: (T=S)'
      WRITE(12,*) '    1=YES  2=NO'
      WRITE(12,*) IEF
      IF(IEF.EQ.1) THEN
      T0=S0
      T1=S1
      T2=S2
      T3=S3
      ENDIF
      DO 995 J=1,5
      ER2(J)=0.0
      ER3(J)=0.0
      ER4(J)=0.0
      T(J)=0.0
      Y1(J)=0.0
      CON(J)=0.0
      C(J)=0.0
      DE(J)=0.0
      DIS(J)=0.0
      B(J)=0.0
      ER1(J)=0.0
      ER2(J)=0.0
      A(J)=0.0
995   CON1(J)=0.0
      ISS=0
```

```
      IJ=0
      AVE=0.0
      IFSAMPLE=FSAMPLE
      WRITE(*,*) 'PRESS ANY NUMBER IF YOU ARE READY TO START'
      READ(*,*) GGG
      DO 31, J=1,10
      CALL AING(LCHAN,PVOLT,ERSTAT)
31    AVE=AVE+PVOLT(2)
      AVE=AVE/10.0
      AVE=10.0*AVE/4095.0-5.0
      AVE=48.43213*AVE+0.404608
      Y(1)=AVE
      Y(2)=AVE
      Y(3)=AVE
      Y(4)=AVE
      Y(5)=AVE
      CALL SETTIM(0,0,0,0)
334   DO 1 J=1,1199
      DO 1 JJ=6,5+IFSAMPLE
      CALL AING(LCHAN,PVOLT,ERSTAT)
      A(JJ)=PVOLT(1)
      B(JJ)=PVOLT(2)
      A(JJ)=10.0*A(JJ)/4095.0-5.0
      B(JJ)=10.0*B(JJ)/4095.0-5.0
      A(JJ)=491.4704*A(JJ)+8.6656+5.0
      B(JJ)=48.43213*B(JJ)+0.4046
      C(JJ)=PVOLT(3)
      C(JJ)=10.0*C(JJ)/4095.0-5.0
      if(j.lt.(6+kdel)) DE(JJ)=DS(J-1)
      if(j.ge.(6+kdel)) DE(JJ)=ds(J)
      IJ=IJ+1
      IF(C(JJ).GT.4.60) GO TO 11
      IF(C(JJ).LT.1.20) GO TO 11
34    Y1(JJ)=-A1*Y1(JJ-1)-A2*Y1(JJ-2)-A3*Y1(JJ-3)
     1 +B1*CON1(JJ-1)+B2*CON1(JJ-2)+B3*CON1(JJ-3)
      Y(JJ)=Y1(JJ)+AVE
      ER1(JJ)=B(JJ)-Y(JJ-KDEL)
      ER2(JJ)=TORREF(J)-ER1(JJ)
      ER3(JJ)=S0/T0*Y(JJ)+S1/T0*Y(JJ-1)+S2/T0*Y(JJ-2)
     1 +S3/T0*Y(JJ-3)-T1/T0*ER3(JJ-1)-T2/T0*ER3(JJ-2)-T3/T0*ER3(JJ-3)
```

```
      ER4(JJ)=ER2(JJ)-ER3(JJ)
      CON1(JJ)=-R1*CON1(JJ-1)-R2*CON1(JJ-2)-R3*CON1(JJ-3)
     1 +T0*ER4(JJ)+T1*ER4(JJ-1)+T2*ER4(JJ-2)+T3*ER4(JJ-3)
      DIS(JJ)=-CA1*DIS(JJ-1)-CA2*DIS(JJ-2)-CA3*DIS(JJ-3)-CA4*
     1 DIS(JJ-4)-CA5*DIS(JJ-5)+CB1*DE(JJ)+CB2*DE(JJ-1)+CB3*DE(JJ-2)
     1 +CB4*DE(JJ-3)+CB5*DE(JJ-4)+CB6*DE(JJ-5)
      CON(JJ)=CON1(JJ)+DIS(JJ)
      ISTEP=CON(JJ)
      ISTEP=ISTEP-ISS
      IMAX=(C(JJ)-1.40)*490.2
      IMIN=(C(JJ)-4.40)*490.2
      IF(ISTEP.GT.IMAX) ISTEP=IMAX
      IF(ISTEP.LT.IMIN) ISTEP=IMIN
      ISS=CON1(JJ)+DIS(JJ)
      WRITE(STEPP(6:9),'(I4.4)') ABS(ISTEP)
      WRITE(STEP(6:9),'(I4.4)') ISTEP
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      IF(ISTEP.GT.0) WRITE (9) STEP,CARR
      IF(ISTEP.LT.0) WRITE (9) STEPP,CARR
      IF(ISTEP.EQ.0) WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      REFSP=245.0*SPREF(J)/2175.0
      REFSP=4095.0*REFSP/255.0
      IOUT=REFSP
      IF(JJ.EQ.(6+kdel))  CALL AOT(LCHAN1,IOUT,ERSTAT)
      IF(JJ.EQ.(5+IFSAMPLE)) THEN
      SPEED(J)=A(6+KDEL)
      TORQUE(J)=B(6+KDEL)
      TIME(J)=T(6+KDEL)
      DO 981 JU=1,5
      Y1(JU)=Y1(JU+IFSAMPLE)
      CON1(JU)=CON1(JU+IFSAMPLE)
      Y(JU)=Y(JU+IFSAMPLE)
      DE(JU)=DE(JU+IFSAMPLE)
      ER4(JU)=ER4(JU+IFSAMPLE)
      ER2(JU)=ER2(JU+IFSAMPLE)
      ER3(JU)=ER3(JU+IFSAMPLE)
981   DIS(JU)=DIS(JU+IFSAMPLE)
      ENDIF
333   CALL GETTIM(IT1,IT2,IT3,IT4)
```

```
      T(JJ)=3600.0*IT1+60.0*IT2+1.0*IT3+0.01*IT4
      IF(T(JJ).LT.(H*IJ)) GO TO 333
1     CONTINUE
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      GO TO 89
11    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      WRITE(*,*) 'LIMIT SWITCH OPERATION'
36    CALL AING(LCHAN,PVOLT,ERSTAT)
      WRITE(*,*) 'ACTUATOR POSITION:'
      ACTPOSI=10.0*PVOLT(3)/4095.0-5.0
      WRITE(*,*) ACTPOSI
      WRITE(*,*) 'DO YOU WANT TO GIVE A STEP INPUT TO THE ACTUATOR?'
      WRITE(*,*) '                  SELECT 1:YES  2:NO'
      READ(*,*) I1
      IF(I1.EQ.2) GO TO 100
      IF(I1.EQ.1) GO TO 35
35    WRITE(*,*) 'INPUT COMMAND FOR ACTUATOR'
      READ(*,555) STEPS
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STEPS,CARR
      CLOSE(UNIT=9)
      GO TO 36

89    WRITE(*,*) 'SELECT: 1)EXIT'
      WRITE(*,*) '        2)TRANSIENT ENGINE CYCLE'
      WRITE(*,*) '        3)CREATE AN OUTPUT FILE'
      WRITE(*,*) '        4)SHUT-DOWN THE ENGINE/DYNO'
      READ(*,*) M1
      IF(M1.EQ.1) GO TO 6
      IF(M1.EQ.2) GO TO 50
      IF(M1.EQ.3) GO TO 88
      IF(M1.EQ.4) CALL STOP
      GO TO 89
      GO TO 6
88    WRITE(*,*) 'AN OUTPUT FILE NAME'
      READ(*,555) FILEN
```

```
555     FORMAT(A)
        OPEN(UNIT=1,FILE=FILEN)
        SPEED(1201)=SPEED(1200)
        DO 39, J=1,1199
        IF(SPREF(J+1).EQ.SPREF(J)) GO TO 39
        IF(TR(J).LT.0.0) GO TO 39
        TORQUE(J+1)=TORQUE(J+1)+0.2748081*(SPEED(J+2)-SPEED(J))
39      WRITE(1,110) TIME(J+1),SPREF(J),TR(J),SPEED(J+1),TORQUE(J+1)
110     FORMAT(1X,F12.5,1X,F12.5,1X,F12.5,1X,F12.5,1X,F12.5)
        CLOSE(UNIT=1)
        GO TO 89
6       continue
        STOP
        end

        SUBROUTINE START(ERCODE)
        INTEGER*2 ERSTAT,LCHAN,LCHAN1,PVOLT(3),iint,ERCODE
        INTEGER*1 CARR
        CHARACTER*9 STO,MOVEP,MOVEN
        DATA CARR /1*13/
        LCHAN=1
        LCHAN1=2
        MOVEP='      M+5'
        MOVEN='      M-5'
        STO='        @'
        iint=2168
50      CALL AOT(LCHAN1,iint,ERSTAT)
        III=0
52      CALL AING(LCHAN,PVOLT,ERSTAT)
        SAFTY=PVOLT(3)
        REFTOR=PVOLT(2)
        SAFTY=10.0*SAFTY/4095.0-5.0
        REFTOR=10.0*REFTOR/4095.0-5.0
        REFTOR=48.43213*REFTOR+0.404608
        ANEG=-10.0
        IF(REFTOR.GT.ANEG) GO TO 51
        IF(SAFTY.LT.2.2) GO TO 53
        IF(III.EQ.1) GO TO 52
        OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
        WRITE (9) MOVEP,CARR
```

```
      CLOSE(UNIT=9)
      III=1
      GO TO 52
51    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      ERCODE=0.0
      GO TO 100
53    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      IEE=0
55    CALL AING(LCHAN,PVOLT,ERSTAT)
      ACTPOS=PVOLT(3)
      ACTPOS=10.0*ACTPOS/4095.0-5.0
      IF(ACTPOS.GT.4.35) GO TO 56
      IF(IEE.EQ.1) GO TO 55
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) MOVEN,CARR
      CLOSE(UNIT=9)
      IEE=1
      GO TO 55
56    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      iint=0
      CALL AOT(LCHAN1,iint,ERSTAT)
      ERCODE=1.0
      WRITE(*,*) '*************************************************'
      WRITE(*,*) '*                !!!  WARNING !!!                *'
      WRITE(*,*) '*   ENGINE DOES NOT FIRE!  SOMETHING IS WRONG    *'
      WRITE(*,*) '*************************************************'
100   CONTINUE
      RETURN
      End

      SUBROUTINE STOP
      INTEGER*2 ERSTAT,LCHAN,LCHAN1,PVOLT(3),iint
      INTEGER*1 CARR
      CHARACTER*9 STO,MOVEP,MOVEN
```

```
      DATA CARR /1*13/
      LCHAN=1
      LCHAN1=2
      MOVEP='      M+5'
      MOVEN='      M-5'
      STO='        @'
      IEE=0
91    CALL AING(LCHAN,PVOLT,ERSTAT)
      ACTPOS=PVOLT(3)
      ACTPOS=10.0*ACTPOS/4095.0-5.0
      IF(ACTPOS.GT.4.35) GO TO 92
      IF(IEE.EQ.1) GO TO 91
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) MOVEN,CARR
      CLOSE(UNIT=9)
      IEE=1
      GO TO 91
92    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      iint=2168
      CALL AOT(LCHAN1,iint,ERSTAT)
      WRITE(*,*) 'ALLOW ENGINE TO OPERATE AT THIS SPEED FOR 5 MINUTES'
      WRITE(*,*) 'WHEN YOU ARE READY PRESS 1'
      READ(*,*) ISHUT
      IF(ISHUT.EQ.1) GO TO 93
93    iint=0
      CALL AOT(LCHAN1,iint,ERSTAT)
      RETURN
      END

      SUBROUTINE IDENT(THETAP,THETAD,NNUM,NDEN,NDNUM,NDDEN,KP,KD)
      real*4 Y(500),yd(500),thetap(6),thetad(6)
      INTEGER*2 ERSTAT,LCHAN,LCHAN1
      INTEGER*2 PVOLT(3)
      INTEGER*2 IT1,IT2,IT3,IT4
      INTEGER*1 CARR
      INTEGER*2 IAC1,ISP1,isp
      CHARACTER*9 STEP,STEPP,STO
      CHARACTER*9 STEP1,MOVEP,MOVEN
```

```
      DATA CARR /1*13/
      LCHAN=1
      LCHAN1=2
      STEP='    +    '
      STEPP='    -    '
      STO='        @'
      MOVEP='      M+5'
      MOVEN='      M-5'
      isp=1.808925*1300.0
      refsp1=1200.0
      ISP1=1.808925*REFSP1
      iac1=100
      IF(IAC1.GT.0) WRITE(STEP(6:9),'(I4.4)') IAC1
      IF(IAC1.LE.0) WRITE(STEPP(6:9),'(I4.4)') ABS(IAC1)
      IF(IAC1.GT.0) STEP1=STEP
      IF(IAC1.LE.0) STEP1=STEPP
      call aot(lchan1,isp,erstat)
      call delay(0,5,0)
      call aing(lchan,pvolt,erstat)
      tor=10.0*pvolt(2)/4095.0-5.0
      tor=48.43213*tor+0.4046
      if(tor.gt.20.0) then
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) MOVEN,CARR
      CLOSE(UNIT=9)
1     CALL AING(LCHAN,PVOLT,ERSTAT)
      TOR1=10.0*PVOLT(2)/4095.0-5.0
      TOR1=48.43213*TOR1+0.4046
      IF(TOR1.GT.20.0) GO TO 1
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      ENDIF
      if(tor.Lt.0.0) then
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) MOVEP,CARR
      CLOSE(UNIT=9)
2     CALL AING(LCHAN,PVOLT,ERSTAT)
      TOR1=10.0*PVOLT(2)/4095.0-5.0
      TOR1=48.43213*TOR1+0.4046
```

