A COMPARISON OF AUTOMATIC CONTROL SCHEMES
FOR CONTROL OF THE SPEED OF A DC MOTOR
USING A DIGITAL MICROCOMPUTER

BY:  D. R. LYMBOURNER
     AND
     A. G. ULSOY

TECHNICAL REPORT NO. UM-MEAM-84-26

THE UNIVERSITY OF MICHIGAN
DEPARTMENT OF MECHANICAL ENGINEERING
AND APPLIED MECHANICS

ME600 PROJECT
JUNE 1984
# TABLE OF CONTENTS

1. LIST OF FIGURES
2. LIST OF TABLES
3. INTRODUCTION 1
4. SYSTEM MODELING 4
5. SYSTEM IDENTIFICATION 17
   A. SYSTEM IDENTIFICATION PROGRAM
   B. ESTIMATE OF OPEN-LOOP GAIN AND TIME CONSTANT 20
   C. PARALLEL MRAS ESTIMATOR 23
   D. RECURSIVE LEAST SQUARES ESTIMATOR 28
   E. TUNABLE ESTIMATOR 32
6. CONTROLLER DESIGN 36
   A. DIGITAL PROPORTIONAL-INTEGRAL CONTROLLER
   B. SELF-TUNING ADAPTIVE CONTROLLER 41
   C. ROBUST FEEDBACK CONTROLLER 43
7. ANALYSIS OF RESULTS 47
   A. DIGITAL PROPORTIONAL-INTEGRAL CONTROLLER
   B. SELF-TUNING ADAPTIVE CONTROLLER 52
   C. ROBUST FEEDBACK CONTROLLER 56
8. SUMMARY AND CONCLUSIONS 59
9. RECOMMENDATIONS 61
10. BIBLIOGRAPHY 62
11. APPENDIX A - FLOWCHARTS AND SOURCE CODE, SYSTEM I.D. PHASE 63
12. APPENDIX B - FLOWCHARTS AND SOURCE CODE, TUNABLE ESTIMATOR 80
13. APPENDIX C - FLOWCHARTS AND SOURCE CODE, DIGITAL PI CONTROLLER 89
14. APPENDIX D - FLOWCHARTS AND SOURCE CODE, SELF-TUNING ADAPTIVE CONTROLLER 98
15. APPENDIX E - FLOWCHARTS AND SOURCE CODE, ROBUST FEEDBACK CONTROLLER 107
16. APPENDIX F - SOURCE CODE, ASSEMBLY LANGUAGE ROUTINES 116
LIST OF FIGURES

1. PHYSICAL SET UP
2. OPEN LOOP SYSTEM
3. AMPLIFIER CHARACTERISTIC
4. DC MOTOR SCHEMATIC
5. TACHOMETER VS. MOTOR VOLTAGES
6. COMBINED OPEN LOOP SYSTEM
7. SYSTEM INPUT SIGNALS
8. SYSTEM RESPONSE
9. ESTIMATION OF DISCRETE TIME PARAMETERS USING A PARALLEL MRAS
10. ESTIMATION OF DISCRETE TIME PARAMETERS USING A PARALLEL MRAS
11. ESTIMATION OF DISCRETE TIME PARAMETERS USING A RECURSIVE LEAST SQUARES ALGORITHM
12. ESTIMATION OF DISCRETE TIME PARAMETERS USING A RECURSIVE LEAST SQUARES ALGORITHM
13. ESTIMATION OF DISCRETE TIME PARAMETERS USING A TUNABLE ESTIMATOR
14. ESTIMATION OF DISCRETE TIME PARAMETERS USING A TUNABLE ESTIMATOR
15. DIGITAL PROPORTIONAL-INTEGRAL (PI) CONTROLLER
16. STABILITY UNIT CIRCLE IN THE Z-PLANE
17. SELF-TUNING ADAPTIVE CONTROLLER
18. ROBUST FEEDBACK CONTROLLER
19. DIGITAL PI CONTROLLER - EFFECT OF DAMPING ON SYSTEM RESPONSE
20. DIGITAL PI CONTROLLER - EFFECT OF DAMPING ON SYSTEM RESPONSE
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>DIGITAL PI CONTROLLER - EFFECT OF USING LOW SPEED PARAMETERS FOR HIGH SPEED CONTROLLER DESIGN</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER</td>
<td>53</td>
</tr>
<tr>
<td>23</td>
<td>COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER</td>
<td>54</td>
</tr>
<tr>
<td>24</td>
<td>COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER</td>
<td>55</td>
</tr>
<tr>
<td>25</td>
<td>COMPARISON OF DIGITAL PI VS. ROBUST FEEDBACK CONTROLLER</td>
<td>57</td>
</tr>
<tr>
<td>26</td>
<td>COMPARISON OF DIGITAL PI VS. ROBUST FEEDBACK CONTROLLER</td>
<td>58</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1. MOTOR-TACHOMETER VOLTAGE DATA
2. CONTINUOUS AND DISCRETE TIME PARAMETERS
INTRODUCTION:

With the recent advances in computer technology and with the increasing popularity and flexibility of the digital microcomputer, the implementation of digital control schemes has become very prevalent in the control of numerous processes or plants. In the past, analog control schemes were implemented by "wiring up" the appropriate hardware and in order to change or modify the control algorithm, the hardware had to be completely rewired. However, with the advent of the digital computer, the control apparatus is now a computer program rather than wired hardware. This allows for the implementation of the control scheme in software rather than hardware. Since the control takes place in the software routine of the computer, a control scheme can be implemented and/or modified quite easily by a few lines of computer code rather than by an extravagant amount of hardware.

Due to the ease of implementation of these digital control schemes, an abundance of control algorithms have been devised. Due to the number of these algorithms it is often difficult to choose the "best" control scheme for a particular application. This report examines three different control schemes implemented on a digital microcomputer to control a particular system. The advantages and disadvantages of each type of control schemes are examined in order to determine the "best" control scheme for the system.

The controlled system or plant used for this study is a DC motor and the variable that is being controlled is the motor speed. In this study the motor is modeled as a first order system in order to simplify the system for control purposes. This is described in the System Modeling Section of this report.

The report consists of four main sections. The first as mentioned previously is the System Modeling Section where the system is modeled for control purposes. The second is the System Identification Section. In this section the discrete time system parameters are obtained using four different methods. The system
parameters are needed to implement the various control algorithms. The first method uses a plot of the system response versus time to calculate the open-loop time constant and the open-loop gain. The discrete time system parameters are calculated from these values and the known sampling period. The second method of obtaining the system parameters is through the use of a Model Reference Adaptive System (MRAS). The third method of parameter estimation used is a Recursive Least Squares Method. The final parameter estimation routine used is a Tunable Estimator which combines the characteristics of both the MRAS and the Recursive Least Squares Method.

The third section of the project is the Controller Design Section. In this section the system parameters obtained from the previous section are used to design the controller for the system. The controller is a computer program written in FORTRAN with Assembly Language routines to read information into the computer from the system and output information to the system. Three different types of controllers are used to control the speed of the DC motor. The first type of controller used is a Digital PI (Proportional-Integral) Controller. The second type of controller is a Self-Tuning Model Reference Adaptive Controller (MRAC). This controller uses a MRAS to continually update the system parameters in order to control the motor speed. The third type of controller design is a Robust Feedback Controller. This is a new type of control scheme designed to be insensitive to changes in the system parameters.

The fourth section of this project analyzes the results of the performance of each type of controller and identifies the benefits and disadvantages of each. The final section of the project summarizes the results of the project.

A significant amount of work has been done in the control of systems such as this one. The first section of this report concerns the modeling of the system, and the motor in particular, in order to determine the characteristics of the system. Two useful references for the modeling of the DC motor were "Dynamics
of Physical Systems" by Robert H. Cannon Jr. (ref. #1) and "Modeling, Analysis, and Control of Dynamic Systems" by William Palm (ref. #2). These books help develop mathematical models for various physical systems and have some specific information about DC motors.

When designing the Digital PI Controller certain tuning rules were used. The Ziegler-Nichols Tuning Rules were used quite extensively along with some information concerning pole placement and system response. This information was obtained from a book called "Introducing Systems and Control" written by Auslander, Takahashi, and Rabins (ref. #3) as well as "Digital Control" by Rolf Isermann (ref. #4).

Due to the nature of the Robust Feedback Controller information concerning system sensitivity was needed. This information was obtained from "Digital Control of Dynamic Systems" written by Franklin (ref. #5). In addition the class notes from ME561 "Design of Automatic Control Systems" (ref. #6) were used extensively throughout this project. The next section of this report examines the modeling of the DC motor system.
SYSTEM MODELING:

In this section the open loop DC motor system is modeled in order to determine the transfer function or characteristics of the open loop system. The open loop system is defined as everything external to the control algorithm in the computer. This includes the digital to analog converter, amplifier, DC motor, tachometer, and analog to digital converter. Each element of the open loop system is studied individually in order to determine its specific characteristics. The elements are then linked together in order to obtain an overall perspective of the characteristics of the open loop system. The physical system is shown in Figure 1 and the open loop system is shown in Figure 2.

It should be understood that the model obtained from this study is only a representation of the physical system. Since it is only a representation, some errors will exist in the model. In order to obtain a model that is convenient to work with certain simplifications or model modifications are made. These simplifications are discussed during the evaluation of each element of the open loop system. If the actual system results differ from the expected or desired results, the simplifications should be recalled in order to determine any error or to account for differences.

The modeling in this section is performed for the continuous time case. Conversion to the discrete time case is described in the System Identification Section of this report.

Digital to Analog Converter

The first element of the open loop system to be considered is the digital to analog converter. This converter takes the analog signal calculated by the control algorithm in the computer and converts it to an analog signal which can be used to drive the motor. This is needed since the DC motor is an analog device rather than a digital device. The D/A converter is nothing more than a simple gain. The value of the D/A depends on the voltage
FIGURE 2

OPEN LOOP SYSTEM

Y

K_{A/D}

TACHOMETER

K_{GEN}

MOTOR

K_{M/A}

AMPLIFIER

K_{DC/A}

D/A

M

\frac{K_u}{T_s+1}
level of the outputs on the computer and the number of bits used for the word size in the computer. Since the computer we are using is a XYCOM which has a 12 bit converter with maximum and minimum output voltage levels of plus and minus ten volts respectively and the maximum word size available is \(2^{12}\), we have that \(2047=10\) volts and \(-2048=-10\) volts. Since the computer restricts us to these maximum and minimum values, the control value calculated in the control algorithm must lie in this range. Thus the D/A converter has a linear gain of \(K_{DAC}=(10/2047)\) for all values between these maximum and minimum values.

**Amplifier**

The second element in the open loop system to be studied is the amplifier. The amplifier takes the relatively small value from the computer and amplifies it in order to drive the motor. The characteristics of a typical amplifier are shown in Figure 3. The graph shows that the amplifier is fairly linear, however some non-linearity does exist. It also shows that the amplifier may reach saturation. This means that beyond a certain range of input the amplifier ceases to be linear and the output signal stays constant for increasing values of the input signal. This saturation point must be noted so that it is not exceeded. Another characteristic of the amplifier is that since it is a dynamic device there will be some time constant associated with it. This means that when a signal is input into the amplifier it takes a finite amount of time to reach the steady state output signal. If we look at the transfer function of the amplifier including the time constant it would appear as:

\[
\frac{E_o(s)}{E_i(s)} = \frac{K_a}{Ts+1}
\]

where \(E_o\) is the output signal, \(E_i\) is the input signal, \(K_a\) is the gain, and \(T\) is the time constant. For most amplifiers the time constant is very small (on the order of nanoseconds), therefore
FIGURE 3
TYPICAL AMPLIFIER CHARACTERISTIC
for all practical purposes it can be neglected. This simplifies the model for the amplifier to the simple gain $\text{Ka (Eo=KaEi)}$.

**DC Motor**

The third element in the open loop system is the DC motor. This is by far the most complex element and the most difficult to model (refs. #1 & #2). Consider the schematic of the DC motor shown in Figure 4. The voltage $V$ is applied across the terminals of the motor. The motor also has some resistance ($R$) and some inductance ($L$) associated with it. The voltage drop across the motor ($em$) is proportional to the motor speed ($\omega$) times a voltage constant ($Kv$). The torque produced by the motor ($Tm$) is equal to the motor current ($I$) times a current constant ($Kt$). The rotor and the load both have certain inertias associated with them. Also, the shaft has some rotational viscous damping ($b$) and some rotational elastic spring ($K$) associated with it. The model does not include any frictional effects. Consider the two systems in Figure 4 separately. First consider the electric circuit on the left half of Figure 4. Applying Kirchoff's Voltage Equation to the DC motor circuit yields:

$$-V+IR+L\frac{dI}{dt}+Kv\omega = 0$$

Therefore:

$$I(s) = \frac{V(s)-Kv\omega(s)}{Ls+R} \quad 1)$$

Applying Newton's Law to the mechanical system on the right hand side of Figure 4 yields:

**Newton's Equation**

$$\sum M_x = (J1+J2)\frac{d\omega}{dt} = Tm-b\omega-K\omega$$

$$(J1+J2)\omega(s) = KtI(s)-b\omega(s)-K\omega(s)/s$$

$$\omega(s)(s(J1+J2)+b+K/s) = KtI(s) \quad 2)$$

Substituting 1) into 2) yields:

$$\omega(s) = \frac{Kts}{s^3(J1+J2)L+s^2(R(J1+J2)+bL)+s(bR+KL+KtKv)+KR} \quad 3)$$

$$V(s)$$
This is a fairly difficult equation to work with since it is third order, however some simplifications can be made. The amount of inductance in the DC motor is fairly small so one simplification that can be made is to let the inductance L equal zero. A second simplification that can be made concerns the motor shaft. If we assume the shaft to be rigid we can neglect the rotational elastic spring forces, and the motor and rotor inertia can be combined into a single inertia. The sum of J1 and J2 is denoted as J. In addition we can assume that the rotational viscous damping is zero. In summary the following assumptions were made:

Motor Inductance (L)=0  
Rotational Elastic Spring (K)=0  
Inertia J=J1+J2  
Rotational Viscous Damping (b)=0

With these assumptions the DC motor transfer function in equation 3 can be simplified to:

\[
\frac{\omega(s)}{V(s)} = \frac{(1/K_v)}{(RJ/K_tK_v)s+1} = \frac{K_m}{Ts+1}
\]

where \(K_m=\text{DC Motor Gain}\) and \(T=\text{DC Motor Time Constant}\)

From the simplifications made to the DC motor the transfer function has been reduced from third order to first order, which is easier to work with when designing a controller for the system. However, the simplifications made to reduce the system from a third to a first order system should be kept in mind when working with the physical system.

**Tachometer**

The tachometer performs the function of the sensor in this study. The tachometer puts out a signal proportional to the DC
motor speed. This signal is fed back into the computer and is used in the control algorithms. The tachometer is modeled as a simple gain Kgen in this study, however some modifications were made to the generator model to obtain this simple gain. First all frictional losses in the gear train of the tachometer were ignored. In addition the time constant of the tach was assumed to be very small so that it could be neglected. With these assumptions the tachometer can be modeled as a simple gain.

**Analog to Digital Converter**

The analog to digital converter takes the analog signal from the tachometer and converts it into a digital signal which the computer can understand. Since the A/D converter performs the same function as the D/A converter only backwards, the gain associated with the A/D converter is the inverse of the D/A gain. Therefore $K_{ADC} = 1/(K_{DAC})$.

Data was obtained from the actual system in order to gain a better understanding of the system characteristics. A voltage was applied to the input of the amplifier and the voltages were measured at the input to the motor and at the output of the tachometer. This data gave the overall gain of the motor-generator set including the motor gain, tach gain, gearing gain, and effects of the inertia. This data is listed in Table 1. A plot of the voltage recorded at the input of the motor vs. the voltage emitted by the tachometer is shown in Figure 5. This plot shows that the gain is fairly linear except at low input (motor) voltages. It also shows the "dead band" or the motor voltage necessary to drive the motor, inertia, and tachometer and produce a voltage at the terminals of the tachometer. For this set up the "dead band" voltage is approximately plus and minus two volts.

The combined open loop system is shown in Figure 6. This figure shows that the open loop transfer function is a number of gains times a first order system. These gains can be combined with the first order to system to yield the open loop transfer
<table>
<thead>
<tr>
<th>Amplifier Voltage Level</th>
<th>Voltage at Motor Terminals</th>
<th>Current</th>
<th>Tachometer Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-9.518</td>
<td>-7.400</td>
<td>-0.61</td>
<td>-6.770</td>
</tr>
<tr>
<td>-8.998</td>
<td>-7.772</td>
<td>-0.62</td>
<td>-6.450</td>
</tr>
<tr>
<td>-8.006</td>
<td>-7.221</td>
<td>-0.60</td>
<td>-5.230</td>
</tr>
<tr>
<td>-6.992</td>
<td>-6.313</td>
<td>-0.60</td>
<td>-4.230</td>
</tr>
<tr>
<td>-6.008</td>
<td>-5.425</td>
<td>-0.60</td>
<td>-3.410</td>
</tr>
<tr>
<td>-5.004</td>
<td>-4.524</td>
<td>-0.58</td>
<td>-2.770</td>
</tr>
<tr>
<td>-4.009</td>
<td>-3.630</td>
<td>-0.55</td>
<td>-2.020</td>
</tr>
<tr>
<td>-3.006</td>
<td>-2.728</td>
<td>-0.50</td>
<td>-1.222</td>
</tr>
<tr>
<td>-2.900</td>
<td>-2.380</td>
<td>-0.45</td>
<td>-0.480</td>
</tr>
<tr>
<td>2.502</td>
<td>2.193</td>
<td>0.75</td>
<td>0.000</td>
</tr>
<tr>
<td>2.807</td>
<td>2.485</td>
<td>0.43</td>
<td>0.716</td>
</tr>
<tr>
<td>3.000</td>
<td>2.654</td>
<td>0.43</td>
<td>0.817</td>
</tr>
<tr>
<td>3.498</td>
<td>3.109</td>
<td>0.45</td>
<td>1.156</td>
</tr>
<tr>
<td>4.002</td>
<td>3.563</td>
<td>0.50</td>
<td>1.510</td>
</tr>
<tr>
<td>4.503</td>
<td>4.018</td>
<td>0.51</td>
<td>1.893</td>
</tr>
<tr>
<td>5.011</td>
<td>4.470</td>
<td>0.54</td>
<td>2.210</td>
</tr>
<tr>
<td>5.500</td>
<td>4.910</td>
<td>0.55</td>
<td>2.589</td>
</tr>
<tr>
<td>6.005</td>
<td>5.361</td>
<td>0.57</td>
<td>2.950</td>
</tr>
<tr>
<td>6.500</td>
<td>5.802</td>
<td>0.58</td>
<td>3.280</td>
</tr>
<tr>
<td>7.001</td>
<td>6.252</td>
<td>0.59</td>
<td>3.660</td>
</tr>
<tr>
<td>7.498</td>
<td>6.701</td>
<td>0.60</td>
<td>4.180</td>
</tr>
<tr>
<td>8.003</td>
<td>7.155</td>
<td>0.61</td>
<td>4.600</td>
</tr>
<tr>
<td>8.499</td>
<td>7.601</td>
<td>0.61</td>
<td>5.120</td>
</tr>
<tr>
<td>9.002</td>
<td>8.055</td>
<td>0.62</td>
<td>5.570</td>
</tr>
<tr>
<td>9.501</td>
<td>8.486</td>
<td>0.63</td>
<td>6.150</td>
</tr>
</tbody>
</table>

**TABLE 1**

**MOTOR-TACHOMETER VOLTAGE DATA**
FIGURE 5
PLOT OF TACHOMETER vs. MOTOR VOLTAGE
function:

\[ Y(s) = \frac{K_{ol}}{M(s) (T_{ol})s + 1} \] 

where: 
- \( M(s) \) = Control Effort 
- \( Y(s) \) = State Variable 
- \( T_{ol} \) = Open Loop Time Constant 
- \( K_{ol} \) = Open Loop Gain

The next section of this report examines ways to evaluate the open loop gain and time constant and their discrete time equivalents.
FIGURE 6
COMBINED OPEN LOOP SYSTEM
SYSTEM IDENTIFICATION:

This section of the report describes methods for obtaining the discrete time system parameters. These values are used in the design of the system controllers in the next section of this report. Four different types of estimation methods are used to obtain the system parameters. The first method discussed involves using a plot of the system response to a step input to determine the open loop gain and time constant. The discrete time parameters are calculated from these values. The second method of parameter estimation involves the use of a parallel Model Reference Adaptive System to calculate the discrete time parameters directly. The third type of estimation procedure is a Recursive Least Squares Method which again calculates the discrete time parameters directly. The final method of parameter estimation involves the use of a Tunable Estimator. This estimator is a more general estimator which can be modified to become either the MRAS or Recursive Least Squares Estimator. However, since it is a general type of procedure, it offers some added flexibility.

