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

BY: D. R. LYMBURNER
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

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

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 coverter, 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

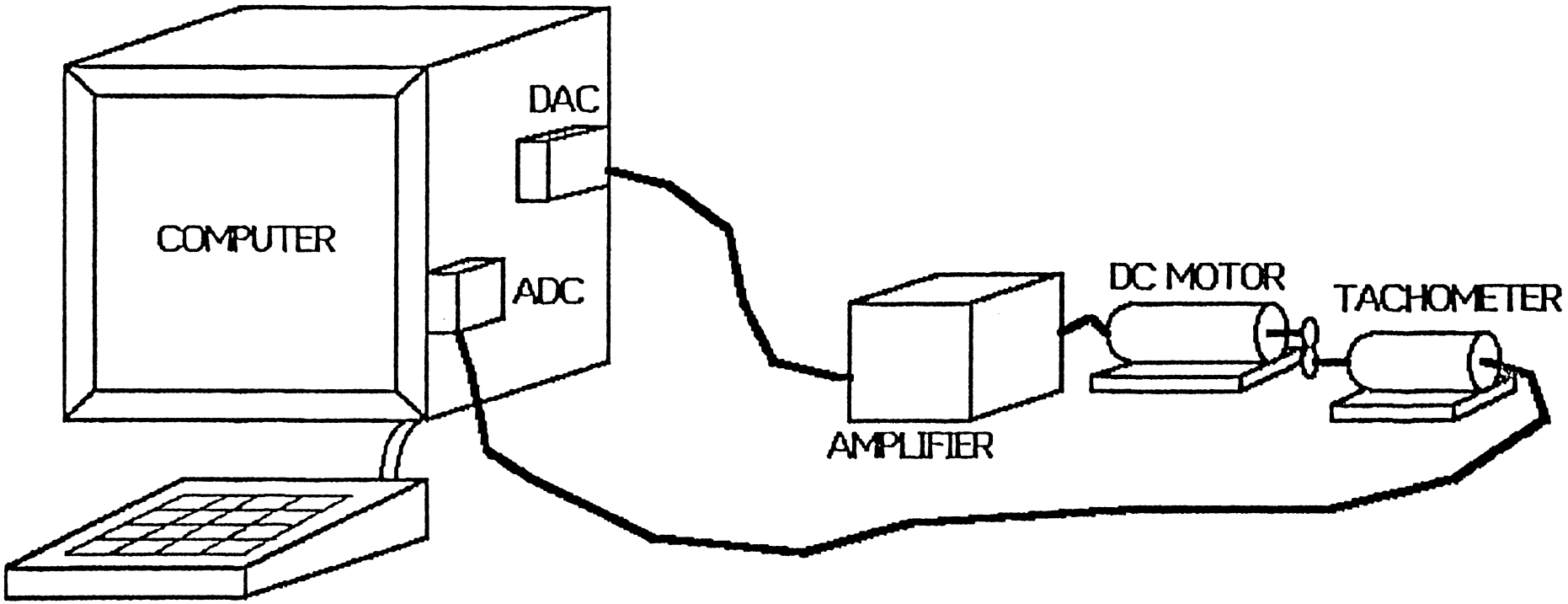


FIGURE 1
PHYSICAL SET UP

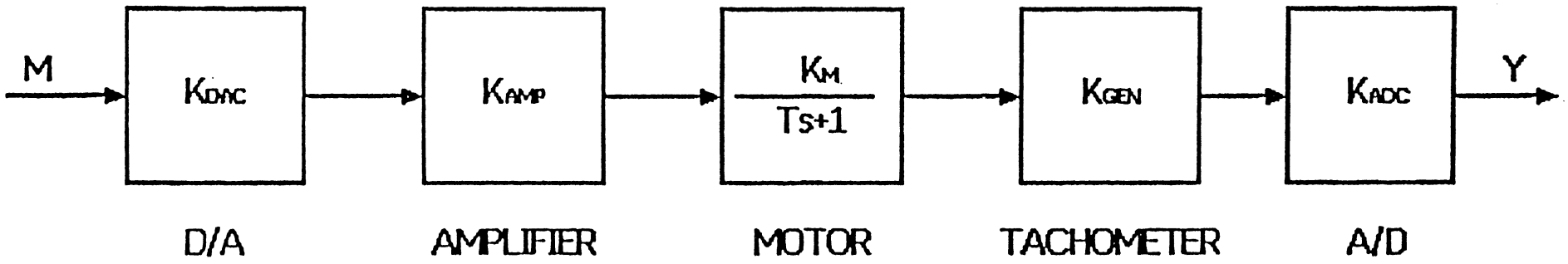


FIGURE 2
OPEN LOOP SYSTEM

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:

$$\frac{Eo(s)}{Ei(s)} = \frac{Ka}{Ts+1}$$

where Eo is the output signal, Ei is the input signal, Ka is the gain, and T is the time constant. For most amplifiers the time constant is very small (on the order of nanoseconds), therefore

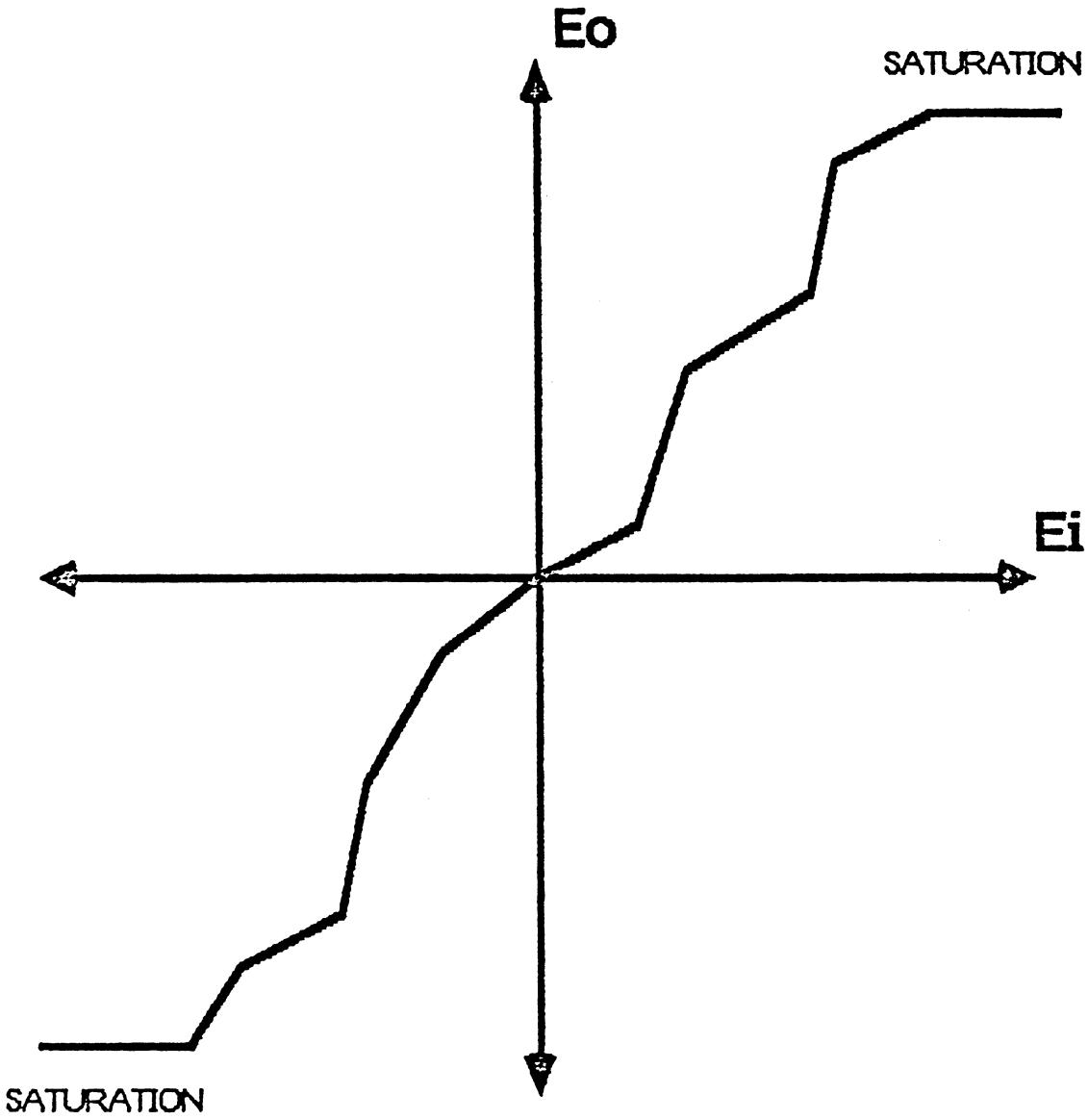


FIGURE 3

TYPICAL AMPLIFIER CHARACTERISTIC

for all practical purposes it can be neglected. This simplifies the model for the amplifier to the simple gain 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 (W) 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 seperately. First consider the electric circuit on the left half of Figure 4. Applying Kirchoff's Voltage Equation to the DC motor circuit yields:

$$-V+IR+L\frac{(dI)}{(dt)}+KvW=0$$

Therefore:

$$I(s)=\frac{V(s)-KvW(s)}{Ls+R} \qquad 1)$$

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

Newton's Equation) $\sum Mx=(J1+J2)\frac{(dW)}{(dt)}=Tm-bW-K\theta$

$$(J1+J2)sW(s)=KtI(s)-bW(s)-KW(s)/s$$

$$W(s)(s(J1+J2)+b+K/s)=KtI(s) \qquad 2)$$

Substituting 1) into 2) yields:

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

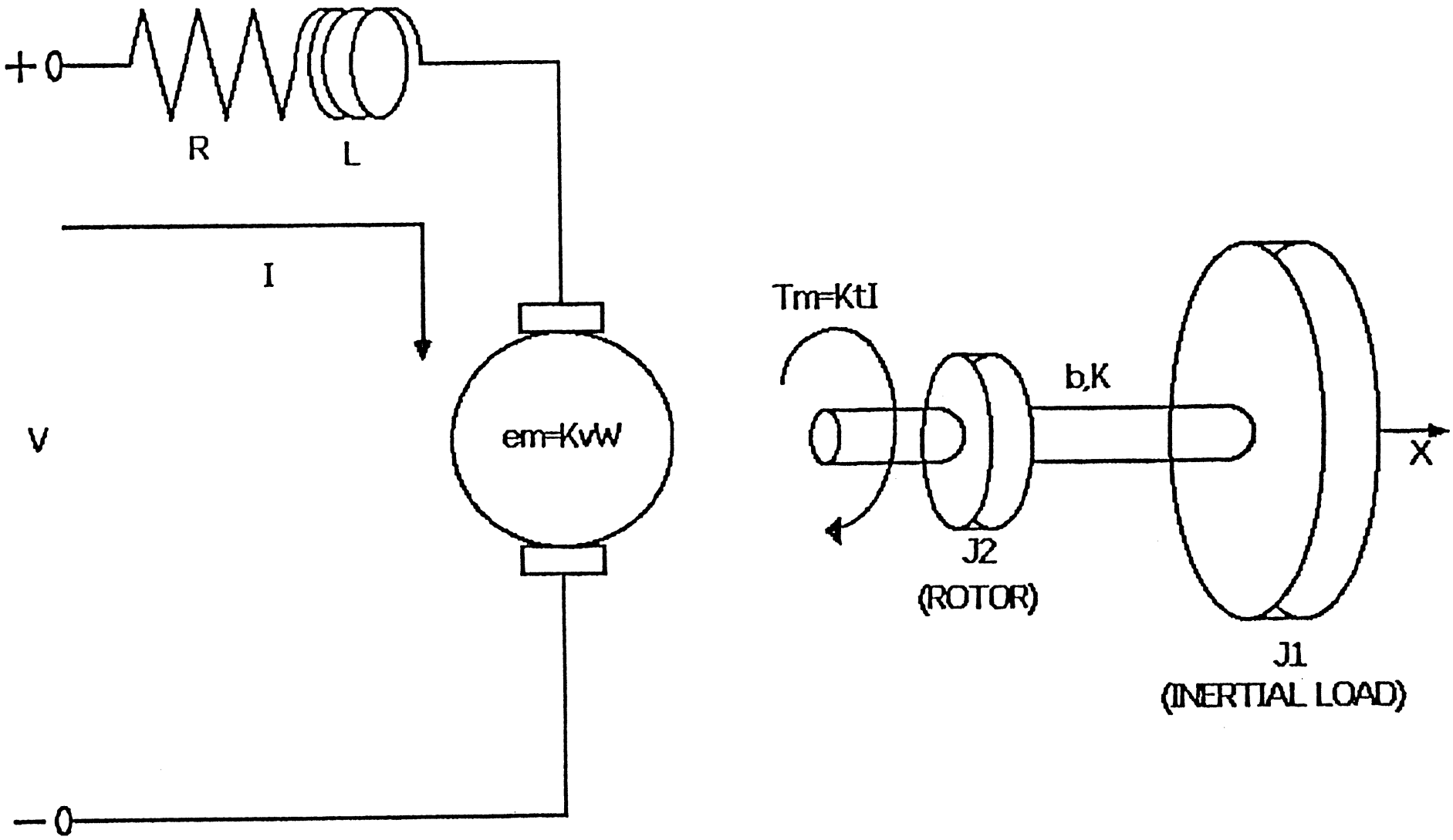


FIGURE 4

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:

$$\frac{W(s)}{V(s)} = \frac{(1/Kv)}{(RJ/KtKv)s+1} = \frac{Km}{Ts+1} \qquad 4)$$

where Km=DC Motor Gain and T=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

plifier Voltage Level	Voltage at Motor Terminals	Current	Tachometer Voltage
-9.518	-7.400	-0.61	-6.770
-8.998	-7.772	-0.62	-6.450
-8.006	-7.221	-0.60	-5.230
-6.992	-6.313	-0.60	-4.230
-6.008	-5.425	-0.60	-3.410
-5.004	-4.524	-0.58	-2.770
-4.009	-3.630	-0.55	-2.020
-3.006	-2.728	-0.50	-1.222
-2.900	-2.380	-0.45	-0.480
2.502	2.193	0.75	0.000
2.807	2.485	0.43	0.716
3.000	2.654	0.43	0.817
3.498	3.109	0.45	1.156
4.002	3.563	0.50	1.510
4.503	4.018	0.51	1.893
5.011	4.470	0.54	2.210
5.500	4.910	0.55	2.589
6.005	5.361	0.57	2.950
6.500	5.802	0.58	3.280
7.001	6.252	0.59	3.660
7.498	6.701	0.60	4.180
8.003	7.155	0.61	4.600
8.499	7.601	0.61	5.120
9.002	8.055	0.62	5.570
9.501	8.486	0.63	6.150

TABLE 1

MOTOR-TACHOMETER VOLTAGE DATA

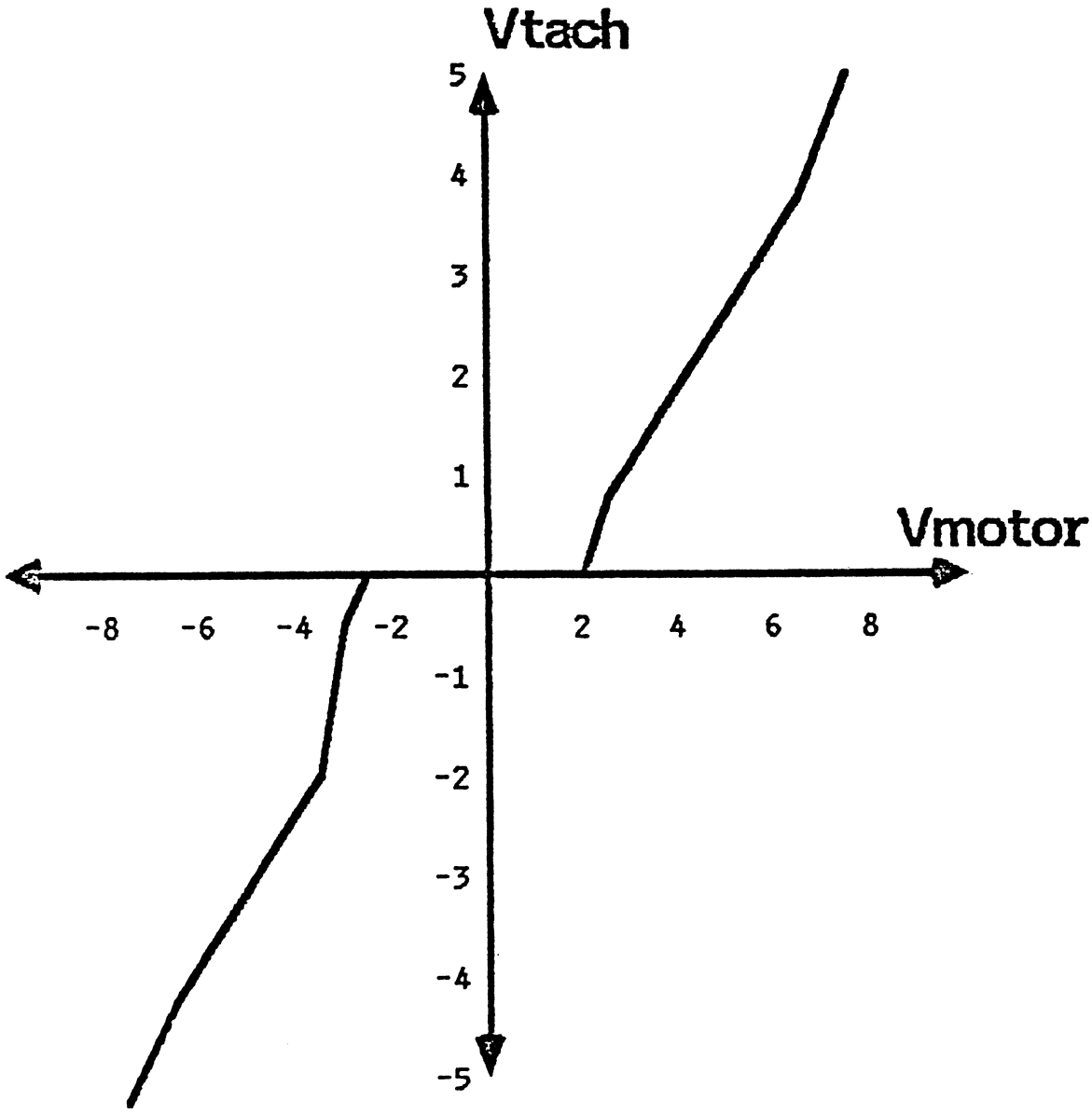


FIGURE 5

PLOT OF TACHOMETER vs. MOTOR VOLTAGE

function:

$$\text{Open Loop Transfer Function} \quad \frac{Y(s)}{M(s)} = \frac{Kol}{(Tol)s+1} \qquad 5)$$

where: M(s)=Control Effort
Y(s)=State Variable
Tol=Open Loop Time Constant
Kol=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.

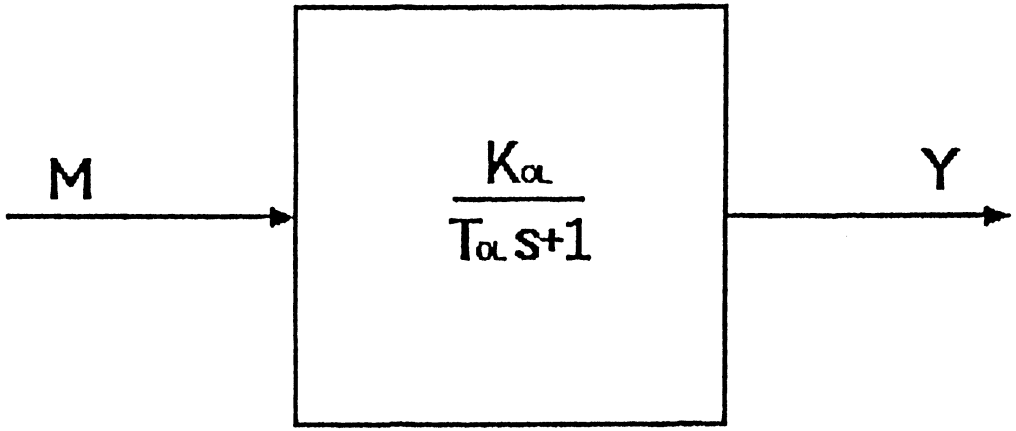


FIGURE 6
COMBINED OPEN LOOP SYSTEM

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 occuring 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.