```
      IF(TOR1.LT.0.0) GO TO 2
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      ENDIF
      call delay(0,5,0)
      CALL SETTIM(0,0,0,0)
      DO 100 J=1,500
      CALL AING(LCHAN,PVOLT,ERSTAT)
      Y(J)=10.0*PVOLT(2)/4095.0-5.0
      Y(J)=48.43213*y(j)+0.4046
      IF(J.EQ.250) THEN
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STEP1,CARR
      CLOSE(UNIT=9)
      ENDIF
9     CALL GETTIM(IT1,IT2,IT3,IT4)
      T=3600.0*IT1+60.0*IT2+1.0*IT3+0.01*IT4
      IF(T.LT.(j*0.02)) GO TO 9
100   CONTINUE
      iac1=-100
      IF(IAC1.GT.0) WRITE(STEP(6:9),'(I4.4)') IAC1
      IF(IAC1.LE.0) WRITE(STEPP(6:9),'(I4.4)') ABS(IAC1)
      IF(IAC1.GT.0) STEP1=STEP
      IF(IAC1.LE.0) STEP1=STEPP
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STEP1,CARR
      CLOSE(UNIT=9)
      call delay(0,5,0)
      CALL SETTIM(0,0,0,0)
      DO 200 J=1,500
      CALL AING(LCHAN,PVOLT,ERSTAT)
      YD(J)=10.0*PVOLT(2)/4095.0-5.0
      YD(J)=48.43213*yD(j)+0.4046
      IF(J.EQ.250) THEN
      CALL AOT(LCHAN1,ISP1,ERSTAT)
      ENDIF
99    CALL GETTIM(IT1,IT2,IT3,IT4)
      T=3600.0*IT1+60.0*IT2+1.0*IT3+0.01*IT4
      IF(T.LT.(j*0.02)) GO TO 99
```

```
200     CONTINUE
        CALL AOT(LCHAN1,ISP1,ERSTAT)
        call delay(0,5,0)
        call aing(lchan,pvolt,erstat)
        tor=10.0*pvolt(2)/4095.0-5.0
        tor=48.43213*tor+0.4046
        if(tor.gt.30.0) then
        OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
        WRITE (9) MOVEN,CARR
        CLOSE(UNIT=9)
3       CALL AING(LCHAN,PVOLT,ERSTAT)
        TOR1=10.0*PVOLT(2)/4095.0-5.0
        TOR1=48.43213*TOR1+0.4046
        IF(TOR1.GT.30.0) GO TO 3
        OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
        WRITE (9) STO,CARR
        CLOSE(UNIT=9)
        ENDIF
        if(tor.Lt.0.0) then
        OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
        WRITE (9) MOVEP,CARR
        CLOSE(UNIT=9)
4       CALL AING(LCHAN,PVOLT,ERSTAT)
        TOR1=10.0*PVOLT(2)/4095.0-5.0
        TOR1=48.43213*TOR1+0.4046
        IF(TOR1.LT.0.0) GO TO 4
        OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
        WRITE (9) STO,CARR
        CLOSE(UNIT=9)
        ENDIF
        ACTIN=100.0
        call parest(y,actin,nnum,nden,thetap,kp,ermin1)
        call parest(yd,actin,ndnum,ndden,thetad,kd,ermin2)
999     continue
        RETURN
        END

        subroutine parest(y,actin,nnum,nden,thetat,k1,ermin1)
        real*4 Y(500),thetat(6)
        real*4 theta1(6),theta2(6),theta3(6),theta4(6)
```

```
      real*4 theta5(6),theta6(6)
      do 1 j=1,500
      IF(J.EQ.1) YY1=Y(1)
      Y(J)=Y(J)-YY1
1     continue
      ermin1=10.0**10
      call tmodel1(y,actin,2,theta1,ermin,k)
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=0
      nden=1
      thetat=theta1
      k1=k
      endif
      call tmodel2(y,actin,3,theta2,ermin,k)
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=0
      nden=2
      thetat=theta2
      k1=k
      endif
      call tmodel3(y,actin,4,theta3,ermin,k)
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=1
      nden=2
      thetat=theta3
      k1=k
      endif
      call tmodel4(y,actin,4,theta4,ermin,k)
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=0
      nden=3
      thetat=theta4
      k1=k
      endif
      call tmodel5(y,actin,5,theta5,ermin,k)
```

```
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=1
      nden=3
      thetat=theta5
      k1=k
      endif
      call tmodel6(y,actin,6,theta6,ermin,k)
      If(ermin.lt.ermin1) then
      ermin1=ermin
      nnum=2
      nden=3
      thetat=theta6
      k1=k
      endif
      RETURN
      END

      subroutine tmodel1(y,actin,n,theta,ermin,k)
      real*4 y(500),theta(6),ofdiag(15),fi(6)
      real*4 err(30),diag(6),u(500),lam,yest(500)
      real*4 b(500),f0(500),f1(500),s1(500),ynew
      real*4 bf1,af1,bs1,bs2,as1,as2
      istop=0
      DO 702 K=1,30
4     DO 300 I=1,n
      diag(i)=10.0**10
      theta(i)=0.0
300   continue
      do 302 i=1,(n*(n-1)/2)
      ofdiag(i)=0.0
302   continue
      do 303 j=1,500
      u(j)=0.0
      if(j.ge.(250+k)) u(j)=actin
      if(j.gt.(250+k)) b(j)=y(j)
      if(j.le.(250+k)) b(j)=0.0
303   continue
      lam=1.0
      bf1=0.0198013267
```

```
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
      DO 1 J=3,470
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      s1(j)=bs1*u(j-1)+bs2*u(j-2)-as1*s1(j-1)-as2*s1(j-2)
      ynew=f0(j)-f1(j)
      fi(1)=-f1(j)
      fi(2)=s1(j)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1     CONTINUE
      if(istop.eq.1) go to 5
      Do 7 j=1,2
      if(theta(j).le.0.0) go to 8
7     continue
      er=0.0
      hsamp=0.02
      call convert0(theta(1),theta(2),
     @hsamp,bz1,az1)
      DO 2 J=250+k,470
      yest(j)=-az1*yest(j-1)+bz1*
     @u(j-1)
      er=er+abs(b(j)-yest(j))**2
2     continue
      err(k)=er
8     do 6 j=1,2
      if(theta(j).le.0.0) er=10.0**10
      if(theta(j).le.0.0) err(k)=10.0**10
6     continue
702   CONTINUE
      istop=1
      ermin=10.0**10
      do 3 j=1,30
      if(err(j).lt.ermin) k=j
      if(err(j).lt.ermin) ermin=err(j)
3     continue
      go to 4
```

```
5       continue
        return
        END

        subroutine tmodel2(y,actin,n,theta,ermin,k)
        real*4 y(500),theta(6),ofdiag(15),fi(6)
        real*4 err(30),diag(6),u(500),lam,yest(500)
        real*4 b(500),f0(500),f1(500),f2(500)
        real*4 s2(500),ynew
        real*4 bf1,af1,bs1,bs2,as1,as2,bt1,bt2,bt3,at1,at2,at3
        istop=0
        DO 702 K=1,30
4       DO 300 I=1,n
        diag(i)=10.0**10
        theta(i)=0.0
300     continue
        do 302 i=1,(n*(n-1)/2)
        ofdiag(i)=0.0
302     continue
        do 303 j=1,500
        u(j)=0.0
        if(j.ge.(250+k)) u(j)=actin
        if(j.gt.(250+k)) b(j)=y(j)
        if(j.le.(250+k)) b(j)=0.0
303     continue
        lam=1.0

        bf1=0.0198013267
        af1=-.980198673306755
        bs1=0.197353227109592e-3
        bs2=0.19473931170301e-3
        as1=-1.960397346613511
        as2=0.9607894391522323
        bt1=0.131349244814061e-5
        bt2=0.517578714848144e-5
        bt3=0.127467285840344e-5
        at1=-2.940596019920266
        at2=2.88236831745697
        at3=-0.941764533584249
        DO 1 J=4,470
```

```
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      f2(j)=bt1*b(j-1)+bt2*b(j-2)+bt3*b(j-3)-at1*f2(j-1)-at2*f2(j-2)
     @-at3*f2(j-3)
      s2(j)=bt1*u(j-1)+bt2*u(j-2)+bt3*u(j-3)-at1*s2(j-1)-at2*s2(j-2)
     @-at3*s2(j-3)
      ynew=f0(j)-2.0*f1(j)+f2(j)
      fi(1)=f2(j)-f1(j)
      fi(2)=-f2(j)
      fi(3)=s2(j)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1     CONTINUE
      if(istop.eq.1) go to 5
      Do 7 j=1,3
      if(theta(j).le.0.0) go to 8
7     continue
      er=0.0
      hsamp=0.02
      call convert1(theta(1),theta(2),theta(3),
     @hsamp,bz1,bz2,az1,az2)
      DO 2 J=250+k,470
      yest(j)=-az1*yest(j-1)-az2*yest(j-2)+bz1*
     @u(j-1)+bz2*u(j-2)
      er=er+abs(b(j)-yest(j))**2
2     continue
      err(k)=er
8     do 6 j=1,3
      if(theta(j).le.0.0) er=10.0**10
      if(theta(j).le.0.0) err(k)=10.0**10
6     continue
702   CONTINUE
      istop=1
      ermin=10.0**10
      do 3 j=1,30
      if(err(j).lt.ermin) k=j
      if(err(j).lt.ermin) ermin=err(j)
3     continue
      go to 4
5     continue
      return
```

```
      END

      subroutine tmodel3(y,actin,n,theta,ermin,k)
      real*4 y(500),theta(6),ofdiag(15),fi(6)
      real*4 err(30),diag(6),u(500),lam,yest(500)
      real*4 b(500),f0(500),f1(500),f2(500)
      real*4 s2(500),s1(500),ynew
      real*4 bf1,af1,bs1,bs2,as1,as2,bt1,bt2,bt3,at1,at2,at3
      istop=0
      DO 702 K=1,30
4     DO 300 I=1,n
      diag(i)=10.0**10
      theta(i)=0.0
300   continue
      do 302 i=1,(n*(n-1)/2)
      ofdiag(i)=0.0
302   continue
      do 303 j=1,500
      u(j)=0.0
      if(j.ge.(250+k)) u(j)=actin
      if(j.gt.(250+k)) b(j)=y(j)
      if(j.le.(250+k)) b(j)=0.0
303   continue
      lam=1.0
      bf1=0.0198013267
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
      bt1=0.131349244814061e-5
      bt2=0.517578714848144e-5
      bt3=0.127467285840344e-5
      at1=-2.940596019920266
      at2=2.88236831745697
      at3=-0.941764533584249
      DO 1 J=4,470
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      f2(j)=bt1*b(j-1)+bt2*b(j-2)+bt3*b(j-3)-at1*f2(j-1)-at2*f2(j-2)
```

```
     @-at3*f2(j-3)
        s1(j)=bs1*u(j-1)+bs2*u(j-2)-as1*s1(j-1)-as2*s1(j-2)
        s2(j)=bt1*u(j-1)+bt2*u(j-2)+bt3*u(j-3)-at1*s2(j-1)-at2*s2(j-2)
     @-at3*s2(j-3)
        ynew=f0(j)-2.0*f1(j)+f2(j)
        fi(1)=f2(j)-f1(j)
        fi(2)=-f2(j)
        fi(3)=s1(j)-s2(j)
        fi(4)=s2(j)
        CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1       CONTINUE
        if(istop.eq.1) go to 5
        Do 7 j=1,4
        if(theta(j).le.0.0) go to 8
7       continue
        er=0.0
        hsamp=0.02
        call convert2(theta(1),theta(2),theta(3),theta(4),
     @hsamp,bz1,bz2,az1,az2)
        DO 2 J=250+k,470
        yest(j)=-az1*yest(j-1)-az2*yest(j-2)+bz1*
     @u(j-1)+bz2*u(j-2)
        er=er+abs(b(j)-yest(j))**2
2       continue
        err(k)=er
8       do 6 j=1,4
        if(theta(j).le.0.0) er=10.0**10
        if(theta(j).le.0.0) err(k)=10.0**10
6       continue
702     CONTINUE
        istop=1
        ermin=10.0**10
        do 3 j=1,30
        if(err(j).lt.ermin) k=j
        if(err(j).lt.ermin) ermin=err(j)
3       continue
        go to 4
5       continue
        return
        END
```

```
      subroutine tmodel4(y,actin,n,theta,ermin,k)
      real*4 y(500),theta(6),ofdiag(15),fi(6)
      real*4 err(30),diag(6),u(500),lam,yest(500)
      real*4 b(500),f0(500),f1(500),f2(500),f3(500)
      real*4 s3(500),ynew
      real*4 bf1,af1,bs1,bs2,as1,as2,bt1,bt2,bt3,at1,at2,at3
      real*4 bfo1,bfo2,bfo3,bfo4,afo1,afo2,afo3,afo4
      istop=0
      DO 702 K=1,30
4     DO 300 I=1,n
      diag(i)=10.0**10
      theta(i)=0.0
300   continue
      do 302 i=1,(n*(n-1)/2)
      ofdiag(i)=0.0
302   continue
      do 303 j=1,500
      u(j)=0.0
      if(j.ge.(250+k)) u(j)=actin
      if(j.gt.(250+k)) b(j)=y(j)
      if(j.le.(250+k)) b(j)=0.0
303   continue
      lam=1.0
      bf1=0.0198013267
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
      bt1=0.131349244814061e-5
      bt2=0.517578714848144e-5
      bt3=0.127467285840344e-5
      at1=-2.940596019920266
      at2=2.88236831745697
      at3=-0.941764533584249
      bfo1=0.065608839472375e-7
      bfo2=0.710248151403192e-7
      bfo3=0.698974599311342e-7
      bfo4=0.062533999506398e-7
```

```
      afo1=-3.920794693227021
      afo2=5.764736634913939
      afo3=-3.767058134336995
      afo4=0.923116346386636
      DO 1 J=5,470
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      f2(j)=bt1*b(j-1)+bt2*b(j-2)+bt3*b(j-3)-at1*f2(j-1)-at2*f2(j-2)
     @-at3*f2(j-3)
      f3(j)=bfo1*b(j-1)+bfo2*b(j-2)+bfo3*b(j-3)+bfo4*b(j-4)-
     @afo1*f3(j-1)-afo2*f3(j-2)-afo3*f3(j-3)-afo4*f3(j-4)
      s3(j)=bfo1*u(j-1)+bfo2*u(j-2)+bfo3*u(j-3)+bfo4*u(j-4)-
     @afo1*s3(j-1)-afo2*s3(j-2)-afo3*s3(j-3)-afo4*s3(j-4)
      ynew=f0(j)-3.0*f1(j)+3.0*f2(j)-f3(j)
      fi(1)=2.0*f2(j)-f1(j)-f3(j)
      fi(2)=f3(j)-f2(j)
      fi(3)=-f3(j)
      fi(4)=S3(j)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1     CONTINUE
      if(istop.eq.1) go to 5
      Do 7 j=1,4
      if(theta(j).le.0.0) go to 8
7     continue
      er=0.0
      hsamp=0.02
      call convert3(theta(1),theta(2),theta(3),theta(4),