All of the estimation procedures mentioned were implemented through the use of a computer program. The majority of the program was written in FORTRAN with Z-80 assembly language routines used to transfer signals to the external DC motor system from the computer and vice versa. An explanation of the computer program follows.

System Identification Program

The flow charts for the computer program used during the System Identification Section are shown in Appendix A along with the FORTRAN and Assembly Language Source Codes. The program performs all calculations "off-line", that is, all the data is collected first and then all calculations are performed. The calculation could have been performed "on-line" (while the data was being collected) however no benefits would have been obtained
from "on-line" calculations so it was decided to use the "off-line" method instead.

The program uses preset interrupts to time the transfer of data both into and out of the computer. The interrupt period was preset by setting appropriate switches on a specified board of the XYCOM. For the System Identification Phase the interrupt frequency was set to .100 seconds. This means that data was output to the DC motor system or read in from the system every .100 seconds for some specified amount of time.

The computer program uses a main program to perform initialization, calculate the output signals, calculate the amount of time to output the signals and read in the response, and initialize the interrupts.

The calculated computer output signal which is the system input signal must be a frequency rich signal so that it sufficiently excites the system. This means that the signal should have some corners on it which are very rich in frequency. In the program the input signal may be a step input, ramp input, or trapezoidal (see Figure 7). The user specifies the maximum magnitude, pulse width, and rise time of the pulse. From this data the appropriate system input signals are calculated.

After the interrupts have been initialized in the assembly language code, a subroutine is called which does nothing more than loop for a certain amount of time specified by the user in the main program. This looping occurs only after the interrupts have been enabled. While this looping is occurring the program is being continually interrupted every .100 seconds. When this interrupt occurs the program jumps to the FORTRAN Interrupt Service Routine specified in the assembly language code. In the Interrupts Service Routine the output signal calculated in the main program is output to the system through the use of an assembly language subroutine and the corresponding system response, measured by the tachometer, is read into the computer through a similar subroutine. This continues until the time specified by the user in the main program expires.
FIGURE 7
SYSTEM INPUT SIGNALS

TRAPEZOIDAL INPUT

RAMP INPUT

STEP INPUT
Now that the signal has been output and the system response measured by the tachometer has been obtained, the estimation procedures can begin. These procedures are now presented.

Open Loop Gain and Time Constant
Using the the open loop gain and time constant is the easiest method of obtaining the open loop parameters, but it has the disadvantage that it cannot be performed on-line. This means the procedure cannot be used to estimate the system parameters while the control algorithm is running. It is also not suitable for more general, higher order systems.

Recall the combined open loop system shown in Figure 6. In the continuous time case the transfer function for a first order system involves one gain and one time constant. These values can be determined by "exciting" the system with an appropriate frequency rich input signal and then measuring the system response. A step input with no ramp was used to excite the system. For a step input the system response can be calculated as:

$$Y(t) = K_0(1 - e^{-t/T_0})M(t)$$

The system input signal along with a typical example of the system response is shown in Figure 8 (ref. #2).

It was noted that the response signal did achieve a steady state value, however the signal was rather erratic and had variations of up to 10% of the steady state value. This "noise" which was of higher frequencies was filtered out of the response signal through the use of an adjustable filter. The filter was a low-pass type which had cut off frequencies of 30, 15, and 6 Hz. It was found that the 30 Hz cut-off was very effective at reducing the "noise" in the response signal. Decreasing the cut-off to 15 and 6 Hz did not improve the signal much further. A check was also made to determine the effect of filtering on the open loop gain and time constant. Even with a cut-off of 6 Hz,
FIGURE 8
SYSTEM RESPONSE

STEP INPUT SIGNAL

\[ y(t) \]

\[ Y_{max} \]

\[ 0.63Y_{max} \]

\[ t \]

Tol
these values did not change from the case with no filter. Filtering this signal helped in obtaining the open loop parameters as well as improving the stability of the other estimation procedures as shall be seen later.

The open loop gain $K_{01}$ is determined by knowing the input magnitude and the steady-state response magnitude. The open loop gain is calculated by:

$$K_{01} = \frac{\text{Steady State Response Magnitude}}{\text{Input Magnitude}}$$

A time constant is described as the amount of time it takes for a system to reach $1/e$ (where $e=2.78$ and $1/e$ is approx. 63%) of its steady state value. This point, shown in Figure 8, can be determined given the system response.

Now that the continuous time parameters have been obtained they must be converted to their discrete time equivalents. Consider a first order system in the continuous time case (see Figure 6):

$$M(s) = \frac{K_{01} Y(s)}{T_{01}s+1} \quad (6a)$$

where: $M(s)$=Control Effort
$Y(s)$=Output Signal
$K_{01}$=Open Loop Gain
$T_{01}$=Open Loop Time Constant

Multiplying through and taking the inverse-LaPlace Transform of both sides yields the state equation:

$$\frac{dY}{dt} = aY + bM \quad (6b)$$

where: $a = -1/T_{01}$
$b = K_{01}/T_{01}$

The discrete time state equation is written as:

$$Y(K+1) = PY(K) + QM(K) \quad (7a)$$

where: $\rho = e^{a\Delta t}$
$Q = (P-1)b/a$
$\Delta t = \text{sampling(interrupt) period} \quad (7b)$
The appropriate values of P and Q are calculated after the open loop gain and time constant are obtained by equation 7b. As discussed previously the system is somewhat non-linear, therefore different values of Kol, Tol, P, and Q are obtained with different magnitudes of the system input values. These values range from -2048 to 2047 which are the max. and min. word sizes determined in the System Modeling Section. The system non-linearity due to different magnitude input values is shown in Table 2.

**Parallel MRAS Estimator**

The second type of estimation algorithm used in this study is a parallel Model Reference Adaptive System (MRAS) (ref. #6). In this instance the MRAS is being used for identification of the system parameters which are unknown, therefore the unknown system is the DC motor system (fixed system) and the adjustable system is a computer model of the DC motor system. The procedure for using the MRAS for parameter identification (estimation) follows.

Consider the motor and the computer model in the discrete time case:

\[
Y(K+1) = P(K)Y(K) + Q(K)M(K) \quad \text{(Motor (Fixed System))} \quad 8)
\]
\[
YH(K+1) = PH(K+1)YH(K) + QH(K+1)M(K) \quad \text{(Model (Adj. System))} \quad 9)
\]

where Y is the motor speed, YH an estimate of the motor speed and M the control effort. The problem is to get the model to follow the fixed system and to get the error \(Y(K) - YH(K)\) to go to zero. To calculate this error and \(YH(K)\) values of \(PH(K)\) and \(QH(K)\) are needed. First an 'a priori' estimate of \(YH(K+1)\) must be obtained. This is performed by:

\[
YHO(K+1) = PH(K)YH(K) + QH(K)M(K) \quad 10)
\]

Next an 'a priori' error is calculated by:

\[
EO(K+1) = Y(K+1) - YHO(K+1) \quad 11)
\]

The error at time \((K+1)\) is calculated by:
### TABLE 2

<table>
<thead>
<tr>
<th>STEP INPUT MAGNITUDE</th>
<th>OPEN LOOP GAIN (Ko1)</th>
<th>OPEN LOOP TIME CONSTANT (To1)</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.39</td>
<td>1.95 sec</td>
<td>.950</td>
<td>.0195</td>
</tr>
<tr>
<td>1000</td>
<td>0.60</td>
<td>1.65 sec</td>
<td>.941</td>
<td>.0353</td>
</tr>
<tr>
<td>1500</td>
<td>0.67</td>
<td>1.60 sec</td>
<td>.939</td>
<td>.0406</td>
</tr>
<tr>
<td>2000</td>
<td>0.71</td>
<td>1.60 sec</td>
<td>.939</td>
<td>.0430</td>
</tr>
</tbody>
</table>

CONTINUOUS AND DISCRETE TIME PARAMETERS WITH VARYING INPUT MAGNITUDES
\[ E(K+1) = ED(K+1) / (1 + K_1 YH(K)^2 + K_2 M(K)^2) \]  
\[ K_1 \] and \[ K_2 \] are gains specified by the user to aid in the rate of convergence of the algorithm. These two gains must be greater than zero for stability purposes.

PH and QH at \( (K+1) \) can now be calculated by:

\[ \text{PH}(K+1) = \text{PH}(K) + K_1 E(K+1) YH(K) \]  
\[ \text{QH}(K+1) = \text{QH}(K) + K_2 E(K+1) M(K) \]  

The model response is calculated from equation 9 and compared to the motor response to see how well the model follows the actual system.

In order to implement this algorithm certain values must be input by the user. The user input values are the gain values \( K_1 \) and \( K_2 \) and the initial values of PH and QH. Although it was found that the values of \( K_1 \) and \( K_2 \) did not significantly affect the rate of convergence of the algorithm, the selection of initial values of PH and QH did significantly affect the outcome. As can be seen from Figures 9 and 10 as different initial values of PH and QH were used the algorithm behaved quite differently. This occurred for all values of \( K_1 \) and \( K_2 \) and for all three different input signals. If the initial values of PH and QH were chosen to be quite close to the actual values of P and Q (motor parameters) the algorithm converged quite nicely. However, if the initial values of PH and QH were even slightly different from P and Q the algorithm converged poorly.

It should also be noted that the MRAS estimation algorithm is a very "alert" estimator. This means that a slight change in the response greatly affects the values of PH and QH. In other words, the algorithm places greater weight on current values of the system response rather than past values of the system response. Filtering the signal did help remove some of the "noise" which affects an "alert" estimator, however this still was not enough to get the estimator to perform as required. The next section examines the Recursive Least Squares Estimator.
<table>
<thead>
<tr>
<th>TIME</th>
<th>PH</th>
<th>QH</th>
<th>X</th>
<th>XH</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.900</td>
<td>.050</td>
<td>7.0</td>
<td>0.0</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>.900</td>
<td>.050</td>
<td>3.0</td>
<td>0.0</td>
<td>3.0</td>
</tr>
<tr>
<td>3</td>
<td>.900</td>
<td>.054</td>
<td>54.0</td>
<td>54.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>.900</td>
<td>.048</td>
<td>97.0</td>
<td>97.0</td>
<td>-.0</td>
</tr>
<tr>
<td>5</td>
<td>.900</td>
<td>.053</td>
<td>140.0</td>
<td>140.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>.900</td>
<td>.055</td>
<td>181.0</td>
<td>181.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>.900</td>
<td>.054</td>
<td>217.0</td>
<td>217.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>.901</td>
<td>.058</td>
<td>254.0</td>
<td>254.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>.901</td>
<td>.057</td>
<td>286.0</td>
<td>286.0</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>.901</td>
<td>.057</td>
<td>315.0</td>
<td>315.0</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>.901</td>
<td>.059</td>
<td>343.0</td>
<td>343.0</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>.901</td>
<td>.059</td>
<td>368.0</td>
<td>368.0</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>.903</td>
<td>.062</td>
<td>394.0</td>
<td>394.0</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>.903</td>
<td>.059</td>
<td>414.0</td>
<td>414.0</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>.903</td>
<td>.062</td>
<td>436.0</td>
<td>436.0</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>.903</td>
<td>.063</td>
<td>457.0</td>
<td>457.0</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>.902</td>
<td>.062</td>
<td>474.0</td>
<td>474.0</td>
<td>0.0</td>
</tr>
<tr>
<td>18</td>
<td>.903</td>
<td>.062</td>
<td>490.0</td>
<td>490.0</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>.902</td>
<td>.063</td>
<td>502.0</td>
<td>502.0</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>.903</td>
<td>.063</td>
<td>517.0</td>
<td>517.0</td>
<td>0.0</td>
</tr>
<tr>
<td>21</td>
<td>.903</td>
<td>.062</td>
<td>529.0</td>
<td>529.0</td>
<td>0.0</td>
</tr>
<tr>
<td>22</td>
<td>.903</td>
<td>.063</td>
<td>541.0</td>
<td>541.0</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>.904</td>
<td>.064</td>
<td>553.0</td>
<td>553.0</td>
<td>0.0</td>
</tr>
<tr>
<td>24</td>
<td>.904</td>
<td>.064</td>
<td>564.0</td>
<td>564.0</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>.903</td>
<td>.063</td>
<td>572.0</td>
<td>572.0</td>
<td>0.0</td>
</tr>
<tr>
<td>26</td>
<td>.905</td>
<td>.066</td>
<td>583.0</td>
<td>583.0</td>
<td>0.0</td>
</tr>
<tr>
<td>27</td>
<td>.904</td>
<td>.065</td>
<td>592.0</td>
<td>592.0</td>
<td>0.0</td>
</tr>
<tr>
<td>28</td>
<td>.905</td>
<td>.066</td>
<td>601.0</td>
<td>601.0</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>.906</td>
<td>.068</td>
<td>612.0</td>
<td>612.0</td>
<td>0.0</td>
</tr>
<tr>
<td>30</td>
<td>.902</td>
<td>.062</td>
<td>614.0</td>
<td>614.0</td>
<td>0.0</td>
</tr>
<tr>
<td>31</td>
<td>.905</td>
<td>.066</td>
<td>621.0</td>
<td>621.0</td>
<td>0.0</td>
</tr>
<tr>
<td>32</td>
<td>.905</td>
<td>.066</td>
<td>628.0</td>
<td>628.0</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>.904</td>
<td>.065</td>
<td>633.0</td>
<td>633.0</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>.905</td>
<td>.065</td>
<td>638.0</td>
<td>638.0</td>
<td>0.0</td>
</tr>
<tr>
<td>35</td>
<td>.906</td>
<td>.067</td>
<td>645.0</td>
<td>645.0</td>
<td>0.0</td>
</tr>
<tr>
<td>36</td>
<td>.905</td>
<td>.066</td>
<td>649.0</td>
<td>649.0</td>
<td>0.0</td>
</tr>
<tr>
<td>37</td>
<td>.904</td>
<td>.065</td>
<td>652.0</td>
<td>652.0</td>
<td>0.0</td>
</tr>
<tr>
<td>38</td>
<td>.904</td>
<td>.065</td>
<td>654.0</td>
<td>654.0</td>
<td>0.0</td>
</tr>
<tr>
<td>39</td>
<td>.906</td>
<td>.066</td>
<td>661.0</td>
<td>661.0</td>
<td>0.0</td>
</tr>
<tr>
<td>40</td>
<td>.904</td>
<td>.065</td>
<td>662.0</td>
<td>662.0</td>
<td>0.0</td>
</tr>
<tr>
<td>41</td>
<td>.904</td>
<td>.065</td>
<td>664.0</td>
<td>664.0</td>
<td>0.0</td>
</tr>
<tr>
<td>42</td>
<td>.915</td>
<td>.067</td>
<td>668.0</td>
<td>668.0</td>
<td>0.0</td>
</tr>
<tr>
<td>43</td>
<td>.937</td>
<td>.067</td>
<td>672.0</td>
<td>672.0</td>
<td>0.0</td>
</tr>
<tr>
<td>44</td>
<td>.957</td>
<td>.067</td>
<td>675.0</td>
<td>675.0</td>
<td>0.0</td>
</tr>
<tr>
<td>45</td>
<td>.975</td>
<td>.067</td>
<td>678.0</td>
<td>678.0</td>
<td>0.0</td>
</tr>
<tr>
<td>46</td>
<td>.850</td>
<td>.066</td>
<td>680.0</td>
<td>680.0</td>
<td>0.0</td>
</tr>
<tr>
<td>47</td>
<td>.933</td>
<td>.064</td>
<td>678.0</td>
<td>678.0</td>
<td>0.0</td>
</tr>
<tr>
<td>48</td>
<td>.906</td>
<td>.060</td>
<td>663.0</td>
<td>663.0</td>
<td>0.0</td>
</tr>
<tr>
<td>49</td>
<td>.905</td>
<td>.067</td>
<td>665.0</td>
<td>665.0</td>
<td>0.0</td>
</tr>
<tr>
<td>50</td>
<td>.934</td>
<td>.065</td>
<td>684.0</td>
<td>684.0</td>
<td>0.0</td>
</tr>
<tr>
<td>51</td>
<td>.968</td>
<td>.068</td>
<td>688.0</td>
<td>688.0</td>
<td>0.0</td>
</tr>
<tr>
<td>52</td>
<td>.906</td>
<td>.066</td>
<td>689.0</td>
<td>689.0</td>
<td>0.0</td>
</tr>
<tr>
<td>53</td>
<td>.936</td>
<td>.067</td>
<td>691.0</td>
<td>691.0</td>
<td>0.0</td>
</tr>
<tr>
<td>54</td>
<td>.906</td>
<td>.067</td>
<td>693.0</td>
<td>693.0</td>
<td>0.0</td>
</tr>
<tr>
<td>55</td>
<td>.904</td>
<td>.065</td>
<td>692.0</td>
<td>692.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**FIGURE 9**