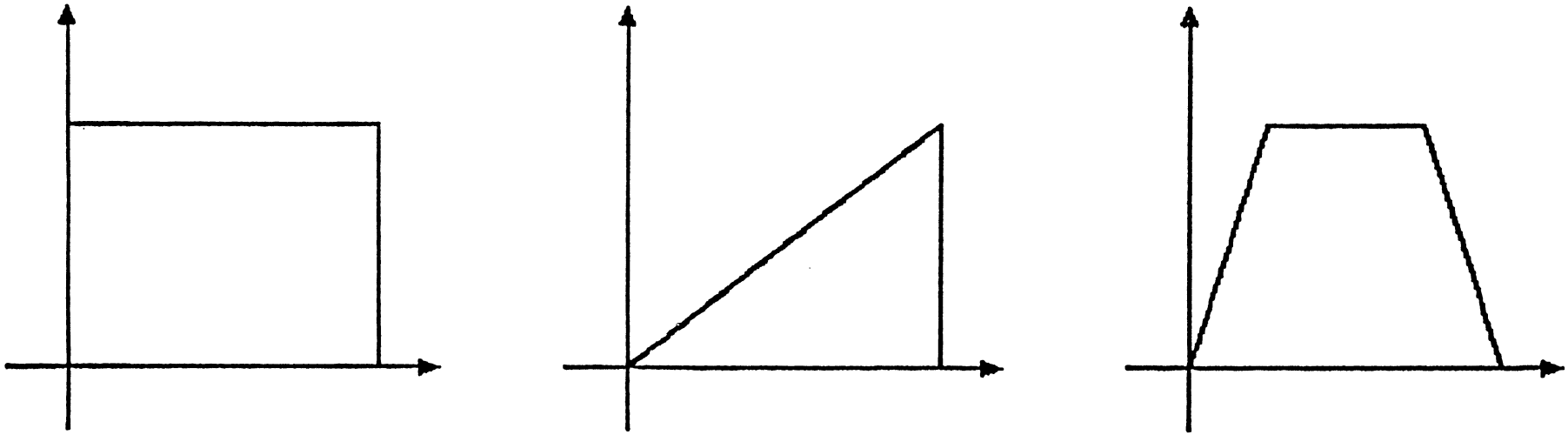


FIGURE 7
SYSTEM INPUT SIGNALS

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:

$$Y(t)=Kol(1 - e^{-t/Tol})M(t)$$

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,

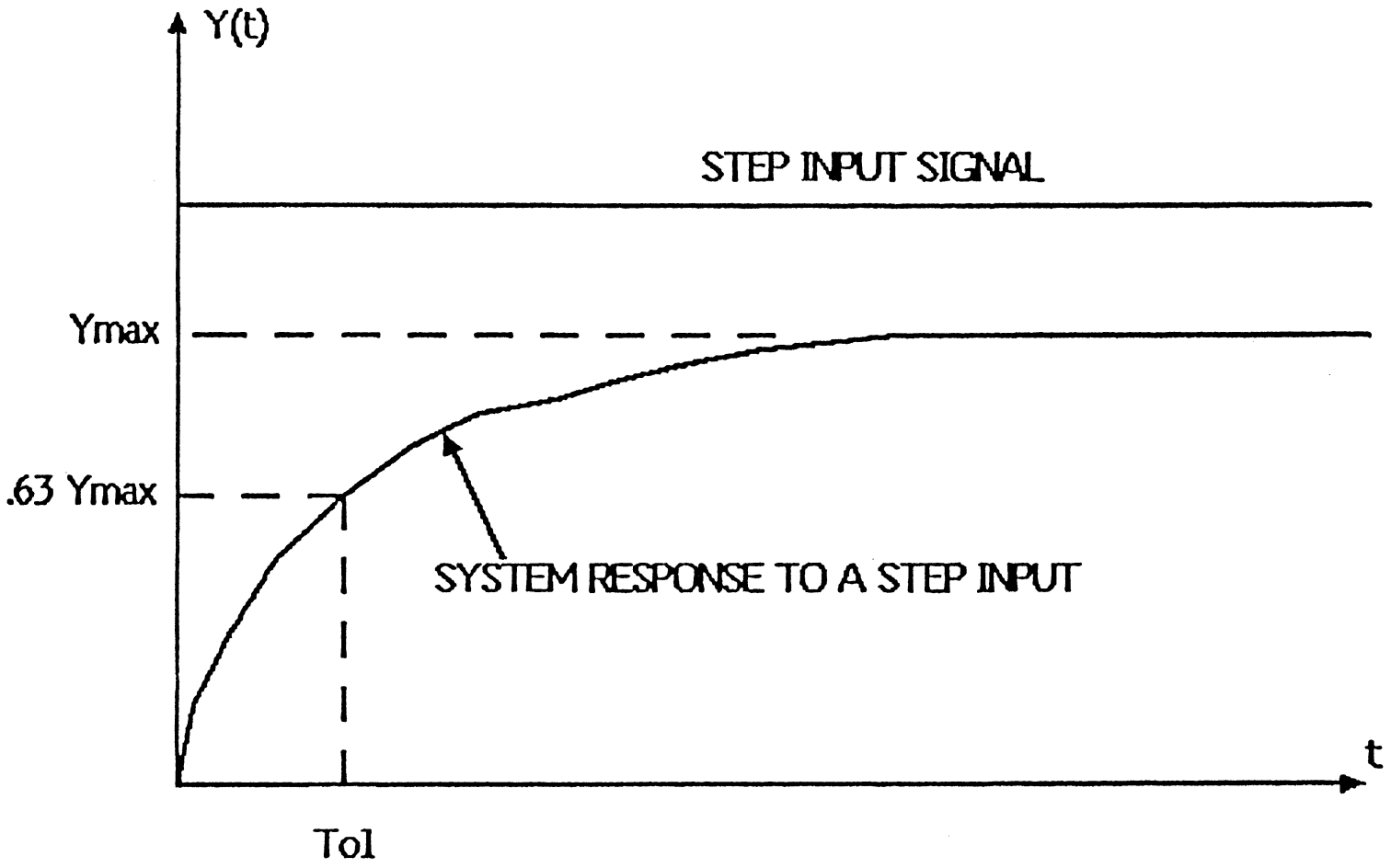


FIGURE 8
SYSTEM RESPONSE

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

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

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):

$$M(s)=\frac{Kol}{Tols+1}\,Y(s) \qquad 6a)$$

where: M(s)=Control Effort
Y(s)=Output Signal
Kol=Open Loop Gain
Tol=Open Loop Time Constant

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

$$dY/dt=aY+bM \qquad 6b)$$

where: a=-1/Tol
b=Kol/Tol

The discrete time state equation is written as:

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

where: $P=e^{a\Delta t}$
$Q=(P-1)b/a$
Δt=sampling(interrupt) period 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:

$$Y(K+1)=P(K)Y(K)+Q(K)M(K) \quad \text{Motor (Fixed System)} \qquad 8)$$

$$YH(K+1)=PH(K+1)YH(K)+QH(K+1)M(K) \quad \text{Model (Adj. System)} \qquad 9)$$

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:

$$YHO(K+1)=PH(K)YH(K)+QH(K)M(K) \qquad 10)$$

Next an 'a priori' error is calculated by:

$$EO(K+1)=Y(K+1)-YHO(K+1) \qquad 11)$$

The error at time (K+1) is calculated by:

TABLE 2

STEP INPUT MAGNITUDE	OPEN LOOP GAIN (Kol)	OPEN LOOP TIME CONSTANT (Tol)	P	Q
500	0.39	1.95 sec	.950	.0195
1000	0.60	1.65 sec	.941	.0353
1500	0.67	1.60 sec	.939	.0406
2000	0.71	1.60 sec	.939	.0430

CONTINUOUS AND DISCRETE TIME PARAMETERS WITH VARYING INPUT MAGNITUDES

$$E(K+1)=EO(K+1)/(1+K_1YH(K)^2+K_2M(K)^2) \quad 12)$$

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:

$$PH(K+1)=PH(K)+K_1E(K+1)YH(K) \quad 13)$$

$$QH(K+1)=QH(K)+K_2E(K+1)M(K) \quad 14)$$

The model response is calculate 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 quited 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.

ESTIMATES OF PH AND QH USING A MRAS
SYSTEM RESULTS FOR INPUT= 1000
K1= 100.00 K2= 100.00

TIME	PH	QH	X	XH	ERROR
1	.900	.050	7.0	0.0	7.0
2	.900	.050	3.0	0.0	3.0
3	.900	.054	54.0	54.0	.0
4	.900	.048	97.0	97.0	-.0
5	.900	.053	140.0	140.0	.0
6	.900	.055	181.0	181.0	.0
7	.900	.054	217.0	217.0	0.0
8	.901	.058	254.0	254.0	.0
9	.901	.057	286.0	286.0	0.0
10	.901	.057	315.0	315.0	0.0
11	.901	.059	343.0	343.0	.0
12	.901	.059	368.0	368.0	.0
13	.903	.062	394.0	394.0	0.0
14	.901	.059	414.0	414.0	-.0
15	.903	.062	436.0	436.0	.0
16	.903	.063	457.0	457.0	-.0
17	.902	.062	474.0	474.0	0.0
18	.903	.062	490.0	490.0	-.0
19	.902	.060	502.0	502.0	-.0
20	.903	.063	517.0	517.0	0.0
21	.903	.062	529.0	529.0	0.0
22	.903	.063	541.0	541.0	0.0
23	.904	.064	553.0	553.0	0.0
24	.904	.064	564.0	564.0	0.0
25	.903	.063	572.0	572.0	0.0
26	.905	.066	583.0	583.0	0.0
27	.904	.065	592.0	592.0	.0
28	.905	.066	601.0	601.0	-.0
29	.906	.068	612.0	612.0	.0
30	.902	.062	614.0	614.0	-.0
31	.905	.066	621.0	621.0	0.0
32	.905	.066	628.0	628.0	.0
33	.904	.065	633.0	633.0	0.0
34	.905	.065	638.0	638.0	0.0
35	.906	.067	645.0	645.0	.0
36	.905	.066	649.0	649.0	0.0
37	.904	.065	652.0	652.0	0.0
38	.904	.065	654.0	654.0	0.0
39	.906	.068	661.0	661.0	.0
40	.904	.065	662.0	662.0	0.0
41	.904	.065	664.0	664.0	0.0
42	.905	.067	668.0	668.0	0.0
43	.905	.067	672.0	672.0	0.0
44	.905	.067	675.0	675.0	.0
45	.905	.067	678.0	678.0	-.0
46	.905	.066	680.0	680.0	-.0
47	.903	.064	678.0	678.0	0.0
48	.906	.068	683.0	683.0	-.0
49	.905	.067	685.0	685.0	.0
50	.904	.065	684.0	684.0	0.0
51	.906	.068	688.0	688.0	.0
52	.905	.066	689.0	689.0	.0
53	.906	.067	691.0	691.0	-.0
54	.906	.067	693.0	693.0	.0
55	.904	.065	692.0	692.0	-.0

FIGURE 9
MRAS Estimator wi
Initial Values of
and QH slightly d
from Actual Value

ESTIMATES OF PH AND QH USING A MRAS
SYSTEM RESULTS FOR INPUT= 1000
K1= 100.00 K2= 100.00

TIME	PH	QH	X	XH	ERROR
1	.950	.040	7.0	0.0	7.0
2	.950	.040	3.0	0.0	3.0
3	.950	.054	54.0	54.0	.0
4	.950	.046	97.0	97.0	-.0
5	.950	.048	140.0	140.0	.0
6	.950	.048	181.0	181.0	0.0
7	.949	.045	217.0	217.0	0.0
8	.950	.048	254.0	254.0	.0
9	.949	.045	286.0	286.0	-.0
10	.949	.044	315.0	315.0	0.0
11	.949	.044	343.0	343.0	0.0
12	.948	.043	368.0	368.0	0.0
13	.949	.045	394.0	394.0	0.0
14	.948	.041	414.0	414.0	-.0
15	.949	.043	436.0	436.0	.0
16	.949	.043	457.0	457.0	-.0
17	.948	.041	474.0	474.0	-.0
18	.948	.041	490.0	490.0	-.0
19	.946	.038	502.0	502.0	-.0
20	.948	.041	517.0	517.0	0.0
21	.947	.039	529.0	529.0	0.0
22	.947	.040	541.0	541.0	0.0
23	.947	.040	553.0	553.0	.0
24	.947	.040	564.0	564.0	.0
25	.946	.038	572.0	572.0	.0
26	.948	.041	583.0	583.0	0.0
27	.947	.040	592.0	592.0	-.0
28	.947	.040	601.0	601.0	.0
29	.948	.042	612.0	612.0	-.0
30	.945	.036	614.0	614.0	0.0
31	.947	.040	621.0	621.0	.0
32	.947	.040	628.0	628.0	0.0
33	.946	.039	633.0	633.0	0.0
34	.946	.039	638.0	638.0	.0
35	.947	.041	645.0	645.0	0.0
36	.946	.039	649.0	649.0	0.0
37	.946	.038	652.0	652.0	0.0
38	.945	.038	654.0	654.0	0.0
39	.948	.041	661.0	661.0	0.0
40	.945	.037	662.0	662.0	0.0
41	.946	.038	664.0	664.0	0.0
42	.947	.039	668.0	668.0	0.0
43	.947	.040	672.0	672.0	0.0
44	.946	.039	675.0	675.0	0.0
45	.947	.039	678.0	678.0	0.0
46	.946	.039	680.0	680.0	0.0
47	.944	.036	678.0	678.0	0.0
48	.948	.041	683.0	683.0	.0
49	.946	.039	685.0	685.0	0.0
50	.945	.037	684.0	684.0	0.0
51	.947	.040	688.0	688.0	0.0
52	.946	.038	689.0	689.0	0.0
53	.946	.039	691.0	691.0	0.0
54	.946	.039	693.0	693.0	.0
55	.945	.037	692.0	692.0	-.0
56	.946	.039	694.0	694.0	0.0

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:

$$\min J(K)=\sum(Y(K)-YHO(K))^2=\sum EO(K)^2 \qquad 15)$$

where YHO(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:

$$EO(K+1)=Y(K+1)-YHO(K+1) \qquad 11)$$

substituting equation 10 into 11 yields:

$$EO(K+1)=Y(K+1)-\theta^T(K)\Delta(K) \qquad 16)$$

where: $\theta^T(K)= (-PH(K)\ QH(K))$ 17)

$\Delta^T(K)= (-YH(K)\ M(K))$ 18)

Substituting 16 into 15 yields:

$$J(K)=\sum(Y(K)-\theta^T(K)\Delta(K))^2 \qquad 19)$$

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

$$\frac{\partial J(K)}{\partial \theta^T(K)} = 0 \qquad 20)$$

After taking the derivative the following result is obtained:

$$\theta(K)=F(K)(\sum Y(K)\Delta(K-1)) \qquad 21)$$

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:

$$\theta(K+1)=\theta(K)+F(K+1)\Delta(K)EO(K+1) \quad 23)$$

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, EO(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 EO(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

RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE
SYSTEM RESULTS FOR INPUT STEP= 1000
INITIAL PH= .94 INITIAL QH= .03

TIME	PH	QH
1	.943	.034
2	.939	.038
3	.939	.038
4	.937	.040
5	.938	.039
6	.941	.036
7	.940	.037
8	.937	.040
9	.939	.038
10	.938	.039
11	.939	.038
12	.938	.039
13	.938	.039
14	.938	.039
15	.939	.038
16	.939	.038
17	.940	.037
18	.938	.039
19	.937	.040
20	.938	.039
21	.940	.037
22	.939	.038
23	.939	.038
24	.938	.039
25	.938	.039
26	.940	.037
27	.939	.038
28	.939	.038
29	.939	.038
30	.938	.039
31	.938	.039
32	.938	.039
33	.939	.038
34	.940	.037
35	.939	.038
36	.938	.039
37	.939	.038
38	.939	.038
39	.939	.038
40	.938	.039
41	.939	.038
42	.938	.039
43	.938	.039
44	.939	.038
45	.938	.039
46	.937	.040
47	.938	.039
48	.939	.038
49	.939	.038
50	.939	.038
51	.939	.038
52	.938	.039
53	.939	.038
54	.938	.039
55	.938	.039
56	.938	.039

FIGURE 11
Recursive Least Squares Esti
with initial values of PH ar
close to the actual values

RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE
SYSTEM RESULTS FOR INPUT STEP= 1000
INITIAL PH= .90 INITIAL QH= .05

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

FIGURE 12
Recursive Least Squares Estimato
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:

$$YHO(K+1)=PH(K)YH(K)+QH(K)M(K) \quad (24)$$

Next an 'a priori' error is calculated:

$$EO(K+1)=Y(K+1)-YHO(K+1) \quad (25)$$

Next an adaptation error is calculated by:

$$E(K+1)=EO(K+1)/(1+\Delta^T(K)F(K)\Delta(K)) \quad (26)$$

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:

$$F(K+1)=\frac{1}{L1}\left(F(K) - \frac{F(K)\Delta(K)\Delta^T(K)F(K)}{(L1/L2)+\Delta T(K)F(K)\Delta(K)}\right) \quad (27)$$

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

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 EO(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.