     @hsamp,bz1,bz2,bz3,az1,az2,az3)
      DO 2 J=250+k,470
      yest(j)=-az1*yest(j-1)-az2*yest(j-2)-az3*yest(j-3)+bz1*
     @u(j-1)+bz2*u(j-2)+bz3*u(j-3)
      er=er+abs(b(j)-yest(j))**2
2     continue
      err(k)=er
8     do 6 j=1,4
      if(theta(j).le.0.0) er=10.0**10
      if(theta(j).le.0.0) err(k)=10.0**10
6     continue
702   CONTINUE
```

```
      istop=1
      ermin=10.0**10
      do 3 j=1,30
      if(err(j).lt.ermin) k=j
      if(err(j).lt.ermin) ermin=err(j)
3     continue
      go to 4
5     continue
      return
      END

      subroutine tmodel5(y,actin,n,theta,ermin,k)
      real*4 y(500),theta(6),ofdiag(15),fi(6)
      real*4 err(30),diag(6),u(500),lam,yest(500)
      real*4 b(500),f0(500),f1(500),f2(500),f3(500)
      real*4 s2(500),s3(500),ynew
      real*4 bf1,af1,bs1,bs2,as1,as2,bt1,bt2,bt3,at1,at2,at3
      real*4 bfo1,bfo2,bfo3,bfo4,afo1,afo2,afo3,afo4
      istop=0
      DO 702 K=1,30
4     DO 300 I=1,n
      diag(i)=10.0**3
      theta(i)=0.0
300   continue
      do 302 i=1,(n*(n-1)/2)
      ofdiag(i)=0.0
302   continue
      do 303 j=1,500
      u(j)=0.0
      if(j.ge.(250+k)) u(j)=actin
      if(j.gt.(250+k)) b(j)=y(j)
      if(j.le.(250+k)) b(j)=0.0
303   continue
      lam=1.0
      bf1=0.0198013267
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
```

```
      bt1=0.131349244814061e-5
      bt2=0.517578714848144e-5
      bt3=0.127467285840344e-5
      at1=-2.940596019920266
      at2=2.88236831745697
      at3=-0.941764533584249
      bfo1=0.065608839472375e-7
      bfo2=0.710248151403192e-7
      bfo3=0.698974599311342e-7
      bfo4=0.062533999506398e-7
      afo1=-3.920794693227021
      afo2=5.764736634913939
      afo3=-3.767058134336995
      afo4=0.923116346386636
      DO 1 J=5,470
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      f2(j)=bt1*b(j-1)+bt2*b(j-2)+bt3*b(j-3)-at1*f2(j-1)-at2*f2(j-2)
     @-at3*f2(j-3)
      f3(j)=bfo1*b(j-1)+bfo2*b(j-2)+bfo3*b(j-3)+bfo4*b(j-4)-
     @afo1*f3(j-1)-afo2*f3(j-2)-afo3*f3(j-3)-afo4*f3(j-4)
      s2(j)=bt1*u(j-1)+bt2*u(j-2)+bt3*u(j-3)-at1*s2(j-1)-at2*s2(j-2)
     @-at3*s2(j-3)
      s3(j)=bfo1*u(j-1)+bfo2*u(j-2)+bfo3*u(j-3)+bfo4*u(j-4)-
     @afo1*s3(j-1)-afo2*s3(j-2)-afo3*s3(j-3)-afo4*s3(j-4)
      ynew=f0(j)-3.0*f1(j)+3.0*f2(j)-f3(j)
      fi(1)=2.0*f2(j)-f1(j)-f3(j)
      fi(2)=f3(j)-f2(j)
      fi(3)=-f3(j)
      fi(4)=S2(j)-S3(j)
      fi(5)=s3(j)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1     CONTINUE
      if(istop.eq.1) go to 5
      do 7 j=1,5
      if(theta(j).le.0.0) go to 8
7     continue
      er=0.0
      hsamp=0.02
      call convert4(theta(1),theta(2),theta(3),theta(4),theta(5),
```

```
     @hsamp,bz1,bz2,bz3,az1,az2,az3)
        DO 2 J=250+k,470
        yest(j)=-az1*yest(j-1)-az2*yest(j-2)-az3*yest(j-3)+bz1*
     @u(j-1)+bz2*u(j-2)+bz3*u(j-3)
        er=er+abs(b(j)-yest(j))**2
2       continue
        err(k)=er
8       do 6 j=1,5
        if(theta(j).le.0.0) err(k)=10.0**10
        if(theta(j).le.0.0) er=10.0**10
6       continue
702     CONTINUE
        istop=1
        ermin=10.0**10
        do 3 j=1,30
        if(err(j).lt.ermin) k=j
        if(err(j).lt.ermin) ermin=err(j)
3       continue
        go to 4
5       continue
        return
        END

        subroutine tmodel6(y,actin,n,theta,ermin,k)
        real*4 y(500),theta(6),ofdiag(15),fi(6)
        real*4 err(30),diag(6),u(500),lam,yest(500)
        real*4 b(500),f0(500),f1(500),f2(500),f3(500)
        real*4 s1(500),s2(500),s3(500),ynew
        real*4 bf1,af1,bs1,bs2,as1,as2,bt1,bt2,bt3,at1,at2,at3
        real*4 bfo1,bfo2,bfo3,bfo4,afo1,afo2,afo3,afo4
        istop=0
        DO 702 K=1,30
4       DO 300 I=1,n
        diag(i)=10.0**10
        theta(i)=0.0
300     continue
        do 302 i=1,(n*(n-1)/2)
        ofdiag(i)=0.0
302     continue
        do 303 j=1,500
```

```
      u(j)=0.0
      if(j.ge.(250+k)) u(j)=actin
      if(j.gt.(250+k)) b(j)=y(j)
      if(j.le.(250+k)) b(j)=0.0
303   continue
      lam=1.0
      bf1=0.0198013267
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
      bt1=0.131349244814061e-5
      bt2=0.517578714848144e-5
      bt3=0.127467285840344e-5
      at1=-2.940596019920266
      at2=2.88236831745697
      at3=-0.941764533584249
      bfo1=0.065608839472375e-7
      bfo2=0.710248151403192e-7
      bfo3=0.698974599311342e-7
      bfo4=0.062533999506398e-7
      afo1=-3.920794693227021
      afo2=5.764736634913939
      afo3=-3.767058134336995
      afo4=0.923116346386636
      DO 1 J=5,470
      f0(j)=bf1*b(j-1)-af1*f0(j-1)
      f1(j)=bs1*b(j-1)+bs2*b(j-2)-as1*f1(j-1)-as2*f1(j-2)
      f2(j)=bt1*b(j-1)+bt2*b(j-2)+bt3*b(j-3)-at1*f2(j-1)-at2*f2(j-2)
     @-at3*f2(j-3)
      f3(j)=bfo1*b(j-1)+bfo2*b(j-2)+bfo3*b(j-3)+bfo4*b(j-4)-
     @afo1*f3(j-1)-afo2*f3(j-2)-afo3*f3(j-3)-afo4*f3(j-4)
      s1(j)=bs1*u(j-1)+bs2*u(j-2)-as1*s1(j-1)-as2*s1(j-2)
      s2(j)=bt1*u(j-1)+bt2*u(j-2)+bt3*u(j-3)-at1*s2(j-1)-at2*s2(j-2)
     @-at3*s2(j-3)
      s3(j)=bfo1*u(j-1)+bfo2*u(j-2)+bfo3*u(j-3)+bfo4*u(j-4)-
     @afo1*s3(j-1)-afo2*s3(j-2)-afo3*s3(j-3)-afo4*s3(j-4)
      ynew=f0(j)-3.0*f1(j)+3.0*f2(j)-f3(j)
      fi(1)=2.0*f2(j)-f1(j)-f3(j)
```

```
      fi(2)=f3(j)-f2(j)
      fi(3)=-f3(j)
      fi(4)=S1(j)-2.0*s2(j)+s3(j)
      fi(5)=s2(j)-s3(j)
      fi(6)=s3(j)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
1     CONTINUE
      if(istop.eq.1) go to 5
      Do 7 j=1,6
      if(theta(j).le.0.0) go to 8
7     continue
      er=0.0
      hsamp=0.02
      call convert5(theta(1),theta(2),theta(3),theta(4),theta(5),
     @theta(6),hsamp,bz1,bz2,bz3,az1,az2,az3)
      DO 2 J=250+k,470
      yest(j)=-az1*yest(j-1)-az2*yest(j-2)-az3*yest(j-3)+bz1*
     @u(j-1)+bz2*u(j-2)+bz3*u(j-3)
      er=er+abs(b(j)-yest(j))**2
2     continue
      err(k)=er
8     do 6 j=1,6
      if(theta(j).le.0.0) er=10.0**10
      if(theta(j).le.0.0) err(k)=10.0**10
6     continue
702   CONTINUE
      istop=1
      ermin=10.0**10
      do 3 j=1,30
      if(err(j).lt.ermin) k=j
      if(err(j).lt.ermin) ermin=err(j)
3     continue
      go to 4
5     continue
      return
      END


      subroutine rud(n,ynew,fi,theta,diag,ofdiag,lam)
      integer n,kf,ku,i,j,jj
```

```
      real*4 ynew
      real*4 fi(n),theta(n),diag(n),ofdiag(n*(n-1)/2),lam
      real*4 perr,fj,vj,alphaj,ajlast,pj,w,k(10)
      perr=ynew
      do 10 i=1,n
      perr=perr-theta(i)*fi(i)
10    continue
      fj=fi(1)
      vj=diag(1)*fj
      k(1)=vj
      alphaj=1.0+vj*fj
      diag(1)=diag(1)/alphaj/lam
      if(n.le.1) goto 40
      kf=0
      ku=0
      do 30 j=2,n
      fj=fi(j)
      jj=j-1
      do 20 i=1,jj
      kf=kf+1
      fj=fj+fi(i)*ofdiag(kf)
20    continue
      vj=fj*diag(j)
      k(j)=vj
      ajlast=alphaj
      alphaj=ajlast+vj*fj
      diag(j)=diag(j)*ajlast/alphaj/lam
      pj=-fj/ajlast
      jj=j-1
      do 30 i=1,jj
      ku=ku+1
      w=ofdiag(ku)+k(i)*pj
      k(i)=k(i)+ofdiag(ku)*vj
      ofdiag(ku)=w
30    continue
40    continue
      do 50 i=1,n
      theta(i)=theta(i)+perr*k(i)/alphaj
50    continue
      return
```

```
      end

      subroutine convert0(t1,t2,h,b1,a1)
      real*4 t1,t2
      a1=-exp(-t1*h)
      b1=t2/t1*(1.0-exp(-t1*h))
      return
      end

      subroutine convert1(t1,t2,t3,h,b1,b2,a1,a2)
      real*4 t1,t2,t3
      w0=sqrt(t2)
      zeta=t1*0.5/w0
      if(zeta.lt.1.0) then
      w=w0*sqrt(1.0-zeta**2)
      alfa=exp(-zeta*w0*h)
      beta=cos(w*h)
      gamma=sin(w*h)
      b1=t3*(1.0-alfa*(beta+w0*gamma*zeta/w))/(w0**2)
      b2=t3*(alfa**2+alfa*(zeta*w0*gamma/w-beta))/(w0**2)
      a1=-2.0*alfa*beta
      a2=alfa**2
      else
      a=0.5*t1-0.5*sqrt(t1**2-4.0*t2)
      b=0.5*t1+0.5*sqrt(t1**2-4.0*t2)
      ab=a*b
      b1=t3*(b*(1.0-exp(-a*h))-a*(1.0-exp(-b*h)))/((b-a)*t2)
      b2=t3*(a*(1.0-exp(-b*h))*exp(-a*h)-b*(1.0-exp(-a*h))*exp(-b*h)
@)/(t2*(b-a))
      a1=-exp(-a*h)-exp(-b*h)
      a2=exp(-a*h-b*h)
      endif
      return
      end

      subroutine convert2(t1,t2,t3,t4,h,b1,b2,a1,a2)
      real*4 t1,t2,t3,t4
      w0=sqrt(t2)
      zeta=t1*0.5/w0
      if(zeta.lt.1.0) then
```

```
      w=w0*sqrt(1.0-zeta**2)
      alfa=exp(-zeta*w0*h)
      beta=cos(w*h)
      gamma=sin(w*h)
      b1=t3*gamma*exp(-zeta*w0*h)/w+t4*(1.0-alfa*(beta+zeta*w0*
     @gamma/w))/(w0**2)
      b2=-t3*gamma*exp(-zeta*w0*h)/w+t4*(alfa**2+alfa*(zeta*w0*
     @gamma/w-beta))/(w0**2)
      a1=-2.0*alfa*beta
      a2=alfa**2
      else
      a=0.5*t1+0.5*sqrt(t1**2-4.0*t2)
      b=0.5*t1-0.5*sqrt(t1**2-4.0*t2)
      ab=a*b
      c=t4/t3
      b1=t3*(exp(-b*h)-exp(-a*h)+c*(1.0-exp(-b*h))/b-c*(1.0
     @-exp(-a*h))/a)/(a-b)
      b2=t3*(c*exp(-a*h-b*h)/ab+(b-c)*exp(-a*h)/(b*(a-b))+
     @(c-a)*exp(-b*h)/(a*(a-b)))
      a1=-exp(-a*h)-exp(-b*h)
      a2=exp(-a*h-b*h)
      endif
      return
      end

      subroutine convert3(t1,t2,t3,t4,h,b1,b2,b3,a1,a2,a3)
      real*4 t1,t2,t3,t4,p1,p2,p3,rn2,rn3
      call roots(t1,t2,t3,p1,p2,p3)
      rn1=t4/(p1**2-p1*p2+p3)
      rn2=-rn1
      rn3=rn1*(p1-p2)
      call convert2(p2,p3,rn2,rn3,h,rb1,rb2,ra1,ra2)
      c=rn1*(1.0-exp(-p1*h))/p1
      d=-exp(-p1*h)
      b1=c+rb1
      b2=c*ra1+rb1*d+rb2
      b3=ra2*c+rb2*d
      a1=d+ra1
      a2=d*ra1+ra2
      a3=ra2*d
```