- Initial Values of PH and QH slightly different from Actual Values.
<table>
<thead>
<tr>
<th>TIME</th>
<th>PH</th>
<th>QH</th>
<th>X</th>
<th>KH</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.950</td>
<td>.640</td>
<td>7.0</td>
<td>0.0</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>.950</td>
<td>.640</td>
<td>3.0</td>
<td>0.0</td>
<td>3.0</td>
</tr>
<tr>
<td>3</td>
<td>.853</td>
<td>.654</td>
<td>54.0</td>
<td>54.0</td>
<td>.0</td>
</tr>
<tr>
<td>4</td>
<td>.650</td>
<td>.646</td>
<td>97.0</td>
<td>97.0</td>
<td>.0</td>
</tr>
<tr>
<td>5</td>
<td>.650</td>
<td>.648</td>
<td>140.0</td>
<td>140.0</td>
<td>.0</td>
</tr>
<tr>
<td>6</td>
<td>.650</td>
<td>.648</td>
<td>181.0</td>
<td>181.0</td>
<td>.0</td>
</tr>
<tr>
<td>7</td>
<td>.650</td>
<td>.645</td>
<td>217.0</td>
<td>217.0</td>
<td>.0</td>
</tr>
<tr>
<td>8</td>
<td>.650</td>
<td>.648</td>
<td>254.0</td>
<td>254.0</td>
<td>.0</td>
</tr>
<tr>
<td>9</td>
<td>.650</td>
<td>.645</td>
<td>286.0</td>
<td>286.0</td>
<td>.0</td>
</tr>
<tr>
<td>10</td>
<td>.650</td>
<td>.644</td>
<td>315.0</td>
<td>315.0</td>
<td>.0</td>
</tr>
<tr>
<td>11</td>
<td>.650</td>
<td>.644</td>
<td>348.0</td>
<td>348.0</td>
<td>.0</td>
</tr>
<tr>
<td>12</td>
<td>.650</td>
<td>.643</td>
<td>368.0</td>
<td>368.0</td>
<td>.0</td>
</tr>
<tr>
<td>13</td>
<td>.650</td>
<td>.645</td>
<td>394.0</td>
<td>394.0</td>
<td>.0</td>
</tr>
<tr>
<td>14</td>
<td>.650</td>
<td>.641</td>
<td>414.0</td>
<td>414.0</td>
<td>.0</td>
</tr>
<tr>
<td>15</td>
<td>.650</td>
<td>.643</td>
<td>436.0</td>
<td>436.0</td>
<td>.0</td>
</tr>
<tr>
<td>16</td>
<td>.650</td>
<td>.643</td>
<td>457.0</td>
<td>457.0</td>
<td>.0</td>
</tr>
<tr>
<td>17</td>
<td>.650</td>
<td>.641</td>
<td>474.0</td>
<td>474.0</td>
<td>.0</td>
</tr>
<tr>
<td>18</td>
<td>.650</td>
<td>.641</td>
<td>490.0</td>
<td>490.0</td>
<td>.0</td>
</tr>
<tr>
<td>19</td>
<td>.650</td>
<td>.638</td>
<td>502.0</td>
<td>502.0</td>
<td>.0</td>
</tr>
<tr>
<td>20</td>
<td>.650</td>
<td>.641</td>
<td>517.0</td>
<td>517.0</td>
<td>.0</td>
</tr>
<tr>
<td>21</td>
<td>.650</td>
<td>.639</td>
<td>529.0</td>
<td>529.0</td>
<td>.0</td>
</tr>
<tr>
<td>22</td>
<td>.650</td>
<td>.640</td>
<td>541.0</td>
<td>541.0</td>
<td>.0</td>
</tr>
<tr>
<td>23</td>
<td>.650</td>
<td>.640</td>
<td>553.0</td>
<td>553.0</td>
<td>.0</td>
</tr>
<tr>
<td>24</td>
<td>.650</td>
<td>.640</td>
<td>564.0</td>
<td>564.0</td>
<td>.0</td>
</tr>
<tr>
<td>25</td>
<td>.650</td>
<td>.638</td>
<td>572.0</td>
<td>572.0</td>
<td>.0</td>
</tr>
<tr>
<td>26</td>
<td>.650</td>
<td>.641</td>
<td>583.0</td>
<td>583.0</td>
<td>.0</td>
</tr>
<tr>
<td>27</td>
<td>.650</td>
<td>.640</td>
<td>592.0</td>
<td>592.0</td>
<td>.0</td>
</tr>
<tr>
<td>28</td>
<td>.650</td>
<td>.640</td>
<td>601.0</td>
<td>601.0</td>
<td>.0</td>
</tr>
<tr>
<td>29</td>
<td>.650</td>
<td>.642</td>
<td>612.0</td>
<td>612.0</td>
<td>.0</td>
</tr>
<tr>
<td>30</td>
<td>.650</td>
<td>.636</td>
<td>614.0</td>
<td>614.0</td>
<td>.0</td>
</tr>
<tr>
<td>31</td>
<td>.650</td>
<td>.640</td>
<td>621.0</td>
<td>621.0</td>
<td>.0</td>
</tr>
<tr>
<td>32</td>
<td>.650</td>
<td>.640</td>
<td>628.0</td>
<td>628.0</td>
<td>.0</td>
</tr>
<tr>
<td>33</td>
<td>.650</td>
<td>.639</td>
<td>633.0</td>
<td>633.0</td>
<td>.0</td>
</tr>
<tr>
<td>34</td>
<td>.650</td>
<td>.638</td>
<td>638.0</td>
<td>638.0</td>
<td>.0</td>
</tr>
<tr>
<td>35</td>
<td>.650</td>
<td>.641</td>
<td>645.0</td>
<td>645.0</td>
<td>.0</td>
</tr>
<tr>
<td>36</td>
<td>.650</td>
<td>.638</td>
<td>649.0</td>
<td>649.0</td>
<td>.0</td>
</tr>
<tr>
<td>37</td>
<td>.650</td>
<td>.638</td>
<td>652.0</td>
<td>652.0</td>
<td>.0</td>
</tr>
<tr>
<td>38</td>
<td>.650</td>
<td>.638</td>
<td>654.0</td>
<td>654.0</td>
<td>.0</td>
</tr>
<tr>
<td>39</td>
<td>.650</td>
<td>.641</td>
<td>661.0</td>
<td>661.0</td>
<td>.0</td>
</tr>
<tr>
<td>40</td>
<td>.650</td>
<td>.637</td>
<td>662.0</td>
<td>662.0</td>
<td>.0</td>
</tr>
<tr>
<td>41</td>
<td>.650</td>
<td>.633</td>
<td>664.0</td>
<td>664.0</td>
<td>.0</td>
</tr>
<tr>
<td>42</td>
<td>.650</td>
<td>.639</td>
<td>668.0</td>
<td>668.0</td>
<td>.0</td>
</tr>
<tr>
<td>43</td>
<td>.650</td>
<td>.640</td>
<td>672.0</td>
<td>672.0</td>
<td>.0</td>
</tr>
<tr>
<td>44</td>
<td>.650</td>
<td>.639</td>
<td>675.0</td>
<td>675.0</td>
<td>.0</td>
</tr>
<tr>
<td>45</td>
<td>.650</td>
<td>.639</td>
<td>676.0</td>
<td>676.0</td>
<td>.0</td>
</tr>
<tr>
<td>46</td>
<td>.650</td>
<td>.639</td>
<td>680.0</td>
<td>680.0</td>
<td>.0</td>
</tr>
<tr>
<td>47</td>
<td>.650</td>
<td>.636</td>
<td>678.0</td>
<td>678.0</td>
<td>.0</td>
</tr>
<tr>
<td>48</td>
<td>.650</td>
<td>.641</td>
<td>683.0</td>
<td>683.0</td>
<td>.0</td>
</tr>
<tr>
<td>49</td>
<td>.650</td>
<td>.639</td>
<td>685.0</td>
<td>685.0</td>
<td>.0</td>
</tr>
<tr>
<td>50</td>
<td>.650</td>
<td>.637</td>
<td>684.0</td>
<td>684.0</td>
<td>.0</td>
</tr>
<tr>
<td>51</td>
<td>.650</td>
<td>.643</td>
<td>688.0</td>
<td>688.0</td>
<td>.0</td>
</tr>
<tr>
<td>52</td>
<td>.650</td>
<td>.638</td>
<td>689.0</td>
<td>689.0</td>
<td>.0</td>
</tr>
<tr>
<td>53</td>
<td>.650</td>
<td>.639</td>
<td>691.0</td>
<td>691.0</td>
<td>.0</td>
</tr>
<tr>
<td>54</td>
<td>.650</td>
<td>.639</td>
<td>693.0</td>
<td>693.0</td>
<td>.0</td>
</tr>
<tr>
<td>55</td>
<td>.650</td>
<td>.637</td>
<td>692.0</td>
<td>692.0</td>
<td>.0</td>
</tr>
</tbody>
</table>

**FIGURE 10**

- MRAS Estimator with Initial Values of PH and QH close to the Actual Values
Recursive Least Square Estimator

The third type of estimator used in this study is a Recursive Least Squares Method. In this procedure it is desired to minimize the square of the error between the actual measured speed and the estimate of the speed. In other words:

\[
\min J(K)=\sum(Y(K)-YH0(K))^2=\sum EO(K)^2
\]  

15)

where \(YH0(K)\) and \(EO(K)\) are 'a priori' estimates of the state (speed) and error values respectively. From equation 11 in the MRAS section it was found:

\[
EO(K+1)=Y(K+1)-YH0(K+1)
\]  

11)

substituting equation 10 into 11 yields:

\[
EO(K+1)=Y(K+1)-\Theta^T(K)\Delta(K)
\]  

16)

where:
\[
\Theta^T(K)=(-PH(K) QH(K))
\]  

17)

\[
\Delta^T(K)=(-YH(K) M(K))
\]  

18)

Substituting 16 into 15 yields:

\[
J(K)=\sum(Y(K)-\Theta^T(K)\Delta(K))^2
\]  

19)

In order to optimize the algorithm the error must be minimized with respect to the parameters \(PH\) and \(QH\), therefore:

\[
\frac{\partial J(K)}{\partial \Theta^T(K)} = 0
\]  

20)

After taking the derivative the following result is obtained:

\[
\Theta(K)=F(K)(\sum Y(K)\Delta(K-1))
\]  

21)

This is the basic Least Squares Solution.

\(F(K)\) must be continually updated at each new time increment. This is performed thru the Matrix Inversion Lemma which allows for:
\[ F(K+1) = F(K) - \frac{F(K)\Delta(K)\Delta^T(K)F(K)}{1 + \Delta^T(K)F(K)\Delta(K)} \quad (22) \]

The basic Least Squares solution is modified to get a Recursive Least Squares solution by:

\[ \theta(K+1) = \theta(K) + F(K+1)\Delta(K)E_0(K+1) \quad (23) \]

As in the case with the MRAS estimator, certain user input values are needed in order to start the Recursive Least Squares Estimator. Initial values of PH and QH are needed as well as initial values of the matrix \( F \). Once the user specifies these values, \( F(K+1) \) is calculated by equation 22, \( E_0(K+1) \) is calculated by equation 16, and the new estimates of PH and QH are calculated in equation 23. If the adaptation gain \( F(K) \) or the error \( E_0(K) \) are large the values of PH and QH change quite a bit. As the adaptation gain decreases and the error goes to zero the algorithm converges to the correct value of PH and QH.

After numerous runs with the Recursive Least Squares Estimator it was found that it was also very sensitive to the initial values of PH and QH. If these values were close to the actual values of P and Q the algorithm performed quite nicely, but as was the case with the MRAS estimator, if the initial values of PH and QH were significantly different from the actual values, the algorithm did not perform well. See Figures 11 and 12.

While the MRAS estimator was found to be a very "alert" system, the Recursive Least Squares Method was found to be fairly insensitive to fluctuations in the system response. Once the algorithm "locked on" to a value for PH and QH, which seemed to occur in the first few steps, the algorithm stayed at that value. The performance of this system was not adequate.

Since neither the MRAS or Recursive Least Squares Estimator performed up to expectations or requirements, it was decided to use a Tunable Estimator which combines the characteristics of both these types of estimators. This allows the user to tune the estimator to further improve the accuracy of the estimation. The algorithm can be tuned to be "alert" like an MRAS estimator then
<table>
<thead>
<tr>
<th>TIME</th>
<th>PH</th>
<th>QH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.943</td>
<td>.034</td>
</tr>
<tr>
<td>2</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>3</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>4</td>
<td>.937</td>
<td>.040</td>
</tr>
<tr>
<td>5</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>6</td>
<td>.941</td>
<td>.036</td>
</tr>
<tr>
<td>7</td>
<td>.940</td>
<td>.037</td>
</tr>
<tr>
<td>8</td>
<td>.937</td>
<td>.040</td>
</tr>
<tr>
<td>9</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>10</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>11</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>12</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>13</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>14</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>15</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>16</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>17</td>
<td>.940</td>
<td>.037</td>
</tr>
<tr>
<td>18</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>19</td>
<td>.937</td>
<td>.040</td>
</tr>
<tr>
<td>20</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>21</td>
<td>.940</td>
<td>.037</td>
</tr>
<tr>
<td>22</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>23</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>24</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>25</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>26</td>
<td>.940</td>
<td>.037</td>
</tr>
<tr>
<td>27</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>28</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>29</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>30</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>31</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>32</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>33</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>34</td>
<td>.940</td>
<td>.037</td>
</tr>
<tr>
<td>35</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>36</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>37</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>38</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>39</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>40</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>41</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>42</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>43</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>44</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>45</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>46</td>
<td>.937</td>
<td>.040</td>
</tr>
<tr>
<td>47</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>48</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>49</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>50</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>51</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>52</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>53</td>
<td>.939</td>
<td>.038</td>
</tr>
<tr>
<td>54</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>55</td>
<td>.938</td>
<td>.039</td>
</tr>
<tr>
<td>56</td>
<td>.938</td>
<td>.039</td>
</tr>
</tbody>
</table>

**FIGURE 11**

Recursive Least Squares Estimation with initial values of PH and QH close to the actual values.
<table>
<thead>
<tr>
<th>TIME</th>
<th>PH</th>
<th>QH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.900</td>
<td>.050</td>
</tr>
<tr>
<td>2</td>
<td>.912</td>
<td>.038</td>
</tr>
<tr>
<td>3</td>
<td>.911</td>
<td>.039</td>
</tr>
<tr>
<td>4</td>
<td>.908</td>
<td>.042</td>
</tr>
<tr>
<td>5</td>
<td>.908</td>
<td>.042</td>
</tr>
<tr>
<td>6</td>
<td>.911</td>
<td>.039</td>
</tr>
<tr>
<td>7</td>
<td>.909</td>
<td>.041</td>
</tr>
<tr>
<td>8</td>
<td>.907</td>
<td>.043</td>
</tr>
<tr>
<td>9</td>
<td>.908</td>
<td>.042</td>
</tr>
<tr>
<td>10</td>
<td>.906</td>
<td>.044</td>
</tr>
<tr>
<td>11</td>
<td>.906</td>
<td>.044</td>
</tr>
<tr>
<td>12</td>
<td>.906</td>
<td>.044</td>
</tr>
<tr>
<td>13</td>
<td>.905</td>
<td>.045</td>
</tr>
<tr>
<td>14</td>
<td>.904</td>
<td>.046</td>
</tr>
<tr>
<td>15</td>
<td>.905</td>
<td>.045</td>
</tr>
<tr>
<td>16</td>
<td>.905</td>
<td>.045</td>
</tr>
<tr>
<td>17</td>
<td>.905</td>
<td>.045</td>
</tr>
<tr>
<td>18</td>
<td>.903</td>
<td>.047</td>
</tr>
<tr>
<td>19</td>
<td>.902</td>
<td>.048</td>
</tr>
<tr>
<td>20</td>
<td>.902</td>
<td>.048</td>
</tr>
<tr>
<td>21</td>
<td>.903</td>
<td>.047</td>
</tr>
<tr>
<td>22</td>
<td>.903</td>
<td>.047</td>
</tr>
<tr>
<td>23</td>
<td>.902</td>
<td>.048</td>
</tr>
<tr>
<td>24</td>
<td>.901</td>
<td>.049</td>
</tr>
<tr>
<td>25</td>
<td>.901</td>
<td>.049</td>
</tr>
<tr>
<td>26</td>
<td>.902</td>
<td>.048</td>
</tr>
<tr>
<td>27</td>
<td>.900</td>
<td>.050</td>
</tr>
<tr>
<td>28</td>
<td>.901</td>
<td>.049</td>
</tr>
<tr>
<td>29</td>
<td>.900</td>
<td>.050</td>
</tr>
<tr>
<td>30</td>
<td>.899</td>
<td>.051</td>
</tr>
<tr>
<td>31</td>
<td>.898</td>
<td>.052</td>
</tr>
<tr>
<td>32</td>
<td>.899</td>
<td>.051</td>
</tr>
<tr>
<td>33</td>
<td>.899</td>
<td>.051</td>
</tr>
<tr>
<td>34</td>
<td>.899</td>
<td>.051</td>
</tr>
<tr>
<td>35</td>
<td>.898</td>
<td>.052</td>
</tr>
<tr>
<td>36</td>
<td>.898</td>
<td>.052</td>
</tr>
<tr>
<td>37</td>
<td>.898</td>
<td>.052</td>
</tr>
<tr>
<td>38</td>
<td>.897</td>
<td>.053</td>
</tr>
<tr>
<td>39</td>
<td>.897</td>
<td>.053</td>
</tr>
<tr>
<td>40</td>
<td>.896</td>
<td>.054</td>
</tr>
<tr>
<td>41</td>
<td>.897</td>
<td>.053</td>
</tr>
<tr>
<td>42</td>
<td>.896</td>
<td>.054</td>
</tr>
<tr>
<td>43</td>
<td>.896</td>
<td>.054</td>
</tr>
<tr>
<td>44</td>
<td>.896</td>
<td>.054</td>
</tr>
<tr>
<td>45</td>
<td>.895</td>
<td>.055</td>
</tr>
<tr>
<td>46</td>
<td>.894</td>
<td>.056</td>
</tr>
<tr>
<td>47</td>
<td>.895</td>
<td>.055</td>
</tr>
<tr>
<td>48</td>
<td>.895</td>
<td>.055</td>
</tr>
<tr>
<td>49</td>
<td>.895</td>
<td>.055</td>
</tr>
<tr>
<td>50</td>
<td>.895</td>
<td>.055</td>
</tr>
<tr>
<td>51</td>
<td>.894</td>
<td>.056</td>
</tr>
<tr>
<td>52</td>
<td>.894</td>
<td>.056</td>
</tr>
<tr>
<td>53</td>
<td>.894</td>
<td>.056</td>
</tr>
<tr>
<td>54</td>
<td>.893</td>
<td>.057</td>
</tr>
<tr>
<td>55</td>
<td>.892</td>
<td>.058</td>
</tr>
</tbody>
</table>

**FIGURE 12**

Recursive Least Squares Estimation with initial values of PH and QH slightly different from actual values.
"lock on" to the values like the Recursive Least Squares estimator. This procedure is now explained.

**Tunable Estimator**

As was the case with the MRAS and the Recursive Least Squares Estimators the goal is to get the error between the system and the model to go to zero. First an 'a priori' estimate of the speed is calculated by:

\[ YH0(K+1) = PH(K)YH(K) + QH(K)M(K) \]  \hspace{1cm} (24)

Next an 'a priori' error is calculated:

\[ E0(K+1) = Y(K+1) - YH0(K+1) \]  \hspace{1cm} (25)

Next an adaptation error is calculated by:

\[ E(K+1) = E0(K+1) / (1 + \Delta^T(K)F(K)\Delta(K)) \]  \hspace{1cm} (26)

where \( \Delta(K) \) and \( F(K) \) are the same as in equation 22.

The next step is to calculate the updated adaptation gain matrix \( F(K) \) by:

\[ F(K+1) = \frac{1}{L1} \left( F(K) - F(K)\Delta(K)\Delta^T(K)F(K) \right) \]  \hspace{1cm} (27)

where \( L1 \) and \( L2 \) are tunable gains which help to yield the desired estimator. For stability purposes \( 0 \leq L1 \leq 1 \) and \( 0 \leq L2 \leq 2 \).

The final step is to calculate the parameters \( PH \) and \( QH \) at the new time increment. This is the same as with the Recursive Least Squares Estimator in equation 23 except that \( E(K+1) \) and \( F(K) \) should be used rather than \( E0(K+1) \) and \( F(K+1) \) respectively.

The gains \( L1 \) and \( L2 \) can be used to effectively tune the estimator to a MRAS estimator, Recursive Least Squares estimator, or some combination of the two. If \( L1 = L2 = 1 \) then the estimator is a Recursive Least Squares estimator. If \( L1 = 1 \) and \( L2 = 0 \) then the MRAS estimator is realized.

It can be seen from equation 27 that if \( L1 \) and \( L2 \) are both small a larger adaptation gain \( F(K) \) will be obtained. This will yield a more "alert" system. If \( L1 \) and \( L2 \) are around 1 then the
system will "lock on" to the value like the Recursive Least Square estimator. The object is to choose L1, L2, and F(0) so that the system is very alert during the first few steps and then locks on to the estimated values.

In order to get the desired tuned estimator, L1 was chosen to be 1 and L2 was chosen to be greater than 0.5. With these values the algorithm was found to converge very well. The algorithm converged to the correct value of PH and QH regardless of the initial values of PH and QH. These results are shown in Figures 13 and 14.

Since the tunable estimator performed the best, it is the on-line estimator used in the Self-Tuning Adaptive Controller and the Robust Feedback Controller in the next section of this report.
<table>
<thead>
<tr>
<th>TIME</th>
<th>PH</th>
<th>QH</th>
<th>F11</th>
<th>F12</th>
<th>F21</th>
<th>F22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.500</td>
<td>.050</td>
<td>.010</td>
<td>0.000</td>
<td>0.000</td>
<td>0.010</td>
</tr>
<tr>
<td>2</td>
<td>.500</td>
<td>.050</td>
<td>.010</td>
<td>0.000</td>
<td>0.000</td>
<td>0.010</td>
</tr>
<tr>
<td>3</td>
<td>.500</td>
<td>.050</td>
<td>.010</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>.813</td>
<td>.049</td>
<td>.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>.858</td>
<td>.049</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>.903</td>
<td>.047</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>.910</td>
<td>.047</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>.917</td>
<td>.047</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>.926</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>.924</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>11</td>
<td>.929</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>12</td>
<td>.955</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>.927</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>14</td>
<td>.921</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>15</td>
<td>.929</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>.929</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>.933</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>.933</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>19</td>
<td>.933</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>20</td>
<td>.933</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>21</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>22</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>23</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>24</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>25</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>26</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>27</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>28</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>29</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>30</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>31</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>32</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>33</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>34</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>35</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>36</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>37</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>38</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>39</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>40</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>41</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>42</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>43</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>44</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>45</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>46</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>47</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>48</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>49</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>50</td>
<td>.938</td>
<td>.046</td>
<td>.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**FIGURE 13**

Tunable Estimator with initial values of PH and QH drastically different from actual values
### Tunable Estimator Estimation Procedure

**System Results for Input = 1000**

**Initial PH = 0.94 Initial QH = 0.03 F = 10.000**

\[
\text{Lambda 1} = 1.0000 \quad \text{Lambda 2} = 0.5000
\]

<table>
<thead>
<tr>
<th>N</th>
<th>PH</th>
<th>QH</th>
<th>E11</th>
<th>E12</th>
<th>E21</th>
<th>E22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.940</td>
<td>0.030</td>
<td>10.000</td>
<td>0.000</td>
<td>0.000</td>
<td>10.000</td>
</tr>
<tr>
<td>2</td>
<td>0.940</td>
<td>0.030</td>
<td>10.000</td>
<td>0.000</td>
<td>0.000</td>
<td>10.000</td>
</tr>
<tr>
<td>3</td>
<td>0.940</td>
<td>0.040</td>
<td>10.000</td>
<td>0.000</td>
<td>0.000</td>
<td>10.000</td>
</tr>
<tr>
<td>4</td>
<td>0.854</td>
<td>0.040</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>0.868</td>
<td>0.040</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>0.907</td>
<td>0.047</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>0.915</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>0.922</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>0.929</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>0.927</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>11</td>
<td>0.926</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>12</td>
<td>0.927</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>0.929</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>14</td>
<td>0.928</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>15</td>
<td>0.928</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>0.931</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>0.931</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>0.934</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>19</td>
<td>0.934</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>20</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>21</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>22</td>
<td>0.930</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>23</td>
<td>0.931</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>24</td>
<td>0.931</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>25</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>26</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>27</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>28</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>29</td>
<td>0.932</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>30</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>31</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>32</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>33</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>34</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>35</td>
<td>0.933</td>
<td>0.045</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Figure 14**

Tunable Estimator with initial values of PH and QH close to the actual values.
CONTROLLER DESIGN

This section of the report details the design of the system controllers. The purpose of the controllers is to control the speed of the DC motor to some steady state value chosen by the user. The three types of controllers that are used to control the motor speed are a Digital Proportional-Integral (PI) Controller, a Self-Tuning Adaptive Controller, and a Robust Feedback Controller. The Digital PI controller uses the discrete time system parameters obtained from the previous section directly in order to obtain the appropriate controller gains. Both the Self-Tuning Adaptive Controller and the Robust Feedback Controller use the Tunable Estimator from the previous section in an on-line manner to control the motor speed. As with the System Identification Phase, all the controllers were implemented through the use of computer programs. Most of the programs were written in FORTRAN with assembly language routines used to transfer data to and from the controller. The next section details the Digital PI Controller Design.