TUNABLE ESTIMATOR ESTIMATION PROCEDURE
SYSTEM RESULTS FOR INPUT= 1000
INITIAL PH= .50 INITIAL QH= .05 F= .010
LAMBDA 1=1.0000 LAMBDA 2= .5000

TIME	PH	QH	F11	F12	F21	F22
1	.500	.050	.010	0.000	0.000	.010
2	.500	.050	.010	0.000	0.000	.010
3	.500	.040	.010	0.000	0.000	.000
4	.813	.049	.001	.000	.000	.000
5	.858	.049	.000	.000	.000	.000
6	.903	.047	.000	.000	.000	.000
7	.910	.047	.000	.000	.000	.000
8	.917	.047	.000	.000	.000	.000
9	.926	.046	.000	.000	.000	.000
10	.924	.046	.000	.000	.000	.000
11	.923	.046	.000	.000	.000	.000
12	.925	.046	.000	.000	.000	.000
13	.927	.046	.000	.000	.000	.000
14	.927	.046	.000	.000	.000	.000
15	.927	.046	.000	.000	.000	.000
16	.929	.046	.000	.000	.000	.000
17	.929	.045	.000	.000	.000	.000
18	.933	.045	.000	.000	.000	.000
19	.933	.045	.000	.000	.000	.000
20	.930	.046	.000	.000	.000	.000
21	.930	.046	.000	.000	.000	.000
22	.929	.046	.000	.000	.000	.000
23	.930	.046	.000	.000	.000	.000
24	.930	.046	.000	.000	.000	.000
25	.931	.045	.000	.000	.000	.000
26	.931	.045	.000	.000	.000	.000
27	.930	.046	.000	.000	.000	.000
28	.931	.045	.000	.000	.000	.000
29	.931	.046	.000	.000	.000	.000
30	.932	.045	.000	.000	.000	.000
31	.931	.045	.000	.000	.000	.000
32	.932	.045	.000	.000	.000	.000
33	.931	.045	.000	.000	.000	.000
34	.932	.045	.000	.000	.000	.000
35	.933	.045	.000	.000	.000	.000
36	.932	.045	.000	.000	.000	.000
37	.933	.045	.000	.000	.000	.000
38	.933	.045	.000	.000	.000	.000
39	.933	.045	.000	.000	.000	.000
40	.933	.045	.000	.000	.000	.000
41	.933	.045	.000	.000	.000	.000
42	.933	.045	.000	.000	.000	.000
43	.933	.045	.000	.000	.000	.000
44	.933	.045	.000	.000	.000	.000
45	.933	.045	.000	.000	.000	.000
46	.933	.045	.000	.000	.000	.000
47	.933	.045	.000	.000	.000	.000
48	.933	.045	.000	.000	.000	.000
49	.933	.045	.000	.000	.000	.000
50	.933	.045	.000	.000	.000	.000

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= .94 INITIAL QH= .03 F= 10.000
LAMBDA 1=1.0000 LAMBDA 2= .5000

TIME	PH	QH	E11	E12	E21	E22
1	.940	.030	10.000	0.000	0.000	10.000
2	.940	.030	10.000	0.000	0.030	10.300
3	.940	.048	10.000	0.000	0.000	.000
4	.854	.040	.002	.000	.000	.000
5	.866	.048	.000	.000	.000	.000
6	.907	.047	.000	.000	.000	.000
7	.915	.046	.000	.000	.000	.000
8	.921	.046	.000	.000	.000	.000
9	.929	.045	.000	.000	.000	.000
10	.927	.046	.000	.000	.000	.000
11	.926	.046	.000	.000	.000	.000
12	.927	.046	.000	.000	.000	.000
13	.929	.045	.000	.000	.000	.000
14	.928	.045	.000	.000	.000	.000
15	.929	.045	.000	.000	.000	.000
16	.931	.045	.000	.000	.000	.000
17	.931	.045	.000	.000	.000	.000
18	.934	.045	.000	.000	.000	.000
19	.934	.045	.000	.000	.000	.000
20	.932	.045	.000	.000	.000	.000
21	.932	.045	.000	.000	.000	.000
22	.930	.045	.000	.000	.000	.000
23	.931	.045	.000	.000	.000	.000
24	.931	.045	.000	.000	.000	.000
25	.932	.045	.000	.000	.000	.000
26	.932	.045	.000	.000	.000	.000
27	.932	.045	.000	.000	.000	.000
28	.932	.045	.000	.000	.000	.000
29	.931	.045	.000	.000	.000	.000
30	.932	.045	.000	.000	.000	.000
31	.932	.045	.000	.000	.000	.000
32	.932	.045	.000	.000	.000	.000
33	.932	.045	.000	.000	.000	.000
34	.933	.045	.000	.000	.000	.000
35	.932	.045	.000	.000	.000	.000
36	.932	.045	.000	.000	.000	.000
37	.932	.045	.000	.000	.000	.000
38	.932	.045	.000	.000	.000	.000
39	.932	.045	.000	.000	.000	.000
40	.932	.045	.000	.000	.000	.000
41	.932	.045	.000	.000	.000	.000
42	.932	.045	.000	.000	.000	.000
43	.932	.045	.000	.000	.000	.000
44	.932	.045	.000	.000	.000	.000
45	.932	.045	.000	.000	.000	.000
46	.932	.045	.000	.000	.000	.000
47	.932	.045	.000	.000	.000	.000
48	.932	.045	.000	.000	.000	.000
49	.932	.045	.000	.000	.000	.000
50	.932	.045	.000	.000	.000	.000

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 Kp and an integral term ($K_i \Delta t Z/(Z-1)$) where the Δt term is the sampling frequency and Ki 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

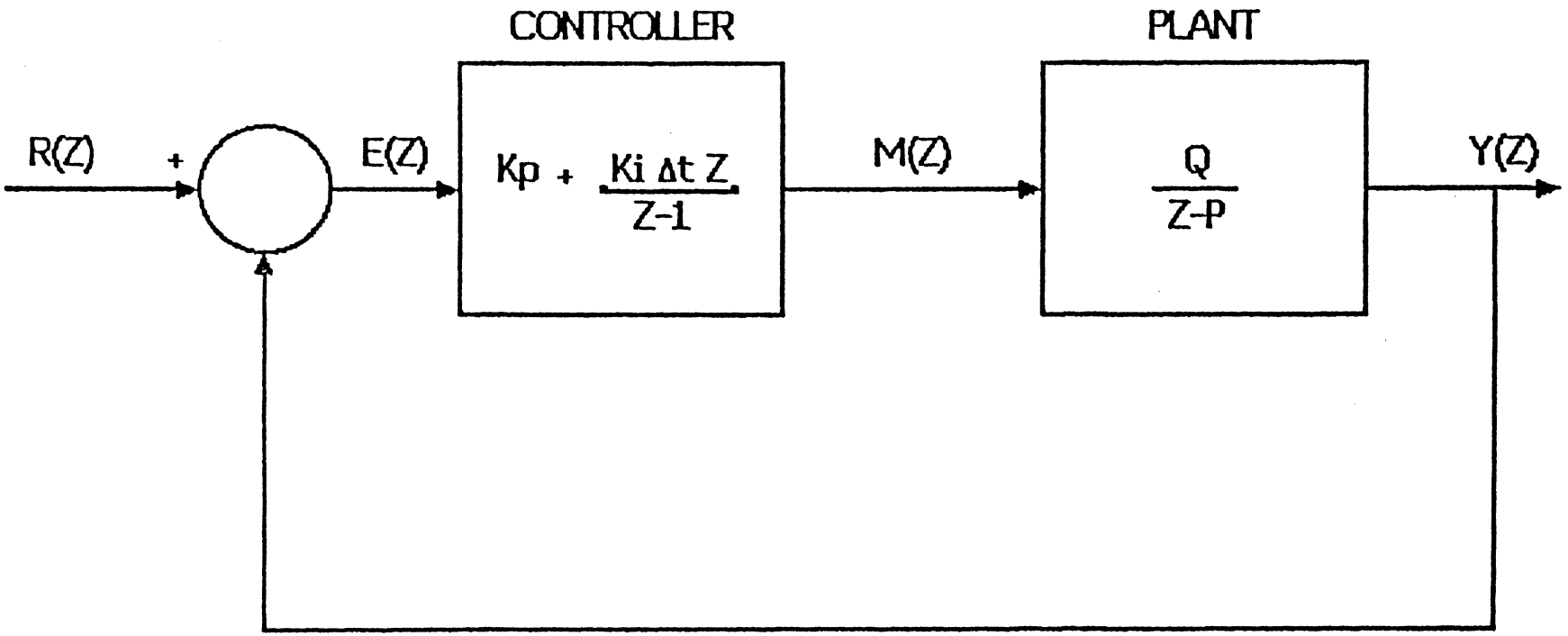


FIGURE 15
DIGITAL PROPORTIONAL - INTEGRAL (PI) CONTROLLER

value M(Z) is equal to the current error multiplied by a proportional term (Kp) 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 Kp and Ki 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 Kp and Ki.

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

$$\frac{Y(Z)}{R(Z)} = \frac{QZ(Kp + Ki\Delta t) - QKp}{Z^2 + (QKp + QKi\Delta t - 1 - P)Z + (P - QKp)} \qquad 28)$$

and the characteristic equation is given by:

$$Z^2 + (QKp+QKi\Delta t-1-P)Z + (P-QKp) = 0 \qquad 29)$$

This equation can be solved using the quadratic equation:

$$Z = -B/2 \pm 1/2\sqrt{B^2-4C} \qquad \text{where: } B=QKP+QKi\Delta t-1-P \quad C=P-QKp \qquad 30)$$

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 Δt 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 amout 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

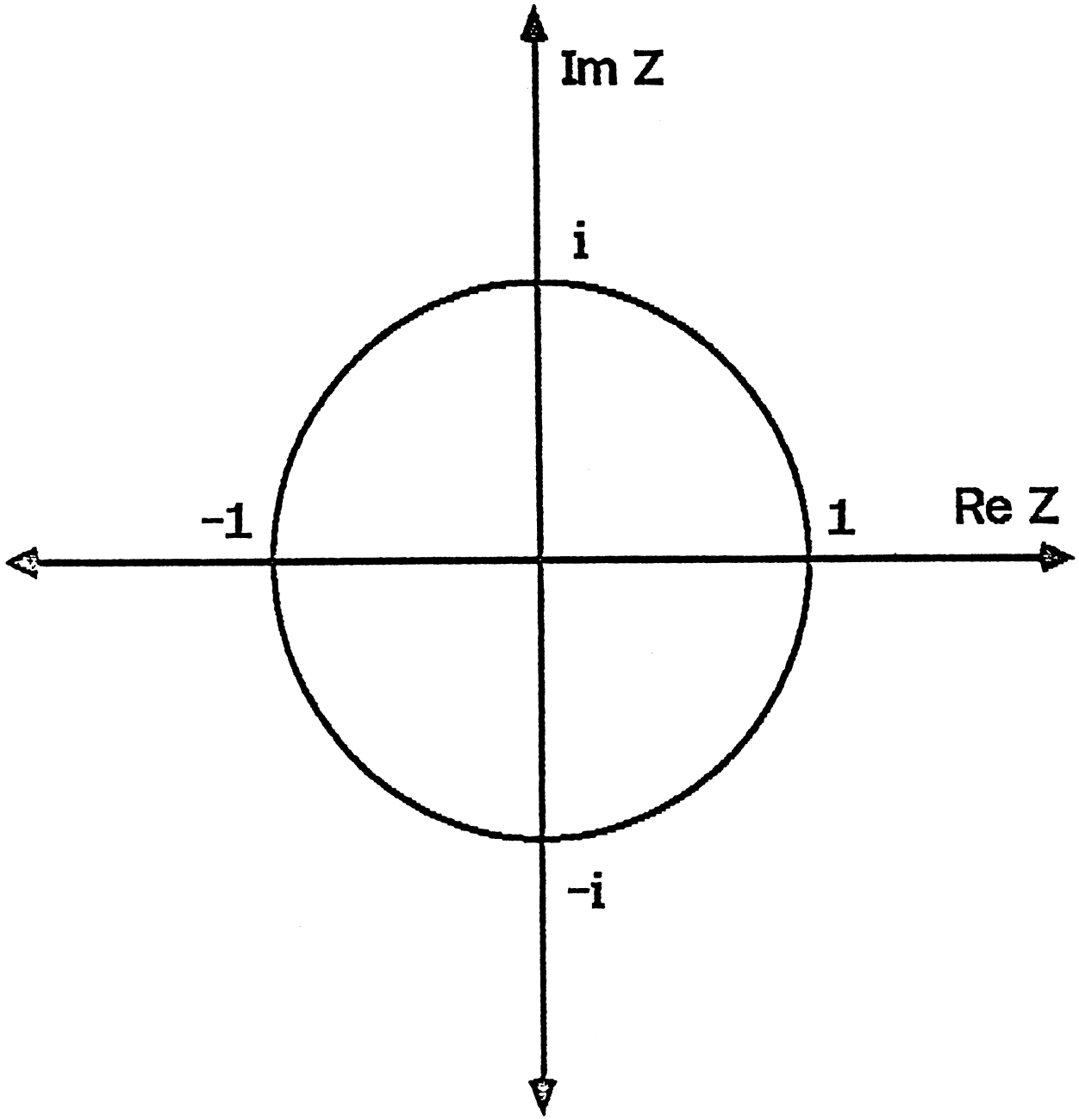


FIGURE 16
STABILITY UNIT CIRCLE IN Z-PLANE

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,

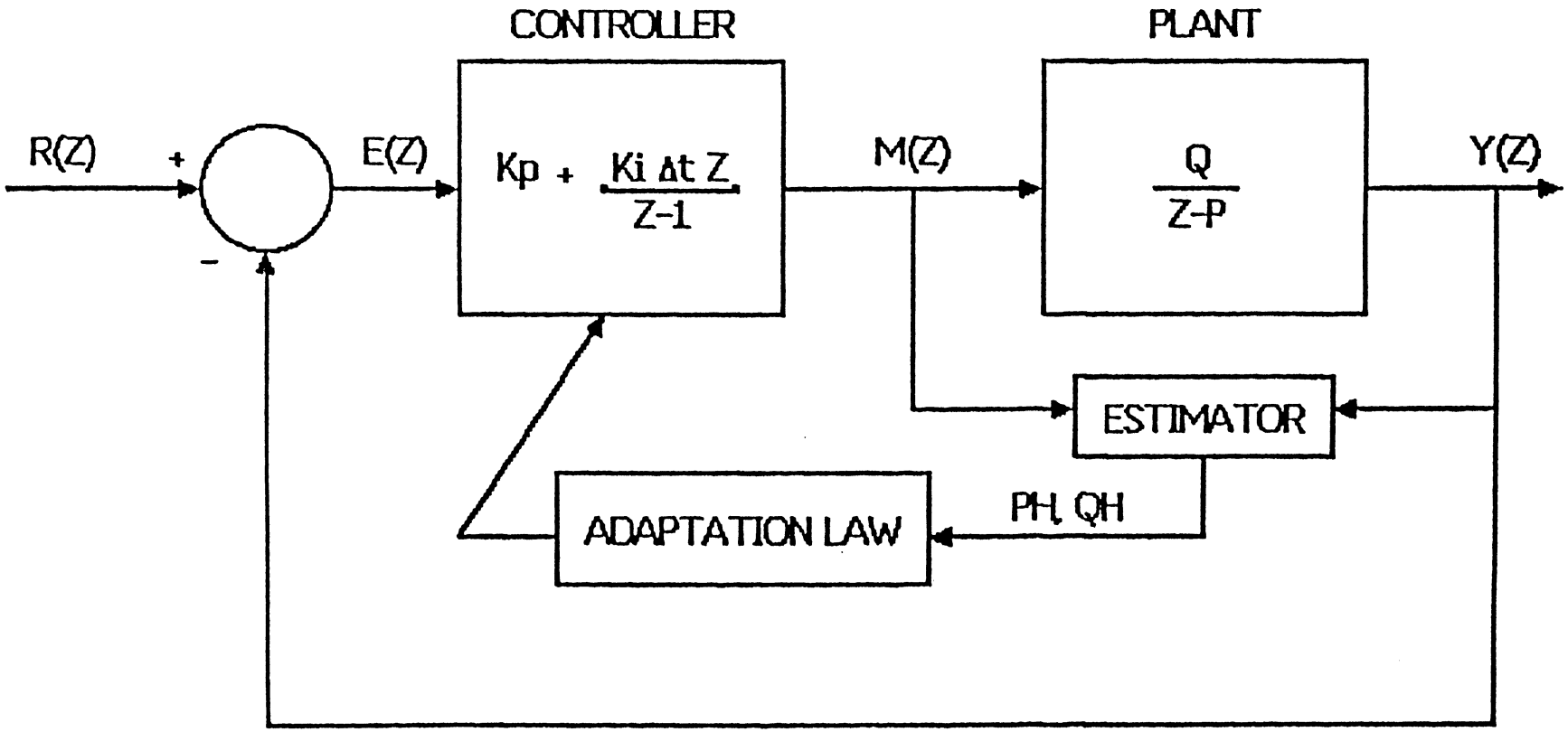


FIGURE 17
SELF - TUNING ADAPTIVE
DIGITAL PROPORTIONAL - INTEGRAL (PI) CONTROLLER

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:

$$M(K)=-k1Y(K)-k2Y(K+1) \qquad 31)$$

From equation 7 we know that $Y(K+1)=PY(K)+QM(K)$ Plugging this into equation 31 yields:

$$(1+Qk2)Y(K+1)=(P-Qk1)Y(K) \qquad 32)$$

Solving for the eigen value yields:

$$\pi=(P-Qk1)/(1+Qk2) \qquad 33)$$

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.

$d\pi/dP = (1+Qk2)^{-1}$ (this equals 1 for k2=0) 34)

$d\pi/dQ = -(K1+PK2)(1+QK2)^{-2}$ (this equals -k1 for K2=0) 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:

$$M(K)=-K1Y(K)-K2YH(K+1) \quad 36)$$

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:

$$M(K)=M(K-1)+(K2-K1\Delta t-Kp)Y(K)+K1\Delta tR(K)-K2YH(K+1)+KpY(K-1) \quad 37)$$

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

where: K3=K1Δt 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:

$$\frac{Y(Z)}{R(Z)} = \frac{K1\Delta tQZ}{Z^2(1+K2Q)+Z(QK1\Delta t-K2Q+KpQ-1-P)+(P-KpQ)} \quad 38)$$

And the characteristic equation is:

$$Z^2(1+K2Q) + Z(QK1\Delta t-K2Q+KpQ-1-P) + (P-KpQ) = 0 \quad 39)$$

Using the quadratic equation Z can be found by:

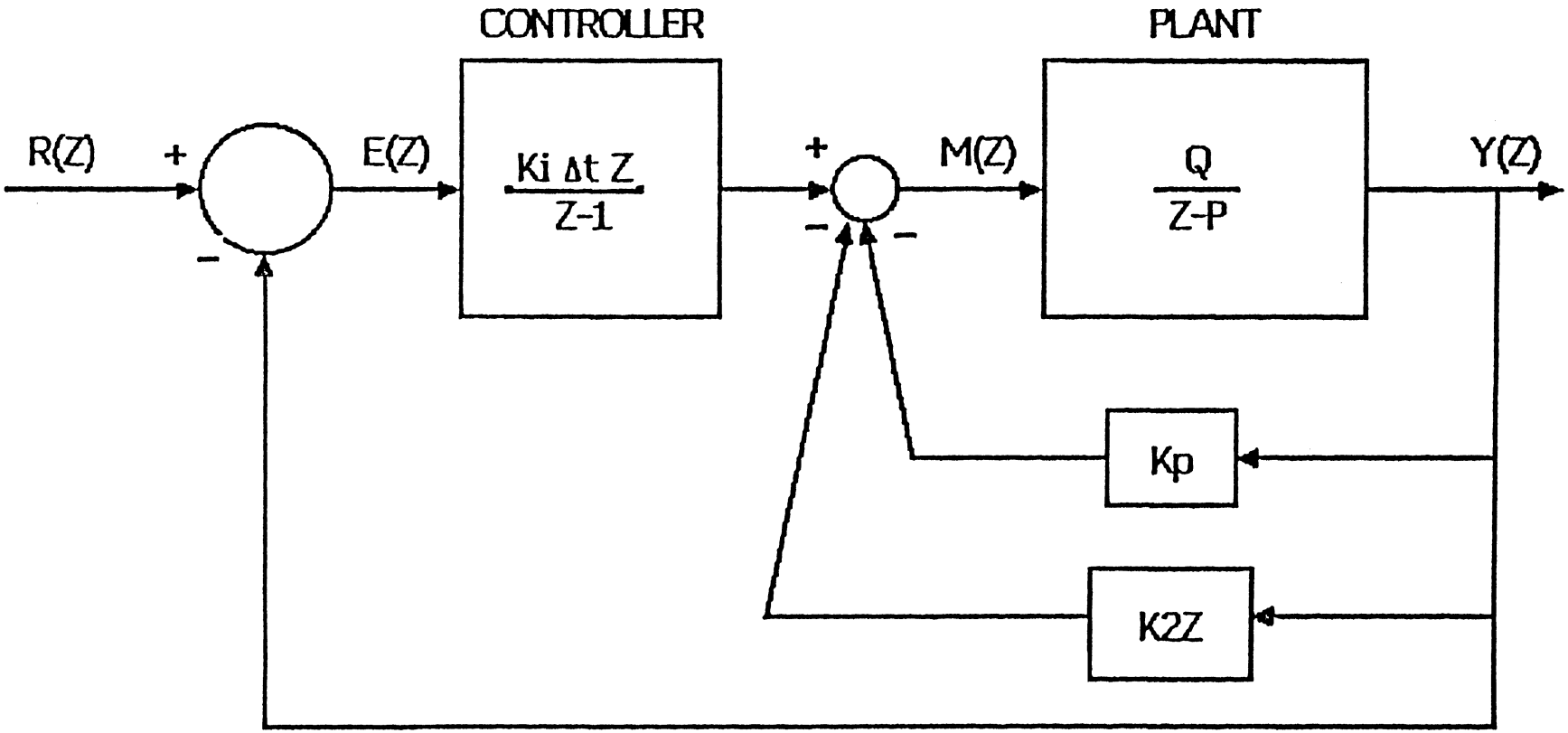


FIGURE 18
ROBUST FEEDBACK CONTROLLER

$$Z=-B/2\pm.5\sqrt{B^2-4C} \qquad \text{where:} \quad B=\frac{(QKi\Delta t-K2Q+KpQ-1-P)}{(1+K2Q)} \quad 40)$$