```
return
end

subroutine convert4(t1,t2,t3,t4,t5,h,b1,b2,b3,a1,a2,a3)
real*4 t1,t2,t3,t4,t5,p1,p2,p3,rn2,rn3
call roots(t1,t2,t3,p1,p2,p3)
rn1=(-t4*p1+t5)/(p1**2-p1*p2+p3)
rn2=-rn1
rn3=(t5-p3*rn1)/p1
call convert2(p2,p3,rn2,rn3,h,rb1,rb2,ra1,ra2)
c=rn1*(1.0-exp(-p1*h))/p1
d=-exp(-p1*h)
b1=c+rb1
b2=c*ra1+rb1*d+rb2
b3=ra2*c+rb2*d
a1=d+ra1
a2=d*ra1+ra2
a3=ra2*d
return
end

subroutine convert5(t1,t2,t3,t4,t5,t6,h,b1,b2,b3,a1,a2,a3)
real*4 t1,t2,t3,t4,t5,t6,p1,p2,p3,rn2,rn3
call roots(t1,t2,t3,p1,p2,p3)
rn1=(t4*p1*p1-t5*p1+t6)/(p1**2-p1*p2+p3)
rn2=t4-rn1
rn3=(t6-p3*rn1)/p1
call convert2(p2,p3,rn2,rn3,h,rb1,rb2,ra1,ra2)
c=rn1*(1.0-exp(-p1*h))/p1
d=-exp(-p1*h)
b1=c+rb1
b2=c*ra1+rb1*d+rb2
b3=ra2*c+rb2*d
a1=d+ra1
a2=d*ra1+ra2
a3=ra2*d
return
end

subroutine roots(b,c,d,p1,p2,p3)
```

```
      real*4 b,c,d,p1,p2,p3
      p=c-b*b/3.0
      q1=(2.0*b*b*b-9.0*b*c+27.0*d)/27.0
      del=0.25*q1*q1+p*p*p/27.0
      if(del.ge.0.0) then
      e1=-0.5*q1+sqrt(del)
      e2=-0.5*q1-sqrt(del)
      e3=1.0
      e4=1.0
      if(e1.lt.0.0) e3=-1.0
      if(e2.lt.0.0) e4=-1.0
      y1=e3*(abs(e1)**(1.0/3.0))+e4*(abs(e2)**(1.0/3.0))
      else
      r=sqrt((-p*p*p)/27.0)
      y1=2.0*sqrt(-p/3.0)*cos(acos(-0.5*q1/r)/3.0)
      endif
      p1=b/3.0-y1
      p2=b-p1
      p3=c-p1*p2
      return
      end

      subroutine design(thetap,thetad,nnum,nden,ndnum,ndden,h,B1,
     @B2,B3,A1,A2,A3,BD1,BD2,BD3,AD1,AD2,AD3,NCN,NCD,CB1,CB2,CB3,CB4,
     @CB5,CB6,CA1,CA2,CA3,CA4,CA5,R1,R2,R3,S0,S1,S2,S3,T0,T1,T2,T3)
      real*4 pn(3),pd(3),dn(3),dd(3),thetap(6),thetad(6)
      do 1 j=nden+1,nden+1+nnum

      pn(j-nden)=thetap(J)
1     continue
      do 2 j=1,nden
      pd(j)=thetap(j)
2     continue
      if(nden.eq.1) call convert0(pd(1),pn(1),h,b1,a1)
      if(nden.eq.2) then
      if(nnum.eq.0) call convert1(pd(1),pd(2),pn(1),h,b1,b2,a1,a2)
      if(nnum.eq.1) call convert2(pd(1),pd(2),pn(1),pn(2),h,b1,b2,a1,
     @a2)
      endif
      if(nden.eq.3) then
```

```
      if(nnum.eq.0) call convert3(pd(1),pd(2),pd(3),pn(1),h,
     @b1,b2,b3,a1,a2,a3)
      if(nnum.eq.1) call convert4(pd(1),pd(2),pd(3),pn(1),pn(2),
     @h,b1,b2,b3,a1,a2,a3)
      if(nnum.eq.2) call convert5(pd(1),pd(2),pd(3),pn(1),pn(2),
     @pn(3),h,b1,b2,b3,a1,a2,a3)
      endif
      do 4 j=ndden+1,ndden+1+ndnum
      dn(j-ndden)=thetad(j)
4     continue
      do 5 j=1,ndden
      dd(j)=thetad(j)
5     continue
      if(ndden.eq.1) call convert0(dd(1),dn(1),h,bd1,ad1)
      if(ndden.eq.2) then
      if(ndnum.eq.0) call convert1(dd(1),dd(2),dn(1),h,bd1,bd2,ad1
     @,ad2)
      if(ndnum.eq.1) call convert2(dd(1),dd(2),dn(1),dn(2),h,bd1,
     @bd2,ad1,ad2)
      endif
      if(ndden.eq.3) then
      if(ndnum.eq.0) call convert3(dd(1),dd(2),dd(3),dn(1),h,
     @bd1,bd2,bd3,ad1,ad2,ad3)
      if(ndnum.eq.1) call convert4(dd(1),dd(2),dd(3),dn(1),dn(2),
     @h,bd1,bd2,bd3,ad1,ad2,ad3)
      if(ndnum.eq.2) call convert5(dd(1),dd(2),dd(3),dn(1),dn(2),
     @dn(3),h,bd1,bd2,bd3,ad1,ad2,ad3)
      endif
      write(*,*) 'Do you want integral action in controller'
      write(*,*) ' 1=yes   2=no'
      read(*,*) int
      write(*,*) 'Do you want to cancel process zeros'
      write(*,*) ' 1=yes  2=no'
      read(*,*) ipz
      tau=exp(-h/0.30)
      p1=-2.0*exp(-0.85*6.0*h)*cos(6.0*h*sqrt(1.0-0.85*0.85))
      p2=exp(-2.0*0.85*6*h)
      if(nden.eq.1) then
      if(int.eq.1) call control1i(b1,a1,tau,r1,s0,s1,t0,t1)
      if(int.eq.2) call control1(b1,a1,tau,s0,t0)
```

```
   endif
   if(nden.eq.2) then
   if(int.eq.1) then
   if(ipz.eq.1) call control2ic(b1,b2,a1,a2,p1,p2,tau,r1,r2,
@s0,s1,s2,t0,t1,t2)
   if(ipz.eq.2) call control2i(b1,b2,a1,a2,p1,p2,tau,r1,r2,
@s0,s1,s2,t0,t1,t2)
   endif
   if(int.eq.2) then
   if(ipz.eq.1) call control2c(b1,b2,a1,a2,p1,p2,r1,s0,s1,t0,t1)
   if(ipz.eq.2) call control2(b1,b2,a1,a2,p1,p2,r1,s0,s1,t0,t1)
   endif
   endif
   if(nden.eq.3) then
   if(int.eq.1) then
   if(ipz.eq.1) call control3ic(b1,b2,b3,a1,a2,a3,p1,p2,tau,r1,
@r2,r3,s0,s1,s2,s3,t0,t1,t2,t3)
   if(ipz.eq.2) call control3i(b1,b2,b3,a1,a2,a3,p1,p2,tau,
@r1,r2,r3,s0,s1,s2,s3,t0,t1,t2,t3)
   endif
   if(int.eq.2) then
   if(ipz.eq.1) call control3c(b1,b2,b3,a1,a2,a3,p1,p2,tau
@,r1,r2,s0,s1,s2,t0,t1,t2)
   if(ipz.eq.2) call control3(b1,b2,b3,a1,a2,a3,p1,p2
@,r1,r2,s0,s1,s2,t0,t1,t2)
   endif
   endif
   call feedcom(nnum,nden,ndnum,ndden,pn(1),pn(2),pn(3),pd(1),
@pd(2),pd(3),dn(1),dn(2),dn(3),dd(1),dd(2),dd(3),b1,b2,b3,a1,a2,
@a3,bd1,bd2,bd3,ad1,ad2,ad3,h,ncn,ncd,cb1,cb2,cb3,cb4,cb5,cb6,
@ca1,ca2,ca3,ca4,ca5)
   return
   end

   subroutine control1i(b1,a1,a,r1,s0,s1,t0,t1)
   s0=(1.0-a-a1)/b1
   s1=a1/b1
   r1=-1.0
   t0=(1.0-a)/b1
   t1=0.0
```

```
      return
      end

      subroutine control1(b1,a1,a,s0,t0)
      s0=(-a-a1)/b1
      t0=(1.0-a)/b1
      return
      end

      subroutine control2i(b1,b2,a1,a2,p1,p2,a,r1,r2,s0,s1,s2,t0,
     @t1,t2)
      r1=(b2*(a1-a2+6.0*a**2-b2/b1*(-4.0*a+1.0-a1))-b1*(a2-4.0*a**3
     @-b1/b2*a**4))/(b2*(a1-1.0-b2/b1)-b1*(a2-a1+b1*a2/b2))
      s0=(1.0-a1-r1-4.0*a)/b1
      s1=(a2-(a2-a1+b1*a2/b2)*r1-4.0*a**3-b1/b2*a**4)/b2
      s2=(a**4+a2*r1)/b2
      t0=(1.0-a)**2/(b1+b2)
      t1=-2*a*t0
      t2=t0*a*a
      r2=-r1
      r1=r1-1.0
      return
      end

      subroutine control2ic(b1,b2,a1,a2,p1,p2,a,r1,r2,s0,s1,s2,
     @t0,t1,t2)
      s0=(-3.0*a+1.0-a1)/b1
      s1=(3.0*a*a+a1-a2)/b1
      s2=(-a**3+a2)/b1
      r1=b2/b1-1.0
      r2=-b2/b1
      t0=(1-a)**2/b1
      t1=-a*t0
      t2=0.0
      return
      end

      subroutine control2(b1,b2,a1,a2,p1,p2,r1,s0,s1,t0,t1)
      r1=(p2-a2+(a1-p1)*b2/b1)/(a1-b1*a2/b2-b2/b1)
      s0=(p1-r1-a1)/b1
```

```
      s1=-a2*r1/b2
      t0=(1.0+p1+p2)/(b1+b2)
      t1=0.0
      return
      end

      subroutine control2c(b1,b2,a1,a2,p1,p2,r1,s0,s1,t0,t1)
      r1=b2/b1
      s0=(p1-a1)/b1
      s1=(p2-a2)/b1
      t0=(1.0+p1+p2)/b1
      t1=0.0
      return
      end

      subroutine control3i(b1,b2,b3,a1,a2,a3,pp1,pp2,a,r1,r2,r3,
     @s0,s1,s2,s3,t0,t1,t2,t3)
      b=0.0
      del=0.0
1     b=b+del
      del=0.01
      q1=a1-1.0-b2/b1-b1/b3*(a3-a2+b2*a3/b3)
      q2=a2-a1-b3/b1-b2/b1*(a1-1.0-b2/b1)+b1*a3/b3
      q3=-b**3-9.0*a*b*b-9.0*b*a*a-a**3
     @+a2-a3-b3/b1*(-3.0*b-3.0*a+1.0-a1)-b2/b1*(3.0*b*b+9.0*a*b+
     @3.0*a*a+a1-
     @a2-b2/b1*(-3.0*b-3.0*a+1.0-a1))+b1/b3*
     @(3.0*a*a*b**3+3.0*b*b*a**3+b2*a**3*b**3/b3)
      q4=a2-a1+b1*a3/b3-b2/b3*(a3-a2+b2*a3/b3)-b3/b1
      q5=a3-a2+b2*a3/b3-b3/b1*(a1-1.0-b2/b1)
      q6=3.0*a*b**3+9.0*a*a*b*b+3.0*a**3*b
     @+a3-b1*a**3*b**3/b3+b2/b3*
     @(3.0*a*a*b**3+3.0*b*b*a**3+b2*a**3*b**3/b3)+
     @b3/b1*(a2-a1-3.0*b*b-9.0*a*b-3.0*a*a+b2/b1*(1.0-a1-3.0*a-3.0*b))
      rr2=(q3*q5-q6*q2)/(q1*q5-q4*q2)
      rr1=(q3-q1*rr2)/q2
      s0=(-3.0*a-3.0*b-rr1+1.0-a1)/b1
      s1=(3.0*b*b+9.0*a*b+3.0*a*a
     @-rr2-(a1-1.0-b2/b1)*rr1-(a2-a1+b2/b1*(-3.0*a-3.0*b+1.0-
     @a1)))/b1
```

```
      s2=(-3.0*a*a*b**3-3.0*b*b*a**3
     @+rr1*a3-rr2*(a3-a2+b2*a3/b3)-b2*a**3*b**3/b3)/b3
      s3=(a**3*b**3+rr2*a3)/b3
      t0=(1.0-a)**3/(b1+b2+b3)
      t1=-3.0*b*t0
      t2=3.0*b*b*t0
      t3=-b**3*t0
      r1=rr1-1.0
      r2=rr2-rr1
      r3=-rr2
      call roots(r1,r2,r3,p1,p2,p3)
      if(p1.lt.-1.0) go to 1
      if(p1.gt.1.0) go to 1
      AI=P2**2-4.0*P3
      IF(AI.GE.0.0) THEN
      ROOT1=-0.5*P2-0.5*SQRT(AI)
      ROOT2=-0.5*P2+0.5*SQRT(AI)
      if(ROOT1.lt.-1.0) go to 1
      if(ROOT1.gt.1.0) go to 1
      if(ROOT2.lt.-1.0) go to 1
      if(ROOT2.gt.1.0) go to 1
      ENDIF
      IF(AI.LT.0.0) THEN
      ROOT1=SQRT((0.5*P2)**2-0.25*AI)
      if(ROOT1.gt.1.0) go to 1
      ENDIF
      return
      end

      subroutine control3ic(b1,b2,b3,a1,a2,a3,p1,p2,a,r1,r2,r3,
     @s0,s1,s2,s3,t0,t1,t2,t3)
      r1=b2/b1-1.0
      r2=b3/b1-b2/b1
      r3=-b3/b1
      s0=(-4.0*a-a1+1.0)/b1
      s1=(6.0*a*a+a1-a2)/b1
      s2=(-4.0*a**3+a2-a3)/b1
      s3=(a**4+a3)/b1
      t0=(1.0-a)**3/b1
      t1=-a*t0
```