Digital Proportional-Integral Controller

The first controller considered is the Digital Proportional-Integral (PI) Controller. This controller is sometimes referred to as the proportional plus reset action controller because the integral action "resets" the position of the proportional band (ref. #3). A schematic of the Digital PI controller is shown in Figure 15.

As shown in Figure 15, a reference value \( R(Z) \) is input by the user. The actual motor speed \( Y(Z) \) is subtracted from the reference value to produce the error value \( E(Z) \). This error is then multiplied by an proportional term \( K_p \) and an integral term \( (K_i \Delta t Z/(Z-1)) \) where the \( \Delta t \) term is the sampling frequency and \( K_i \) is the integral gain. In the continuous time case an integration is denoted by \( 1/S \), but in the discrete time case an integration corresponds to \( Z/(Z-1) \). Also, in the discrete time case an integration refers to the past value, therefore the manipulated
FIGURE 15
DIGITAL PROPORTIONAL - INTEGRAL (PI) CONTROLLER
value $M(Z)$ is equal to the current error multiplied by a proportional term ($K_p$) and the integrated past error. The manipulated value $M(Z)$ is then used to drive the system.

The plant or DC Motor was modeled in the System Modeling Section as a first order system (see Figure 6). The relationship between the continuous time and discrete time parameters was then discussed in the System Identification Section of this report. In the discrete time case the representation of a first order system is given by $Q/(Z-P)$ where $Q$ and $P$ are the discrete time parameters discussed in the previous section. The manipulated value $M(Z)$ then drives the system to produce the desired motor speed. The motor speed $Y(Z)$ is continually fed back and the procedure repeated.

The only values which may be chosen by the user to control the motor speed are the reference value $R(Z)$ and the proportional and integral gains $K_p$ and $K_i$ respectively. Once the reference value has been selected, the values of $P$ and $Q$ of the system are fixed, therefore the only way to control the system is through the selection of $K_p$ and $K_i$.

In order to determine the values for $K_p$ and $K_i$ the closed loop system shown in Figure 15 is examined. The transfer function is given by:

$$Y(Z) = \frac{QZ(K_p + Ki\Delta t) - QKp}{R(Z)Z^2 + (QKp + QK_i\Delta t - 1 - P)Z + (P - QKp)} \quad 28$$

and the characteristic equation is given by:

$$Z^2 + (QK_p + QK_i\Delta t - 1 - P)Z + (P - QK_p) = 0 \quad 29$$

This equation can be solved using the quadratic equation:

$$Z = \frac{-B/2 \pm \sqrt{B^2 - 4C}}{2} \quad 30$$

where: $B = QK_p + QK_i\Delta t - 1 - P$

$C = P - QK_p$

The restriction for $Z$ is that the magnitude of the value of $Z$ must be less than one for stability purposes. In the s-plane the poles must be on the left-hand side of the plane for the
system to be stable and in the Z-plane the magnitude of the pole must be less than one for a stable system (see Figure 16). This allows for the poles to be all real, all imaginary, or both real and imaginary. Including some imaginary part in the selection of poles introduces some damping into the system which causes oscillation. If the pole magnitude is less than one this oscillation dies out eventually, but if the magnitude is equal to one the oscillation will continue (ref. # 3). A small amount of damping in the system may be desired since it can actually speed up the convergence of the speed to the steady state value.

Appendix C contains the flowcharts and source code for the Digital PI Controller program written to control the motor speed. In the program the user inputs the desired speed, pole locations, and values for P and Q. This means that the user must have measured the system parameters P and Q previously for the desired reference speed. From the pole locations and the values of P and Q entered by the user, the proportional and integral gains are calculated using equation 30. The proportional and integral gains are then used to calculate the gains for the control law where the control law is:  

\[ M(K) = M(K-1) + (Kp + Ki \Delta t) E(K) - KpE(K-1). \]

In the control law \( \Delta t \) is the interrupt period preset by the user, \( E(K) \) is the current speed error and \( E(K-1) \) is the previous error, and \( M(K) \) is the manipulated value or control effort output to the system. The error is measured in an Interrupt Service Routine similar to the one used in the System Identification Programs. The control effort is also calculated in the Interrupt Service Routine and output to the system. This procedure occurs for a certain amount of time predetermined by the user. When completed the user may print out the system response to the control, repeat the control procedure, or quit.

The problem with the Digital PI Controller is that since the values of P and Q change for different output speeds and reference values, calculating a single set of gains based on one particular speed may control the system very well for that particular speed, but may do a poor job of controlling the system
FIGURE 16
STABILITY UNIT CIRCLE IN Z-PLANE
at a different speed. Therefore, a single set of gains may not control the system for all speeds due to the non-linearities in the system. The next section of this report examines a way of using a parameter estimator to overcome the problem of non-linearities in the system.

**Self-Tuning Adaptive Controller**

The schematic for the Self-Tuning Adaptive Controller is shown in Figure 17. This controller is the same as the Digital PI Controller except that it contains one additional loop. This is an Adaptation Loop which continually measures the system output and calculates the system parameters P and Q through the use of the Tunable Estimator described in the System Identification Section. These system parameters are used in the control laws for the Digital PI Controller to calculate the proportional and integral gains needed to obtain the desired performance. An explanation of the program to control the system using a Self-Tuning Adaptive Controller follows. The flowcharts and source code are listed in Appendix D.

The user inputs the reference speed and the desired pole placement in the Z-Plane along with information pertaining to the Tunable Estimator. From the pole placement, values of B and C can be calculated from equation 30. This data is then transferred to the Interrupt Service Routine where the control takes place. First a signal is output to the system and the system response is measured. The system response measurement is used in the Tunable Estimator Algorithm to calculate the system parameters. The Tunable Estimator is included in the Interrupt Service Routine of the Self-Tuning Adaptive Controller so that the system parameters P and Q can be continually updated. After every updating of P and Q, the proportional and integral gains for the Digital PI Controller are calculated and used in the control law just like in the Digital PI Controller Design Section.

Due to the time limitations imposed by the interrupt period,
FIGURE 17
SELF-TUNING ADAPTIVE
DIGITAL PROPORTIONAL-INTEGRAL (PD) CONTROLLER
the Tunable Estimator in the Interrupt Service Routine was modified to save time. As seen from Figures 13 and 14, the off diagonal terms of the adaptation gain matrix $F$ are initially zero and remain zero for all time. By eliminating these two terms in the ISR of the controller, quite a bit of time was saved. In addition, it was found that the estimator performed best when the gains Lambda 1 and Lambda 2 were found to be 1.0 and 0.5 respectively. By including these values directly in the estimator calculations, additional time was saved.

The advantage of the Self-Tuning Controller is that the system parameters are calculated on-line so that any reference value can be used without determining $P$ and $Q$ prior to implementing the controller. This controller compensates for non-linearities in the system by calculating the discrete time parameters on-line and using them directly in the control laws.

**Robust Feedback Controller**

The third type of controller used in this study is a Robust Feedback Controller. This controller is designed to be insensitive to changes in the system parameters $P$ and $Q$. Since it is insensitive to parameter variations, the non-linearities in the system do not affect it as greatly.

Considering initially only a proportional controller, the basis of the Robust Feedback Controller is the assumption that the manipulated value $M(K)$ can be calculated by:

$$M(K) = -k1Y(K) - k2Y(K+1)$$  \hspace{1cm} (31)

From equation 7 we know that $Y(K+1) = PY(K) + QM(K)$

Plugging this into equation 31 yields:

$$(1+Qk2)Y(K+1) = (P-Qk1)Y(K)$$  \hspace{1cm} (32)

Solving for the eigen value yields:

$$\lambda = (P-Qk1)/(1+Qk2)$$  \hspace{1cm} (33)

In order to determine the sensitivity to changes in $P$ and $Q$, the partial derivative of the eigen value with respect to these two parameters is needed.
\[
\frac{d\pi}{dP} = (1 + Qk2)^{-1} \quad \text{(this equals 1 for } k2 = 0) \tag{34}
\]

\[
\frac{d\pi}{dQ} = -(k1 + Pk2)(1 + Qk2)^{-2} \quad \text{(this equals } -k1 \text{ for } k2 = 0) \tag{35}
\]

By selecting a large value for \( k2 \) it is possible to reduce these sensitivity measures and still select \( k1 \) (given \( k2 \)) to achieve the desired eigen value (response). The problem is that \( Y(K+1) \) is not available at time step \( K \) therefore an 'a priori' estimate of \( Y(K+1) \) must be used. Therefore the control law becomes:

\[
M(K) = -k1Y(K) - k2YH(K+1) \tag{36}
\]

where: \( YH(K+1) = PH(K)Y(K) + QH(K)M(K) \)

The values \( PH \) and \( QH \) are estimates of \( P \) and \( Q \) which must be obtained from an on-line estimator. In this case the modified Tunable Estimator discussed previously in the Self-Tuning Controller section is used.

The case considered above for the Robust Feedback Controller is a Proportional action case. If this is extended to the Proportional-Integral case the controller becomes:

\[
M(K) = M(K-1) + (K2 - K1\Delta t - Kp)Y(K) + K1\Delta t R(K) - k2YH(K+1) + k2P(K-1) \tag{37}
\]

\[
M(K) = (M(K-1) + (K2(1 - PH) - K3 - K1)Y(K) + K3R(K) + K1Y(K-1))/(1 + K2QH(K))
\]

where: \( K3 = K1\Delta t \) and \( K1 = Kp \)

Once again \( k2 \) can be used to reduce the sensitivity to changes in \( P \) and \( Q \) and \( k1 \) and \( K3 \) can be used to assign pole placement given \( k2 \). The Digital PI Robust Feedback Controller is shown in Figure 18.

Consider the Robust Feedback Controller in Figure 18. The transfer function is given by:

\[
\frac{Y(Z)}{R(Z)} = \frac{K1\Delta t QZ}{Z^2(1 + K2Q) + Z(QK1\Delta t - K2Q + KpQ - 1 - P) + (P - KpQ)} \tag{38}
\]

And the characteristic equation is:

\[
Z^2(1 + K2Q) + Z(QK1\Delta t - K2Q + KpQ - 1 - P) + (P - KpQ) = 0 \tag{39}
\]

Using the quadratic equation \( Z \) can be found by:
FIGURE 18
ROBUST FEEDBACK CONTROLLER
\[ Z = -B/2 \pm \frac{1}{2} \sqrt{B^2 - 4C} \quad \text{where:} \quad B = \frac{(QKi_\Delta t - K2Q + KpQ - 1 - P)}{(1 + K2Q)} \]
\[ C = (P - KpQ)/(1 + K2Q) \]

The Tunable Estimator is used in this controller to again update the values of \( P \) and \( Q \). With the values of \( P \) and \( Q \) and the user specified value of \( K2 \) and pole locations, the integral and proportional gains can be calculated from equation 40.

The flowcharts and source code for the Robust Feedback Controller are given in Appendix E.
ANALYSIS OF RESULTS

This section of the report analyzes the performance results of the three different types of controllers used in this study.

As mentioned earlier, the maximum voltage that can be applied is ten volts which corresponds to an output signal of 2047. From Table 2 it was found that the gain at high speeds was approximately 0.70. This means that the controller can control references speeds where (Ref. speed/.70)≤2047. Therefore for this analysis all reference speed values will satisfy this condition. The results for the Digital PI Controller follow.

Digital Proportional-Integral Controller

The Digital PI Controller was found to work very well for all cases where P and Q were known except at low speeds (less than 400). The first results that are shown examine the effect of damping on the response of the DC motor system. Figure 19 shows the results of changing the pole locations from Z=0±0i to Z=.5±.5i for a speed value of 650 which is a mid-range speed. This graph shows that both pole locations seem to yield good performance and the error (difference between desired and actual speed) approaches zero at about the same rate. However, if the data is examined closer it can be seen that some difference do exist between the responses. Figure 20 shows the same data as Figure 19 only it examines the data between 1 and 5 seconds on a more detailed scale. This plot shows that the response of the controller with poles at Z=0±0i is much more erratic than the response of the system with poles at Z=.5±.5i. Placing the poles at Z=0±0i is called a "dead beat" controller. This method is the fastest way of obtaining the steady state value with no overshoot. It does however have the disadvantage of requiring high effort values. This high effort causes the system to be somewhat erratic or jumpy. Placing the poles at Z=.5±.5i introduces some damping into the system. This allows the system to overshoot the desired speed initially, however since the
DIGITAL PI CONTROLLER

EFFECT OF DAMPING ON SYSTEM RESPONSE
SPEED = 650

POLE LOCATIONS

\[ Z = 0 \pm 0.01 \]

\[ Z = 0.5 \pm 0.5 \]

TIME (SEC)
DIGITAL PI CONTROLLER

EFFECT OF DAMPING ON SYSTEM RESPONSE
SPEED=650

POLE LOCATIONS
Z=0±01
Z=.5±.51

ERROR (DESired-ACTUAL SPEED)

TIME (SEC)

FIGURE 20
magnitude of the poles is less than one, the system eventually converges to the desired steady state speed. Since there is some damping present, the system approaches the desired speed in a much gentler manner and thus the steady state error is much less erratic than with the dead beat controller.

As was shown in the System Identification Section, the values of the open loop gain, open loop time constant, and discrete time parameters P and Q were not constant for all speeds due to non-linearities in the system. Problems can occur with the controller and system response if the parameters are assumed to be constant for all speed ranges. Consider the problem of using the parameters for a speed of 200 to design a controller for a speed of 1340. From Table 2 it can be seen that for an output speed of 200 (input signal of 500) P and Q are found to be 0.95 and 0.02 respectively. If the desired pole locations are at \( Z = 0 \pm 0.01 \) and using values of P and Q for a speed of 200 it can be found from equation 30 that \( K_p = 50.0 \) and \( K_1 = 526.3 \). If these values are used for a controller to control a speed of 1340 problems can arise. From Table 2 for a speed of 1340 (input signal of 2000) the actual values of P and Q are 0.939 and 0.043 rather than 0.95 and 0.02. Plugging the values of P and Q for a speed of 1340 along with the gain values for a speed of 200 and working through equation 30 it can be shown that the actual pole location is \( Z = 3.78, -1.98 \). Clearly both poles have magnitudes greater that one therefore instability will result. This instability is shown in Figure 21 where the system response of a Digital PI Controller for a high speed range application using low speed parameters is compared to a high speed range application using high speed parameters. When low speed parameters are used for a high speed controller design instability can result.

Since a single set of parameters cannot control the system over all speed ranges, alternative controller designs are needed which compensate for changing parameters. The results for such a controller are examined next.
DIGITAL PI CONTROLLER

EFFECT OF USING LOW SPEED PARAMETERS FOR A HIGH SPEED CONTROLLER

SPEED = 1340

TIME (SEC)

FIGURE 21
Self-Tuning Adaptive Controller

Since the parameters are not constant over all speed ranges and system instability can result, there is a need for a controller such as a Self-Tuning Controller which uses an on-line method to measure the discrete time parameters directly and use them in the controller.

Figure 22 examines the response of the DC motor system to a Digital PI Controller and a Self-Tuning Adaptive Controller. This figure shows that both controllers have basically the same response with the Self-Tuning Controller being approximately one step (0.100 seconds) behind the Digital PI Controller. This slight lag in the response of the system using the Self-Tuning Controller could possibly be caused by the parameter estimation algorithm in the Self-Tuning Controller.

As mentioned previously, the Digital PI Controller does a rather poor job of controlling the system at low speeds. Figure 23 compares the response of the DC motor system at low speed with a Digital PI Controller and a Self-Tuning Controller. As can be seen from Figure 23, the response of the system with the Self-Tuning Controller is far better than the response with the straight PI Controller. The Self-Tuning Controller is able to obtain a stable steady state speed while the Digital PI Controller response is very erratic. The erratic response of the system using the Digital PI Controller is probably due to system non-linearities such as the dead band shown in Figure 5. The Self-Tuning Controller is able to compensate for these system non-linearities and control the speed better.

Figure 24 shows the result of using the Self-Tuning Controller on a system where instability can result with a Digital PI Controller. As mentioned before, if low speed parameters are used to design a controller for a high speed Digital PI Controller instability can result (see Figure 21). However, if a Self-Tuning Controller is applied to the system under the same circumstances the system is not unstable and the
COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER

SPEED=650
Z=0.5±0.51

ERROR
(DESIRED-ACTUAL SPEED)

TIME (SEC)
FIGURE 22
COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER

SPEED = 200
Z = 0 ± 0.01

ERROR
(DESIRE-D ACTUAL SPEED)

SELF-TUNING CONTROLLER

DIGITAL PI CONTROLLER

TIME (SEC)
COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER

EFFECT OF USING LOW SPEED PARAMETERS FOR A HIGH SPEED CONTROLLER

SPEED=1340

ERROR (DESIRED-ACTUAL SPEED)

SELF-TUNING CONTROLLER

DIGITAL PI CONTROLLER

TIME (SEC)

FIGURE 24
speed error converges to zero.

Robust Feedback Controller

The Robust Feedback Controller was found to perform quite well when compared to both the Digital PI Controller and the Self-Tuning Controller.

Figure 25 shows a comparison of the Digital PI to the Robust Feedback Controller. The graph shows that both controllers control the system quite well, but as was the case with the Self-Tuning Controller, the Robust Feedback Controller appears to be approximately one step (.100 seconds) behind the Digital PI Controller probably due to the Tunable Estimator in the Interrupt Service Routine of the Robust Feedback Controller.

Figure 26 shows the performance of the Robust Feedback Controller for a low speed application. The Digital PI Controller was found to perform rather poorly for low speed applications, but the Robust Controller performed very well. While the Digital PI Controller performance was very erratic, the Robust Controller performance was very good. It was even better that the Self-Tuning Controller for this low speed application (see Figure 23). Due to the continual updating of the system parameters and its insensitivity to changes in these parameters the Robust Feedback Controller appears to be the best controller for low speed applications.

Runs were also made to determine if the Robust Controller could control the system when the system was unstable with the Digital PI Controller. This could not be completed due to conversion overflow errors in the software routines during high speed applications. It is believed that the Robust Controller can control the system where the Digital PI Controller failed, but due to these errors this data is not available.
COMPARISON OF DIGITAL PI VS. ROBUST CONTROLLER

SPEED = 650

$z = 0.5 \pm 0.51$

ERROR
(DESIRED - ACTUAL SPEED)

TIME (SEC)

FIGURE 25
COMPARISON OF DIGITAL PI VS. ROBUST CONTROLLER

SPEED=200
Z=0±0.1

ERROR
(DESIRE ACTUAL SPEED)

ROBUST
CONTROLLER

DIGITAL PI
CONTROLLER

TIME (SEC)

FIGURE 26
SUMMARY AND CONCLUSIONS

This final section of the report summarizes the results for the three different controllers used in this project.

Digital Proportional-Integral Controller
1. The Digital PI Controller still appears to be the fastest way of obtaining the desired steady state speed (see Figure 22).
2. Increasing the damping yields better steady state error and eliminates much of the erratic steady state behavior (see Figure 20).
3. Instability can result if the exact values of the discrete time parameters P and Q are not known (see Figure 21).
4. The Digital PI Controller does not appear to be the best controller for controlling low speeds (less than 400) due to erratic steady state behavior (see Figure 23).

Self-Tuning Adaptive Controller
1. The Self-Tuning Adaptive Controller is good for applications where the discrete time parameters are not known exactly or are changing.
2. The Self-Tuning Controller appears to be slightly slower than the Digital PI Controller (see Figure 22).
3. The Self-Tuning Controller appears to be better than the Digital PI Controller for low speed applications (see Figure 23).
4. The Self-Tuning Controller can control a system which is unstable when a Digital PI Controller is used (see Figure 24).

Robust Feedback Controller
1. The Robust Feedback Controller is good for applications where the discrete time parameters are unknown or changing.
2. The Robust Feedback Controller appears to be slightly slower than the Digital PI Controller (see Figure 25).
3. The Robust Feedback Controller appears to be the best for
controlling the system at low speeds (see Figure 26).