$$C=(P-KpQ)/(1+K2Q)$$

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

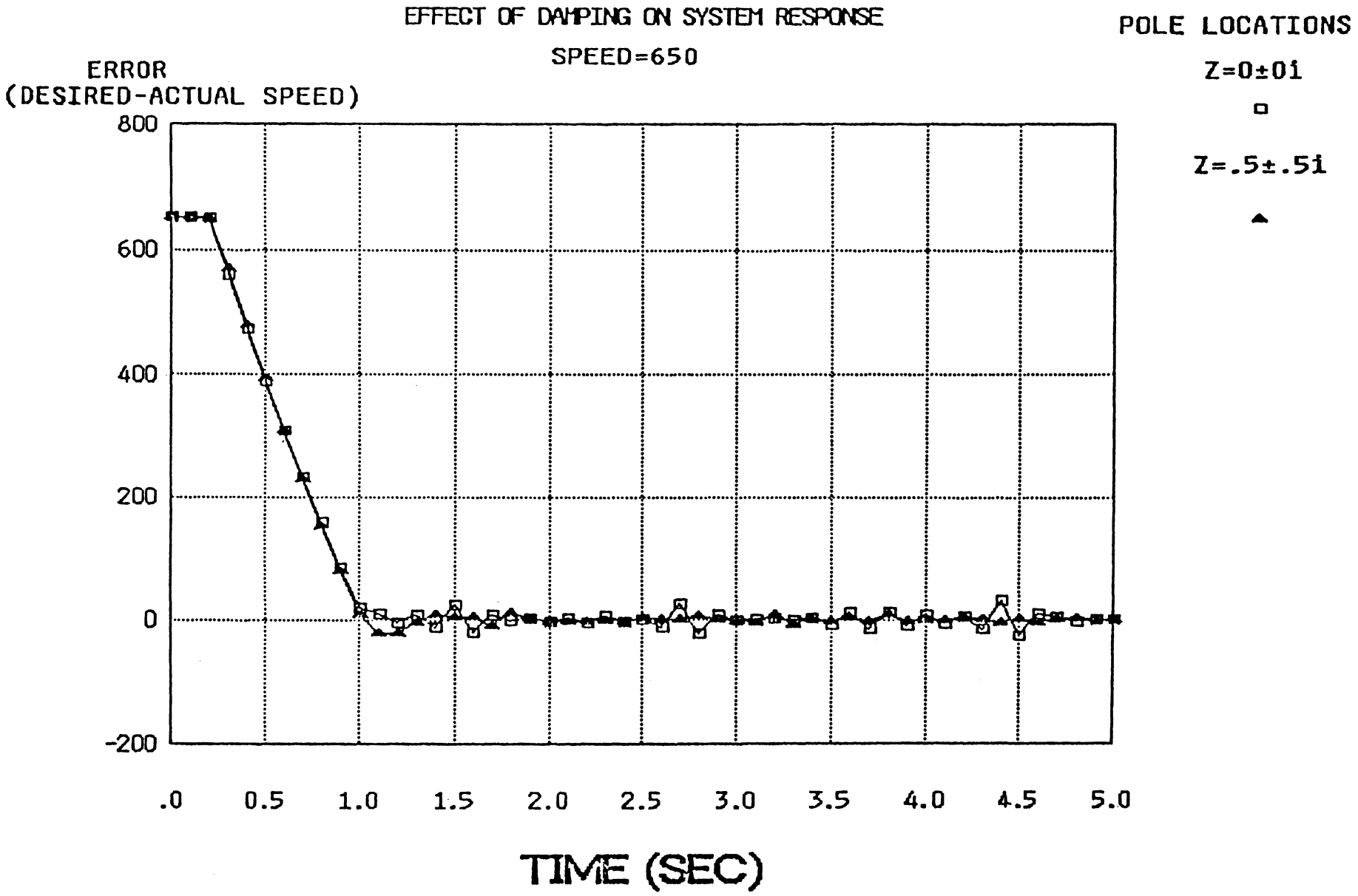
This section of the report analyzes the performance results of the three different types of controllers used in this study.

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 approximatly 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



DIGITAL PI CONTROLLER

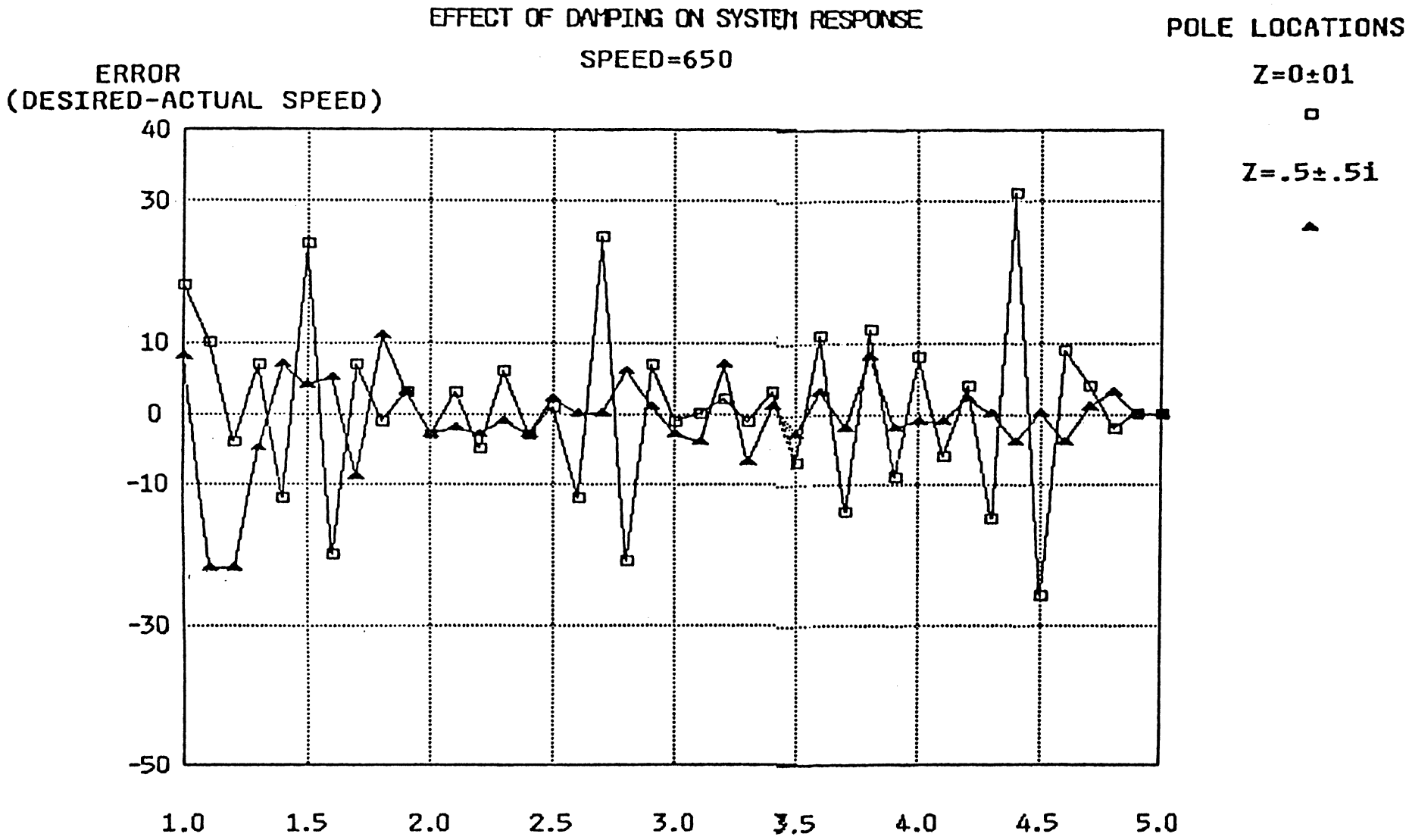


FIGURE 20

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±0i and using values of P and Q for a speed of 200 it can be found from equation 30 that Kp=50.0 and Ki=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

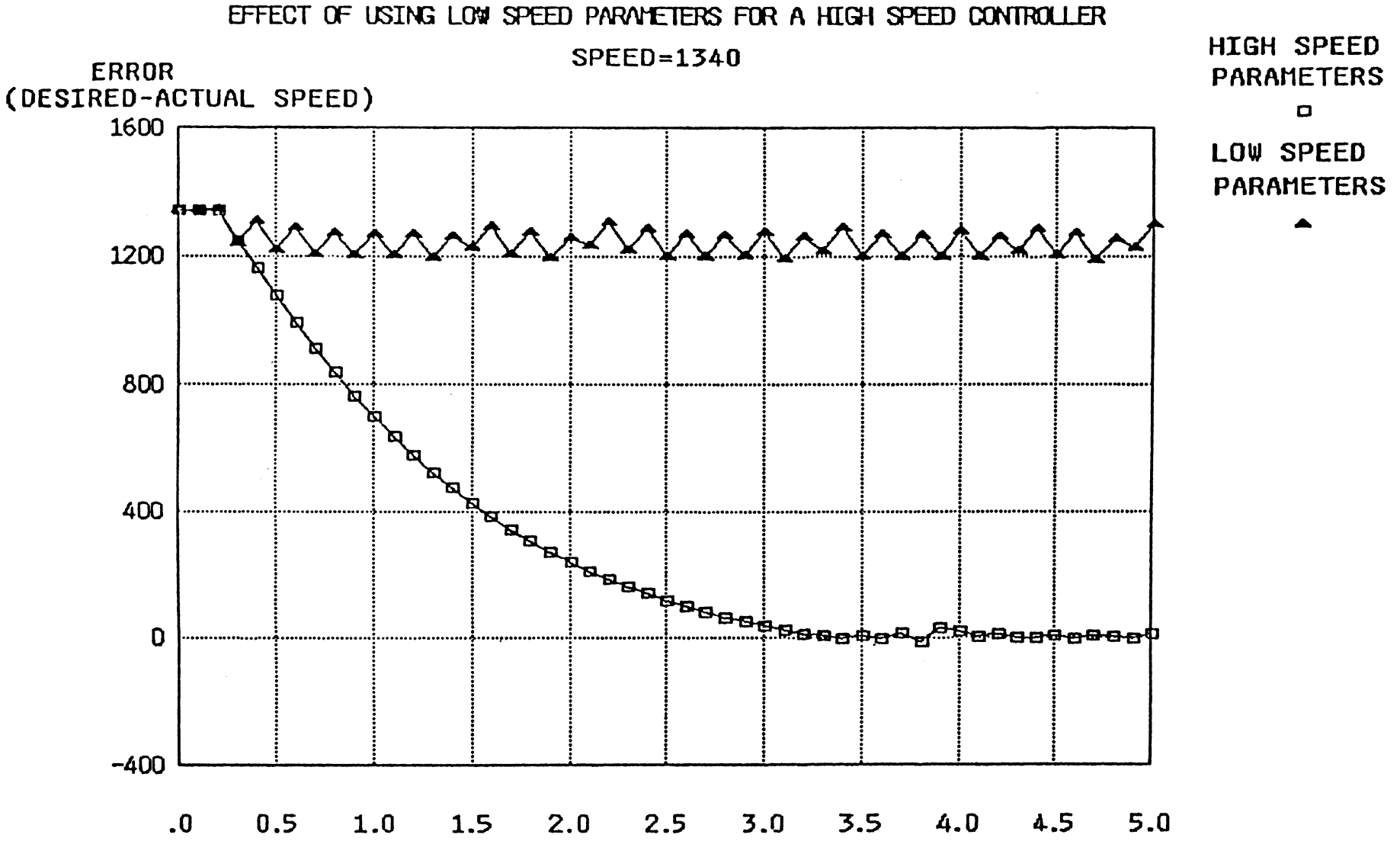


FIGURE 21

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 mention 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-Tunining 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

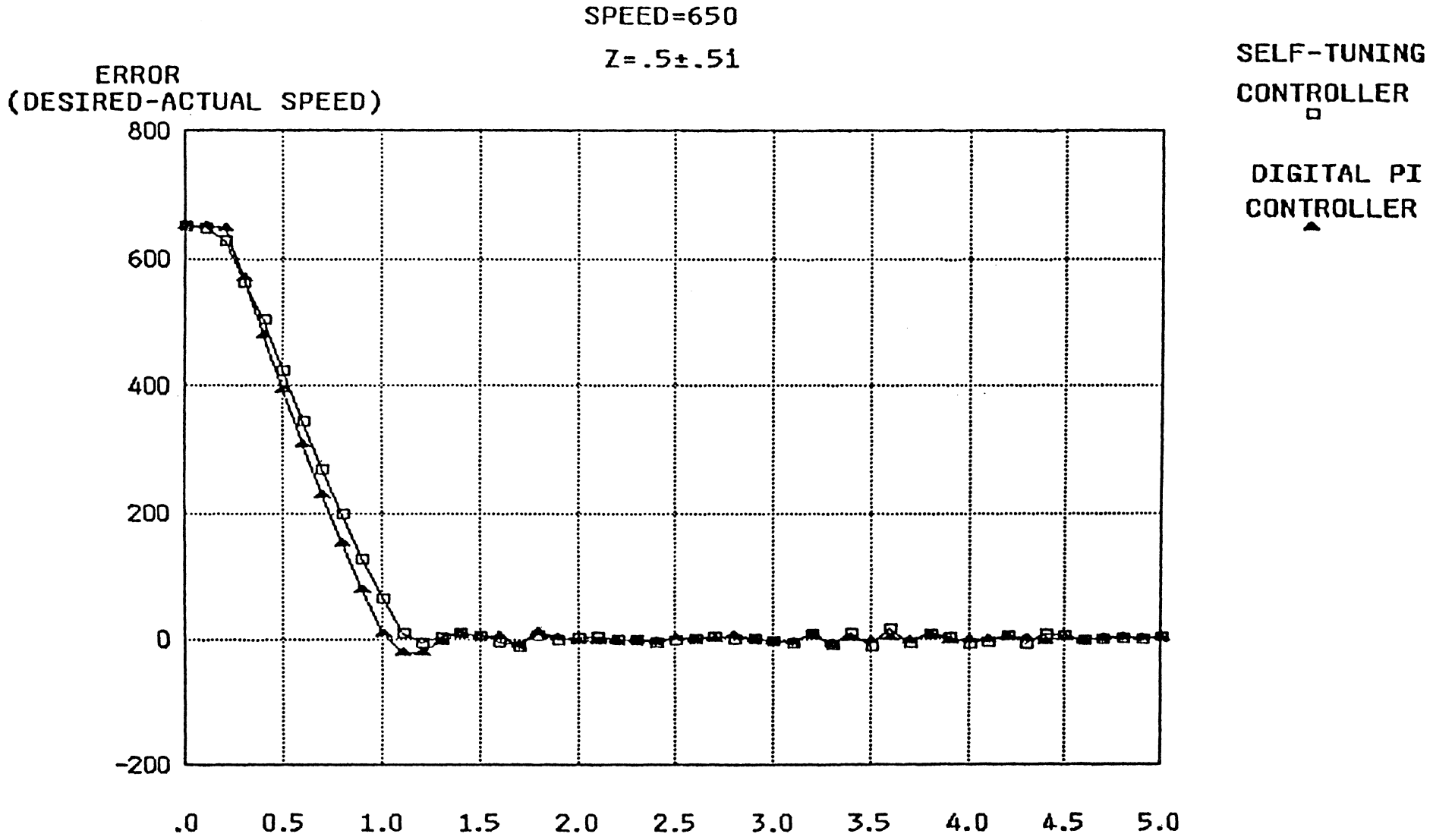
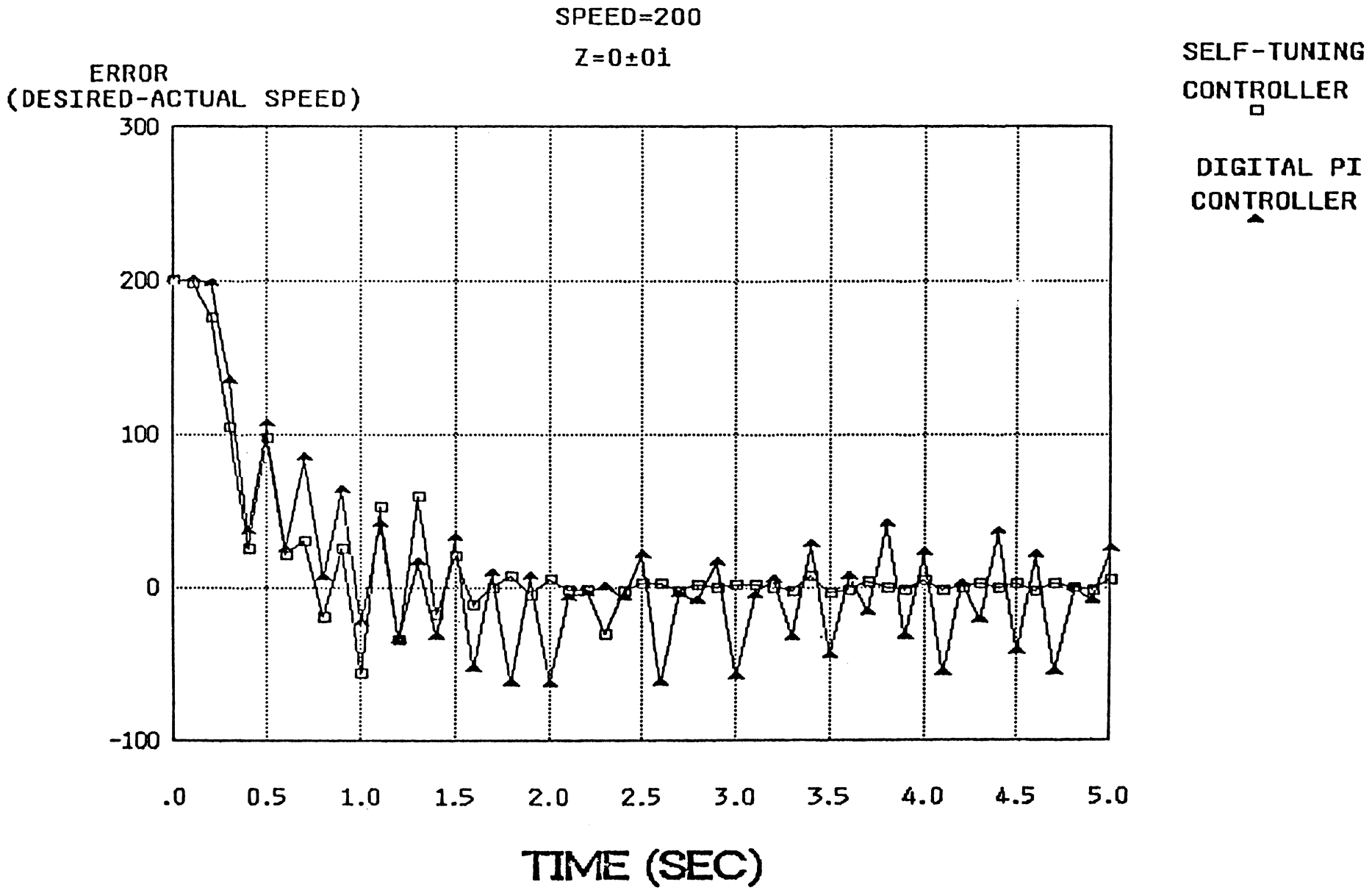