```
   t2=0.0
   t3=0.0
   return
   end

   subroutine control3(b1,b2,b3,a1,a2,a3,p1,p2,r1,r2,s0,s1,s2,
@t0,t1,t2)
   q1=-a3-b3/b1*(p1-a1)-(a1-b1*a3/b3+b2*b2*a3/(b3**2)-b2*a2/b3)*
@(p2-a2-b2/b1*(p1-a1))/(1.0+b1*b2*a3/(b3*b3)-b1*a2/b3)
   q2=a2-b2*a3/b3-b3/b1-(a1-b1*a3/b3+b2*b2*a3/(b3*b3)-b2*a2/b3)*
@(a1-b1*a3/b3-b2/b1)/(1.0+b1*b2*a3/(b3*b3)-b1*a2/b3)
   r1=q1/q2
   r2=(p2-a2-b2/b1*(p1-a1)-r1*(a1-b1*a3/b3-b2/b1))/(1.0
@+b1*b2*a3/(b3*b3)-b1*a2/b3)
   s0=(p1-r1-a1)/b1
   s1=(b2*a3/b3*r2-a2*r2-a3*r1)/b3
   s2=-a3*r2/b3
   t0=(1.0+p1+p2)/(b1+b2+b3)
   t1=0.0
   t2=0.0
   return
   end

   subroutine control3c(b1,b2,b3,a1,a2,a3,p1,p2,a,r1,r2,s0,s1,s2,
@t0,t1,t2)
   r1=b2/b1
   r2=b3/b1
   s0=(p1-a-a1)/b1
   s1=(p2-a*p1-a2)/b1
   s2=(-a*p2-a3)/b1
   t0=(1.0-a)*(1.0+p1+p2)/b1
   t1=0.0
   t2=0.0
   return
   end

   subroutine feedcom(nnum,nden,ndnum,ndden,pn1,pn2,pn3,pd1,pd2,
@pd3,dn1,dn2,dn3,dd1,dd2,dd3,b1,b2,b3,a1,a2,a3,bd1,bd2,bd3,ad1,
@ad2,ad3,h,ncn,ncd,cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
   if(nden.eq.2) go to 2
```

```
      if(nden.eq.3) go to 3

      if(nden.eq.1) then
      if(ndden.eq.1) then
      cb1=bd1/b1
      cb2=bd1*a1/b1
      ca1=ad1
      ncn=1
      ncd=1
      endif
      if(ndden.eq.2) then
      cb1=bd1/b1
      cb2=bd1/b1*(bd2/bd1+a1)
      cb3=bd2*a1/b1
      ca1=ad1
      ca2=ad2
      ncn=2
      ncd=2
      endif
      if(ndden.eq.3) then
      cb1=bd1/b1
      cb2=bd1/b1*(bd2/bd1+a1)
      cb3=bd1/b1*(bd3/bd1+a1*bd2/bd1)
      cb4=a1*bd3/b1
      ca1=ad1
      ca2=ad2
      ca3=ad3
      ncn=3
      ncd=3
      endif
      endif
      go to 1
2     if(ndden.eq.2) go to 4
      if(ndden.eq.3) go to 5
      zero=-b2/b1
      if(zero.lt.-0.9) zero=2.0
      if(zero.le.1.0) then
      cb1=bd1/b1
      cb2=bd1*a1/b1
      cb3=bd1*a2/b1
```

```
      ca1=ad1+b2/b1
      ca2=ad1*b2/b1
      ncn=2
      ncd=2
      else
      if(nnum.eq.1) then
      ta1=pd1+dn2/dn1
      ta2=pd1*dn2/dn1
      tb1=dd1-(pd1+dn2/dn1)
      tb2=dd2-pd1*dn2/dn1
      call convert2(ta1,ta2,tb1,tb2,h,bc1,bc,ac1,ac2)
      cb1=pn1/dn1
      cb2=pn1/dn1*(ac1+bc1)
      cb3=pn1/dn1*(ac2+bc2)
      ca1=ac1
      ca2=ac2
      ncn=2
      ncd=2
      endif
      if(nnum.eq.0) then
      tb1=dd1-50.0-pd1
      tb2=dd2-50.0*pd1
      ta1=50.0+pd1
      ta2=50.0*pd1
      call convert2(ta1,ta2,tb1,tb2,h,bc1,bc2,ac1,ac2)
      cb1=50.0*pn1/dn1
      cb2=50.0*pn1/dn1*(ac1+bc1)
      cb3=50.0*pn1/dn1*(ac2+bc2)
      ca1=ac1
      ca2=ac2
      ncn=2
      ncd=2
      endif
      endif
      go to 1
 4    continue
c     nden=2 ndden=2
      zero=-b2/b1
      if(zero.lt.-0.9) zero=2.0
      if(zero.le.1.0) then
```

```
      cb1=bd1/b1
      cb2=(bd1*a1+bd2)/b1
      cb3=(bd1*a2+bd2*a1)/b1
      cb4=bd2*a2/b1
      ca1=b2/b1+ad1
      ca2=ad2+ad1*b2/b1
      ca3=ad2*b2/b1
      ncn=3
      ncd=3
      else
      if(nnum.eq.0) then
      if(ndnum.eq.0) then
      tb1=pd1-dd1
      tb2=pd2-dd2
      ta1=dd1
      ta2=dd2
      call convert2(ta1,ta2,tb1,tb2,h,bc1,bc2,ac1,ac2)
      cb1=dn1/pn1
      cb2=dn1/pn1*(ac1+bc1)
      cb3=dn1/pn1*(ac2+bc2)
      ca1=ac1
      ca2=ac2
      ncn=2
      ncd=2
      endif
      if(ndnum.eq.1) then
      ta1=50.0+dd1
      ta2=50.0*dd1+dd2
      ta3=50.0*dd2
      tb1=dn2/dn1+pd1-50.0-dd1
      tb2=pd2+pd1*dn2/dn1-50.0*dd1-dd2
      tb3=dn2*pd2/dn1-50.0*dd2
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,ac1,ac2,
     @ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=50.0*dn1/pn1
      cb2=50.0*dn1/pn1*(ac1+bc1)
      cb3=50.0*dn1/pn1*(ac2+bc2)
```

```
   cb4=50.0*dn1/pn1*(ac3+bc3)
   ncn=3
   ncd=3
   endif
   endif
   if(nnum.eq.1) then
   if(ndnum.eq.0) then
   ta1=dd1+pn2/pn1
   ta2=dd1*pn2/pn1+dd2
   ta3=dd2*pn2/pn1
   tb1=dn1/pn1
   tb2=pd1*dn1/pn1
   tb3=pd2*dn1/pn1
   call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,cb1,cb2,cb3,ca1,ca2,
@ca3)
   ncn=2
   ncd=3
   endif
   if(ndnum.eq.1) then
   ta1=dd1+pn2/pn1
   ta2=dd1*pn2/pn1+dd2
   ta3=dd2*pn2/pn1
   tb1=pd1+dn2/dn1-ta1
   tb2=pd2+dn2*pd1/dn1-ta2
   tb3=dn2*pd2/dn1-ta3
   call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,ac1,ac2,
@ac3)

   ca1=ac1
   ca2=ac2
   ca3=ac3
   cb1=dn1/pn1
   cb2=dn1/pn1*(ac1+bc1)
   cb3=DN1/pn1*(ac2+bc2)
   cb4=dn1/pn1*(ac3+bc3)
   ncn=3
   ncd=3
   endif
   endif
   endif
```

```
      go to 1
5     continue
      zero=-b2/b1
      if(zero.lt.-0.9) zero=2.0
      if(zero.le.1.0) then
      ca1=b2/b1+ad1
      ca2=ad1*b2/b1+ad2
      ca3=ad3+ad2*b2/b1
      ca4=b2*ad3/b1
      cb1=bd1/b1
      cb2=(bd2+a1*bd1)/b1
      cb3=(bd3+a1*bd2+a2*bd1)/b1
      cb4=(a1*bd3+a2*bd2)/b1
      cb5=a2*bd3/b1
      ncn=4
      ncd=4
      else
      if(nnum.eq.0) then
      if(ndnum.eq.0) then
      ta1=dd1
      ta2=dd2
      ta3=dd3
      tb1=dn1/pn1
      tb2=dn1*pd1/pn1
      tb3=dn1*pd2/pn1
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,cb1,cb2,cb3,ca1,
    @ca2,ca3)
      ncn=2
      ncd=3
      endif
      if(ndnum.eq.1) then
      ta1=dd1
      ta2=dd2
      ta3=dd3
      tb1=dn2/dn1+pd1-ta1
      tb2=pd2+dn2*pd1/dn1-ta2
      tb3=dn2*pd2/dn1-ta3
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,ac1,ac2,
    @ac3)
      ca1=ac1
```

```
   ca2=ac2
   ca3=ac3
   cb1=dn1/pn1
   cb2=dn1/pn1*(ac1+bc1)
   cb3=dn1/pn1*(ac2+bc2)
   cb4=dn1/pn1*(ac3+bc3)
   ncn=3
   ncd=3
   endif
   if(ndnum.eq.2) then
   ta1=50.0+dd1
   ta2=dd2+50.0*dd1
   ta3=dd3+50.0*dd2
   ta4=50.0*dd3
   tb1=50.0*dn1/pn1
   tb2=50.0/pn1*(dn1*pd1+dn2)
   tb3=50.0/pn1*(dn1*pd2+dn2*pd1+dn3)
   tb4=50.0/pn1*(dn2*pd2+dn3*pd1)
   tb5=50.0/pn1*(dn3*pd2)
   call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,cb1,
@cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
   ncn=4
   ncd=4
   endif
   endif
   if(nnum.eq.1) then
   if(ndnum.eq.0) then
   ta1=dd1+pn2/pn1
   ta2=dd1*pn2/pn1+dd2
   ta3=dd2*pn2/pn1+dd3
   ta4=dd3*pn2/pn1
   tb1=dn1/pn1
   tb2=dn1*pd1/pn1
   tb3=dn1*pd2/pn1
   call tustin24(ta1,ta2,ta3,ta4,tb1,tb2,tb3,h,cb1,cb2,cb3,cb4,
@cb5,ca1,ca2,ca3,ca4)
   ncn=4
   ncd=4
   endif
   if(ndnum.eq.1) then
```

```
      ta1=dd1+pn2/pn1
      ta2=dd1*pn2/pn1+dd2
      ta3=dd2*pn2/pn1+dd3
      ta4=dd3*pn2/pn1
      tb1=dn1/pn1
      tb2=(dn1*pd1+dn2)/pn1
      tb3=(dn1*pd2+dn2*pd1)/pn1
      tb4=dn2*pd2/pn1
      call tustin34(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,h,cb1,cb2,cb3,
     @cbd,cb5,ca1,ca2,ca3,ca4)
      ncn=4
      ncd=4
      endif
      if(ndnum.eq.2) then
      ta1=dd1+pn2/pn1
      ta2=dd1*pn2/pn1+dd2
      ta3=dd2*pn2/pn1+dd3
      ta4=dd3*pn2/pn1
      tb1=dn1/pn1
      tb2=(dn1*pd1+dn2)/pn1
      tb3=(dn1*pd2+dn2*pd1+dn3)/pn1
      tb4=(dn2*pd2+dn3*pd1)/pn1
      tb5=dn3*pd2/pn1
      call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,cb1,cb2,
     @cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      ncn=4
      ncd=4
      endif
      endif
      endif
      go to 1
3     continue
      ss1=b2/b1
      ss2=b3/b1
      if(abs(ss1*0.5).gt.1.0) go to 10
      ss3=ss1*ss1-4.0*ss2
      if(ss3.ge.0.0)  root1=-0.5*ss1+sqrt(ss3)*0.5
      if(ss3.ge.0.0)  root2=-0.5*ss1-sqrt(ss3)*0.5
      if(ss3.le.0.0)  root1=sqrt((0.5*ss1)**2+(-ss3*0.5*0.5))
      if(ss3.le.0.0)  root2=sqrt((0.5*ss1)**2+(-ss3*0.5*0.5))
```

```
if(abs(root1).gt.1.0) go to 10
if(abs(root2).gt.1.0) go to 10
if(ndden.eq.1) then
cb1=bd1/b1
cb2=bd1*a1/b1
cb3=bd1*a2/b1
cb4=bd1*a3/b1
ca1=b2/b1+ad1
ca2=b3/b1+ad1*b2/b1
ca3=ad1*b3/b1
ncn=3
ncd=3
endif
if(ndden.eq.2) then
cb1=bd1/b1
cb2=(bd1*a1+bd2)/b1
cb3=(bd1*a2+bd2*a1)/b1
cb4=(bd1*a3+bd2*a2)/b1
cb5=bd2*a3/b1
ca1=ad1+b2/b1
ca2=ad2+b3/b1+ad1*b2/b1
ca3=ad1*b3/b1+ad2*b2/b1
ca4=ad2*b3/b1
ncn=4
ncd=4
endif
if(ndden.eq.3) then
cb1=bd1/b1
cb2=(bd1*a1+bd2)/b1
cb3=(bd1*a2+bd2*a1+bd3)/b1
cb4=(bd1*a3+bd2*a2+bd3*a1)/b1
cb5=(bd2*a3+bd3*a2)/b1
cb6=bd3*a3/b1
ca1=ad1+b2/b1
ca2=ad2+b2*ad1/b1+b3/b1
ca3=ad3+ad2*b2/b1+ad1*b3/b1
ca4=ad3*b2/b1+ad2*b3/b1
ca5=ad3*b3/b1
ncn=5
ncd=5
```