4. It is suspected that the Robust Feedback Controller can control a system which is unstable when a Digital PI Controller is used, however this cannot be verified at this time due to some minor software errors.
RECOMMENDATIONS

The following recommendations are made:

1. Some additional work should be done to locate the conversion overflow error which can occur in the Interrupt Service Routine of both the Self-Tuning Controller and the Robust Controller during high speed applications.

2. Different sampling periods should be used to determine the effect on controller performance.

3. Different size inertial loads should be used to determine the effect on controller performance.

4. A gain scheduling algorithm should be implemented to determine its effect on performance.
BIBLIOGRAPHY

McGraw-Hill, 1967

2. MODELING, ANALYSIS, AND CONTROL OF DYNAMIC SYSTEM,
William J. Palm III, John Wiley & Sons, 1976

3. INTRODUCING SYSTEMS AND CONTROL, Auslander, Rabins, Takahashi
McGraw-Hill, 1974

4. DIGITAL CONTROL, Rolf Isermann, Springer-Verlag, Berlin,
Heidelberg, 1981

5. DIGITAL CONTROL OF DYNAMIC SYSTEMS, Franklin,
Addison-Wesley Publishing Co., 1980

6. ME561 CLASS NOTES, A.G. Ulsoy, University of Michigan,
1983
SYSTEM IDENTIFICATION PHASE
LEVEL 1 FLOWCHART

START

INITIALIZATION

SEND A ZERO TO THE MOTOR

USER INPUT VALUES

PRINT BACK VALUES

VALUES CORRECT?

NO

YES

CALCULATE THE TIME VALUE

CALL INTNL

SEND OUT PULSE

SUBROUTINE TO WASTE TIME

SEND A ZERO TO SHUT OFF MOTOR

TUNABLE ESTIMATOR

ESTIMATION PROCEDURE

RECURSIVE LEAST SQUARES

ESTIMATE K & T

STOP

MRAS
APPENDIX A

SYSTEM IDENTIFICATION PHASE
FLOWCHARTS AND SOURCE CODE
START

INITIALIZATION

ENABLE INTERRUPTS

LOOP TO KILL TIME DONE?

NO

YES

DISABLE INTERRUPTS

RETURN
**SYSTEM IDENTIFICATION PHASE**

**LEVEL 2 FLOWCHART**

**SUBROUTINE TO ESTIMATE**

**SYSTEM PARAMETERS**

**USING A MRAS**

1. **START**
2. **INITIALIZATION**
3. **INPUT USER VALUES**
4. **PRINT BACK VALUES**
5. **VALUES CORRECT?**
   - NO
   - YES
     - **DECISION**
     - **CALCULATE AND PRINT OUT SYSTEM PARAMETERS**
     - **AGAIN**
6. **RETURN**
SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO ESTIMATE
SYSTEM PARAMETERS USING A
RECURSIVE LEAST SQUARES METHOD

START

INITIALIZATION

DECISION

RETURN

INPUT USER VALUES

PRINT BACK VALUES

VALUES CORRECT?

YES

SPECIFY INITIAL VALUES

CALCULATE F(K+1)

CALCULATE EQ(K+1)

CALCULATE PH & QH

PRINT OUT VALUES

NO
START

INITIALIZATION

READ THE VALUE FROM THE TACHOMETER

RETURN
THIS PROGRAM IS USED TO DETERMINE THE OPEN LOOP
TIME CONSTANT AND THE OPEN LOOP GAIN FOR A DC MOTOR SO
THAT DIGITAL PI AND PI2 CONTROL ALGORITHMS CAN BE IMPLEMENTED.
THIS PROGRAM ALSO USES A MODEL REFERENCE ADAPTIVE SYSTEM
(MRAS) SO THAT SYSTEM PARAMETERS CAN BE ESTIMATED IN ORDER
FOR AN ADAPTIVE CONTROL ALGORITHM TO BE IMPLEMENTED. FINALLY,
THE PROGRAM USES A RECURSIVE LEAST SQUARES METHOD TO OBTAIN
ESTIMATES OF THE SYSTEM PARAMETERS.

THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.

DIMENSION ADCVAL(300), AICHN(8), DACHN(4), IU(300)
INTEGER ADCVAL,DACVAL
LOGICAL#1 NAIC,NDAC,AICHN,NACHN
COMMON/ISRB/;I
COMMON/AICBLK/NADIC,AICHN
COMMON/DACBLK,NDAC,DACHN
COMMON/A1/DACVAL,IU;ISEC,IMAX
COMMON/A2/ITIM
AICHN(1)=2
DACHN(1)=0
NAIC=1
NDAC=1
I=0

CALL DAC(0)
WRITE(3,10)
10 FORMAT(/' THIS PROGRAM IS TO BE USED DURING THE SYSTEM /
C' IDENTIFICATION PHASE OF THE ME600 PROJECT "A COMPARISON /
C' OF AUTOMATIC CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A /
C' DC MOTOR USING A DIGITAL MICROCOMPUTER". '/
C' THE PROGRAM CONSISTS OF THREE SEPERATE SECTIONS WHICH ARE /
C' LISTED BELOW: '/
C'  1. ESTIMATION OF THE OPEN-LOOP GAIN (K) AND OPEN-LOOP /
C'   TIME CONSTANT (T) TO BE USED IN A DIGITAL PI AND PID /
C'   CONTROL ALGORITHM. '/
C'  2. ESTIMATION OF THE SYSTEM PARAMETERS USING A MODEL /
C'   REFERENCE ADAPTIVE SYSTEM (MRAS) SO THAT AN ADAPTIVE /
C'   CONTROL ALGORITHM CAN BE USED. '/
C'  3. ESTIMATION OF THE SYSTEM PARAMETERS USING A RECURSIVE /
C'   LEAST SQUARES METHOD. '/)
15 FORMAT(I1)
20 WRITE(3,20)
20 FORMAT( ' PRESS RETURN TO CONTINUE ')
READ(2,15) KL
WRITE(3,30)
30 FORMAT( '/ FIRST THE SYSTEM RESPONSE MUST BE OBTAINED. THIS WILL /
C' BE DONE BY PUTTING OUT A STEP OUTPUT ON THE DIGITAL TO ANALOG /
C' PORT. THE USER WILL NOW SPECIFY THE MAGNITUDE AND DURATION OF /
C' THE STEP FUNCTION. ')
35 FORMAT(15)

*****************************************************************************
* INPUT THE USER VALUES *
*****************************************************************************

40 WRITE(3,50)
50 FORMAT( '/ INPUT THE MAGNITUDE OF THE INPUT FOLLOWED BY A ' /
C' COMMA (-2048< X<2047) AN INTEGER VALUE ')
READ(2,35) IMAX
55 WRITE(3,60)
60 FORMAT( '/ INPUT THE PULSE WIDTH OF THE STEP VALUE IN SECONDS. ' /
C' FOLLOW THIS VALUE WITH A COMMA ( 1 < X< 25 ) ')
READ(2,35) ISEC
1 WRITE(3,65)
65 FORMAT( '/ INPUT THE TIME IN SECONDS TO REACH THE MAXIMUM ' /
C' VALUE FROM ZERO. FOLLOW THIS VALUE WITH A COMMA. ')
READ(2,35) IR

*****************************************************************************
* PRINT BACK THE VALUES *
*****************************************************************************

WRITE(3,70) IMAX
70 FORMAT( '/ THE MAGNITUDE OF THE STEP INPUT IS ',I5 )
WRITE(3,80) ISEC
80 FORMAT( '/ THE PULSE WIDTH IS ',I3, ', ' SECONDS ')
WRITE(3,85) IR
85 FORMAT( '/ THE TIME TO REACH MAX. MAGNITUDE IS ',I5, ', ' SECONDS ')

*****************************************************************************
* ARE THE VALUES CORRECT? *
*****************************************************************************
WRITE(3,90)
90 FORMAT( '/ ' ARE THE VALUES CORRECT? (1=YES, 2=NO) ' )
READ(2,15) J
IF (J .EQ. 2) GO TO 40

C
C ************************************************************
C * CALCULATE THE TIME VALUES *
C ************************************************************

C TIM=530.*ISEC/15.
C ITIM=TIM
C L=ISEC*10.
C M=IR*10.
C N=L-M
C IU(1)=0
C DO 95 K=2,L
C IF (M .EQ. 0) GO TO 92
C IA=FLOAT(IMAX/M)+0.5
C IF (K .LT. M) GO TO 91
C IF (K .GT. N) GO TO 93
C 92 IU(K)=IMAX
C GO TO 95
C 91 IU(K)=IU(K-1)+IA
C GO TO 95
C 93 IU(K)=IU(K-1)-IA
C 95 CONTINUE
C DO 97 K=1,L
C 96 WRITE(3,96) IU(K)
C 97 CONTINUE
C
C ************************************************************
C * COLLECT DATA FROM THE MOTOR *
C ************************************************************

C WRITE(3,100)
C 100 FORMAT( '/ ' PRESS RETURN TO COLLECT THE DATA FROM THE MOTOR ' )
READ(2,15) JL

C
C ************************************************************
C * CALL SUBROUTINE TO INITIALIZE *
C ************************************************************

CALL INTNIL

C
C ************************************************************
C * CALL SUBROUTINE TO WASTE TIME *
C ************************************************************

CALL WASTE

C
C ************************************************************
C * SEND A ZERO TO SHUT OFF THE MOTOR *
C ************************************************************

CALL DAC(0)

C
C ************************************************************
C * WHICH TYPE OF ESTIMATION RULE? *
C
C

WRITE(3,110)

110 FORMAT(/' THE DATA HAS BEEN COLLECTED AND IS READY TO BE '/
C' PROCESSED TO DETERMINE THE PARAMETER ESTIMATES. '/)

115 WRITE(3,120)

120 FORMAT(/' WHICH DO YOU WISH TO PERFORM?'/
C' 1=ESTIMATION OF OPEN-LOOP GAIN AND TIME CONSTANT '/
C' 2=ESTIMATION FOR USE IN AN ADAPTIVE CONTROLLER '/
C' 3=RECURSIVE LEAST SQUARES ESTIMATION METHOD '/
C' 4=OBTAIN ANOTHER SET OF DATA FROM THE MOTOR '/
C' 5=QUIT '/
C' ENTER 1 - 5 '/)

READ(2,15) IE
IF (IE .EQ. 1) CALL KTAU
IF (IE .EQ. 2) CALL MRAS
IF (IE .EQ. 3) CALL RCLS
IF (IE .EQ. 4) GO TO 40
IF (IE .EQ. 5) GO TO 150
GO TO 115

150 STOP
END

C

C ******************************
C  * SUBROUTINE TO WASTE TIME WHILE * 
C  * INTERUPTS OCCUR  * 
C  *
C  ********************************
C
C SUBROUTINE WASTE
COMM/ISRBLK/IJ
COMM/A2/ITIM
WRITE(3,600)

600 FORMAT(/' THE DATA COLLECTION PROCEDURE HAS BEGUN '/)
IJ=0

C

C  ******************************
C  * ENABLE THE INTERRUPTS  * 
C  ****************************
C
CALL ENABLE

C

C  ******************************
C  * WASTE TIME  * 
C  ****************************
C
DO 700 I=1,ITIM
DO 650 J=1,1425
650 CONTINUE
700 CONTINUE

C

C  ******************************
C  * DISABLE THE INTERRUPTS  * 
C  ****************************
C
CALL DISABL
RETURN
END
SUBROUTINE ISR
DIMENSION ADCVAL(300), ADCHN(8), DACHN(4), IU(300)
INTEGER ADCVAL, DACVAL
LOGICAL #1 NADC, NDAC, ADCHN, DACHN
COMMON/ISRBLK/IJ
COMMON/ADCBLK/NADC, ADCHN
COMMON/BLK/NDAC, DACHN
COMMON/A1/ADCVAL, IU, ISEC, IMAX

** GET THE VALUE FROM THE TACHOMETER **

IJ=IJ+1
CALL ADC(ADCVAL(IJ))
CALL DAC(IU(IJ))
RETURN
END

** SUBROUTINE TO ESTIMATE OPEN-LOOP GAIN AND OPEN-LOOP TIME CONSTANT **

** INITIALIZATION **
IOT=ISEC*10.

** DECISION **
149 WRITE(3,151)
151 FORMAT(3,151)
' YOU HAVE ENTERED THE SUBROUTINE TO DETERMINE THE '/
' OPEN-LOOP GAIN AND TIME CONSTANT. YOU MAY NOW '/
' 1=RETURN TO THE MAIN PROGRAM '/
' 2=PRINT OUT THE SYSTEM RESPONSE FOR THIS PROCEDURE '/
' ENTER 1 OR 2 ')
IF ( LJ .EQ. 2 ) GO TO 153
GO TO 149

C********************************************************************
C* PRINT OUT THE SYSTEM RESPONSE *
C********************************************************************

153 WRITE(1,154)
154 FORMAT('/**15X,' 'OUTPUT FOR SYSTEM IDENTIFICATION PHASE ')
WRITE(1,155) IMAX
155 FORMAT(20X,' SYSTEM RESPONSE TO INPUT=',I5)
WRITE(1,156)
156 FORMAT(8X,' USE FOR CALCULATING OPEN-LOOP GAIN AND TIME CONSTANT')
WRITE(1,160)
156 FORMAT(/14X,' TIME INCREMENT',6X,' TACHOMETER VALUE',6X,'INPUT')
DO 180 I=1,IOT
WRITE(1,170) I,ADCVAL(I),IU(I)
170 FORMAT(20X,I3,19X,I5,9X,I5)
180 CONTINUE
WRITE(1,185)
185 FORMAT('/** PRINT OUT COMPLETE ')
GO TO 149
190 RETURN
END

********************************************************************
C********************************************************************
C* SUBROUTINE TO OBTAIN PARAMETER ESTIMATES *
C* USING A MODEL REFERENCE ADAPTIVE SYSTEM (MRAS) *
C********************************************************************

********************************************************************
C* INITIALIZATION *
********************************************************************

C SUBROUTINE MRAS
DIMENSION ADCVAL(300), IU(300)
DIMENSION X(300), XHO(300), XHH(300), PH(300), GH(300)
DIMENSION EQ(300), DEN(300), E(300), ER(300)
INTEGER ADCVAL, IADCVAL
COMMON/A1/ADCVAL, IU, ISEC, IMAX

C********************************************************************
C* USER INPUT VALUES *
********************************************************************

WRITE(3,200)
200 FORMAT(/' YOU HAVE ENTERED THE SUBROUTINE TO DETERMINE '/
C' THE SYSTEM PARAMETERS USING A MODEL REFERENCE ADAPTIVE '/
C' SYSTEM (MRAS). THE USER MUST NOW INPUT VALUES NEEDED TO '/
C' COMPLETE THE ESTIMATION PROCEDURE. ')
210 FORMAT(F10.0)
215 WRITE(3,220)
220 FORMAT(/' INPUT THE GAIN K1. INCLUDE A PERIOD ')
READ(2,210) AK1
WRITE(3,230)
230 FORMAT(/' INPUT THE GAIN K2. INCLUDE A PERIOD ')
READ(2,210) AK2
WRITE(3,271)
271 FORMAT(// INPUT THE INITIAL VALUE OF THE COMPUTER MODEL /)
C PARAMETER "PH". INCLUDE A PERIOD )
READ(2,210) APH
WRITE(3,272)

272 FORMAT(// INPUT THE INITIAL VALUE OF THE COMPUTER MODEL /)
C PARAMETER "QH". INCLUDE A PERIOD )
READ(2,210) AQH
WRITE(3,273)

273 FORMAT(// INPUT THE INITIAL VALUE OF "XH". INCLUDE A PERIOD )
READ(2,210) XA

************
PRINT BACK THE VALUES
************

WRITE(3,275) AK1
WRITE(3,280) AK2
WRITE(3,301) APH
WRITE(3,302) AQH
WRITE(3,303) XA

************
ARE THESE THE CORRECT VALUES? *
************

WRITE(3,310)
310 FORMAT(// ARE THESE THE CORRECT VALUES? (1=YES, 2=NO) )
READ(2,320) LP
IF (LP .EQ. 2) GO TO 215

************
DECISION
************

WRITE(3,322)
322 FORMAT(// YOU ARE NOW READY TO OBTAIN PARAMETER ESTIMATES /
C DO YOU WISH TO: /
C 1=RETURN TO THE MAIN PROGRAM /
C 2=REPEAT THIS ESTIMATE PROCEDURE /
C 3=PRINT OUT THE ESTIMATES FOR THIS PROCEDURE /
WRITE(3,324)

324 FORMAT(// ENTER 1,2, OR 3 )
READ(2,330)LJ
IF (LJ .EQ. 1) GO TO 360
IF (LJ .EQ. 2) GO TO 215
IF (LJ .EQ. 3) GO TO 325
GO TO 323

************
SPECIFY INITIAL VALUE OF THE RESPONSE *
************

************
AND COMPUTER MODEL PARAMETERS *
************

325 XH(1)=XA
PH(1) = APH
QH(1) = AQB

*************************************************************************
* CALCULATE COMPUTER MODEL PARAMETERS *
*************************************************************************

IOT = ISEC*10.
DO 330 K = 1, IOT
X(K) = ADCVAL(K)
X(K + 1) = ADCVAL(K + 1)
XH(K + 1) = PH(K) - XH(K) + QH(K) * IU(K)
EO(K + 1) = X(K + 1) - XH(K + 1)
DEN(K) = (1 + AK1 * XH(K))**2 + (AK2 * (IU(K))**2)
EC(K + 1) = EO(K + 1) / DEN(K)
PH(K + 1) = PH(K) + AK1 * EC(K + 1) * XH(K)
QH(K + 1) = QH(K) + AK2 * EC(K + 1) * IU(K)
XH(K + 1) = PH(K + 1) * XH(K) + QH(K + 1) * IU(K)
ER(K) = X(K) - XH(K)
330 CONTINUE

*************************************************************************
* PRINT OUT THE SYSTEM RESULTS *
*************************************************************************
WRITE(1, 335)
335 FORMAT(//, 15X, 'ESTIMATES OF PH AND QH USING A HRAS')
WRITE(1, 336) IMAX
336 FORMAT(15X, 'SYSTEM RESULTS FOR INPUT=', I5)
WRITE(1, 337) AK1, AK2
337 FORMAT(23X, 'K1=', F7.2, ' K2=', F7.2)
WRITE(1, 339)
DO 340 K = 1, IOT
WRITE(1, 340) K, PH(K), QH(K), X(K), XH(K), ER(K)
340 CONTINUE
WRITE(1, 345)
345 FORMAT(//, 'PRINT OUT COMPLETE')
GO TO 321
360 RETURN
END

*************************************************************************
* SUBROUTINE FOR PARAMETER ESTIMATION UTILIZING A RECURSIVE LEAST SQUARES METHOD *
*************************************************************************

SUBROUTINE RCLS
DIMENSION ADCVAL(300), THET1(300), IU(300), YH(300)
DIMENSION THET2(300), F12(300), F11(300)
DIMENSION F21(300), F22(300), C11(300), C12(300), C21(300)
DIMENSION C22(300), DEN(300), EO(300), PH(300)
INTEGER AICVAL
COMMON/A1/ADCVAL, IU, ISEC, IMAX
IOT = ISEC*10.
395 WRITE(3,400) 400 FORMAT(/ ' YOU ARE IN THE SUBROUTINE TO DETERMINE THE SYSTEM '/' C' PARAMETERS USING A RECURSIVE LEAST SQUARES TECHNIQUE. THE '/' C' USER NOW HAS THE OPTION OF: '/' C' 1=RETURNING TO THE MAIN PROGRAM '/' C' 2=ESTIMATING THE PARAMETERS ') 405 WRITE(3,410) 410 FORMAT(/ ' ENTER 1 OR 2 ') 411 FORMAT(I1) READ(2,411) IL IF (IL .EQ. 1) GO TO 560 IF (IL .EQ. 2) GO TO 412 GO TO 405 412 WRITE(3,413) 413 FORMAT(/ ' INPUT AN INITIAL ESTIMATE OF "PH", INCLUDE A PERIOD ') 414 FORMAT(F10.0) READ(2,414) P WRITE(3,415) 415 FORMAT(/ ' INPUT AN INITIAL ESTIMATE OF "QH", INCLUDE A PERIOD ') READ(2,414) Q WRITE(3,409) 409 FORMAT(/ ' INPUT THE INITIAL VALUE OF THE "F" MATRIX, INCLUDE '/' C' A PERIOD. ALL INITIAL VALUES ARE ASSUMED TO BE THE SAME. ') READ(2,414) F 416 FORMAT(/ ' THE INITIAL ESTIMATE OF "PH" IS',F4.2) WRITE(3,417) P 417 FORMAT(/ ' THE INITIAL ESTIMATE OF "QH" IS',F4.2) WRITE(3,408) F 408 FORMAT(/ ' THE INITIAL VALUE OF "F" IS',F8.3) 418 FORMAT(' ARE THE VALUES CORRECT? (1=YES,2=NO) ') READ(2,411) IH IF (IH .EQ. 2) GO TO 412 419 FORMAT(/ ' SPECIFY INITIAL VALUES OF THE ') 420 FORMAT(/ ' COMPUTER MODEL PARAMETERS ') 421 WRITE(3,418) 418 PH(1)=P
THETI(1) = -P
THET2(1) = Q
F11(1) = F
F12(1) = F
F21(1) = F
F22(1) = F
YH(1) = 0.