FIGURE 22

COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER



COMPARISON OF DIGITAL PI VS. SELF-TUNING CONTROLLER

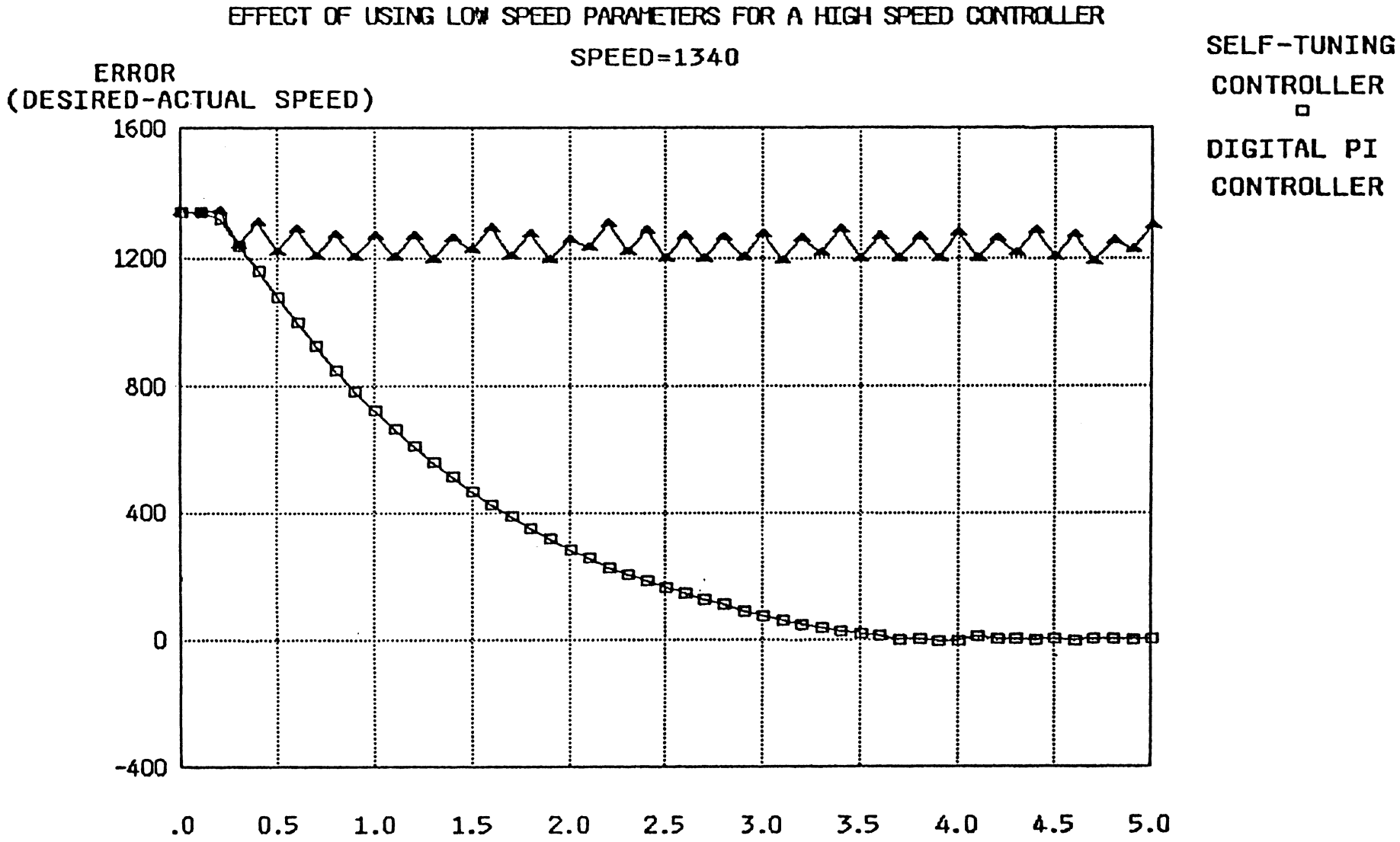


FIGURE 24

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 the 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

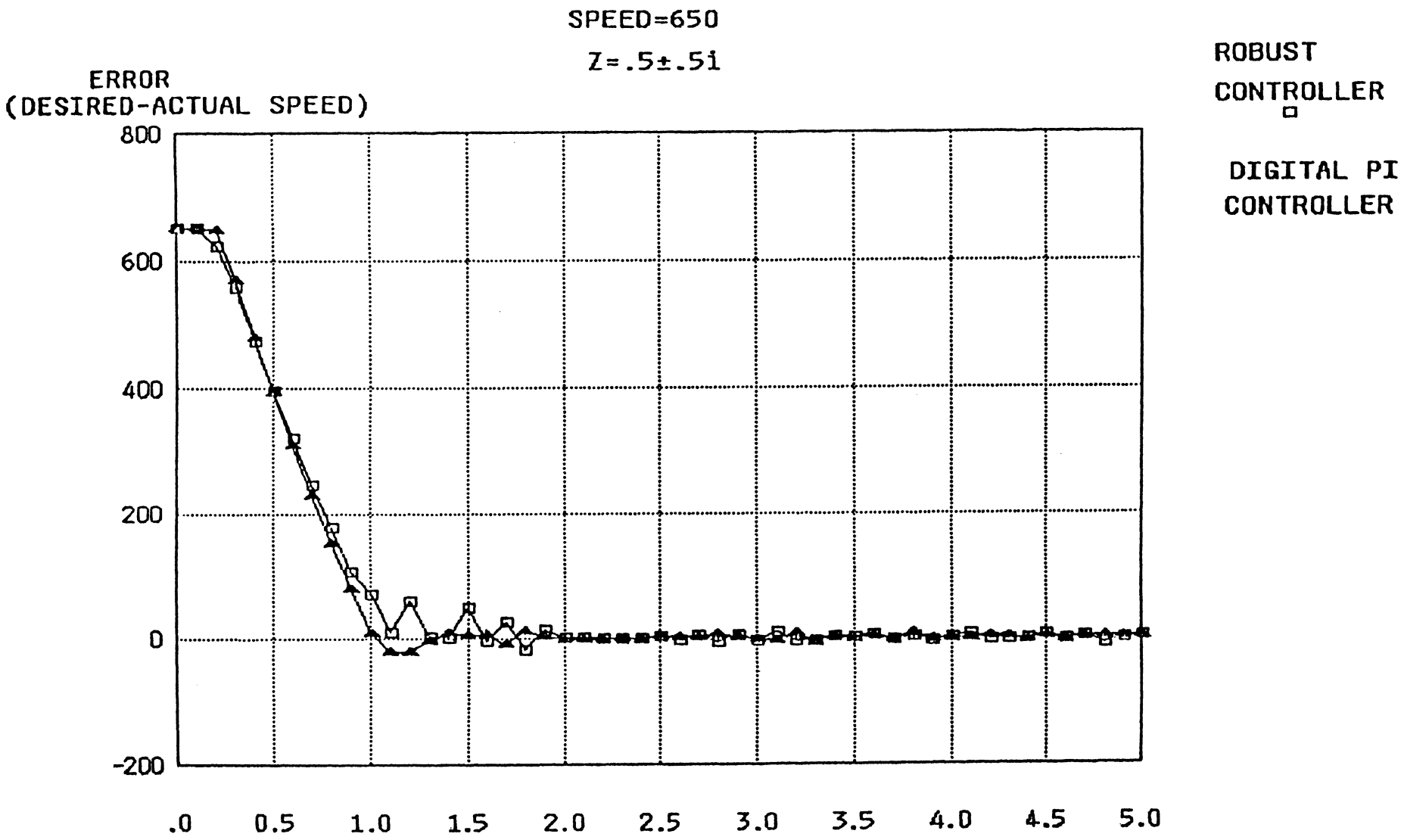
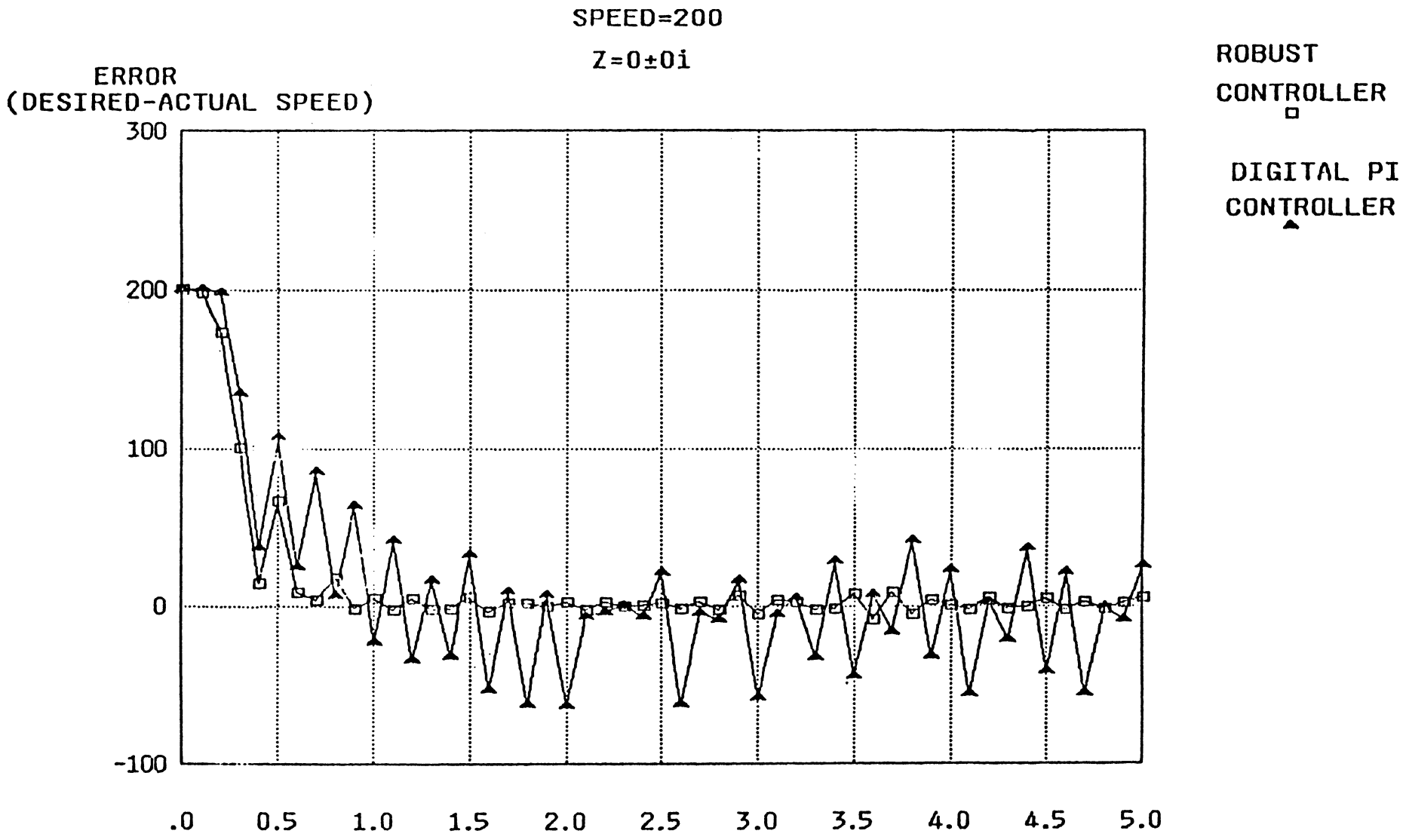


FIGURE 25

COMPARISON OF DIGITAL PI VS. ROBUST CONTROLLER



TIME (SEC)

FIGURE 26

SUMMARY AND CONCLUSIONS

This final section of the report summarizes the results for the three diferent 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

1. DYNAMICS OF PHYSICAL SYSTEMS, Robert H. Cannon Jr. 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

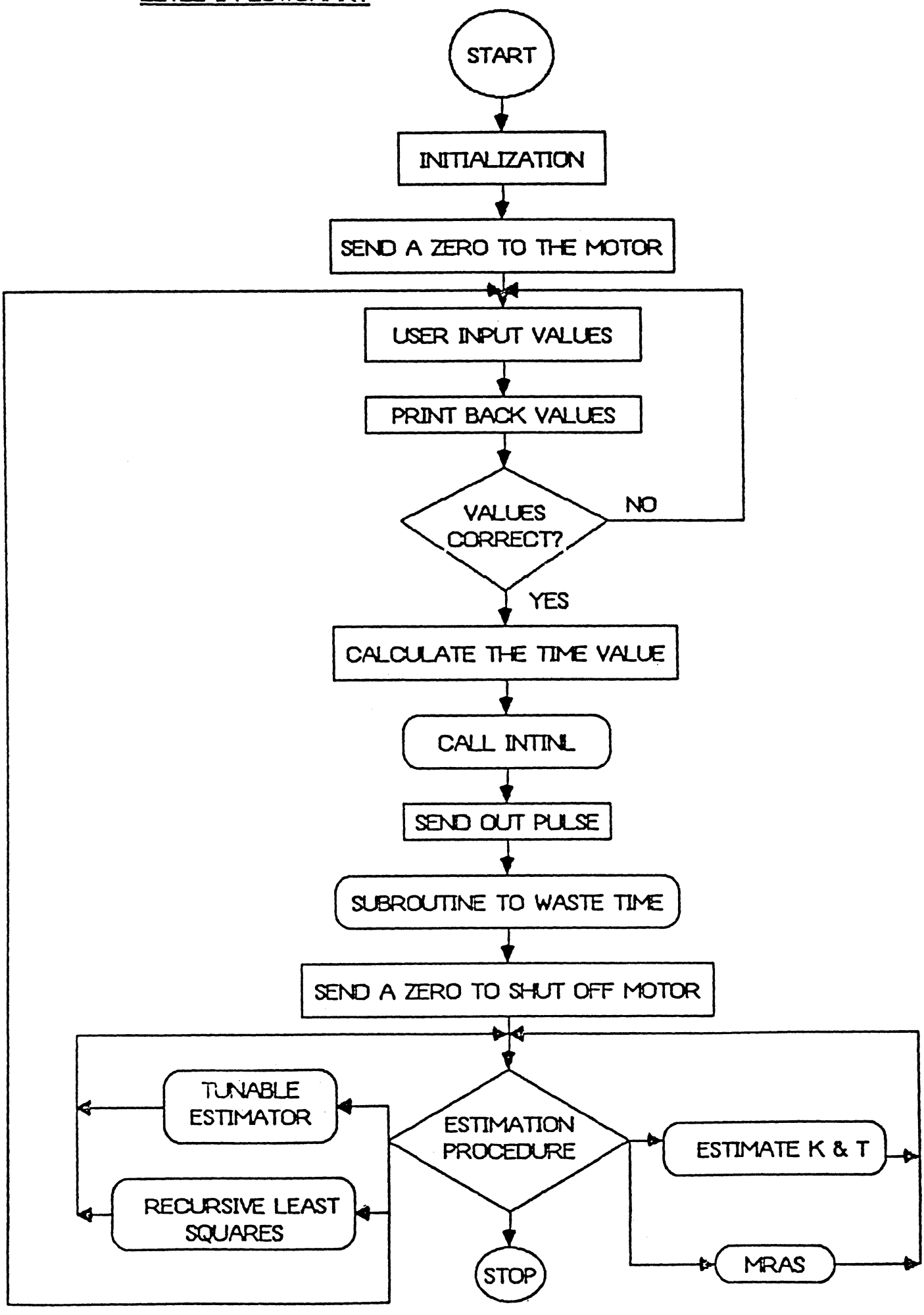
4. DIGITAL CONTROL, Rolf Isermann, Springer-Verlag, Berlin, Heidelberg, 1981

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

MAIN PROGRAM

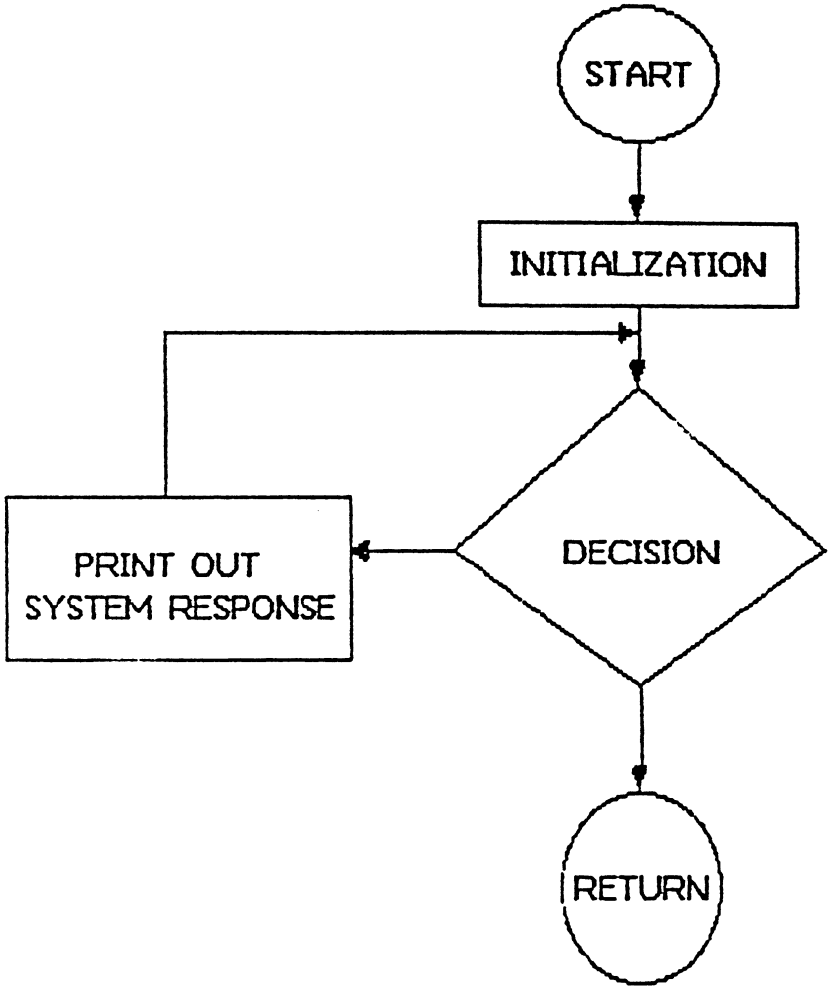


APPENDIX A

SYSTEM IDENTIFICATION PHASE FLOWCHARTS AND SOURCE CODE

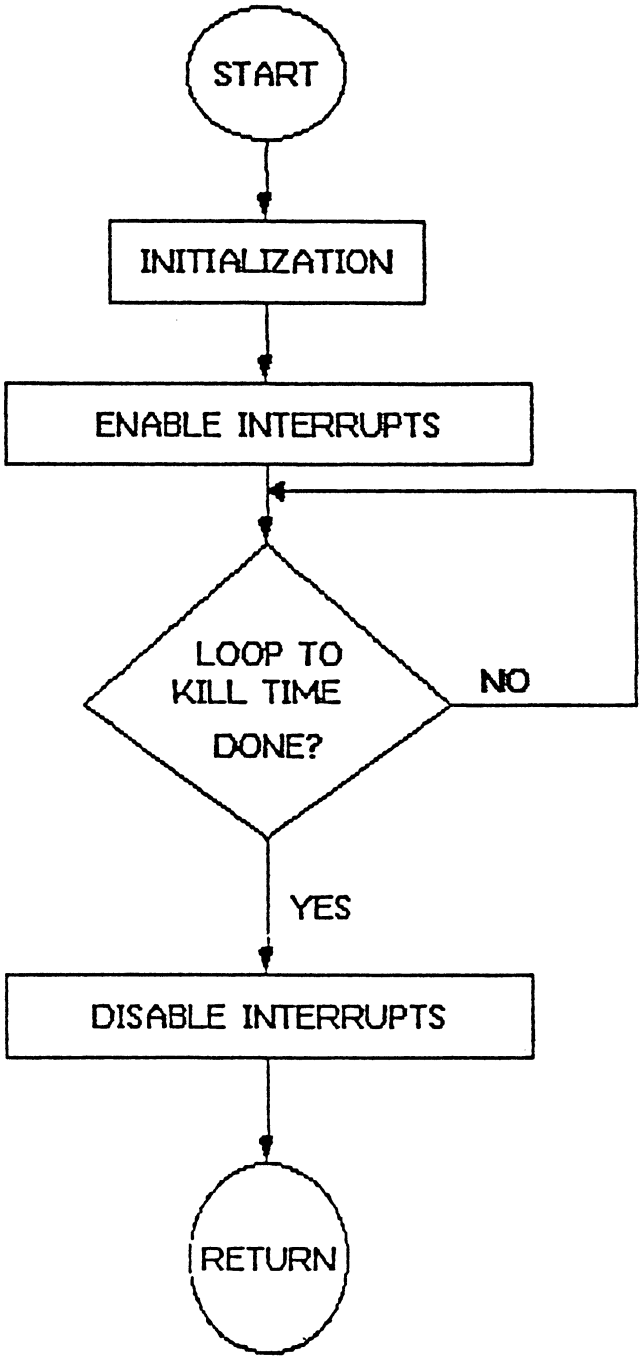
SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO ESTIMATE
OPEN-LOOP GAIN AND
TIME CONSTANT