```
      endif
      go to 1
10    continue
      if(ndden.eq.2) go to 6
      if(ndden.eq.3) go to 7
      if(nnum.eq.0) then
      ta1=100.0+dd1
      ta2=100.0*dd1+2500.0
      ta3=2500.0*dd1
      tb1=pd1-ta1
      tb2=pd2-ta2
      tb3=pd3-ta3
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,ac1,
     @ac2,ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=dn1*2500.0/pn1
      cb2=dn1*2500.0/pn1*(ac1+bc1)
      cb3=dn1*2500.0/pn1*(ac2+bc2)
      cb4=dn1*2500.0/pn1*(ac3+bc3)
      ncn=3
      ncd=3
      endif
      if(nnum.eq.1) then
      ta1=50.0+dd1+pn2/pn1
      ta2=(50.0+dd1)*pn2/pn1+50*dd1
      ta3=50.0*dd1*pn2/pn1
      tb1=pd1-ta1
      tb2=pd2-ta2
      tb3=pd3-ta3
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,
     @ac1,ac2,ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=50.0*dn1/pn1
      cb2=50.0*dn1/pn1*(ac1+bc1)
      cb3=50.0*dn1/pn1*(ac2+bc2)
      cb4=50.0*dn1/pn1*(ac3+bc3)
```

```
      ncn=3
      ncd=3
      endif
      if(nnum.eq.2) then
      ta1=pn2/pn1+dd1
      ta2=dd1*pn2/pn1+pn3/pn1
      ta3=dd1*pn3/pn1
      tb1=pd1-ta1
      tb2=pd2-ta2
      tb3=pd3-ta3
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,
     @ac1,ac2,ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=dn1/pn1
      cb2=dn1/pn1*(ac1+bc1)
      cb3=dn1/pn1*(ac2+bc2)
      cb4=dn1/pn1*(ac3+bc3)
      ncn=3
      ncd=3
      endif
      go to 1
6     continue
      if(nnum.eq.0) then
      if(ndnum.eq.0) then
      ta1=50.0+dd1
      ta2=dd2+50.0*dd1
      ta3=50.0*dd2
      tb1=pd1-ta1
      tb2=pd2-ta2
      tb3=pd3-ta3

      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,
     @ac1,ac2,ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=50.0*dn1/pn1
      cb2=50.0*dn1/pn1*(ac1+bc1)
```

```
   cb3=50.0*dn1/pn1*(ac2+bc2)
   cb4=50.0*dn1/pn1*(ac3+bc3)
   ncn=3
   ncd=3
   endif
   if(ndnum.eq.1) then
   ta1=dd1+100.0
   ta2=2500.0+dd1*100.0+dd2
   ta3=2500.0*dd1+100.0*dd2
   ta4=2500*dd2
   tb1=2500.0/pn1*dn1
   tb2=2500.0/pn1*(dn1*pd1+dn2)
   tb3=2500.0/pn1*(dn1*pd2+dn2*pd1)
   tb4=2500.0/pn1*(dn1*pd3+dn2*pd2)
   tb5=2500.0/pn1*dn2*pd3
   call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,
@cb1,cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
   ncn=4
   ncd=4
   endif
   endif
   if(nnum.eq.1) then
   if(ndnum.eq.0) then
   ta1=pn2/pn1+dd1
   ta2=pn2*dd1/pn1+dd2
   ta3=pn2*dd2/pn1
   tb1=pd1-ta1
   tb2=pd2-ta2
   tb3=pd3-ta3
   call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,
@ac1,ac2,ac3)
   ca1=ac1
   ca2=ac2
   ca3=ac3
   cb1=dn1/pn1
   cb2=dn1/pn1*(ac1+bc1)
   cb3=dn1/pn1*(ac2+bc2)
   cb4=dn1/pn1*(ac3+bc3)
   ncn=3
   ncd=3
```

```
   endif
   if(ndnum.eq.1) then
   ta1=dd1+50.0+pn2/pn1
   ta2=50.0*pn2/pn1+dd1*(50.0+pn2/pn1)+dd2
   ta3=dd1*50.0*pn2/pn1+dd2*(50.0+pn2/pn1)
   ta4=50.0*dd2*pn2/pn1
   tb1=50.0*dn1/pn1
   tb2=50.0/pn1*(dn1*pd1+dn2)
   tb3=50.0/pn1*(dn1*pd2+dn2*pd1)
   tb4=50.0/pn1*(dn1*pd3+dn2*pd2)
   tb5=50.0/pn1*dn2*pd3
   call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,
@cb1,cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
   ncn=4
   ncd=4
   endif
   endif
   if(nnum.eq.2) then
   if(ndnum.eq.0) then
   ta1=dd1+pn2/pn1
   ta2=pn3/pn1+dd1*pn2/pn1+dd2
   ta3=dd1*pn3/pn1+dd2*pn2/pn1
   ta4=dd2*pn3/pn1
   tb1=dn1/pn1
   tb2=dn1*pd1/pn1
   tb3=dn1*pd2/pn1
   tb4=dn1*pd3/pn1
   call tustin34(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,h,cb1,cb2,
@cb3,cb4,cb5,ca1,ca2,ca3,ca4)
   ncn=4
   ncd=4
   endif
   if(ndnum.eq.1) then
   ta1=dd1+pn2/pn1
   ta2=pn3/pn1+dd1*pn2/pn1+dd2
   ta3=dd1*pn3/pn1+dd2*pn2/pn1
   ta4=dd2*pn3/pn1
   tb1=dn1/pn1
   tb2=(dn1*pd1+dn2)/pn1
   tb3=(dn1*pd2+dn2*pd1)/pn1
```

```
      tb4=(dn1*pd3+dn2*pd2)/pn1
      tb5=dn2*pd3/pn1
      call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,cb1,cb2,
     @cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      ncn=4
      ncd=4
      endif
      endif
      go to 1
 7    continue
c     nden=3 ndden=3
c     nnum=0
      if(nnum.eq.1) go to 8
      if(nnum.eq.2) go to 9
      if(ndnum.eq.0) then
      ta1=dd1
      ta2=dd2
      ta3=dd3
      tb1=pd1-ta1
      tb2=pd2-ta2
      tb3=pd3-ta3
      call convert5(ta1,ta2,ta3,tb1,tb2,tb3,h,bc1,bc2,bc3,
     @ac1,ac2,ac3)
      ca1=ac1
      ca2=ac2
      ca3=ac3
      cb1=dn1/pn1
      cb2=dn1/pn1*(ac1+bc1)
      cb3=dn1/pn1*(ac2+bc2)
      cb4=dn1/pn1*(ac3+bc3)
      NCN=3
      ncd=3
      endif
      if(ndnum.eq.1) then
      ta1=50.0+dd1
      ta2=dd2+50.0*dd1
      ta3=dd3+50.0*dd2
      ta4=50.0*dd3
      tb1=50.0*dn1/pn1
      tb2=50.0/pn1*(dn1*pd1+dn2)
```

```
      tb3=50.0/pn1*(dn1*pd2+dn2*pd1)
      tb4=50.0/pn1*(dn1*pd3+dn2*pd2)
      tb5=50.0*dn2*pd3/pn1
      call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,cb1,cb2,
     @cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      ncn=4
      ncd=4
      endif
      if(ndnum.eq.2) then
      ta1=dd1+100.0
      ta2=dd2+100.0*dd1+2500.0
      ta3=dd3+100.0*dd2+2500.0*dd1
      ta4=100.0*dd3+2500.0*dd2
      ta5=2500.0*dd3
      tb1=2500.0*dn1/pn1
      tb2=2500.0/pn1*(dn1*pd1+dn2)
      tb3=2500.0/pn1*(dn1*pd2+dn2*pd1+dn3)
      tb4=2500.0/pn1*(dn1*pd3+dn2*pd2+dn3*pd1)
      tb5=2500.0/pn1*(dn2*pd3+dn3*pd2)
      tb6=2500.0/pn1*(dn3*pd3)
      call tustin55(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,tb5,tb6,h,
     @cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
      ncn=5
      ncd=5
      endif
      go to 1
8     continue

      if(ndnum.eq.0) then
      ta1=dd1+pn2/pn1
      ta2=dd1*pn2/pn1+dd2
      ta3=dd2*pn2/pn1+dd3
      ta4=dd3*pn2/pn1
      tb1=dn1/pn1
      tb2=dn1*pd1/pn1
      tb3=dn1*pd2/pn1
      tb4=dn1*pd3/pn1
      call tustin34(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,h,cb1,cb2,
     @cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      ncn=4
```

```
      ncd=4
      endif
      if(ndnum.eq.1) then
      ta1=dd1+pn2/pn1
      ta2=dd1*pn2/pn1+dd2
      ta3=dd2*pn2/pn1+dd3
      ta4=dd3*pn2/pn1
      tb1=dn1/pn1
      tb2=(dn1*pd1+dn2)/pn1
      tb3=(dn1*pd2+dn2*pd1)/pn1
      tb4=(dn1*pd3+dn2*pd2)/pn1
      tb5=dn2*pd3/pn1
      call tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,h,cb1,cb2,
     @cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      ncn=4
      ncd=4
      endif
      if(ndnum.eq.2) then
      ta1=dd1+50.0+pn2/pn1
      ta2=50.0*pn2/pn1+dd1*(50.0+pn2/pn1)+dd2
      ta3=dd1*50.0*pn2/pn1+dd2*(50.0+pn2/pn1)+dd3
      ta4=dd2*50.0*pn2/pn1+dd3*(50.0+pn2/pn1)
      ta5=dd3*50.0*pn2/pn1
      tb1=50.0*dn1/pn1
      tb2=50.0/pn1*(dn1*pd1+dn2)
      tb3=50.0/pn1*(dn1*pd2+dn2*pd1+dn3)
      tb4=50.0/pn1*(dn1*pd3+dn2*pd2+dn3*pd1)
      tb5=50.0/pn1*(dn2*pd3+dn3*pd2)
      tb6=50.0*dn3*pd3/pn1
      call tustin55(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,tb5,tb6,h,
     @cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
      ncn=5
      ncd=5
      endif
      go to 1
9     continue
      if(ndnum.eq.0) then
      ta1=dd1+pn2/pn1
      ta2=pn2/pn1+dd1*pn2/pn1+dd2
      ta3=dd1*pn3/pn1+dd2*pn2/pn1+dd3
```

```
   ta4=dd2*pn3/pn1+dd3*pn2/pn1
   ta5=dd3*pn3/pn1
   tb1=dn1/pn1
   tb2=dn1*pd1/pn1
   tb3=dn1*pd2/pn1
   tb4=dn1*pd3/pn1
   call tustin35(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,h,
@cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
   ncn=5
   ncd=5
   endif
   if(ndnum.eq.1) then
   ta1=dd1+pn2/pn1
   ta2=pn2/pn1+dd1*pn2/pn1+dd2
   ta3=dd1*pn3/pn1+dd2*pn2/pn1+dd3
   ta4=dd2*pn3/pn1+dd3*pn2/pn1
   ta5=dd3*pn3/pn1
   tb1=dn1/pn1
   tb2=(dn1*pd1+dn2)/pn1
   tb3=(dn1*pd2+dn2*pd1)/pn1
   tb4=(dn1*pd3+dn2*pd2)/pn1
   tb5=dn2*pd3/pn1
   call tustin45(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,tb5,h,cb1,
@cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
   ncn=5
   ncd=5
   endif
   if(ndnum.eq.2) then
   ta1=dd1+pn2/pn1
   ta2=pn2/pn1+dd1*pn2/pn1+dd2
   ta3=dd1*pn3/pn1+dd2*pn2/pn1+dd3
   ta4=dd2*pn3/pn1+dd3*pn2/pn1
   ta5=dd3*pn3/pn1
   tb1=1.0*dn1/pn1
   tb2=1.0/pn1*(dn1*pd1+dn2)
   tb3=1.0/pn1*(dn1*pd2+dn2*pd1+dn3)
   tb4=1.0/pn1*(dn1*pd3+dn2*pd2+dn3*pd1)
   tb5=1.0/pn1*(dn2*pd3+dn3*pd2)
   tb6=1.0*dn3*pd3/pn1
   call tustin55(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,tb5,
```

```
     @tb6,h,cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
        ncn=5
        ncd=5
        endif
1       continue
        return
        end

        Subroutine tustin24(ta1,ta2,ta3,ta4,tb1,tb2
     @,tb3,h,cb1,cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
        r=2.0/h
        s=r**4+ta1*r**3+ta2*r*r+ta3*r+ta4
        r2=r*r
        r3=r*r*r
        r4=r*r*r*r
        s=r4+ta1*r3+ta2*r2+ta3*r+ta4
        ca1=(-4.0*r4-2.0*ta1*r3+2.0*ta3*r+4.0*ta4)/s
        ca2=(6.0*r4-2.0*ta2*r2+6.0*ta4)/s
        ca3=(-4.0*r4+2.0*ta1*r3-2.0*ta3*r+4.0*ta4)/s
        ca4=(r4-ta1*r3+ta2*r2-ta3*r+ta4)/s
        cb1=(tb1*r2+tb2*r+tb3)/s
        cb2=(2.0*tb2*r+4.0*tb3)/s
        cb3=(-2.0*tb1*r2+6.0*tb3)/s
        cb4=(-2.0*tb2*r+4.0*tb3)/s
        cb5=(tb1*r2-tb2*r+tb3)/s
        return
        end

        subroutine tustin34(ta1,ta2,ta3,ta4,tb1,
     @tb2,tb3,tb4,h,cb1,cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
        r=2.0/h
        s=r**4+ta1*r**3+ta2*r*r+ta3*r+ta4
        r2=r*r
        r3=r*r*r
        r4=r*r*r*r
        s=r4+ta1*r3+ta2*r2+ta3*r+ta4
        ca1=(-4.0*r4-2.0*ta1*r3+2.0*ta3*r+4.0*ta4)/s
        ca2=(6.0*r4-2.0*ta2*r2+6.0*ta4)/s
        ca3=(-4.0*r4+2.0*ta1*r3-2.0*ta3*r+4.0*ta4)/s
        ca4=(r4-ta1*r3+ta2*r2-ta3*r+ta4)/s
```