**********************************************************************
* ESTIMATION PROCEDURE * 
**********************************************************************

WRITE(3,420)
420 FORMAT(/' STARTING RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE'
) DO 425 K = 1,10T
YH(K+1) = (PH(K)*YH(K)) + (THET2(K)*I(U(K))

* CALCULATE F(K+1) *
C11(K) = (F11(K)*(-YH(K)) + F12(K)*I(U(K)))*(-YH(K))
C12(K) = (F11(K)*(-YH(K)) + F12(K)*I(U(K)))
C21(K) = (F21(K)*(-YH(K)) + F22(K)*I(U(K)))
C22(K) = (F21(K)*(-YH(K)) + F22(K)*I(U(K)))
DEN(K) = C11(K) + C22(K) + 1.
F11(K+1) = F11(K) - (C11(K)*F11(K) + C12(K)*F21(K))/DEN(K)
F12(K+1) = F12(K) - (C11(K)*F12(K) + C12(K)*F21(K))/DEN(K)
F21(K+1) = F21(K) - (C21(K)*F11(K) + C22(K)*F21(K))/DEN(K)
F22(K+1) = F22(K) - (C21(K)*F12(K) + C22(K)*F22(K))/DEN(K)

* CALCULATE EO(K+1) *
EO(K+1) = AICVAL(K+1) + THETI(K)*YH(K) - THET2(K)*I(U(K))

* CALCULATE THETA HAT AT K+1 *
THETI(K+1) = THETI(K) + (F11(K+1)*(-YH(K)) + F12(K+1)*I(U(K)))*EO(K+1)
THET2(K+1) = THET2(K) + (F21(K+1)*(-YH(K)) + F22(K+1)*I(U(K)))*EO(K+1)

* CALCULATE PH AND GH *
PH(K+1) = -THETI(K+1)
GH(K+1) = THET2(K+1)

**********************************************************************
* WRITE OUT THE VALUES * 
**********************************************************************

425 CONTINUE
WRITE(3,430)
430 FORMAT(/' ESTIMATION PROCEDURE COMPLETE. PRINTING OUT VALUES'
) WRITE(1,440)
440 FORMAT(//'15X,' RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE '
) WRITE(1,450) IMAX
450 FORMAT(20X,' SYSTEM RESULTS FOR INPUT=',I5)
WRITE(1,460) P,F
460 FORMAT(11X,' INITIAL PH=',F4.2,' INITIAL GH=',F4.2,' F=',F8.3)
WRITE(1,470)
470 FORMAT(//'20X,' TIME',7X,' PH',8X,' GH')
10 DO 420 K = 1,10T
WRITE(1,480) K, PH(K), THET2(K)
480 FORMAT(20X, I3, 2X, F8.3, 2X, F8.3)
490 CONTINUE
WRITE(1,495)
495 FORMAT(/,
' PRINT OUT COMPLETE
GO TO 395
560 RETURN
END
EOF
APPENDIX B

TUNABLE ESTIMATOR
FLOWCHARTS AND SOURCE CODE
START

INITIALIZATION

DECISION

INPUT USER VALUES

PRINT BACK VALUES

VALUES CORRECT?

NO

YES

SPECIFY INITIAL VALUES

CALCULATE E0(K+1)

CALCULATE E(K+1)

CALCULATE F(K+1)

CALCULATE PH & QH

PRINT OUT VALUES

RETURN
**ME600 PROJECT**

**COMPARISON OF AUTOMATIC CONTROL SCHEMES**

**SYSTEM IDENTIFICATION PHASE**

**RESEARCH ASSISTANT: DAN LYMBURNER**

**RESEARCH ADVISOR: PROF. G. ULSOY**

THIS PROGRAM IS USED DURING THE SYSTEM IDENTIFICATION PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES". A TUNABLE ESTIMATOR IS USED TO OBTAIN THE DISCRETE TIME SYSTEM PARAMETERS.

THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.

**MAIN PROGRAM**

**INITIALIZATION**

**DIMENSION ADCVAL(300), AICHN(8), DACHN(4), IU(300)**

**INTEGER ADCVAL, DACVAL**

**LOGICAL NADC, NDAC, AICHN, DACHN**

**COMMON/ISRBLK/IJ**

**COMMON/AICBLK/NADC, AICHN**

**COMMON/DACBLK/NDAC, DACHN**

**COMMON/AL/ADCVAL, IU, ISEC, IMAX**

**COMMON/A2/ITIM**

**AICHN(1)=2**

**DACHN(1)=0**

**NADC=1**

**NDAC=1**

**I=0**

**CALL DAC(0)**

**PROGRAM EXPLANATION**

**WRITE(3,10)**

**FORMAT(' THIS PROGRAM IS TO BE USED DURING THE SYSTEM IDENTIFICATION PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES." A TUNABLE ESTIMATOR IS USED TO OBTAIN THE DISCRETE TIME SYSTEM PARAMETERS. THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.')**
C' IDENTIFICATION PHASE OF THE ME600 PROJECT "A COMPARISON \\
C' OF AUTOMATIC CONTROL SCHEMES FOR CONTROL OF THE SPEED \\
C' OF IC MOTOR USING A DIGITAL MICROCOMPUTER". \\
C' THE PROGRAM CONSISTS OF TWO SEPERATE SECTIONS WHICH ARE \\
C' LISTED BELOW: \\
C' 1. ESTIMATION OF THE OPEN-LOOP GAIN (K) AND OPEN-LOOP \\
C' TIME CONSTANT (T) TO BE USED IN A DIGITAL PID AND PID \\
C' CONTROL ALGORITHM. \\
C' 2. ESTIMATION OF THE DISCRETE TIME SYSTEM PARAMETERS THROUGH \\
C' THE USE OF A TUNABLE ESTIMATION PROCEDURE. \\
15 FORMAT(I1) 
WRITE(3,20) 
20 FORMAT( ' PRESS RETURN TO CONTINUE ' ) 
READ(2,15) KL 
WRITE(3,30) 
30 FORMAT( ' FIRST THE SYSTEM RESPONSE MUST BE OBTAINED. THIS WILL \\
C' BE DONE BY PUTTING OUT A STEP OUTPUT ON THE DIGITAL TO ANALOG \\
C' PORT. THE USER WILL NOW SPECIFY THE MAGNITUDE AND DURATION ( \\
C' THE STEP FUNCTION. ' ) 
35 FORMAT(I5) 
C 
C***************************************************************************** 
* INPUT THE USER VALUES * 
***************************************************************************** 
C***************************************************************************** 
40 WRITE(3,50) 
50 FORMAT( ' INPUT THE MAGNITUDE OF THE INPUT FOLLOWED BY A ' \\
C' COMMA (-2048<X<2047) AN INTEGER VALUE ' ) 
READ(2,35) IMAX 
55 WRITE(3,60) 
60 FORMAT( ' INPUT THE PULSE WIDTH OF THE STEP VALUE IN SECONDS. \\
C' FOLLOW THIS VALUE WITH A COMMA ( 1<X<25 ) ' ) 
READ(2,35) ISEC 
65 WRITE(3,65) 
70 FORMAT( ' INPUT THE TIME IN SECONDS TO REACH THE MAXIMUM ' \\
C' VALUE FROM ZERO. FOLLOW THIS VALUE WITH A COMMA. ' ) 
READ(2,35) IR 
75 WRITE(3,70) IMAX 
80 FORMAT( ' THE MAGNITUDE OF THE STEP INPUT IS ',I5 ) 
WRITE(3,80) ISEC 
85 FORMAT( ' THE PULSE WIDTH IS ',I3,' SECONDS ' ) 
WRITE(3,85) IR 
90 FORMAT( ' THE TIME TO REACH MAX. MAGNITUDE IS ',I5,' SECONDS ' ) 
C***************************************************************************** 
* ARE THE VALUES CORRECT? * 
***************************************************************************** 
C***************************************************************************** 
90 WRITE(3,90) 
95 FORMAT( ' ARE THE VALUES CORRECT? (1=YES, 2=NO) ' ) 
READ(2,15) J 
100 IF ( J .EQ. 2 ) GO TO 40 
C***************************************************************************** 
* CALCULATE THE TIME VALUES * 
*****************************************************************************
TIM=530.*ISEC/15.
ITIM=TIM
L=ISEC*10.0
M=IR*10.
N=L-M
IU(1)=0
DO 95 K=2,L
IF (M .EQ. 0) GO TO 92
IA=FLOAT(IMAX/M)+0.5
IF (K .LT. M) GO TO 91
IF (K .GT. N) GO TO 93
92 IU(K)=IMAX
GO TO 95
91 IU(K)=IU(K-1)+IA
GO TO 95
93 IU(K)=IU(K-1)-IA
95 CONTINUE
DO 97 K=1,L
WRITE(3,96) IU(K)
96 FORMAT('///15X,I5')
97 CONTINUE

***********************************************************************************************
* COLLECT DATA FROM THE MOTOR *
***********************************************************************************************

WRITE(3,100)
100 FORMAT('/// PRESS RETURN TO COLLECT THE DATA FROM THE MOTOR///)
READ(2,15) JL

***********************************************************************************************
* CALL SUBROUTINE TO INITIALIZE * 
* THE INTERRUPTS * 
***********************************************************************************************

CALL INTINIT

***********************************************************************************************
* CALL SUBROUTINE TO WASTE TIME * 
***********************************************************************************************

CALL WASTE

***********************************************************************************************
* SEND A ZERO TO SHUT OFF THE MOTOR * 
***********************************************************************************************

CALL DAC(0)

***********************************************************************************************
* WHICH TYPE OF ESTIMATION RULE? * 
***********************************************************************************************

WRITE(3,110)
110 FORMAT('/// THE DATA HAS BEEN COLLECTED AND IS READY TO BE ///
C' PROCESSED TO DETERMINE THE PARAMETER ESTIMATES.///)
115 WRITE(3,120)
120 FORMAT('/// WHICH DO YOU WISH TO PERFORM///')
1 = ESTIMATION OF OPEN-LOOP GAIN AND TIME CONSTANT
2 = ESTIMATION USING THE TUNABLE ESTIMATION PROCEDURE
3 = OBTAIN ANOTHER SET OF DATA FROM THE MOTOR
4 = QUIT

READ (2, 15) IE
IF (IE .EQ. 1) CALL KTAU
IF (IE .EQ. 2) CALL RCLS
IF (IE .EQ. 3) GO TO 40
IF (IE .EQ. 4) GO TO 150
GO TO 115

150 STOP
END

*****************************************************************
* SUBROUTINE TO WASTE TIME WHILE INTERRUPTS OCCUR *
*****************************************************************

SUBROUTINE WASTE
COMMON/ISRLK/IJ
COMMON/A2/ITIM
WRITE (3,600)
600 FORMAT (/ ' THE DATA COLLECTION PROCEDURE HAS BEGUN ')

IJ = 0

*****************************************************************
* ENABLE THE INTERRUPTS *
*****************************************************************

CALL ENABLE

*****************************************************************
* WASTE TIME *
*****************************************************************

DO 700 I = 1, ITIM
DO 650 J = 1, 1425
650 CONTINUE
600 CONTINUE

*****************************************************************
* DISABLE THE INTERRUPTS *
*****************************************************************

CALL DISABL
RETURN
END

*****************************************************************
* INTERRUPT SERVICE *
* ROUTINE FOR DATA COLLECTION *
*****************************************************************
SUBROUTINE ISR
DIMENSION ADCVAL(300), ADCHN(8), DACHN(4), IU(300)
INTEGER ADCVAL, DACVAL
LOGICAL NAIC, NDAC, ADC, DACHN
COMMON/ISRBLK/NIJ
COMMON/ADCBLK/NAIC, DACHN
COMMON/DACBLK/NDAC, DACHN
COMMON/A1/ADCVAL, IU, ISEC, IMAX

******************************************************************************
* GET THE VALUE FROM THE TACHOMETER *
******************************************************************************

IJ=IJ+1
CALL DAC(IU(IJ))
CALL AIC(ADCVAL(IJ))
RETURN
END

******************************************************************************
* SUBROUTINE TO ESTIMATE OPEN-LOOP GAIN AND *
* OPEN-LOOP TIME CONSTANT *
******************************************************************************

******************************************************************************
* INITIALIZATION *
******************************************************************************

SUBROUTINE KTAU
DIMENSION ADCVAL(300), IU(300)
INTEGER ADCVAL
COMMON/A1/ADCVAL, IU, ISEC, IMAX
IOT=ISEC*10.

******************************************************************************
* DECISION *
******************************************************************************

WRITE(3,151)
FORMAT(/'YOU HAVE ENTERED THE SUBROUTINE TO DETERMINE THE /
C' OPEN-LOOP GAIN AND TIME CONSTANT. YOU MAY NOW: '/
C' 1=RETURN TO THE MAIN PROGRAM '/
C' 2=PRINT OUT THE SYSTEM RESPONSE FOR THIS PROCEDURE '/
C' ENTER 1 OR 2 '/)

152 FORMAT(I1)
READ(2,152) LJ
IF (LJ .EQ. 1) GO TO 190
IF (LJ .EQ. 2) GO TO 153
GO TO 149

******************************************************************************
* PRINT OUT THE SYSTEM RESPONSE *
******************************************************************************

153 WRITE(1,154)
154 FORMAT(/,15X,'OUTPUT FOR SYSTEM IDENTIFICATION PHASE ')
WRITE(1,155) IMAX

155 FORMAT(20X,' SYSTEM RESPONSE TO INPUT=' ,I5 ')

WRITE(1,156)

156 FORMAT(8X,' USE FOR CALCULATING OPEN-LOOP GAIN AND TIME CONSTANTS')

WRITE(1,160)

160 FORMAT(/,14X,' TIME INCREMENT',6X,' TACHOMETER VALUE',6X,'INPUT')

DO 180 I=1,IOT

WRITE(1,170) I,ADICVAL(I),IU(I)

170 FORMAT(20X,I3,19X,I5,9X,I5)

180 CONTINUE

WRITE(1,185)

185 FORMAT(/,' PRINT OUT COMPLETE ')

GO TO 149

190 RETURN

END

C

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

C

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

C

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

C

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

C

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

SUBROUTINE RCLS

DIMENSION ADICVAL(300), THET1(300), IU(300), YH(300)

DIMENSION THET2(300), F12(300), F11(300)

DIMENSION F21(300), F22(300), C11(300), C12(300), C21(300)

DIMENSION C22(300), IEN(300), ED(300), PH(300)

DIMENSION YHD(300), Y(300), E(300), IF(300)

INTEGER ADICVAL

COMMON/A1/ADICVAL,IU,ISEC,IMAX

IOT=ISEC*10.

C

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

C

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

C

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

395 WRITE(3,400)

400 FORMAT(/' YOU ARE IN THE SUBROUTINE TO DETERMINE THE SYSTEM '/

C' PARAMETERS USING A TUNABLE ESTIMATION ALGORITHM. THE '/

C' USER NOW HAS THE OPTION OF: '/

C' 1=RETURNING TO THE MAIN PROGRAM '/

C' 2=ESTIMATING THE PARAMETERS '/

405 WRITE(3,410)

410 FORMAT(/' ENTER 1 OR 2 ')

411 FORMAT(I1)

READ(2,411) IL

IF (IL .EQ. 1) GO TO 560

IF (IL .EQ. 2) GO TO 412

GO TO 405

C

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

C

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

C

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

C

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

412 WRITE(3,413)

413 FORMAT(/' INPUT AN INITIAL ESTIMATE OF "PH". INCLUDE A PERIOD'

414 FORMAT(F10.0)

415 READ(2,414) P

416 WRITE(3,415)
415 FORMAT(/' INPUT AN INITIAL ESTIMATE OF "QH". INCLUDE A PERIOD ')
READ(2,414) Q
WRITE(3,409)

409 FORMAT(/' INPUT THE INITIAL VALUE OF THE "F" MATRIX. INCLUDE C' A PERIOD. ALL INITIAL VALUES ARE ASSUMED TO BE THE SAME. ')
READ(2,414) F
WRITE(3,350)

350 FORMAT(/' INPUT THE VALUE LAMBDA 1. INCLUDE A PERIOD ')
READ(2,414) AL1
352 WRITE(3,354)
354 FORMAT(/' INPUT THE VALUE LAMBDA 2. INCLUDE A PERIOD ')
READ(2,414) AL2

C

***********************************************************************
C
* PRINT BACK THE VALUES *
C
***********************************************************************

WRITE(3,416) P

416 FORMAT(/' THE INITIAL ESTIMATE OF "PH" IS',F4.2)
WRITE(3,417) Q

417 FORMAT(/' THE INITIAL ESTIMATE OF "QH" IS',F4.2)
WRITE(3,408) F

408 FORMAT(/' THE INITIAL VALUE OF "F" IS',F8.3)
WRITE(3,300) AL1

300 FORMAT(/' THE VALUE LAMBDA 1 IS',F6.4)
WRITE(3,310) AL2

310 FORMAT(/' THE VALUE LAMBDA 2 IS',F6.4)

C

***********************************************************************
C
* VALUES CORRECT? *
C
***********************************************************************

WRITE(3,418)

418 FORMAT(/' ARE THE VALUES CORRECT? (1=YES, 2=NO)' )
READ(2,411) IH
IF (IH .EQ. 2) GO TO 412

C

***********************************************************************
C
* SPECIFY INITIAL VALUES OF THE *
C
* COMPUTER MODEL PARAMETERS *
C
***********************************************************************

PH(1)=P
THET1(1)=-P
THET2(1)=Q
F11(1)=F
F12(1)=0.
F21(1)=0.
F22(1)=F
YH(1)=0.

C

***********************************************************************
C
* ESTIMATION PROCEDURE *
C
***********************************************************************

WRITE(3,420)

420 FORMAT(/' STARTING RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE' )
DO 425 K=1,10T
YHD(K+1)=(F(H(K))*YH(K))+(THET2(K)*IH(K))
YH(K)=ANGUAL(K)

425
Y(K+1) = ADCVAL(K+1)
ED(K+1) = Y(K+1) - YHO(K+1)

* CALCULATE E(K+1) *
C11(K) = (F11(K)*(-YH(K)) + F12(K)*IUK(K))*(YH(K))
C12(K) = (F11(K)*(-YH(K)) + F12(K)*IUK(K))*IUK(K)
C21(K) = (F21(K)*(-YH(K)) + F22(K)*IUK(K))*(-YH(K))
C22(K) = (F21(K)*(-YH(K)) + F22(K)*IUK(K))*IUK(K)
DEN(K) = C11(K) + C22(K+1)
E(K+1) = ED(K+1)/DEN(K)

* CALCULATE F(K+1) *
DF(K) = C11(K) + C22(K) + (AL1/AL2)
F11(K+1) = (F11(K) - (C11(K)*F11(K) + C12(K)*F21(K))/DF(K))/AL1
F12(K+1) = (F12(K) - (C11(K)*F12(K) + C22(K)*F22(K))/DF(K))/AL1
F21(K+1) = (F21(K) - (C21(K)*F11(K) + C22(K)*F21(K))/DF(K))/AL1
F22(K+1) = (F22(K) - (C21(K)*F12(K) + C22(K)*F22(K))/DF(K))/AL1

* CALCULATE THETA HAT AT K+1 *
THET1(K+1) = THET1(K) + (F11(K)*(-YH(K)) + F12(K)*IUK(K))*E(K+1)
THET2(K+1) = THET2(K) + (F21(K)*(-YH(K)) + F22(K)*IUK(K))*E(K+1)

* CALCULATE PH AND QH *
PH(K+1) = -THET1(K+1)
QH(K+1) = THET2(K+1)
YH(K+1) = PH(K+1)*YH(K) + THET2(K+1)*IUK(K)

* *************** *
* WRITE OUT THE VALUES *
* ***************

CONTINUE
WRITE(3,430)
WRITE(1,440)
WRITE(/'ESTIMATION PROCEDURE COMPLETE. PRINTING OUT VALUES/')
WRITE(/'TUNABLE ESTIMATOR ESTIMATION PROCEDURE/')
WRITE(/'SYSTEM RESULTS FOR INPUT='//,I5)
WRITE(1,460) P,Q,F
WRITE(/'INITIAL PH='//,F4.2,'INITIAL QH='//,F4.2,'F='//,F8.3)
WRITE(1,465) AL1,AL2
WRITE(1,466) AL1,AL2
WRITE(1,467)
WRITE(/'TIME','/,'PH','/,'QH')
DO 490 K = 1, 100
WRITE(1,480) K,PH(K),THET2(K),F11(K),F12(K),F21(K),F22(K)
490 CONTINUE
WRITE(1,495)
WRITE(/'PRINT OUT COMPLETE/')
GO TO 395
END
APPENDIX C

DIGITAL PI CONTROLLER FLOWCHARTS AND SOURCE CODE
START

INITIALIZATION

INPUT USER VALUES

PRINT BACK VALUES

VALUES CORRECT?