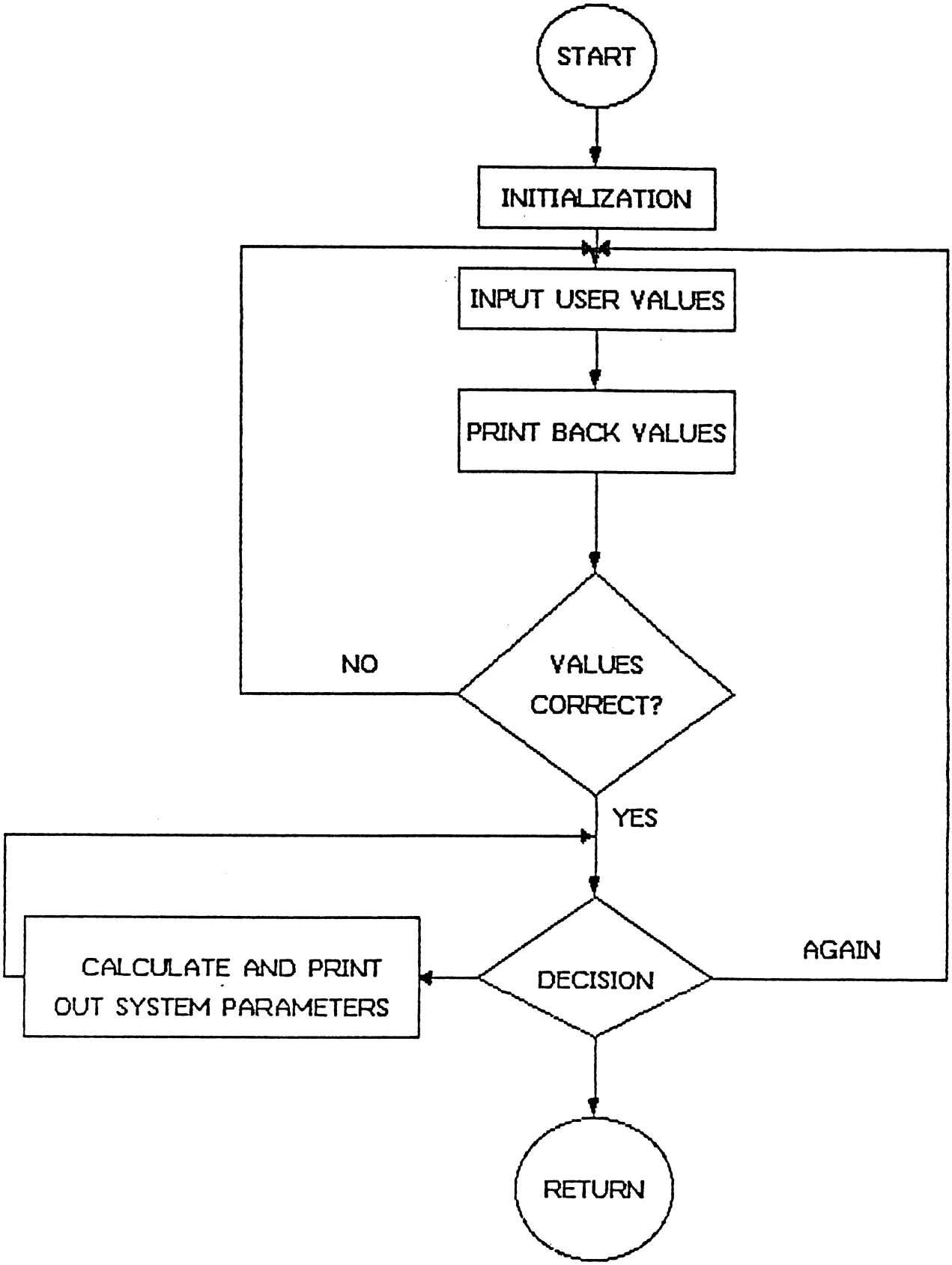
SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO
WASTE TIME



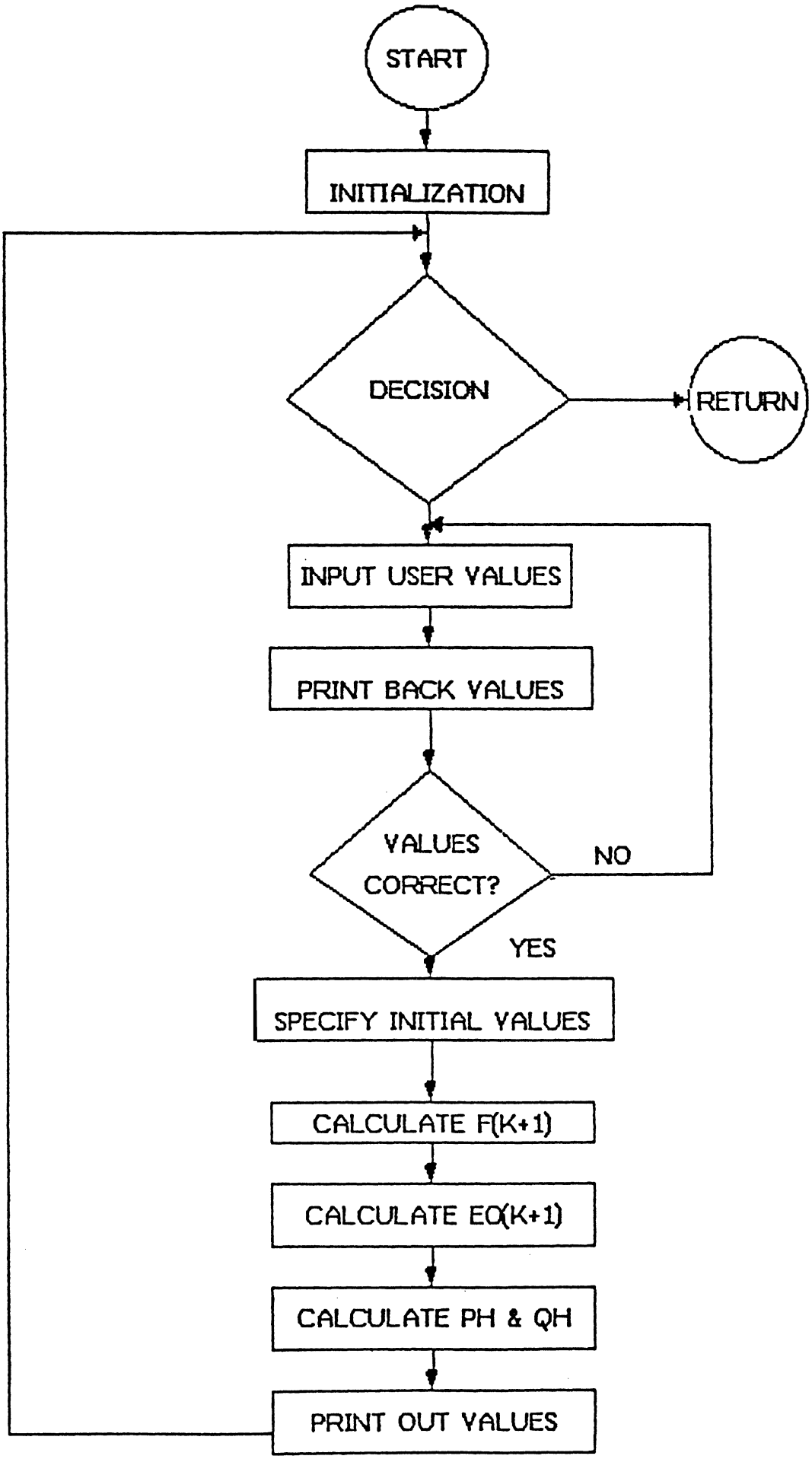
SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO ESSTIMATE
SYSTEM PARAMETERS
USING A MRAS



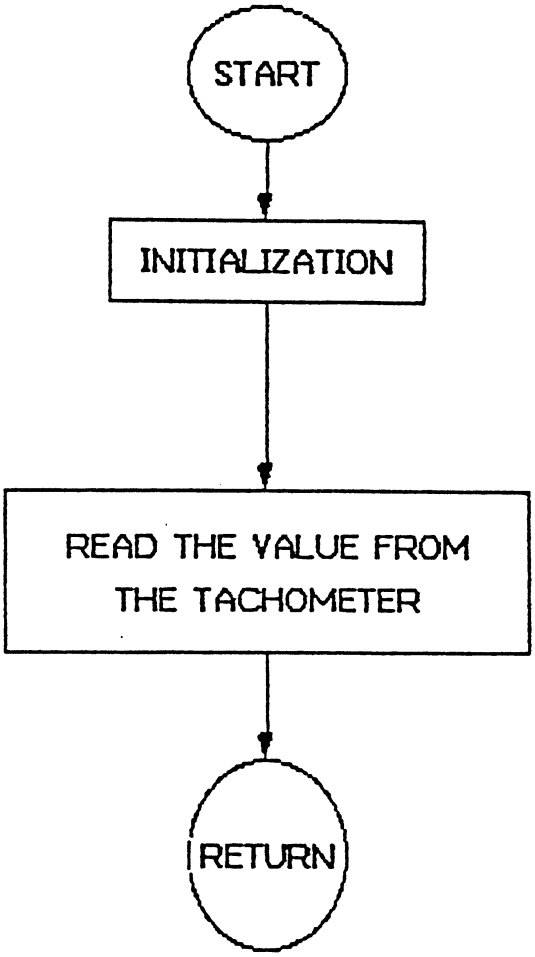
SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO ESTIMATE
SYSTEM PARAMETERS USING A
RECURSIVE LEAST SQUARES METHOD



SYSTEM IDENTIFICATION PHASE
LEVEL 3 FLOWCHART

INTERRUPT SERVICE ROUTINE
FOR DATA COLLECTION



```
C
C
C          ******************************************
C          *                                        *
C          *            ME600 PROJECT               *
C          * COMPARISON OF AUTOMATIC CONTROL SCHEMES *
C          *                                        *
C          *     SYSTEM IDENTIFICATION PHASE        *
C          *                                        *
C          *  RESEARCH ASSISTANT: DAN LYMBURNER     *
C          *  RESEARCH ADVISOR: PROF. G. ULSOY      *
C          *                                        *
C          ******************************************
C
C        THIS PROGRAM IS USED TO DETERMINE THE OPEN LOOP
C        TIME CONSTANT AND THE OPEN LOOP GAIN FOR A DC MOTOR SO
C        THAT DIGITAL PI AND PID CONTROL ALGORITHMS CAN BE IMPLEMENTED.
C        THIS PROGRAM ALSO USES A MODEL REFERENCE ADAPTIVE SYSTEM
C        (MRAS) SO THAT SYSTEM PARAMETERS CAN BE ESTIMATED IN ORDER
C        FOR AN ADAPTIVE CONTROL ALGORITHM TO BE IMPLEMENTED.  FINALLY,
C        THE PROGRAM USES A RECURSIVE LEAST SQUARES METHOD TO OBTAIN
C        ESTIMATES OF THE SYSTEM PARAMETERS.
C
C        THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
C        READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.
C
C
C ***************************************************************
C *                                                             *
C *                        MAIN PROGRAM                         *
C *                                                             *
C ***************************************************************
C
C
C          ****************************************
C          *          INITIALIZATION              *
C          ****************************************
C
      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/DACBLK/NDAC,DACHN
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
      COMMON/A2/ITIM
      ADCHN(1)=2
      DACHN(1)=0
      NADC=1
      NDAC=1
      IJ=0
C
C          ****************************************
C          *  SEND A ZERO TO THE MOTOR            *
C          ****************************************
C
      CALL DAC(0)
C
C          ****************************************
C          *        PROGRAM EXPLANATION           *
```

```
C          ****************************************
C
      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)
      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(I5)
C
C          ****************************************
C          *     INPUT THE USER VALUES            *
C          ****************************************
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
      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
C
C          ****************************************
C          *        PRINT BACK THE VALUES         *
C          ****************************************
C
      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 ')
C
C          ****************************************
C          *     ARE THE VALUES CORRECT?          *
C          ****************************************
C
C
```

```
0121          WRITE(3,90)
0122       90 FORMAT(/' ARE THE VALUES CORRECT?  (1=YES, 2=NO)  ')
0123          READ(2,15) J
0124          IF (J .EQ. 2) GO TO 40
0125    C
0126    C          ************************************
0127    C          *   CALCULATE THE TIME VALUES      *
0128    C          ************************************
0129    C
0130          TIM=530.*ISEC/15.
0131          ITIM=TIM
0132          L=ISEC*10.
0133          M=IR*10.
0134          N=L-M
0135          IU(1)=0
0136          DO 95 K=2,L
0137          IF (M .EQ. 0) GO TO 92
0138          IA=FLOAT(IMAX/M)+0.5
0139          IF (K .LT. M) GO TO 91
0140          IF (K .GT. N) GO TO 93
0141       92 IU(K)=IMAX
0142          GO TO 95
0143       91 IU(K)=IU(K-1)+IA
0144          GO TO 95
0145       93 IU(K)=IU(K-1)-IA
0146       95 CONTINUE
0147          DO 97 K=1,L
0148          WRITE(3,96) IU(K)
0149       96 FORMAT(//,15X,I5)
0150       97 CONTINUE
0151    C
0152    C          ************************************
0153    C          *   COLLECT DATA FROM THE MOTOR    *
0154    C          ************************************
0155    C
0156          WRITE(3,100)
0157      100 FORMAT(/' PRESS RETURN TO COLLECT THE DATA FROM THE MOTOR ')
0158          READ(2,15)JL
0159    C
0160    C          ************************************
0161    C          *  CALL SUBROUTINE TO INITIALIZE   *
0162    C          *        THE INTERRUPTS            *
0163    C          ************************************
0164    C
0165          CALL INTINL
0166    C
0167    C          ************************************
0168    C          *    CALL SUBROUTINE TO WASTE TIME *
0169    C          ************************************
0170    C
0171          CALL WASTE
0172    C
0173    C          ************************************
0174    C          * SEND A ZERO TO SHUT OFF THE MOTOR *
0175    C          ************************************
0176    C
0177          CALL DAC(0)
0178    C
0179    C          ************************************
0180    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 ****************************************************
C *                                                  *
C *         SUBROUTINE TO WASTE TIME WHILE           *
C *                  INTERRUPTS OCCUR                *
C *                                                  *
C ****************************************************
C
      SUBROUTINE WASTE
      COMMON/ISRBLK/IJ
      COMMON/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
```

```
0241   C
0242   C
0243   C ***************************************************
0244   C *                                                 *
0245   C *                 INTERRUPT SERVICE               *
0246   C *         ROUTINE FOR DATA COLLECTION             *
0247   C *                                                 *
0248   C ***************************************************
0249   C
0250   C
0251         SUBROUTINE ISR
0252         DIMENSION ADCVAL(300), ADCHN(8), DACHN(4), IU(300)
0253         INTEGER ADCVAL,DACVAL
0254         LOGICAL*1 NADC,NDAC,ADCHN,DACHN
0255         COMMON/ISRBLK/IJ
0256         COMMON/ADCBLK/NADC,ADCHN
0257         COMMON/DACBLK/NDAC,DACHN
0258         COMMON/A1/ADCVAL,IU,ISEC,IMAX
0259   C
0260   C          ****************************************
0261   C          *  GET THE VALUE FROM THE TACHOMETER   *
0262   C          ****************************************
0263   C
0264         IJ=IJ+1
0265         CALL ADC(ADCVAL(IJ))
0266         CALL DAC(IU(IJ))
0267         RETURN
0268         END
0269   C
0270   C
0271   C ***************************************************************
0272   C *                                                             *
0273   C *      SUBROUTINE TO ESTIMATE OPEN-LOOP GAIN AND              *
0274   C *                 OPEN-LOOP TIME CONSTANT                     *
0275   C *                                                             *
0276   C ***************************************************************
0277   C
0278   C          ************************************
0279   C          *          INITIALIZATION          *
0280   C          ************************************
0281   C
0282         SUBROUTINE KTAU
0283         DIMENSION ADCVAL(300), IU(300)
0284         INTEGER ADCVAL
0285         COMMON/A1/ADCVAL,IU,ISEC,IMAX
0286         IDT=ISEC*10.
0287   C
0288   C          ************************************
0289   C          *          DECISION                *
0290   C          ************************************
0291   C
0292       149 WRITE(3,151)
0293       151 FORMAT(/' YOU HAVE ENTERED THE SUBROUTINE TO DETERMINE THE '/
0294          C' OPEN-LOOP GAIN AND TIME CONSTANT. YOU MAY NOW: '/
0295          C'        1=RETURN TO THE MAIN PROGRAM '/
0296          C'        2=PRINT OUT THE SYSTEM RESPONSE FOR THIS PROCEDURE '/
0297          C'        ENTER 1 OR 2  ')
0298       152 FORMAT(I1)
0299         READ(2,152)LJ
0300         IF (LJ .EQ. 1) GO TO 190
```

```
      IF (LJ .EQ. 2) GO TO 153
      GO TO 149
C
C          ****************************************
C          *  PRINT OUT THE SYSTEM RESPONSE       *
C          ****************************************
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)
  160 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*****************************************************************
C *                                                               *
C *           SUBROUTINE TO OBTAIN PARMETER ESTIMATES             *
C *         USING A MODEL REFERENCE ADAPTIVE SYSTEM (MRAS)        *
C *                                                               *
C*****************************************************************
C
C          ****************************************
C          *              INITIALIZATION          *
C          ****************************************
C
      SUBROUTINE MRAS
      DIMENSION ADCVAL(300), IU(300)
      DIMENSION X(300), XHO(300), XH(300), PH(300), QH(300)
      DIMENSION EO(300), DEN(300), E(300), ER(300)
      INTEGER ADCVAL,DACVAL
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
C
C          ****************************************
C          *     USER INPUT VALUES                *
C          ****************************************
C
      WRITE(3,200)
  200 FORMAT(/' YOU HAVE ENTEED 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)
```

```
0361     271 FORMAT(/' INPUT THE INITIAL VALUE OF THE COMPUTER MODEL '/
0362        C' PARAMETER "PH". INCLUDE A PERIOD   ')
0363         READ(2,210) APH
0364         WRITE(3,272)
0365     272 FORMAT(/' INPUT THE INITIAL VALUE OF THE COMPUTER MODEL '/
0366        C' PARAMETER "QH".  INCLUDE A PERIOD   ')
0367         READ(2,210) AQH
0368         WRITE(3,273)
0369     273 FORMAT(/' INPUT THE INITIAL VALUE OF "XH".  INCLUDE A PERIOD ')
0370         READ(2,210) XA
0371   C
0372   C          ************************************
0373   C          *        PRINT BACK THE VALUES       *
0374   C          ************************************
0375   C
0376         WRITE(3,275) AK1
0377     275 FORMAT(/' THE GAIN K1 IS ',F8.2 )
0378         WRITE(3,280) AK2
0379     280 FORMAT(/' THE GAIN K2 IS ',F8.2 )
0380         WRITE(3,301) APH
0381     301 FORMAT(/' THE INITIAL VALUE OF "PH" IS',F7.4)
0382         WRITE(3,302) AQH
0383     302 FORMAT(/' THE INITIAL VALUE OF "QH" IS',F7.4)
0384         WRITE(3,303) XA
0385     303 FORMAT(/' THE INITIAL VALUE OF "XH" IS',F7.4)
0386   C
0387   C          ************************************
0388   C          *  ARE THESE THE CORRECT VALUES?   *
0389   C          ************************************
0390   C
0391         WRITE(3,310)
0392     310 FORMAT(/' ARE THESE THE CORRECT VALUES? (1=YES, 2=NO)  ')
0393     320 FORMAT(I1)
0394         READ(2,320) LP
0395         IF (LP .EQ. 2) GO TO 215
0396   C
0397   C          ************************************
0398   C          *           DECISION               *
0399   C          ************************************
0400   C
0401     321 WRITE(3,322)
0402     322 FORMAT(/' YOU ARE NOW READY TO OBTAIN PARAMETER ESTIMATES  '/
0403        C' DO YOU WISH TO:    '/
0404        C'          1=RETURN TO THE MAIN PROGRAM  '/
0405        C'          2=REPEAT THIS ESTIMATE PROCEDURE   '/
0406        C'          3=PRINT OUT THE ESTIMATES FOR THIS PROCEDURE   ')
0407     323 WRITE(3,324)
0408     324 FORMAT(' ENTER 1,2, OR 3  ')
0409         READ(2,320)LJ
0410         IF (LJ .EQ. 1) GO TO 360
0411         IF (LJ .EQ. 2) GO TO 215
0412         IF (LJ .EQ. 3) GO TO 325
0413         GO TO 323
0414   C
0415   C          ****************************************
0416   C          *  SPECIFY INITIAL VALUE OF THE RESPONSE *
0417   C          *        AND COMPUTER MODEL PARAMETERS   *
0418   C          ****************************************
0419   C
0420     325 XH(1)=XA
```

```
      PH(1)=APH
      QH(1)=AQH
C
C          ****************************************
C          *   CALCULATE COMPUTER MODEL PARAMETERS  *
C          ****************************************
C
      IOT=ISEC*10.
      DO 330 K=1,IOT
      X(K)=ADCVAL(K)
      X(K+1)=ADCVAL(K+1)
      XHO(K+1)=PH(K)*XH(K)+QH(K)*IU(K)
      EO(K+1)=X(K+1)-XHO(K+1)
      DEN(K)=(1.+AK1*XH(K)**2.)+(AK2*(IU(K))**2.)
      E(K+1)=EO(K+1)/DEN(K)
      PH(K+1)=PH(K)+AK1*E(K+1)*XH(K)
      QH(K+1)=QH(K)+AK2*E(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
C
C          ****************************************
C          *  PRINT OUT THE SYSTEM RESULTS        *
C          ****************************************
C
      WRITE(1,335)
  335 FORMAT(//,15X,' ESTIMATES OF PH AND QH USING A MRAS')
      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)
  339 FORMAT(//,3X,'TIME',7X,'PH',8X,'QH',7X,'X',9X,'XH',6X,'ERROR')
      DO 350 K=1,IOT
      WRITE(1,340) K,PH(K),QH(K),X(K),XH(K),ER(K)
  340 FORMAT(3X,I3,2X,F8.3,2X,F8.3,2X,F8.1,2X,F8.1,2X,F8.1)
  350 CONTINUE
      WRITE(1,355)
  355 FORMAT(/,'       PRINT OUT COMPLETE  ')
      GO TO 321
  360 RETURN
      END
C
C ***************************************************************
C *                                                             *
C *   SUBROUTINE FOR PARAMETER ESTIMATION UTILIZING A           *
C *        RECURSIVE LEAST SQUARES METHOD                       *
C *                                                             *
C ***************************************************************
C
C
      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 ADCVAL
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
      IOT=ISEC*10.
C
```

```
C          ****************************************
C          *          DECISION                    *
C          ****************************************
C
  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
C
C          ****************************************
C          *       USER INPUT VALUES              *
C          ****************************************
C
  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
C
C          ****************************************
C          *          PRINT BACK THE VALUES       *
C          ****************************************
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)
C
C          ****************************************
C          *              VALUES CORRECT?         *
C          ****************************************
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          ****************************************
C          *  SPECIFY INITIAL VALUES OF THE       *
C          *  COMPUTER MODEL PARAMETERS           *
C          ****************************************
C
      PH(1)=P
```

```
      THET1(1)=-P
      THET2(1)=Q
      F11(1)=F
      F12(1)=F
      F21(1)=F
      F22(1)=F
      YH(1)=0.
C
C          ****************************************
C          *         ESTIMATION PROCEDURE         *
C          ****************************************
C
      WRITE(3,420)
  420 FORMAT(/' STARTING RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE')
      DO 425 K=1,IOT
      YH(K+1)=(PH(K)*YH(K))+(THET2(K)*IU(K))
C
C          *     CALCULATE F(K+1)         *
C
      C11(K)=(F11(K)*(-YH(K))+F12(K)*IU(K))*(-YH(K))
      C12(K)=(F11(K)*(-YH(K))+F12(K)*IU(K))*IU(K)
      C21(K)=(F21(K)*(-YH(K))+F22(K)*IU(K))*(-YH(K))
      C22(K)=(F21(K)*(-YH(K))+F22(K)*IU(K))*IU(K)
      DEN(K)=C11(K)+C22(K)+1.
C
      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)*F22(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)
C
C        *     CALCULATE EO(K+1)     *
C
      EO(K+1)=ADCVAL(K+1)+THET1(K)*(YH(K))-THET2(K)*IU(K)
C
C        *      CALCULATE THETA HAT AT K+1     *
C
      THET1(K+1)=THET1(K)+(F11(K+1)*(-YH(K))+F12(K+1)*IU(K))*EO(K+1)
      THET2(K+1)=THET2(K)+(F21(K+1)*(-YH(K))+F22(K+1)*IU(K))*EO(K+1)
C
C        *  CALCULATE PH AND QH     *
C
      PH(K+1)=-THET1(K+1)
      QH(K+1)=THET2(K+1)
C
C          ****************************************
C          *           WRITE OUT THE VALUES       *
C          ****************************************
C
  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,Q,F
  460 FORMAT(11X,'INITIAL PH=',F4.2,'  INITIAL QH=',F4.2,'  F=',F8.3)
      WRITE(1,470)
  470 FORMAT(//,20X,'TIME',7X,'PH',8X,'QH')
      DO 490 K=1,IOT
```