```
      cb1=(tb1*r3+tb2*r2+tb3*r+tb4)/s
      cb2=(-2.0*tb1*r3+2.0*tb3*r+4.0*tb4)/s
      cb3=(-2.0*tb2*r2+6.0*tb4)/s
      cb4=(2.0*tb1*r3-2.0*tb3*r+4.0*tb4)/s
      cb5=(-tb1*r3+tb2*r2-tb3*r+tb4)/s
      return
      end

      subroutine tustin44(ta1,ta2,ta3,ta4,tb1,tb2,tb3,tb4,tb5,
     @h,cb1,cb2,cb3,cb4,cb5,ca1,ca2,ca3,ca4)
      r=2.0/h
      s=r**4+ta1*r**3+ta2*r*r+ta3*r+ta4
      r2=r*r
      r3=r*r*r
      r4=r*r*r*r
      s=r4+ta1*r3+ta2*r2+ta3*r+ta4
      ca1=(-4.0*r4-2.0*ta1*r3+2.0*ta3*r+4.0*ta4)/s
      ca2=(6.0*r4-2.0*ta2*r2+6.0*ta4)/s
      ca3=(-4.0*r4+2.0*ta1*r3-2.0*ta3*r+4.0*ta4)/s
      ca4=(r4-ta1*r3+ta2*r2-ta3*r+ta4)/s
      cb1=(tb1*r4+tb2*r3+tb3*r2+tb4*r+tb5)/s
      cb2=(-4.0*tb1*r4-2*tb2*r3+2.0*tb4*r+4.0*tb5)/s
      cb3=(6.0*tb1*r4-2.0*tb3*r2+6.0*tb5)/s
      cb4=(-4.0*tb1*r4+2.0*tb2*r3-2.0*tb4*r+4.0*tb5)/s
      cb5=(tb1*r4-tb2*r3+tb3*r2-tb4*r+tb5)/s
      return
      end

      subroutine tustin35(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,
     @h,cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
      r=2.0/h
      r2=r*r
      r3=r*r*r
      r4=r*r*r*r
      r5=r*r*r*r*r
      s=r5+ta1*r4+ta2*r3+ta3*r2+ta4*r+ta5
      ca1=(-5.0*r5-3.0*r4*ta1-r3*ta2+r2*ta3+3.0*r*ta4+5.0*ta5)/s
      ca2=(10.0*r5+2.0*r4*ta1-2.0*r3*ta2-2.0*r2*ta3+2.0*r*ta4+
     @10.0*ta5)/s
      ca3=(-10.0*r5+2.0*r4*ta1+2.0*r3*ta2-2.0*r2*ta3-2.0*r*ta4+
```

```
@10.0*ta5)/s
   ca4=(5.0*r5-3.0*r4*ta1+r3*ta2+r2*ta3-3.0*r*ta4+5.0*ta5)/s
   ca5=(-r5+ta1*r4-ta2*r3+ta3*r2-ta4*r+ta5)/s
   cb1=(tb1*r3+tb2*r2+tb3*r+tb4)/s
   cb2=(-tb1*r3+tb2*r2+3.0*tb3*r+5.0*tb4)/s
   cb3=(-2.0*tb1*r3-2.0*tb2*r2+2.0*tb3*r+10.0*tb4)/s
   cb4=(2.0*tb1*r3-2.0*tb2*r2-2.0*tb3*r+10.0*tb4)/s
   cb5=(tb1*r3+tb2*r2-3.0*tb3*r+5.0*tb4)/s
   cb6=(-tb1*r3+tb2*r2-tb3*r+tb4)/s
   return
   end

   subroutine tustin45(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,
@tb5,h,cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
   r=2.0/h
   r2=r*r
   r3=r*r*r
   r4=r*r*r*r
   r5=r*r*r*r*r
   s=r5+ta1*r4+ta2*r3+ta3*r2+ta4*r+ta5
   ca1=(-5.0*r5-3.0*r4*ta1-r3*ta2+r2*ta3+3.0*r*ta4+5.0*ta5)/s
   ca2=(10.0*r5+2.0*r4*ta1-2.0*r3*ta2-2.0*r2*ta3+2.0*r*ta4+
@10.0*ta5)/s
   ca3=(-10.0*r5+2.0*r4*ta1+2.0*r3*ta2-2.0*r2*ta3-2.0*r*ta4+
@10.0*ta5)/s
   ca4=(5.0*r5-3.0*r4*ta1+r3*ta2+r2*ta3-3.0*r*ta4+5.0*ta5)/s
   ca5=(-r5+ta1*r4-ta2*r3+ta3*r2-ta4*r+ta5)/s
   cb1=(tb1*r4+tb2*r3+tb3*r2+tb4*r+tb5)/s
   cb2=(-3.0*tb1*r4-tb2*r3+tb3*r2+3.0*tb4*r+5.0*tb5)/s
   cb3=(2.0*tb1*r4-2.0*tb2*r3-2.0*tb3*r2+2.0*tb4*r+10.0*tb5)/s
   cb4=(2.0*tb1*r4+2.0*tb2*r3-2.0*tb3*r2-2.0*tb4*r+10.0*tb5)/s
   cb5=(-3.0*tb1*r4+tb2*r3+tb3*r2-3.0*tb4*r+5.0*tb5)/s
   cb6=(tb1*r4-tb2*r3+tb3*r2-tb4*r+tb5)/s
   return
   end

   subroutine tustin55(ta1,ta2,ta3,ta4,ta5,tb1,tb2,tb3,tb4,
@tb5,tb6,h,cb1,cb2,cb3,cb4,cb5,cb6,ca1,ca2,ca3,ca4,ca5)
   r=2.0/h
   r2=r*r
```

```
      r3=r*r*r
      r4=r*r*r*r
      r5=r*r*r*r*r
      s=r5+ta1*r4+ta2*r3+ta3*r2+ta4*r+ta5
      ca1=(-5.0*r5-3.0*r4*ta1-r3*ta2+r2*ta3+3.0*r*ta4+5.0*ta5)/s
      ca2=(10.0*r5+2.0*r4*ta1-2.0*r3*ta2-2.0*r2*ta3+2.0*r*ta4+
     @10.0*ta5)/s
      ca3=(-10.0*r5+2.0*r4*ta1+2.0*r3*ta2-2.0*r2*ta3-2.0*r*ta4+
     @10.0*ta5)/s
      ca4=(5.0*r5-3.0*r4*ta1+r3*ta2+r2*ta3-3.0*r*ta4+5.0*ta5)/s
      ca5=(-r5+ta1*r4-ta2*r3+ta3*r2-ta4*r+ta5)/s
      cb1=(tb1*r5+tb2*r4+tb3*r3+tb4*r2+tb5*r+tb6)/s
      cb2=(-5.0*r5*tb1-3.0*r4*tb2-r3*tb3+r2*tb4+3.0*r*tb5+5.0*tb6)/s
      cb3=(10.0*r5*tb1+2.0*r4*tb2-2.0*r3*tb3-2.0*r2*tb4
     @+2.0*r*tb5+10.0*tb6)/s
      cb4=(-10.0*r5*tb1+2.0*r4*tb2+2.0*r3*tb3-2.0*r2*tb4
     @-2.0*r*tb5+10.0*tb6)/s
      cb5=(5.0*r5*tb1-3.0*r4*tb2+r3*tb3+r2*tb4-3.0*r*tb5+5.0*tb6)/s
      cb6=(-r5*tb1+r4*tb2-r3*tb3+r2*tb4-r*tb5+tb6)/s
      return
      end
```

C.2 Program Listing for Adaptive Regulator with Continuous Update of Process and Controller Parameters

The computer program for the full adaptive regulator contains some of the subroutines same as given in the previous section, so they are not included here. These subroutines are: SUBROUTINE START, SUBROUTINE STOP, SUBROUTINE RUD, SUBROUTINE CONVERT1, and SUBROUTINE CONVERT2I.

```
      PROGRAM ENGINE
      implicit real*4 (a-h)
      implicit real*4 (o-z)
      INTEGER*2 ERSTAT,LCHAN,BOARD,COUNT,PCHAN,LCHAN1
      INTEGER*2 CHA(0:3),PVOLT(3)
      INTEGER*2 IT1,IT2,IT3,IT4
      INTEGER*1 CARR
```

```
      INTEGER*2 IOUT,I1,ISTEP,ISS
      INTEGER*2 IMIN,IMAX
      real*4 LAM,theta(4),diag(4),ofdiag(6),FI(4),lammin
      CHARACTER*9 VVH,VVST,STO
      CHARACTER*9 FILEN,STEP,STEPP,STEPS,file1
      DIMENSION A(15),B(15),Y(15),T(15),CON(15),C(15)
      DIMENSION Y1(15),DE(15),CON1(15),DIS(15),DS(1200)
      DIMENSION SPEED(1201),SPREF(1200),TORREF(1200),TR(1200)
      DIMENSION TORQUE(1200),TIME(1200),ER1(15),ER2(15),ER3(15)
      DIMENSION ER4(15),f0(15),f1(15),f2(15),si1(15),si2(15),ri1(15)
      dimension ri2(15),bb(15),us(15),ut(15),as(15),bt(15)
      DATA CARR /1*13/
      write(*,*) 'input the data filename to store parameter vector'
      read(*,555) file1
      open(unit=32,file=file1,status='new')
      CHA(0)=3
      CHA(1)=0
      CHA(2)=1
      CHA(3)=2
      LCHAN=1
      LCHAN1=2
      BOARD=1
      COUNT=3
      PCHAN=0
      ERSTAT=0
      lammin=0.95
      sigma=0.5
      CALL INIT(ERSTAT)
      IF(ERSTAT.NE.0) GO TO 666
      CALL AIG820 (LCHAN,BOARD,CHA,COUNT,ERSTAT)
      CALL AOT820 (LCHAN1,BOARD,PCHAN,ERSTAT)
      STEP='    +    '
      STEPP='    -    '
      STO='       @'
100   WRITE(*,*) '************************************************'
      WRITE(*,*) '*                                              *'
      WRITE(*,*) '*   SPEED AND TORQUE CONTROL OF A DIESEL ENGINE   *'
      WRITE(*,*) '*                                              *'
      WRITE(*,*) '*   SELECT ONE OF THE ITEMS BELOW:             *'
      WRITE(*,*) '*          1)START-UP THE ENGINE/DYNO          *'
```

```
      WRITE(*,*) '*            (SPEED ~ 1200 RPM, TORQUE ~ 70 lbft2)*'
      WRITE(*,*) '*          2)TRANSIENT ENGINE CYCLE                *'
      WRITE(*,*) '*          3)SHUT-DOWN THE ENGINE/DYNO             *'
      WRITE(*,*) '*          4)EXIT                                  *'
      WRITE(*,*) '*                                                  *'
      WRITE(*,*) '**************************************************'
      READ(*,*) ISELE
      IF(ISELE.EQ.1) CALL START(ERCODE)
      IF(ISELE.EQ.2) GO TO 50
      IF(ISELE.EQ.3) CALL STOP
      IF(ISELE.EQ.4) GO TO 6
      IF(ERCODE.EQ.1.0) GO TO 6
      GO TO 100
50    OPEN(UNIT=4,FILE='inputdoc',STATUS='OLD')
      DO 999 J=1,1199
      READ(4,*) SECT,SPREF(J),TORREF(J)
999   continue
      CLOSE(UNIT=4)
      DO 998 J=1,1199
      SPREF(J)=9.0*SPREF(J)+1200.0
      IF(SPREF(J).LT.1400.0) TORREF(J)=TORREF(J)*2.15
      IF(SPREF(J).LT.1400.0) GO TO 997
      IF(SPREF(J).LT.1700.0) TORREF(J)=TORREF(J)*2.05
      IF(SPREF(J).LT.1700.0) GO TO 997
      IF(SPREF(J).LT.1900.0) TORREF(J)=TORREF(J)*2.0
      IF(SPREF(J).LT.1900.0) GO TO 997
      IF(SPREF(J).LT.2100.0) TORREF(J)=TORREF(J)*1.9
      IF(SPREF(J).LT.2100.0) GO TO 997
      IF(SPREF(J).LT.2150.0) TORREF(J)=TORREF(J)*1.8
      IF(SPREF(J).LT.2150.0) GO TO 997
      IF(SPREF(J).LT.2175.0) TORREF(J)=TORREF(J)*1.7
      IF(SPREF(J).LT.2175.0) GO TO 997
      IF(SPREF(J).LT.2190.0) TORREF(J)=TORREF(J)*1.55
      IF(SPREF(J).LT.2190.0) GO TO 997
      IF(SPREF(J).LT.2200.0) TORREF(J)=TORREF(J)*1.40
      IF(SPREF(J).LT.2200.0) GO TO 997
      TORREF(J)=TORREF(J)*1.30
997   IF(TORREF(J).LT.0.0) TORREF(J)=-20.0
998   CONTINUE
      SPREF(1200)=SPREF(1199)
```

```
      TORREF(1200)=TORREF(1199)
      TR(1)=TORREF(1)
      TR(1200)=TORREF(1200)
      DO 994 J=2,1199
      TR(J)=TORREF(J)
      IF(SPREF(J+1).EQ.SPREF(J)) GO TO 994
      IF(TORREF(J).LT.0.0) GO TO 994
      TORREF(J)=TORREF(J)-0.2748081*(SPREF(J+1)-SPREF(J-1))
994   CONTINUE
      DO 996 J=1,1200
996   DS(J)=SPREF(J)-SPREF(1)
      DO 60 I=1,4
      diag(i)=10.0**6
      theta(i)=0.0
60    continue
      do 61 i=1,6
      ofdiag(i)=0.0
61    continue
      N=4
      SIGMA=0.5
      LAM=0.9999
      H=0.2
      bf1=0.0198013267
      af1=-.980198673306755
      bs1=0.197353227109592e-3
      bs2=0.19473931170301e-3
      as1=-1.960397346613511
      as2=0.9607894391522323
      bt1=0.131349244814061e-5
      bt2=0.517578714848144e-5
      bt3=0.127467285840344e-5
      at1=-2.940596019920266
      at2=2.88236831745697
      at3=-0.941764533584249
      a1=-1.3509
      a2=0.4711
      b1=0.0879
      b2=0.0684
      cb1=0.06464/b1
      cb2=0.050272/b1
```