NO

YES

CALCULATE CONTROL VALUES

CALL INTNL

CALL WASTE

PRINT OUT VALUES

DECISION

RETURN
CONTROLLER DESIGN PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO
WASTE TIME
DIGITAL PI CONTROLLER

START

INITIALIZATION

ENABLE INTERRUPTS

LOOP TO KILL TIME
DONE?

NO

YES

DISABLE INTERRUPTS

RETURN
CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

INTERRUPT SERVICE ROUTINE
DIGITAL PI CONTROLLER

START

INITIALIZATION

CALL ADC
(GET TACH VALUE)

CALCULATE THE ERROR

CALCULATE THE CONTROL

CALL DAC
(OUTPUT THE CONTROL)

RETURN
**ME600 PROJECT**
**COMPARISON OF AUTOMATIC CONTROL SCHEMES**
**CONTROLLER DESIGN PHASE**
**DIGITAL PI CONTROLLER**
**RESEARCH ASSISTANT: DAN LYMBOURNER**
**RESEARCH ADVISOR: PROF. G. ULSOY**

THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL MICROCOMPUTER". THE ALGORITHM USED IN THIS PARTICULAR PROGRAM IS A DIGITAL PROPORTIONAL-INTEGRAL (PI) CONTROLLER.

THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.

**MAIN PROGRAM**

**INITIALIZATION**

DIMENSION ADCVAL(300), AICHN(8), DACHN(4), EK(300), MK(300)
INTEGER ADCVAL, I, CO, C1, MK, EK
LOGICAL I, NAIC, NDAC, AICHN, DACHN
COMMON/ISRBLK/I, D, CO, C1
COMMON/A1/ADCVAL, I, ISEK
COMMON/ADCBLK/NAIC, AICHN
COMMON/DACBLK/NDAC, DACHN
COMMON/A2/ITIM
COMMON/OUTBLK/MK, EK
AICHN(1)=2
DACHN(1)=0
NAIC=1
NDAC=1

**SEND A ZERO TO THE MOTOR**

CALL DAC(0)

**PROGRAM EXPLANATION**
C

******

0063 WRITE(3,10)
0064 10 FORMAT(// THIS PROGRAM IS TO BE USED DURING THE CONTROLLER /)
0065 C' DESIGN PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC
0066 C' CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A DC MOTOR USING
0067 C' A DIGITAL MICROCOMPUTER". THIS PROGRAM UTILIZES A PROPORTIONAL
0068 C' INTEGRAL (PI) CONTROLLER TO CONTROL THE MOTOR SPEED. ')
0069 12 WRITE(3,13)
0070 13 FORMAT(// THE USER NOW HAS THE OPTION OF: ')
0071 C' 1=QUITTING
0072 C' 2=IMPLEMENTING THE CONTROL ALGORITHM
0073 C' /
0074 C' ENTER 1 OR 2 ')
0075 15 FORMAT(I1)
0076 READ(2,15) KL
0077 IF (KL .EQ. 1) GO TO 150
0078 WRITE(3,30)
0079 30 FORMAT(// YOU HAVE Elected TO IMPLEMENT THE PI CONTROL ')
0080 C' ALGORITHM. THE USER WILL NOW BE ASKED TO INPUT THE ')
0081 C' NECESSARY CONTROL VALUES. ')
0082 35 FORMAT(15)
0083 C

0084 C

0085 C

0086 C

0087 C

0088 40 WRITE(3,50)
0089 50 FORMAT(// INPUT THE MAGNITUDE OF THE SPEED VALUE FOLLOWED BY A
0090 C' COMMA (-2048<X<2047) An INTEGER VALUE ')
0091 READ(2,35) IU
0092 55 WRITE(3,60)
0093 60 FORMAT(// INPUT THE TIME TO CONTROL THE MOTOR SPEED IN SECONDS.
0094 C' FOLLOW THIS VALUE WITH A COMMA ( 1<X<25 ) ')
0095 READ(2,35) ISEC
0096 IF (ISEC .GT. 25) GO TO 55
0097 WRITE(3,62)
0098 62 FORMAT(// INPUT THE REAL PART OF Z1. INCLUDE A PERIOD ')
0099 64 FORMAT(F10.0)
0100 READ(2,64) Z1R
0101 WRITE(3,66)
0102 66 FORMAT(// INPUT THE IMAGINARY PART OF Z1 & Z2. INCLUDE A PERIOD
0103 C' WITH A COMMA ( |Z|<100 ) ')
0104 READ(2,64) Z1I
0105 WRITE(3,67)
0106 67 FORMAT(// INPUT THE REAL PART OF Z2. INCLUDE A PERIOD ')
0107 READ(2,64) Z2R
0108 WRITE(3,68)
0109 68 FORMAT(// INPUT THE INTERRUPT PERIOD IN SEC. INCLUDE A PERIOD.
0110 C' WITH A COMMA ( |Z|<255 ) ')
0111 READ(2,64) Z
0112 WRITE(3,72)
0113 72 FORMAT(// INPUT THE MOTOR PARAMETER "P". INCLUDE A PERIOD. ')
0114 READ(2,64) P
0115 WRITE(3,74)
0116 74 FORMAT(// INPUT THE MOTOR PARAMETER "Q". INCLUDE A PERIOD. ')
0117 RED(2,64) Q
0118 C

0119 C

0120 C
WRITE(3,70) IU
70 FORMAT(/' THE MAGNITUDE OF THE SPEED VALUE IS ',I5 )
WRITE(3,80) ISEC
80 FORMAT(/' THE CONTROL TIME IS ',I3,' SECONDS ')
WRITE(3,82) Z1R
82 FORMAT(/' THE REAL PART OF Z1 IS ',F7.3 )
WRITE(3,84) Z1I
84 FORMAT(/' THE IMAGINARY PART OF Z1 & Z2 IS ',F7.3 )
WRITE(3,85) Z2R
85 FORMAT(/' THE REAL PART OF Z2 IS ',F7.3 )
WRITE(3,86) Z2I
86 FORMAT(/' THE INTERRUPT PERIOD IS ',F7.4,' SECONDS ')
WRITE(3,87) P
87 FORMAT(/' THE MOTOR PARAMETER "P" IS ',F7.3 )
WRITE(3,88) Q
88 FORMAT(/' THE MOTOR PARAMETER "Q" IS ',F7.3 )

*****************************************************************************
* ARE THE VALUES CORRECT? *
*****************************************************************************

WRITE(3,90)
90 FORMAT(/' ARE THE VALUES CORRECT? (1=YES, 2=NO) ')
READ(2,15) J
IF (J.EQ. 2) GO TO 40

*****************************************************************************
* CALCULATE THE TIME AND THE *
* CONTROL PARAMETERS *
*****************************************************************************

ITIM=530.*ISEC/15.
ISEC=ISEC#10.
B=-Z1R+Z2R
IF (ZI.EQ.0.) GO TO 92
C=(2.*ZI)*R2.+B**2.)/4.
GO TO 94

X=ABS(Z1R-Z2R)
C=(B**2.-X**2.)/4.
WRITE(3,96) B,C
96 FORMAT(/,5X,F7.4,5X,F7.4)
AKP=(P-C)/Q
AKI=(B+1.+P-Q*AKP)/(Z#Q)
WRITE(3,98) AKP,AKI
98 FORMAT(/,5X,F10.3,5X,F10.3)
I=1
I=1.
CO=AKP+(AKI#Z)
C1=AKP

*****************************************************************************
* CALL SUBROUTINE TO INITIALIZE *
* THE INTERRUPTS *
*****************************************************************************

CALL INITN

*****************************************************************************
* START THE CONTROL *
*****************************************************************************
WRITE(3,100)
100 FORMAT(/' PRESS RETURN TO BEGIN MOTOR CONTROL ')
READ(2,15) KA

CALL WASTE

CALL DISABLE
CALL RAC(0)

125 WRITE(3,130)
130 FORMAT(/' THE MOTOR CONTROL IS COMPLETE. YOU NOW HAVE /
C' THE OPTION OF: '
C' 1=PRINTING OUT THE CONTROL DATA '
C' 2=REPEATING THE PI CONTROL PROCEDURE '
C' 3=QUITTING ')

135 WRITE(3,140)
140 FORMAT(/' ENTER 1,2, OR 3 ')
READ(2,15) JD
IF (JD .EQ. 1) GO TO 142
IF (JD .EQ. 2) GO TO 40
IF (JD .EQ. 3) GO TO 150
GO TO 135

142 WRITE(1,143)
143 FORMAT(/'15X,'SYSTEM RESPONSE FOR CONTROL USING A PI ALGORITHM /
WRITE(1,144) IU, AKP, AKI
144 FORMAT(20X, 'SPEED=', F8.2, ' KI=', F8.2,)
141 FORMAT(20X, 'REAL Z=', F5.3, 3X, F5.3, 5X, 'IMAG. Z=', F5.3//)
143. WRITE(1,145)
145 FORMAT(5X, 'TIME', 5X, 'CONTROL EFFORT', 5X, 'ACTUAL SPEED', 5X, 'ERROR', 5X, 20X, L=1, IDT
146 WRITE(1,146) L, JK(L), ALICVAL(L), EK(L)
146 FORMAT(6X, I3, 9X, I5, 11X, I5, 7X, I8)
147 CONTINUE
GO TO 125
150 STOP
END

SUBROUTINE TO WASTE TIME WHILE
INTERRUPTS OCCUR

END
SUBROUTINE WASTE
COMMON/A2/ITIM
WRITE(3,600)
600 FORMAT(/' THE DATA COLLECTION PROCEDEURE HAS BEGUN ')

CALL ENABLE

DO 700 JX=1,ITIM
DO 650 J=1,1425
650 CONTINUE
700 CONTINUE

RETURN
END

SUBROUTINE ISR
DIMENSION AICVAL(300), AICHN(8), DACNH(4), EK(300), MK(300)
INTEGER AICVAL, I, CO, C1, MK, EK:
LOGICAL#1 NABC, NDAC, AICHN, DACNH
COMMON/ISRBLK/I, CO, C1
COMMON/AICBLK/NABC, AICHN
COMMON/DACBLK/NDAC, DACNH
COMMON/A1/AICVAL, IU, ISEC
COMMON/OUTBLK/MK, EK
EK(1)=IU
MK(1)=2047
AICVAL(1)=0

I=I+1
IM1=I-1
CALL ADC(AICVAL(I))
EK(I) = IU - ADCVAL(I)

C C C
C ***************************************************************
C C CALCULATE THE CONTROL
C C ***************************************************************
C
MK(I) = (MK(IM1) * D + CO * EK(I) - C1 * EK(IM1))

IF (MK(I) .GT. 2047) MK(I) = 2047
IF (MK(I) .LT. -2048) MK(I) = -2048
CALL DAC(MK(I))
RETURN
END
APPENDIX D

SELF-TUNING ADAPTIVE CONTROLLER
FLOWCHARTS AND SOURCE CODE
START

INITIALIZATION

ENABLE INTERRUPTS

LOOP TO KILL TIME DONE?

NO

YES

DISABLE INTERRUPTS

RETURN
CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

START

INITIALIZATION

SPECIFY INITIAL CONDITIONS

CALCULATE CONTROL GAINS

CALL ADC
(GET TACH VALUE)

CALCULATE THE ERROR

CALCULATE THE CONTROL

CALL DAC
(OUTPUT THE CONTROL)

CALCULATE NEW VALUES OF P AND Q FOR NEXT OUTPUT

RETURN

INTERRUPT SERVICE ROUTINE
SELF-TUNING ADAPTIVE CONTROLLER
COMPARISON OF AUTOMATIC CONTROL SCHEMES

CONTROLLER DESIGN PHASE

SELF-TUNING ADAPTIVE CONTROLLER

RESEARCH ASSISTANT: DAN LYMBURNER

RESEARCH ADVISOR: PROF. G. ULSOY

---------------------------------------------------------------------

THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A
DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600
PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR
CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL
MICROCOMPUTER". THE ALGORITHM USED IN THIS PARTICULAR
PROGRAM IS A SELF-TUNING ADAPTIVE CONTROLLER.

THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.

---------------------------------------------------------------------

** MAIN PROGRAM **

** --------------------------- **

** INITIALIZATION **

** --------------------------- **

DIMENSION Y(300), AICHN(8), DACHN(4), EK(300), MK(300)
INTEGER Y,MK,Ek
LOGICAL*1 NADC,NDIAC,AICHN,DACHN
COMMON/ISRLK/I,B,C,F,Q,F
COMMON/A1/Y,IU,ISEC
COMMON/AICBLK/NAIC,ADCHN
COMMON/DACBLK/NDIC,DACHN
COMMON/A2/ITIM
COMMON/OUTBLK/MK,Ek
AICHN(1)=2
DACHN(1)=0
NADC=1
NDIAC=1

** SEND A ZERO TO THE MOTOR **

** --------------------------- **

CALL DAC(0)

** --------------------------- **

* PROGRAM EXPLANATION *

* --------------------------- **
WRITE(3,10)

10 FORMAT(10)  
   THIS PROGRAM IS TO BE USED DURING THE CONTROLLER /
   DESIGN PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC
   CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A DC MOTOR USING
   A DIGITAL MICROCOMPUTER". THIS PROGRAM UTILIZES A SELF TUNING
   ADAPTIVE CONTROLLER TO CONTROL THE MOTOR SPEED. ')

12 WRITE(3,13)

13 FORMAT(10)  THE USER NOW HAS THE OPTION OF: '/
   1=QUITTING '/
   2=IMPLEMENTING THE CONTROL ALGORITHM '/
   '/
   C' ENTER 1 OR 2 '

15 FORMAT(11)

16 READ(2,15) KL

17 IF (KL .EQ. 1) GO TO 150

18 WRITE(3,30)

30 FORMAT(10)  YOU HAVE ELECTED TO IMPLEMENT THE PI CONTROL '/
   ALGORITHM. THE USER WILL NOW BE ASKED TO INPUT THE '/
   NECESSARY CONTROL VALUES. ')

35 FORMAT(16)

C

********;**************************************
C
*       INPUT THE USER VALUES

********;**************************************

C

40 WRITE(3,50)

50 FORMAT(10)  INPUT THE MAGNITUDE OF THE SPEED VALUE FOLLOWED BY A
   INTEGER VALUE ')

51 READ(2,35) IU

55 WRITE(3,60)

60 FORMAT(10)  INPUT THE TIME TO CONTROL THE MOTOR SPEED IN SECONDS.
   FOLLOW THIS VALUE WITH A COMMA (1 <X< 25) '

61 READ(2,35) ISEC

65 IF (ISEC .LT. 25) GO TO 55

66 WRITE(3,62)

62 FORMAT(10)  INPUT THE REAL PART OF Z1. INCLUDE A PERIOD '

63 READ(2,64) ZIR

64 FORMAT(F10.0)

65 WRITE(3,66)

66 FORMAT(10)  INPUT THE IMAGINARY PART OF Z1 & Z2. INCLUDE A PERIOD
   INCLUDE A PERIOD '

67 READ(2,64) ZI

68 FORMAT(F10.0)

68 WRITE(3,67)

67 FORMAT(10)  INCLUDE A PERIOD '

69 READ(2,64) Z2R

70 FORMAT(F10.0)

70 WRITE(3,68)

70 FORMAT(10)  INCLUDE A PERIOD '

71 READ(2,64) Z2I

72 FORMAT(F10.0)

72 WRITE(3,69)

72 FORMAT(10)  INCLUDE A PERIOD '

73 READ(2,64) EMBED

74 FORMAT(F10.0)

74 WRITE(3,70)

74 FORMAT(10)  INCLUDE A PERIOD '

75 READ(2,64) P

76 FORMAT(F10.0)

76 WRITE(3,71)

76 FORMAT(10)  INCLUDE A PERIOD '

77 READ(2,64) Q

78 FORMAT(F10.0)

78 WRITE(3,72)

78 FORMAT(10)  INCLUDE A PERIOD '

79 READ(2,64) F

80 C
WRITE(3,70) IU
70 FORMAT(/' THE MAGNITUDE OF THE SPEED VALUE IS ',I5 )
WRITE(3,80) ISEC
80 FORMAT(/' THE CONTROL TIME IS ',I3,' SECONDS ') WRITE(3,82) Z1R
82 FORMAT(/' THE REAL PART OF Z1 IS ',F7.3 ) WRITE(3,84) ZI
84 FORMAT(/' THE IMAGINARY PART OF Z IS ',F7.3 ) WRITE(3,85) Z2R
85 FORMAT(/' THE REAL PART OF Z2 IS ',F7.3 ) WRITE(3,86) Z
86 FORMAT(/' THE INTERRUPT PERIOD IS ',F7.4,' SECONDS ') WRITE(3,230) P
230 FORMAT(/' THE INITIAL VALUE OF PH IS ',F7.4 ) WRITE(3,240) Q
240 FORMAT(/' THE INITIAL VALUE OF QH IS ',F7.4 ) WRITE(3,250) F
250 FORMAT(/' THE INITIAL VALUE OF F IS ',F8.3 )

*************************************************************
* ARE THE VALUES CORRECT?  *
*************************************************************

WRITE(3,90)
90 FORMAT(/' ARE THE VALUES CORRECT? (1=YES, 2=NO) ')
READ(2,15) J
IF (J .EQ. 2) GO TO 40

*************************************************************
* CALCULATE THE TIME AND THE  *
* CONTROL PARAMETERS  *
*************************************************************

ITIM=530.*ISEC/15.
IOT=ISEC*10.
I=1
B=-(Z1R+Z2R)
IF (ZI .EQ. 0.) GO TO 95
GO TO 98
95 X=ABS(Z1R-Z2R)
C=(B**2.-X**2.)/4.
98 WRITE(3,96) B,C
96 FORMAT(/,5X,F7.4,5X,F7.4)

*************************************************************
* CALL SUBROUTINE TO INITIALIZE  *
* THE INTERRUPTS  *
*************************************************************

CALL INTINIT

*************************************************************
* START THE CONTROL  *
*************************************************************
WRITE(3, 100)
100 FORMAT(’/ PRESS RETURN TO BEGIN MOTOR CONTROL ’)
READ(2, 15) KA

************
* CALL SUBROUTINE TO WASTE TIME *
************
CALL WASTE

************
* SEND A ZERO TO SHUT OFF THE MOTOR *
************
CALL DISABL
CALL DAC(0)

************
TRY AGAIN?
************

125 WRITE(3, 130)
130 FORMAT(’/ THE MOTOR CONTROL IS COMPLETE. YOU NOW HAVE’/
C’ THE OPTION OF: ’/
C’ 1=PRINTING OUT THE CONTROL DATA ’/
C’ 2=REPEATING THE SELF TUNING CONTROL PROCEDURE ’/
C’ 3=QUITTING ’)

135 WRITE(3, 140)
140 FORMAT(’/ ENTER 1, 2, OR 3 ’)
READ(2, 15) JD
IF (JD .EQ. 1) GO TO 142
IF (JD .EQ. 2) GO TO 40
IF (JD .EQ. 3) GO TO 150
GO TO 135

142 WRITE(1, 143)
143 FORMAT(’/ 15X, ’SYSTEM RESPONSE USING A SELF TUNING CONTROLLER ’)
WRITE(1, 144) IU, Z1R, Z2R, ZI
WRITE(1, 145)
DO 147 L=1, 10T
WRITE(1, 146) L, HK(L), Y(L), EK(L)
146 FORMAT(6X, I3, 9X, I5, 11X, I5, 7X, I8)
147 CONTINUE
GO TO 125

150 STOP
END

************
SUBROUTINE TO WASTE TIME WHILE *
************
SUBROUTINE WASTE
COMMON/A2/ITIM
WRITE(3,600)
600 FORMAT(/' THE MOTOR CONTROL HAS BEGUN '/)  
C
C******************************************************************************
C* ENABLE THE INTERRUPTS *
C******************************************************************************
C
CALL ENABLE

C******************************************************************************
C* WASTE TIME *
C******************************************************************************
C
IO 700 JX=1,ITIM
IO 650 J=1,1425
650 CONTINUE
700 CONTINUE
RETURN
END