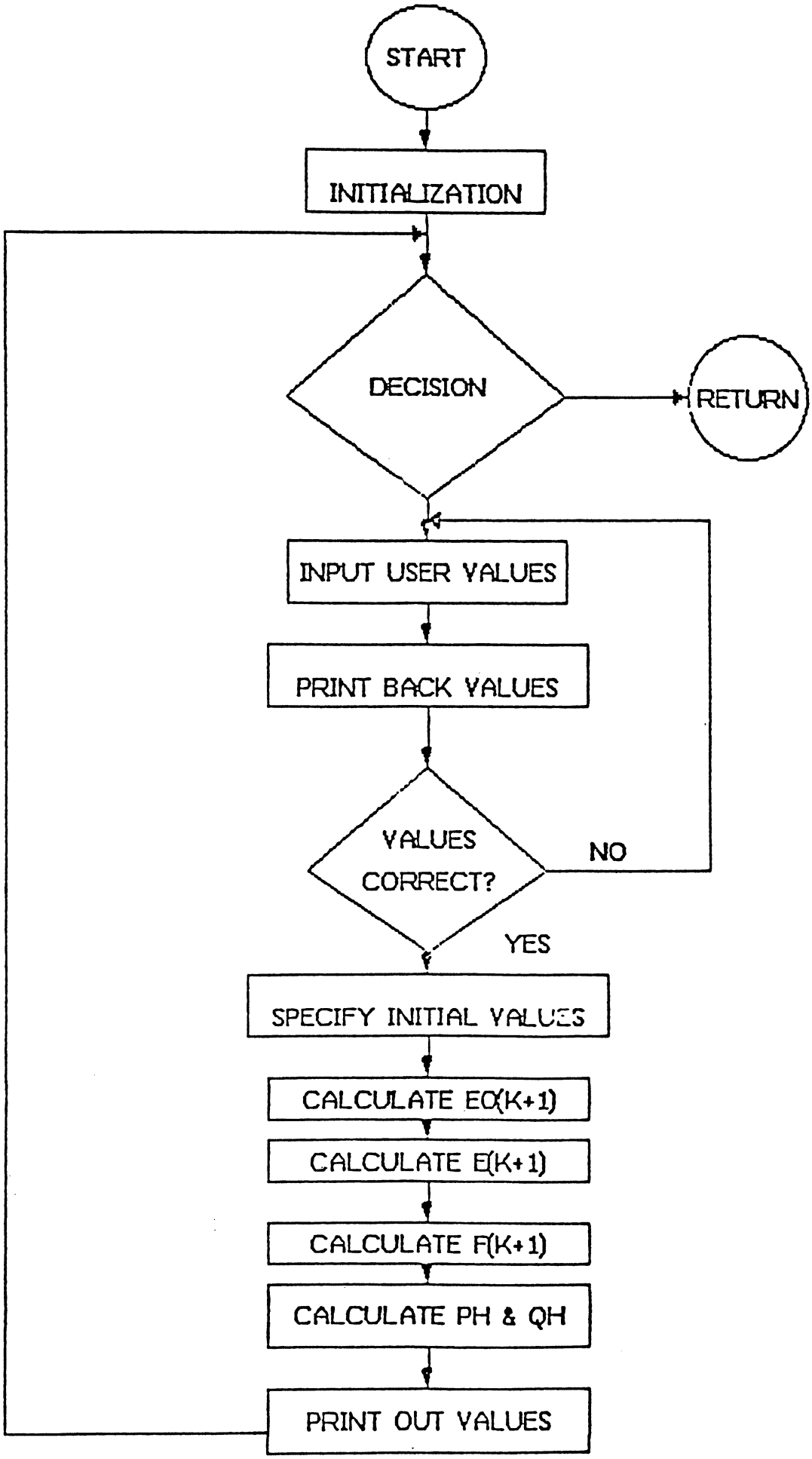
```
0601          WRITE(1,480) K,PH(K),THET2(K)
0602      480 FORMAT(20X,I3,2X,F8.3,2X,F8.3)
0603      490 CONTINUE
0604          WRITE(1,495)
0605      495 FORMAT(/,'      PRINT OUT COMPLETE
0606          GO TO 395
0607      560 RETURN
0608          END
0609
EOF
```

APPENDIX B

TUNABLE ESTIMATOR FLOWCHARTS AND SOURCE CODE

SYSTEM IDENTIFICATION PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO ESTIMATE
SYSTEM PARAMETERS USING A
TUNABLE ESTIMATOR



```
C
C
C       ****************************************
C       *                                      *
C       *           ME600 PROJECT              *
C       * COMPARISON OF AUTOMATIC CONTROL SCHEMES *
C       *                                      *
C       *     SYSTEM IDENTIFICATION PHASE      *
C       *                                      *
C       *  RESEARCH ASSISTANT: DAN LYMBURNER   *
C       *  RESEARCH ADVISOR: PROF. G. ULSOY    *
C       *                                      *
C       ****************************************
C
C      THIS PROGRAM IS USED DURING THE SYSTEM IDENTIFICATION
C      PHASE OF THE ME600 PROJECT "A COMPARISON OF AUTOMATIC
C      CONTROL SCHEMES".  A TUNABLE ESTIMATOR IS USED TO OBTAIN THE
C      DISCRETE TIME SYSTEM PARAMETERS.
C
C      THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
C      READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.
C
C
C ****************************************************************
C *                                                              *
C *                       MAIN PROGRAM                           *
C *                                                              *
C ****************************************************************
C
C
C          ************************************
C          *          INITIALIZATION          *
C          ************************************
C
      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/DACBLK/NDAC,DACHN
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
      COMMON/A2/ITIM
      ADCHN(1)=2
      DACHN(1)=0
      NADC=1
      NDAC=1
      IJ=0
C
C          ************************************
C          *  SEND A ZERO TO THE MOTOR        *
C          ************************************
C
      CALL DAC(0)
C
C          ************************************
C          *        PROGRAM EXPLANATION       *
C          ************************************
C
      WRITE(3,10)
   10 FORMAT(// THIS PROGRAM IS TO BE USED DURING THE SYSTEM //
```

```
0061        C' IDENTIFICATION PHASE OF THE ME600 PROJECT "A COMPARISON '/
0062        C' OF AUTOMATIC CONTROL SCHEMES FOR CONTROL OF THE SPEED OF A '/
0063        C' DC MOTOR USING A DIGITAL MICROCOMPUTER".  '/
0064        C' THE PROGRAM CONSISTS OF TWO SEPERATE SECTIONS WHICH ARE '/
0065        C' LISTED BELOW: '/
0066        C'      1. ESTIMATION OF THE OPEN-LOOP GAIN (K) AND OPEN-LOOP '/
0067        C'         TIME CONSTANT (T) TO BE USED IN A DIGITAL PI AND PID
0068        C'         CONTROL ALGORITHM. '/
0069        C'      2. ESTIMATION OF THE DISCRETE TIME SYSTEM PARAMETERS THRU
0070        C'         THE USE OF A TUNABLE ESTIMATION PROCEDURE. '///)
0071      15 FORMAT(I1)
0072         WRITE(3,20)
0073      20 FORMAT(' PRESS RETURN TO CONTINUE ')
0074         READ(2,15) KL
0075         WRITE(3,30)
0076      30 FORMAT(/' FIRST THE SYSTEM RESPONSE MUST BE OBTAINED.  THIS WIL
0077        C' BE DONE BY PUTTING OUT A STEP OUTPUT ON THE DIGITAL TO ANALOG
0078        C' PORT.  THE USER WILL NOW SPECIFY THE MAGNITUDE AND DURATION O
0079        C' THE STEP FUNCTION. ')
0080      35 FORMAT(I5)
0081   C
0082   C          ****************************************
0083   C          *     INPUT THE USER VALUES            *
0084   C          ****************************************
0085   C
0086      40 WRITE(3,50)
0087      50 FORMAT(/' INPUT THE MAGNITUDE OF THE INPUT FOLLOWED BY A '/
0088        C' COMMA (-2048<X<2047)  AN INTEGER VALUE   ')
0089         READ(2,35) IMAX
0090      55 WRITE(3,60)
0091      60 FORMAT(/' INPUT THE PULSE WIDTH OF THE STEP VALUE IN SECONDS.
0092        C' FOLLOW THIS VALUE WITH A COMMA   ( 1 <X< 25 )   ')
0093         READ(2,35) ISEC
0094         WRITE(3,65)
0095      65 FORMAT(/' INPUT THE TIME IN SECONDS TO REACH THE MAXIMUM  '/
0096        C'  VALUE FROM ZERO. FOLLOW THIS VALUE WITH A COMMA.    ')
0097         READ(2,35) IR
0098   C
0099   C          ****************************************
0100   C          *        PRINT BACK THE VALUES         *
0101   C          ****************************************
0102   C
0103         WRITE(3,70) IMAX
0104      70 FORMAT(/' THE MAGNITUDE OF THE STEP INPUT IS ',I5 )
0105         WRITE(3,80) ISEC
0106      80 FORMAT(/' THE PULSE WIDTH IS ',I3,' SECONDS ')
0107         WRITE(3,85) IR
0108      85 FORMAT(/' THE TIME TO REACH MAX. MAGNITUDE IS ',I5,' SECONDS '
0109   C
0110   C          ****************************************
0111   C          *      ARE THE VALUES CORRECT?         *
0112   C          ****************************************
0113   C
0114         WRITE(3,90)
0115      90 FORMAT(/' ARE THE VALUES CORRECT?  (1=YES, 2=NO)  ')
0116         READ(2,15) J
0117         IF (J .EQ. 2) GO TO 40
0118   C
0119   C          ****************************************
0120   C          *   CALCULATE THE TIME VALUES          *
```

```
C       ************************************
C
      TIM=530.*ISEC/15.
      ITIM=TIM
      L=ISEC*10.
      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
C
C         ****************************************
C         *   COLLECT DATA FROM THE MOTOR        *
C         ****************************************
C
      WRITE(3,100)
  100 FORMAT(/' PRESS RETURN TO COLLECT THE DATA FROM THE MOTOR ')
      READ(2,15)JL
C
C         ****************************************
C         *  CALL SUBROUTINE TO INITIALIZE       *
C         *        THE INTERRUPTS                *
C         ****************************************
C
      CALL INTINL
C
C         ****************************************
C         *    CALL SUBROUTINE TO WASTE TIME     *
C         ****************************************
C
      CALL WASTE
C
C         ****************************************
C         * SEND A ZERO TO SHUT OFF THE MOTOR    *
C         ****************************************
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?'/
```

```
0181        C'      1=ESTIMATION OF OPEN-LOOP GAIN AND TIME CONSTANT '/
0182        C'      2=ESTIMATION USING THE TUNABLE ESTIMATION PROCEDURE '/
0183        C'      3=OBTAIN ANOTHER SET OF DATA FROM THE MOTOR '/
0184        C'      4=QUIT '/
0185        C'      ENTER 1 - 4   ')
0186          READ(2,15) IE
0187          IF (IE .EQ. 1) CALL KTAU
0188          IF (IE .EQ. 2) CALL RCLS
0189          IF (IE .EQ. 3) GO TO 40
0190          IF (IE .EQ. 4) GO TO 150
0191          GO TO 115
0192      150 STOP
0193          END
0194    C
0195    C
0196    C ***************************************************************
0197    C *                                                             *
0198    C *          SUBROUTINE TO WASTE TIME WHILE                     *
0199    C *                  INTERRUPTS OCCUR                           *
0200    C *                                                             *
0201    C ***************************************************************
0202    C
0203          SUBROUTINE WASTE
0204          COMMON/ISRBLK/IJ
0205          COMMON/A2/ITIM
0206          WRITE(3,600)
0207      600 FORMAT(/' THE DATA COLLECTION PROCEDURE HAS BEGUN ')
0208          IJ=0
0209    C
0210    C          ***************************************
0211    C          *    ENABLE THE INTERRUPTS            *
0212    C          ***************************************
0213    C
0214          CALL ENABLE
0215    C
0216    C          ***************************************
0217    C          *          WASTE TIME                 *
0218    C          ***************************************
0219    C
0220          DO 700 I=1,ITIM
0221          DO 650 J=1,1425
0222      650 CONTINUE
0223      700 CONTINUE
0224    C
0225    C          ***************************************
0226    C          *         DISABLE THE INTERRUPTS      *
0227    C          ***************************************
0228    C
0229          CALL DISABL
0230          RETURN
0231          END
0232    C
0233    C
0234    C ***************************************************************
0235    C *                                                             *
0236    C *                  INTERRUPT SERVICE                          *
0237    C *         ROUTINE FOR DATA COLLECTION                         *
0238    C *                                                             *
0239    C ***************************************************************
0240    C
```

```
C
      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/DACBLK/NDAC,DACHN
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
C
C          ****************************************
C          *  GET THE VALUE FROM THE TACHOMETER   *
C          ****************************************
C
      IJ=IJ+1
      CALL DAC(IU(IJ))
      CALL ADC(ADCVAL(IJ))
      RETURN
      END
C
C
C ***************************************************************************
C *                                                                         *
C *      SUBROUTINE TO ESTIMATE OPEN-LOOP GAIN AND                          *
C *                    OPEN-LOOP TIME CONSTANT                              *
C *                                                                         *
C ***************************************************************************
C
C           ****************************************
C           *          INTIALIZATION               *
C           ****************************************
C
      SUBROUTINE KTAU
      DIMENSION ADCVAL(300), IU(300)
      INTEGER ADCVAL
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
      IOT=ISEC*10.
C
C           ****************************************
C           *                DECISION              *
C           ****************************************
C
149   WRITE(3,151)
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
C
C          ****************************************
C          *   PRINT OUT THE SYSTEM RESPONSE      *
C          ****************************************
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 CONSTAN
      WRITE(1,160)
  160 FORMAT(//,14X,' TIME INCREMENT',6X,' TACHOMETER VALUE',6X,'INPU
      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 ***************************************************************
C *                                                             *
C *       SUBROUTINE FOR TUNABLE ESTIMATION PROCEDURE           *
C *                                                             *
C ***************************************************************
C
C
      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), ED(300), PH(300)
      DIMENSION YHO(300), Y(300), E(300), DF(300)
      INTEGER ADCVAL
      COMMON/A1/ADCVAL,IU,ISEC,IMAX
      IOT=ISEC*10.
C
C          ***************************************
C          *          DECISION                   *
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          *       USER INPUT VALUES             *
C          ***************************************
C
  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
      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          ****************************************
C          *         PRINT BACK THE VALUES        *
C          ****************************************
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          ****************************************
C          *              VALUES CORRECT?         *
C          ****************************************
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          ****************************************
C          *  SPECIFY INITIAL VALUES OF THE       *
C          *  COMPUTER MODEL PARAMETERS           *
C          ****************************************
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          ****************************************
C          *         ESTIMATION PROCEDURE         *
C          ****************************************
C
      WRITE(3,420)
  420 FORMAT(/' STARTING RECURSIVE LEAST SQUARES ESTIMATION PROCEDURE')
      DO 425 K=1,IOT
      YHO(K+1)=(PH(K)*YH(K))+(THET2(K)*IU(K))
      Y(K)=ACTUAL(K)
```