```
      ca1=b2/b1
      r1=-0.822
      r2=-0.178
      s0=4.271
      s1=-5.506
      s2=1.823
      t0=1.94
      t1=-1.744
      t2=0.3917
      DO 995 J=1,5
      ER2(J)=0.0
      ER3(J)=0.0
      ER4(J)=0.0
      T(J)=0.0
      Y1(J)=0.0
      CON(J)=0.0
      C(J)=0.0
      DE(J)=0.0
      DIS(J)=0.0
      B(J)=0.0
      ER1(J)=0.0
      A(J)=0.0
      f0(j)=0.0
      f1(j)=0.0
      f2(j)=0.0
      si1(j)=0.0
      si2(j)=0.0
      ri1(j)=0.0

      ri2(j)=0.0
      ut(j)=0.0
      us(j)=0.0
      bb(j)=0.0
995   CON1(J)=0.0
      ISS=0
      IJ=0
      AVE=0.0
      WRITE(*,*) 'PRESS ANY NUMBER IF YOU ARE READY TO START'
      READ(*,*) GGG
      DO 31, J=1,10
```

```
      CALL AING(LCHAN,PVOLT,ERSTAT)
31    AVE=AVE+PVOLT(2)
      AVE=AVE/10.0
      AVE=10.0*AVE/4095.0-5.0
      AVE=48.43213*AVE+0.404608
      Y(1)=AVE
      Y(2)=AVE
      Y(3)=AVE
      Y(4)=AVE
      Y(5)=AVE
      CALL SETTIM(0,0,0,0)
334   DO 1 J=1,1199
      DO 1 JJ=6,10
      do 2 kk=6,15
      CALL AING(LCHAN,PVOLT,ERSTAT)
      A(JJ)=PVOLT(1)
      B(JJ)=PVOLT(2)
      A(jj)=10.0*A(jj)/4095.0-5.0
      B(jj)=10.0*B(jj)/4095.0-5.0
      A(jj)=491.4704*A(jj)+8.6656
      B(jj)=48.43213*B(jj)+0.4046
      bb(kk)=b(jj)-ave
      C(JJ)=PVOLT(3)
      C(JJ)=10.0*C(JJ)/4095.0-5.0
      DE(JJ)=DS(J)
      if(jj.lt.8) de(jj)=ds(j-1)
      IF(C(JJ).GT.4.80) GO TO 11
      IF(C(JJ).LT.0.80) GO TO 11
      if(kk.eq.6) then
      if(j.gt.20) call design(theta,h,b1,b2,
     @a1,a2,r1,r2,s0,s1,s2,t0,t1,t2,cb1,cb2,ca1)
      as(jj)=a(jj)
      bt(jj)=b(jj)
      Y1(JJ)=-A1*Y1(JJ-1)-A2*Y1(JJ-2)
     1 +B1*CON1(JJ-1)+B2*CON1(JJ-2)
      Y(JJ)=Y1(JJ)+AVE
      ER1(JJ)=B(JJ)-Y(JJ-2)
      ER2(JJ)=TORREF(J)-ER1(JJ)
      ER3(JJ)=S0/T0*Y(JJ)+S1/T0*Y(JJ-1)+S2/T0*Y(JJ-2)
     1 -t1/t0*er3(jj-1)-t2/t0*er3(jj-2)
```

```
   ER4(JJ)=ER2(JJ)-ER3(JJ)
   CON1(JJ)=-R1*CON1(JJ-1)-R2*CON1(JJ-2)
1 +T0*ER4(JJ)+t1*er4(jj-1)+t2*er4(jj-2)
   DIS(JJ)=-CA1*DIS(JJ-1)+CB1*DE(JJ)+CB2*DE(JJ-1)
   if(abs(con1(jj)+dis(jj)-con1(jj-1)-dis(jj-1)).lt.5.0) then
   con1(jj)=con1(jj-1)
   dis(jj)=dis(jj-1)
   endif
   CON(JJ)=CON1(JJ)+DIS(JJ)
   ISTEP=CON(JJ)
   ISTEP=ISTEP-ISS
   IMAX=(C(JJ)-1.40)*490.2
   IMIN=(C(JJ)-4.00)*490.2
   IF(ISTEP.GT.IMAX) ISTEP=IMAX
   IF(ISTEP.LT.IMIN) ISTEP=IMIN
   ISS=CON(jj)
   WRITE(STEPP(6:9),'(I4.4)') ABS(ISTEP)
   WRITE(STEP(6:9),'(I4.4)') ISTEP
   OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
   IF(ISTEP.GT.0) WRITE (9) STEP,CARR
   IF(ISTEP.LT.0) WRITE (9) STEPP,CARR
   IF(ISTEP.EQ.0) WRITE (9) STO,CARR
   CLOSE(UNIT=9)
   REFSP=245.0*SPREF(J)/2175.0
   REFSP=4095.0*REFSP/255.0
   IOUT=REFSP
   IF(JJ.EQ.8)  CALL AOT(LCHAN1,IOUT,ERSTAT)
   IF(JJ.EQ.(10)) THEN
   SPEED(J)=As(8)
   TORQUE(J)=Bt(8)
   TIME(J)=T(8)
   write(32,*) j,theta,con(jj),-de(jj)
   endif
   endif
   ut(kk)=con(jj-2)
   us(kk)=-de(jj-2)
   f0(kk)=bf1*bb(kk-1)-af1*f0(kk-1)
   f1(kk)=bs1*bb(kk-1)+bs2*bb(kk-2)-as1*f1(kk-1)-as2*f1(kk-2)
   f2(kk)=bt1*bb(kk-1)+bt2*bb(kk-2)+bt3*bb(kk-3)-at1*f2(kk-1)
@-at2*f2(kk-2)-at3*f2(kk-3)
```

```
      si1(kk)=bs1*us(kk-1)+bs2*us(kk-2)-as1*si1(kk-1)-as2*si1(kk-2)
      si2(kk)=bt1*us(kk-1)+bt2*us(kk-2)+bt3*us(kk-3)-at1*si2(kk-1)-
     @at2*si2(kk-2)-at3*si2(kk-3)
      ri1(kk)=bs1*ut(kk-1)+bs2*ut(kk-2)-as1*ri1(kk-1)-as2*ri1(kk-2)
      ri2(kk)=bt1*ut(kk-1)+bt2*ut(kk-2)+bt3*ut(kk-3)-
     @at1*ri2(kk-1)-at2*ri2(kk-2)-at3*ri2(kk-3)
      ynew=f0(kk)-2.0*f1(kk)+f2(kk)
      fi(1)=f2(kk)-f1(kk)
      fi(2)=-f2(kk)
      fi(4)=si2(kk)
      fi(3)=ri2(kk)
      if(abs(ut(kk)+us(kk)).lt.20.) go to 4
      if(bb(kk).lt.20.0) go to 4
      call forget(n,ynew,fi,theta,diag,ofdiag,sigma,
     @lammin,lam)
      CALL Rud(n,ynew,fi,theta,diag,ofdiag,lam)
4     do 5 ji=1,4
      if(theta(ji).lt.2.0) then
      theta(1)=3.76
      theta(2)=4.34
      theta(3)=5.55
      theta(4)=4.15
      endif
      if(theta(ji).gt.7.0) then
      theta(1)=3.76
      theta(2)=4.34
      theta(3)=5.55
      theta(4)=4.15
      endif
5     continue
      if((theta(1)/(2.0*sqrt(theta(2)))).lt.0.7) then
      theta(1)=3.76
      theta(2)=4.34
      theta(3)=5.55
      theta(4)=4.15
      endif
      if((theta(1)/(2.0*sqrt(theta(2)))).gt.1.2) then
      theta(1)=3.76
      theta(2)=4.34
      theta(3)=5.55
```

```
      theta(4)=4.15
      endif
2     continue
      DO 981 JU=1,5
      f0(ju)=f0(ju+10)
      f1(ju)=f1(ju+10)
      f2(ju)=f2(ju+10)
      si1(ju)=si1(ju+10)
      si2(ju)=si2(ju+10)
      ri1(ju)=ri1(ju+10)
      ri2(ju)=ri2(ju+10)
      us(ju)=us(ju+10)
981   ut(ju)=ut(ju+10)
      if(jj.eq.10) then
      DO 3 JU=1,5
      Y1(JU)=Y1(JU+5)
      CON1(JU)=CON1(JU+5)
      Y(JU)=Y(JU+5)
      C(JU)=C(JU+5)
      DE(JU)=DE(JU+5)
      ER4(JU)=ER4(JU+5)
      ER2(JU)=ER2(JU+5)
      con(ju)=con(ju+5)
      ER3(JU)=ER3(JU+5)
3     DIS(JU)=DIS(JU+5)
      endif
      ij=ij+1
51    CALL GETTIM(IT1,IT2,IT3,IT4)
      T(JJ)=3600.0*IT1+60.0*IT2+1.0*IT3+0.01*IT4
      IF(T(JJ).LT.(0.2*IJ)) GO TO 51
1     CONTINUE
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      GO TO 89
11    OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STO,CARR
      CLOSE(UNIT=9)
      WRITE(*,*) 'LIMIT SWITCH OPERATION'
36    CALL AING(LCHAN,PVOLT,ERSTAT)
```

```
      WRITE(*,*) 'ACTUATOR POSITION:'
      ACTPOSI=10.0*PVOLT(3)/4095.0-5.0
      WRITE(*,*) ACTPOSI
      WRITE(*,*) 'DO YOU WANT TO GIVE A STEP INPUT TO THE ACTUATOR?'
      WRITE(*,*) '                  SELECT 1:YES  2:NO'
      READ(*,*) I1
      IF(I1.EQ.2) GO TO 100
      IF(I1.EQ.1) GO TO 35
35    WRITE(*,*) 'INPUT COMMAND FOR ACTUATOR'
      READ(*,555) STEPS
      OPEN(UNIT=9,FILE='COM1',FORM='BINARY')
      WRITE (9) STEPS,CARR
      CLOSE(UNIT=9)
      GO TO 36
89    WRITE(*,*) 'SELECT: 1)EXIT'
      WRITE(*,*) '        2)TRANSIENT ENGINE CYCLE'
      WRITE(*,*) '        3)CREATE AN OUTPUT FILE'
      WRITE(*,*) '        4)SHUT-DOWN THE ENGINE/DYNO'
      READ(*,*) M1
      IF(M1.EQ.1) GO TO 6
      IF(M1.EQ.2) GO TO 50
      IF(M1.EQ.3) GO TO 88
      IF(M1.EQ.4) CALL STOP
      GO TO 89
      GO TO 6
88    WRITE(*,*) 'AN OUTPUT FILE NAME'
      READ(*,555) FILEN
555   FORMAT(A)
      OPEN(UNIT=1,FILE=FILEN)
      SPEED(1201)=SPEED(1200)

      DO 39, J=1,1199
      IF(SPREF(J+1).EQ.SPREF(J)) GO TO 39
      IF(TR(J).LT.0.0) GO TO 39
      TORQUE(J+1)=TORQUE(J+1)+0.2748081*(SPEED(J+2)-SPEED(J))
39    WRITE(1,110) TIME(J+1),SPREF(J),TR(J),SPEED(J+1),TORQUE(J+1)
110   FORMAT(1X,F12.5,1X,F12.5,1X,F12.5,1X,F12.5,1X,F12.5)
      CLOSE(UNIT=1)
      GO TO 89
666   WRITE(*,*) 'ERROR OCCURRED IN INITIALIZATION THE RTI-820'
```

```
6       continue
        close(unit=32)
        STOP
        end

        subroutine design(theta,h,b1,b2,a1,a2,r1,r2,
     @s0,s1,s2,t0,t1,t2,cb1,cb2,ca1)
        real*4 theta(4)
        call convert1(theta(1),theta(2),theta(3),
     @h,b1,b2,a1,a2)
        call convert1(theta(1),theta(2),theta(4),
     @h,bd1,bd2,ad1,ad2)
        tau=exp(-h/0.25)
        ss1=b2/b1
        if(abs(ss1).gt.1.0) go to 1
        call control2i(b1,b2,a1,a2,p1,p2,tau,r1,
     @r2,s0,s1,s2,t0,t1,t2)
        cb1=bd1/b1
        cb2=bd2/b1
        ca1=b2/b1
        go to 2
1       a1=-1.3509
        a2=0.4711
        b1=0.0879
        b2=0.0684
        cb1=0.06464/b1
        cb2=0.050272/b1
        ca1=b2/b1
        r1=-0.822
        r2=-0.178
        s0=4.271
        s1=-5.506
        s2=1.823
        t0=1.94
        t1=-1.744
        t2=0.3917
2       continue
        return
        end
```

```
      subroutine forget(n,ynew,fi,theta,diag,ofdiag,sigma,
     @lammin,lam)
      integer n,jf,jl,i,j,k
      real*4 ynew,fi(n),theta(n),diag(n),ofdiag(n*(n-1)/2)
      real*4 error,w,npar,tv(10),lam,sigma,lammin
      error=ynew
      do 1 i=1,n
      error=error-fi(i)*theta(i)
1     continue
      w=0.0
      do 2 i=1,n
      tv(i)=fi(i)
      k=i-1
      do 3 j=1,i-1
      tv(i)=tv(i)+ofdiag(k)*fi(j)
      k=k+n-j-1
3     continue
      w=w+tv(i)*diag(i)*tv(i)
2     continue
      npar=1.0-w-error**2/sigma
      lam=(npar+sqrt(npar**2+4.0*w))/2.0
      if(lam.lt.lammin) lam=lammin
      return
      end
```