C******************************************************************************
C* INTERRUPT SERVICE *
C* ROUTINE FOR SELF TUNING ADAPTIVE CONTROLLER *
C******************************************************************************
C
SUBROUTINE ISK
DIMENSION Y(300), AICHN(8), IACHN(4), EK(300), MK(300)
INTEGER Y, MK, EK
REAL KPN, KIN
LOGICAL*1 NAIC, NdAC, AICHN, IACHN
COMMON/ISRBLK/I,R,C,P,O,F
COMMON/AICBLK/NAIC, AICHN
COMMON/IACBLK/NdAC, IACHN
COMMON/A1Y, IU, ISEC
COMMON/OUTBLK/MK, EK

******************************************************************************
C* SPECIFY INITIAL VALUES *
******************************************************************************
C
IF (I NE 1) GO TO 500
MK(1)=2047
YH=0.
QHF=Q
PHP=P
F11P=F
F22P=F
CALL AICH(Y(1))
CALL IACH(MK(1))
EK(1)=IU-Y(1)
500 I=I+1
J=I-1
KPN=(PHP-C)/QHF
KIN=(E+1+PHP-QHF*KPN)/( .1*QHF)
C
******************************************************************************
CALL ADC(Y(I))
EC(I)=IU-Y(I)

MK(I)=MK(J)+(KPN+KIN/10.)*EK(I)-KPN*EK(J)
IF (MK(I) .GT. 2047) MK(I)=2047
IF (MK(I) .LT. -2048) MK(I)=-2048
CALL DAC(MK(I))

** CALCULATE PH AND QH **

** CALCULATE EN **
C11P=(F11P*(-YHP))*(YHP)
C22P=(F22P*MK(J))*MK(J)
EN=(Y(I)-PHP*YHP-QHP*MK(J))/(C11P+C22P+1.)

** CALCULATE FN **
DFF=C11P+C22P+2.
F11N=(F11P-(C11P*F11P)/DFF)
F22N=(F22P-(C22P*F22P)/DFF)

** CALCULATE NEW VALUES OF PH AND QH **
PHN=PHP-(F11P*(-YHP))*EN
QHN=QHP+(F22P*MK(J))*EN
YHN=PHN*YHP+QHN*MK(J)
PHP=PHN
QHP=QHN
F11P=F11N
F22P=F22N
YHP=YHN
RETURN
END
APPENDIX E

ROBUST FEEDBACK CONTROLLER
FLOWCHARTS AND SOURCE CODE
CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

START

INITIALIZATION

SPECIFY INITIAL CONDITIONS

CALCULATE CONTROL GAINS

CALL ADC
(GET TACH VALUE)

CALCULATE THE ERROR

CALCULATE THE CONTROL

CALL DAC
(OUTPUT THE CONTROL)

CALCULATE NEW VALUES OF P AND Q FOR NEXT OUTPUT

RETURN

INTERRUPT SERVICE ROUTINE
ROBUST FEEDBACK CONTROLLER
**ME600 PROJECT**

**COMPARISON OF AUTOMATIC CONTROL SCHEMES**

**CONTROLLER DESIGN PHASE**

**ROBUST FEEDBACK CONTROLLER**

**RESEARCH ASSISTANT: IAN LYMBOURNER**

**RESEARCH ADVISOR: PROF. G. ULSBY**

**THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL MICROCOMPUTER". THE ALGORITHM USED IN THIS PARTICULAR PROGRAM IS A ROBUST FEEDBACK CONTROLLER.**

**THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.**

**MAIN PROGRAM**

**INITIALIZATION**

**DIMENSION Y(300), AICHN(8), IACHN(4), EK(300), HK(300)**

**INTEGER Y,MK,EK**

**LOGICAL NADC,NIAC,AICHN,IACHN**

**COMMON/ISRBLK/I,B,C,F,Q,F,A2**

**COMMON/A1/Y,IU,ISEC**

**COMMON/A1CBLK/NADC,IACHN**

**COMMON/A2/IITIM**

**COMMON/OUTBLK/HK,EK**

**AICHN(1)=2**

**IACHN(1)=0**

**NADC=1**

**NIAC=1**

**SEND A ZERO TO THE MOTOR**

**CALL DAC(0)**

**PROGRAM EXPLANATION**
C WRITE(3,10)

10 FORMAT(' THIS PROGRAM IS TO BE USED DURING THE CONTROLLER /
C DESIGN PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC
C CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A DC MOTOR USING
C A DIGITAL MICROCOMPUTER". THIS PROGRAM UTILIZES A ROBUST/
C FEEDBACK CONTROLLER TO CONTROL THE MOTOR SPEED. ')

12 WRITE(3,13)

13 FORMAT(' THE USER NOW HAS THE OPTION OF: /
C 1=QUITTING /
C 2=IMPLEMENTING THE CONTROL ALGORITHM /
C /
C ENTER 1 OR 2 ')

15 FORMAT(I1)

READ(2,15) KL

IF (KL .EQ. 1) GO TO 150

WRITE(3,30)

30 FORMAT(' YOU HAVE ELECTED TO IMPLEMENT THE CONTROL /
C ALGORITHM. THE USER WILL NOW BE ASKED TO INPUT THE /
C NECESSARY CONTROL VALUES. ')

35 FORMAT(I6)

C

C ********************************************************************
C * INPUT THE USER VALUES *
C ********************************************************************

40 WRITE(3,50)

50 FORMAT(' INPUT THE MAGNITUDE OF THE SPEED VALUE FOLLOWED BY A
C COMMA (-2048<X<2047) AN INTEGER VALUE ')

READ(2,35) IU

55 WRITE(3,60)

60 FORMAT(' INPUT THE TIME TO CONTROL THE MOTOR SPEED IN SECONDS.
C FOLLOW THIS VALUE WITH A COMMA ( 1 <X< 25 ) ')

READ(2,35) ISEC

IF (ISEC .GT. 25) GO TO 55

64 FORMAT(F10.0)

WRITE(3,65)

65 FORMAT(' INPUT THE REAL PART OF Z1. INCLUDE A PERIOD ')

READ(2,64) Z1R

WRITE(3,66)

66 FORMAT(' INPUT THE IMAGINARY PART OF Z1 & Z2. INCLUDE A PERIOD
C READ (2,64) Z1I
C WRITE (3,67)

67 FORMAT(' INPUT THE REAL PART OF Z2. INCLUDE A PERIOD ')

READ(2,64) Z2R

WRITE(3,68)

68 FORMAT(' INPUT THE INTERRUPT PERIOD IN SEC. INCLUDE A PERIOD.
C READ (2,64) Z
C WRITE (3,69)

69 FORMAT(' INPUT THE SENSITIVITY GAIN. INCLUDE A PERIOD. ')

READ(2,64) AK2

WRITE(3,200)

200 FORMAT(' INPUT THE INITIAL VALUE OF PH. INCLUDE A PERIOD. ')

READ(2,64) P

WRITE(3,210)

210 FORMAT(' INPUT THE INITIAL VALUE OF RH. INCLUDE A PERIOD. ')

READ(2,64) Q

WRITE(3,220)

220 FORMAT(' INPUT THE INITIAL VALUE OF THE MATRIX F. INCLUDE '/
READ(2,64) F

*** PRINT BACK THE VALUES ***

WRITE(3,70) IU
70 FORMAT(///' THE MAGNITUDE OF THE SPEED VALUE IS ',I5 )
WRITE(3,80) ISEC
80 FORMAT( ' THE CONTROL TIME IS ',I3,' SECONDS ')
WRITE(3,82) Z1R
82 FORMAT( ' THE REAL PART OF Z1 IS ',F7.3 )
WRITE(3,84) Z1I
84 FORMAT( ' THE IMAGINARY PART OF Z1 IS ',F7.3 )
WRITE(3,85) Z2R
85 FORMAT( ' THE REAL PART OF Z2 IS ',F7.3 )
WRITE(3,245) AK2
245 FORMAT( ' THE SENSITIVITY GAIN IS ',F9.3 )
WRITE(3,86) Z
86 FORMAT( ' THE INTERRUPT PERIOD IS ',F7.4,' SECONDS ')
WRITE(3,230) P
230 FORMAT( ' THE INITIAL VALUE OF PH IS ',F7.4 )
WRITE(3,240) Q
240 FORMAT( ' THE INITIAL VALUE OF QH IS ',F7.4 )
WRITE(3,250) F
250 FORMAT( ' THE INITIAL VALUE OF F IS ',F8.3 )

*** ARE THE VALUES CORRECT? ***

WRITE(3,90)
90 FORMAT(///' ARE THE VALUES CORRECT? (1=YES, 2=NO) ')
READ(2,15) J
IF (J .EQ. 2) GO TO 40

*** CALCULATE THE TIME AND THE CONTROL PARAMETERS ***

ITIM=530.*ISEC/15.
IT1=ISEC#10.
I=1
B=-(Z1R+Z2R)
IF (Z1 .EQ. 0.) GO TO 95
GO TO 98
95 X=ABS(Z1R-Z2R)
C=(B**2.-X**2.)/4.
98 WRITE(3,96) B,C
96 FORMAT(/,5X,F7.4,5X,F7.4)

*** CALL SUBROUTINE TO INITIALIZE ***

*** THE INTERRUPTS ***

...
CALL INITIAL

WRITE(3,100)
100 FORMAT(/' PRESS RETURN TO BEGIN MOTOR CONTROL ') READ(2,15) KA

CALL WASTE

WRITE(3,130)
130 FORMAT(/' THE MOTOR CONTROL IS COMPLETE. YOU NOW HAVE/')
      C' THE OPTION OF:' /
      C' 1=PRINTING OUT THE CONTROL DATA' /
      C' 2=REPEATING THE ROBUST FEEDBACK CONTROL PROCEDURE' /
      C' 3=QUITTING')

WRITE(3,140)
140 FORMAT(/' ENTER 1,2, OR 3 ') READ(2,15) JD IF (JD .EQ. 1) GO TO 142 IF (JD .EQ. 2) GO TO 40 IF (JD .EQ. 3) GO TO 150 GO TO 135

WRITE(1,143)
143 FORMAT(/' SYSTEM RESPONSE USING A ROBUST FEEDBACK CONTROLLER') WRITE(1,144) IU,Z1R,Z2R,Z1
144 FORMAT(15X,'SPEED=','15X,'REAL Z=','F5.3,2X,F5.3,' IM Z=','F5.3)
145 FORMAT(15X,'TIME',5X,'CONTROL EFFORT',5X,'ACTUAL SPEED',5X,'ERRO
DC 147 L=1,10T
148 WRITE(1,146) L, MK(L), Y(L), EK(L)
146 FORMAT(2X,'I3,9X,I5,11X,I5,7X,18)
147 CONTINUE
150 STOP
END

C
C SUBROUTINE TO WASTE TIME WHILE
C INTERRUPTS OCCUR
C
SUBROUTINE WASTE
COMMON/A2/ITIM,
WRITE(3,600)
600 FORMAT( 'THE MOTOR CONTROL HAS BEGUN' )

* ENABLE THE INTERRUPTS *
CALL ENABLE

* WASTE TIME *
DO 700 JX=1,ITIM
DO 650 J=1,1425
650 CONTINUE
700 CONTINUE
RETURN
END

SUBROUTINE ISR
DIMENSION Y(300), AICHN(8), IACHN(4), EK(300), HK(300)
INTEGER Y,HK,EK
REAL KPN, K3
LOGICAL%1 NADC,NIAC,AICHN,IACHN
COMMON/AIC_BLK/NADC,AICHN
COMMON/IACHN/NIAC,IACHN
COMMON/A1/Y,IIU,ISEC
COMMON/OUT_BLK/HK,EK

* SPECIFY INITIAL VALUES *
IF (I .NE. 1) GO TO 500
HK(1)=2047
YHP=0.
QHP=Q
PHP=P
F11P=F
F22P=F
CALL AIC(Y(1))
CALL IAC(HK(1))
EK(1)=IU-Y(1)
500 I=I+1
J=I-1

KPN=(PHP-C*(1.+AK2*QHP))/QHP
K3=(B+AK2*QHP*(B+1.)*R1+PHP-KPN*QHP)/(QHP)

CALL ADC(Y(I))
EK(I)=IU-Y(I)

* CALCULATE THE CONTROL *

S=AK2*(1-PHP)-KPN-K3
T=KPN*Y(J)
MK(I)=(MK(J)+S*Y(I)+K3*IU+T)/(1.+AK2*QHP)
IF (HK(I) .GT. 2047) MK(I)=2047
IF (HK(I) .LT. -2048) MK(I)=-2048
CALL DAC(MK(I))

* CALCULATE PH AND QH *

* CALCULATE EN *

C11P=(F11P*(-YHP))*(-YHP)
C22P=(F22P*HK(J))*MK(J)
EN=(Y(I)-PHP*YHP-QHP*MK(J))/(C11P+C22P+1.)

* CALCULATE FN *

IFP=C11P+C22P+2.
F11N=(F11P-(C11P*F11P)/IFP)
F22N=(F22P-(C22P*F22P)/IFP)

* CALCULATE NEW VALUES OF PH AND QH *

PHN=PHP-(F11P*(-YHP))*EN
QHN=QHP+(F22P*HK(J))*EN
YHN=PHN*YHP+QHN*MK(J)
PHP=PHN
QHP=QHN
F11P=F11N
F22P=F22N
YHP=YHN
RETURN
END
APPENDIX F

ASSEMBLY LANGUAGE ROUTINES
SOURCE CODE
ASSEMBLY LANGUAGE CODE FOR ME600 PROJECT

INTERRUPT CONTROLLER COMMAND WORDS

ICW1 EQU 010H ; INTERRUPT INITIALIZATION
ICW2 EQU 011H ; COMMAND WORDS
OCW1 EQU 011H ; INTERRUPT OPERATION
OCW2 EQU 010H ; CONTROL WORDS
MASK EQU 0FEH ; INTERRUPT MASK
EOI EQU 020H ; END OF INTERRUPT

CLKEOI MACRO

This macro issues the interrupt acknowledge command to the real time clock, and is designed to be physically last in an interrupt service routine

PUSH AF
LD A, EOI ; ISSUE AN END OF INTERRUPT
OUT (OCW2), A ; TO THE CONTROLLER
POP AF
EI ; ENABLE INTERRUPTS
RETI ; AND RETURN
ENDM

THE REAL TIME CLOCK

ASEG
PUBLIC INTVEC
ORG 04000H ; MUST START AT A 64 BYTE BOUNDARY; HERE IT IS LOADED IN HIGH MEMORY

INTERRUPT VECTOR

INTVEC: ; ONLY FUNCTION IS TO
JP SERVICE ; JUMP TO SERVICE ROUTINE

CSEG
PUBLIC SERVICE, INTINL, ENABLE, DISABLE
EXT ISR
INTNL: PUSH HL
    PUSH AF

INITIALIZE THE INTERRUPT FACILITY AND
SET THE MASK REGISTER TO ENABLE THE REAL TIME
CLOCK (BIT 0 = CLOCK)

LD HL, INTVEC ; INTERRUPT VECTOR -> HL
LD A, L ; LOWER ROUTINE Addr -> A
AND 11100000B ; MASK OFF BITS 0 - 5
OR 00010010B ; MERGE OS VECTOR INTERVAL = 8 BYTES
OUT (ICW1), A ; INITIALIZE
LD A, H ; INTERRUPT
OUT (ICW2), A ; ADDRESS

LD A, MASK ; INITIALIZE MASK
OUT (OCW1), A ; FOR REAL TIME CLOCK

SETS INTERRUPT MODE 0 FOR THE CPU
THE USER MUST ISSUE A ENABLE INTERRUPTS
COMMAND TO ACTUALLY START THE CLOCK

EI
Clock is ticking from this point on

IM 0
RETURN FROM SUBROUTINE

POP AF
POP HL
RET

SERVICE ROUTINE THAT JUST RETURNS AFTER AN INTERRUPT
OCCURRED. THIS MEANS THAT AFTER YOU ENABLE INTERRUPTS
AND YOU ISSUE A HALT COMMAND, THE PROCESSOR STARTS
EXECUTING AGAIN AFTER 100 MILLISECONDS, RELOADS THE CLOCK
AND EXECUTES THE NEXT STATEMENT AFTER HALT

SERVICE: CALL ISR ; CALL INTERRUPT SERVICE ROUTINE
CLKEO1 ; ACKNOWLEDGE INTERRUPT
; AND RETURN
This routine is designed to be called from a FORTRAN program and will perform analog to digital conversions using the XYCOM 1850B board. The mode of operation is random channel addressing with interrupts disabled. The number and order of addressing is specified by the labeled common block /AICBLK/. The subroutine returns values in an INTEGER*2 array which is the argument to the call.

A sample calling sequence is:

```
INTEGER*2 AICVAL(27)
LOGICAL*1 NAIC,AICHN(8)

DO 100 I=1,27,3
CALL AIC(AICVAL(I))
CONTINUE
```

```
SUBTTL COMMON /AICBLK/ NAIC,AICHN(8)
```

```
SYMBOLS
```

```
SLOTNO EQU OFF70H ; SLOT CODE WHERE 1850B IS INSTALLED AND IS THE BASE FOR THE FOLLOWING!
```

```
STATUS EQU SLOTNO ; STATUS WORD
```

```
MULPLX EQU SLOTNO+1 ; MULTIPLEX ADDRESS
```
DATA1 EQU SLOTNO+2;ADDRESS OF FIRST DATA BYTE
DATA2 EQU SLOTNO+3;ADDRESS OF SECOND DATA BYTE
ADDONE EQU 7 ;BIT SET WHEN DONE

;----------------------------------
;
CODE
;
;----------------------------------
;
SAVE REGISTERS ON ENTRY
;
ADCl: PUSH IX
PUSH IY
PUSH HL
PUSH DE
PUSH BC
PUSH AF

ASSUME DESTINATION ADDRESS OF DATA IS
POINTED TO BY HL

LD A,(NADC) ;NUMBER OF
LD B,A ;CHANNELS IN B
LD IX,AIDCHN ;IX FOR MUltIPLEXING
LD IY,STATUS ;IY TO CHECK STATUS

THE FOLLOWING LOOP MULTIPLEXES, WAITS, & STORES

ADCBl: LD A,(HULPLX),A ;ADDRESS A CHANNEL &
INC IX ;INCREMENT FOR NEXT TIME
LD A,(IX) ;ACCUMULATOR HAS CHANNEL

WAIT: BIT ADDONE,(IY) ;IF THE DONE BIT NOT SET
JR Z,WAIT ;WAIT TIL IT IS

LD DE,(DATA1) ;(DATA1) -> E
LD (HL),I ;STORE HIGH BYTE
INC HL ;AND INCREMENT

LD (HL),E ;STORE LOW BYTE
INC HL ;AND INCREMENT HL
DIJNZ AICLP ; LOOP IF MORE CHANNELS

RESTOR REGISTERS

POPOP AF
POPOP BC
POPOP DE
POPOP HL
POPOP IY
POPOP IX
RET

TITLE DAC SUBROUTINE
SUBTTL DOCUMENTATION

This routine is designed to be called from a FORTRAN program and will perform digital to analog conversions using the XYCOM 1850B board. The mode of operation is random channel addressing with interrupts disabled. The number and order of addressing is specified by the labeled common block /DACBLK/. The subroutine sends values in an INTEGER*2 array which is the argument to the call.

A sample calling sequence is:

INTEGER*2 DACVAL(27)
LOGICAL*1 NIAC,DACHN(4)
COMMON /DACBLK/ NIAC,DACHN

DO 100 I=1,27,3
CALL DAC(DACVAL(I))
CONTINUE

CSEG
PUBLIC DAC

------------------------

---
SAVE REGISTERS ON ENTRY

DAC:
    PUSH IX
    PUSH HL
    PUSH DE
    PUSH BC
    PUSH AF

ASSUME SOURCE ADDRESS OF DATA IS POINTED TO BY HL

LD A,(NDAC)    ;NUMBER OF
LD B,A          ;CHANNELS IN B
LD IX,DACHN     ;IX FOR MULTIPLEXING

DAACL:
    LD D,(HL)      ;GET LOW BYTE
    INC HL         ;AND INCREMENT
    LD A,OFH       ;MASK FOR BITS 4-7
    AND (HL)       ;WITH RESULT IN A
    INC HL         ;AND INCREMENT
    LD E,A         ;AND MOV TO E

CHECK FOR ODD OR EVEN CHANNEL

BIT 0,(IX)      ;CHECK BIT ZERO
JR Z,SENDIT     ;IF EVEN, SEND NOW
SET 7,E         ;OTHERWISE SELECT DAC1

SENDIT: LD (DATA1),DE
    INC IX

DJNZ DAACL      ;LOOP IF MORE CHANNELS

RESTOR REGISTERS

POP AF
POP BC
POP DE
POP HL
POP IX
RET
END