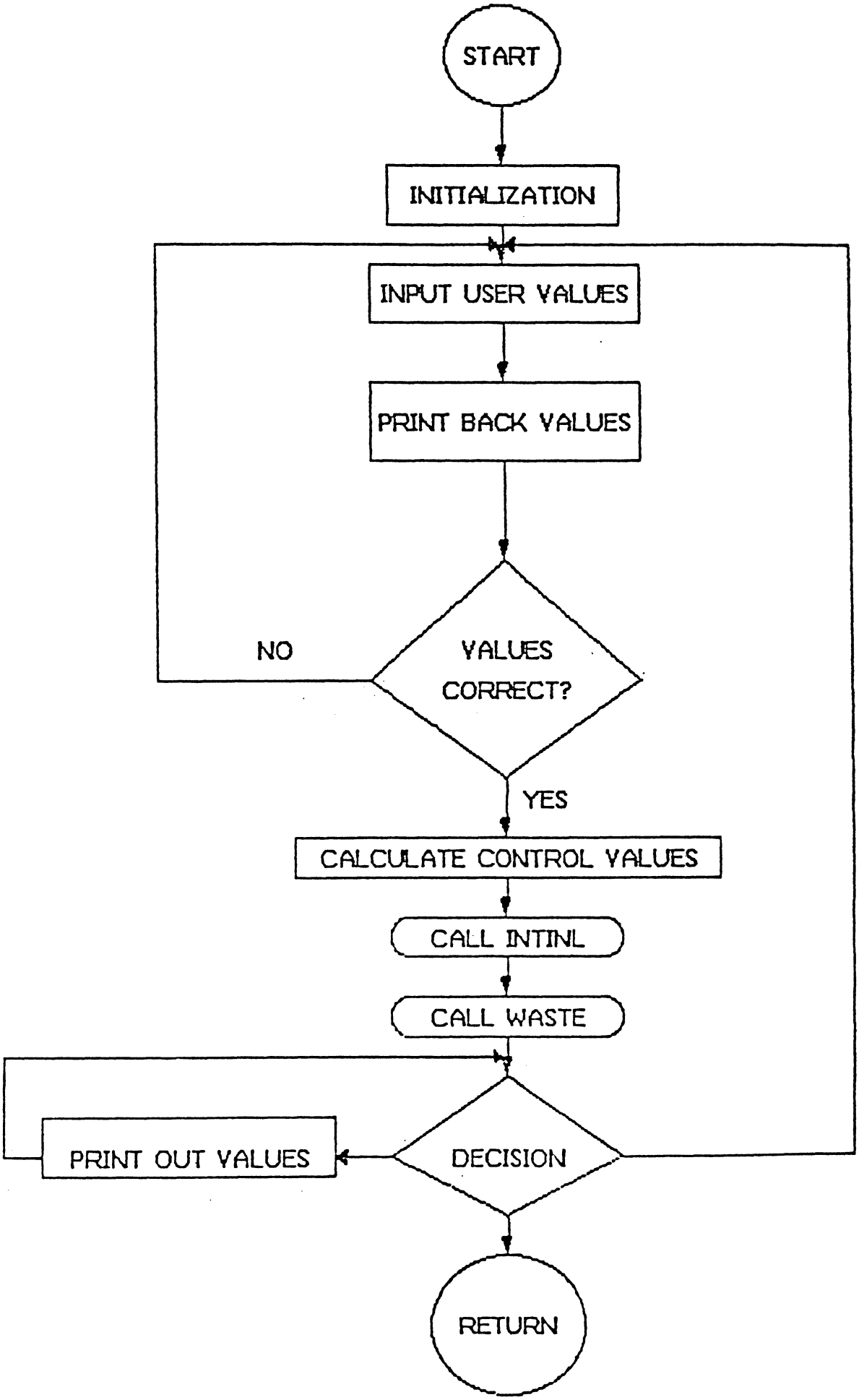
```
0421          Y(K+1)=ADCVAL(K+1)
0422          ED(K+1)=Y(K+1)-YHO(K+1)
0423    C
0424    C          *     CALCULATE E(K+1)        *
0425    C
0426          C11(K)=(F11(K)*(-YH(K))+F12(K)*IU(K))*(-YH(K))
0427          C12(K)=(F11(K)*(-YH(K))+F12(K)*IU(K))*IU(K)
0428          C21(K)=(F21(K)*(-YH(K))+F22(K)*IU(K))*(-YH(K))
0429          C22(K)=(F21(K)*(-YH(K))+F22(K)*IU(K))*IU(K)
0430          DEN(K)=C11(K)+C22(K)+1.
0431          E(K+1)=ED(K+1)/DEN(K)
0432    C
0433    C          *     CALCULATE F(K+1)
0434    C
0435          DF(K)=C11(K)+C22(K)+(AL1/AL2)
0436          F11(K+1)=(F11(K)-(C11(K)*F11(K)+C12(K)*F21(K))/DF(K))/AL1
0437          F12(K+1)=(F12(K)-(C11(K)*F12(K)+C12(K)*F22(K))/DF(K))/AL1
0438          F21(K+1)=(F21(K)-(C21(K)*F11(K)+C22(K)*F21(K))/DF(K))/AL1
0439          F22(K+1)=(F22(K)-(C21(K)*F12(K)+C22(K)*F22(K))/DF(K))/AL1
0440    C
0441    C        *      CALCULATE THETA HAT AT K+1     *
0442    C
0443          THET1(K+1)=THET1(K)+(F11(K)*(-YH(K))+F12(K)*IU(K))*E(K+1)
0444          THET2(K+1)=THET2(K)+(F21(K)*(-YH(K))+F22(K)*IU(K))*E(K+1)
0445    C
0446    C        *  CALCULATE PH AND QH     *
0447    C
0448          PH(K+1)=-THET1(K+1)
0449    C     QH(K+1)=THET2(K+1)
0450          YH(K+1)=PH(K+1)*YH(K)+THET2(K+1)*IU(K)
0451    C
0452    C          ************************************
0453    C          *           WRITE OUT THE VALUES       *
0454    C          ************************************
0455    C
0456      425 CONTINUE
0457          WRITE(3,430)
0458      430 FORMAT(/' ESTIMATION PROCEDURE COMPLETE. PRINTING OUT VALUES ')
0459          WRITE(1,440)
0460      440 FORMAT(//,18X,' TUNABLE ESTIMATOR ESTIMATION PROCEDURE ')
0461          WRITE(1,450) IMAX
0462      450 FORMAT(20X,' SYSTEM RESULTS FOR INPUT=',I5)
0463          WRITE(1,460) P,Q,F
0464      460 FORMAT(11X,'INITIAL PH=',F4.2,'  INITIAL QH=',F4.2,'  F=',F8.3)
0465          WRITE(1,465) AL1,AL2
0466      465 FORMAT(20X,'LAMBDA 1=',F6.4,10X,'LAMBDA 2=',F6.4)
0467          WRITE(1,470)
0468      470 FORMAT(//,2X,'TIME',7X,'PH',8X,'QH')
0469          DO 490 K=1,IOT
0470          WRITE(1,480) K,PH(K),THET2(K),F11(K),F12(K),F21(K),F22(K)
0471      480 FORMAT(2X,I3,2X,F8.3,2X,F8.3,2X,F10.3,2X,F10.3,2X,F10.3,2X,F10.
0472      490 CONTINUE
0473          WRITE(1,495)
0474      495 FORMAT(/,'      PRINT OUT COMPLETE  ')
0475          GO TO 395
0476      560 RETURN
0477          END
0478
 OF
```

APPENDIX C

DIGITAL PI CONTROLLER FLOWCHARTS AND SOURCE CODE

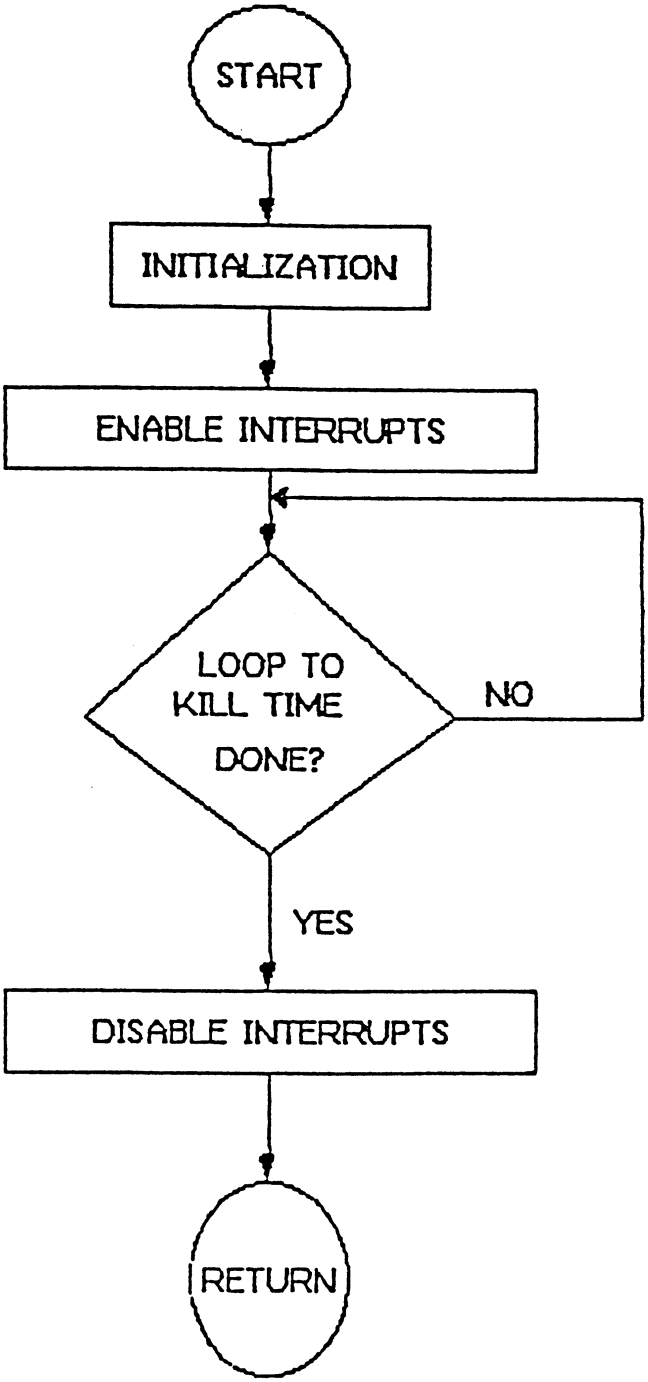
CONTROLLER DESIGN PHASE
LEVEL 1 FLOWCHART

MAIN PROGRAM
DIGITAL PI CONTROLLER



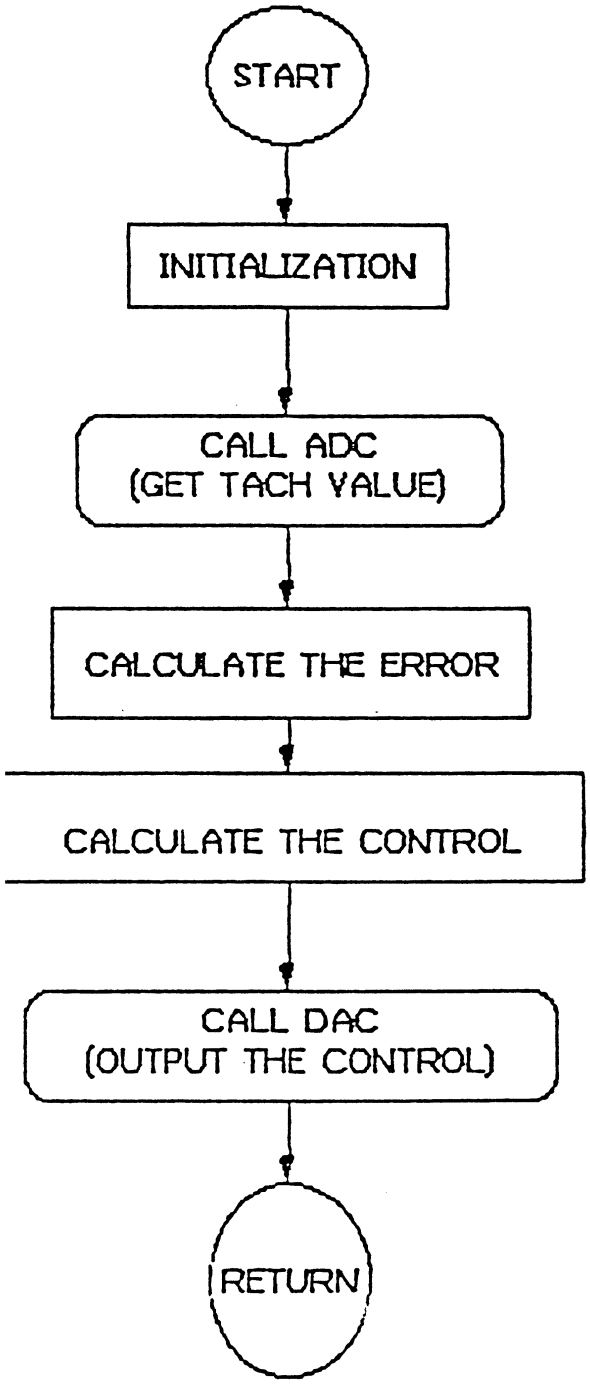
CONTROLLER DESIGN PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO
WASTE TIME
DIGITAL PI CONTROLLER



CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

INTERRUPT SERVICE ROUTINE
DIGITAL PI CONTROLLER



```
C
C
C          ********************************************
C          *                                          *
C          *            ME600 PROJECT                 *
C          * COMPARISON OF AUTOMATIC CONTROL SCHEMES  *
C          *                                          *
C          *      CONTROLLER DESIGN PHASE             *
C          *                                          *
C          *       DIGITAL PI CONTROLLER              *
C          *                                          *
C          *  RESEARCH ASSISTANT: DAN LYNBURNER       *
C          *  RESEARCH ADVISOR: PROF. G. ULSOY        *
C          *                                          *
C          ********************************************
C
C        THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A
C        DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600
C        PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR
C        CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL
C        MICROCOMPUTER".  THE ALGORITHM USED IN THIS PARTICULAR
C        PROGRAM IS A DIGITAL PROPORTIONAL-INTEGRAL (PI) CONTROLLER.
C
C        THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
C        READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.
C
C
C  ***************************************************************
C  *                                                             *
C  *                        MAIN PROGRAM                         *
C  *                                                             *
C  ***************************************************************
C
C
C           ****************************************
C           *          INITIALIZATION              *
C           ****************************************
C
      DIMENSION ADCVAL(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER ADCVAL,D,C0,C1,MK,EK
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,D,C0,C1
      COMMON/A1/ADCVAL,IU,ISEC
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A2/ITIM
      COMMON/OUTBLK/MK,EK
      ADCHN(1)=2
      DACHN(1)=0
      NADC=1
      NDAC=1
C
C           ****************************************
C           *  SEND A ZERO TO THE MOTOR            *
C           ****************************************
C
      CALL DAC(0)
C
C           ****************************************
C           *        PROGRAM EXPLANATION           *
```

```
C          ****************************************
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 PROPORTIONAL
     C' INTEGRAL (PI) 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 PI CONTROL '/
     C' ALGORITHM. THE USER WILL NOW BE ASKED TO INPUT THE '/
     C' NECESSARY CONTROL VALUES. ')
35    FORMAT(I5)
C
C          ************************************
C          *     INPUT THE USER VALUES        *
C          ************************************
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
      WRITE(3,62)
62    FORMAT(/' INPUT THE REAL PART OF Z1. INCLUDE A PERIOD ')
64    FORMAT(F10.0)
      READ(2,64) Z1R
      WRITE(3,66)
66    FORMAT(/' INPUT THE IMAGINARY PART OF Z1 & Z2. INCLUDE A PERIOD
      READ(2,64) ZI
      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.
      READ(2,64) Z
      WRITE(3,72)
72    FORMAT(/' INPUT THE MOTOR PARAMETER "P". INCLUDE A PERIOD. ')
      READ(2,64) P
      WRITE(3,74)
74    FORMAT(/' INPUT THE MOTOR PARAMETER "Q". INCLUDE A PERIOD. ')
      READ(2,64) Q
C
C          ************************************
C          *        PRINT BACK THE VALUES     *
C          ************************************
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 Z1 & Z2 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(3,87) P
   87 FORMAT(/' THE MOTOR PARAMETER "P" IS ',F7.3 )
      WRITE(3,88) Q
   88 FORMAT(/' THE MOTOR PARAMETER "Q" IS ',F7.3 )
C
C          ************************************
C          *     ARE THE VALUES CORRECT?      *
C          ************************************
C
      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 AND THE     *
C          *     CONTROL PARAMETERS           *
C          ************************************
C
      ITIM=530.*ISEC/15.
      IOT=ISEC*10.
      B=-(Z1R+Z2R)
      IF (ZI .EQ. 0.) GO TO 92
      C=((2.*ZI)**2.+B**2.)/4.
      GO TO 94
   92 X=ABS(Z1R-Z2R)
      C=(B**2.-X**2.)/4.
   94 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
      D=1.
      C0=AKP+(AKI*Z)
      C1=AKP
C
C          ************************************
C          *  CALL SUBROUTINE TO INITIALIZE   *
C          *        THE INTERRUPTS            *
C          ************************************
C
      CALL INTINL
C
C          ************************************
C          *    START THE CONTROL             *
C          ************************************
C
```

```
0181  C
0182        WRITE(3,100)
0183    100 FORMAT(/' PRESS RETURN TO BEGIN MOTOR CONTROL  ')
0184        READ(2,15) KA
0185  C
0186  C          ************************************
0187  C          *    CALL SUBROUTINE TO WASTE TIME *
0188  C          ************************************
0189  C
0190        CALL WASTE
0191  C
0192  C          ****************************************
0193  C          * SEND A ZERO TO SHUT OFF THE MOTOR *
0194  C          ****************************************
0195  C
0196        CALL DISABL
0197        CALL DAC(0)
0198  C
0199  C
0200  C          ************************************
0201  C          *              TRY AGAIN?          *
0202  C          ************************************
0203  C
0204    125 WRITE(3,130)
0205    130 FORMAT(/' THE MOTOR CONTROL IS COMPLETE.  YOU NOW HAVE'/
0206       C' THE OPTION OF: '/
0207       C'       1=PRINTING OUT THE CONTROL DATA  '/
0208       C'       2=REPEATING THE PI CONTROL PROCEDURE '/
0209       C'       3=QUITTING ')
0210    135 WRITE(3,140)
0211    140 FORMAT(/' ENTER 1,2, OR 3   ')
0212        READ(2,15) JD
0213        IF (JD .EQ. 1) GO TO 142
0214        IF (JD .EQ. 2) GO TO 40
0215        IF (JD .EQ. 3) GO TO 150
0216        GO TO 135
0217    142 WRITE(1,143)
0218    143 FORMAT(//,15X,'SYSTEM RESPONSE FOR CONTROL USING A PI ALGORITHM
0219        WRITE(1,144) IU,AKP,AKI
0220    144 FORMAT(20X,' SPEED=',I5,'     KP=',F8.2,'     KI=',F8.2,)
0221        WRITE(1,141) Z1R,Z2R,Z1
0222    141 FORMAT(20X,' REAL Z=',F5.3,3X,F5.3,5X,'IMAG. Z=',F5.3///)
0223        WRITE(1,145)
0224    145 FORMAT(5X,'TIME',5X,'CONTROL EFFORT',5X,'ACTUAL SPEED',5X,'ERRO
0225        DO 147 L=1,IOT
0226        WRITE(1,146) L,MK(L),ADCVAL(L),EK(L)
0227    146 FORMAT(6X,I3,9X,I5,11X,I5,7X,I8)
0228    147 CONTINUE
0229        GO TO 125
0230    150 STOP
0231        END
0232  C
0233  C
0234  C ***************************************************
0235  C *                                                 *
0236  C *         SUBROUTINE TO WASTE TIME WHILE          *
0237  C *                INTERRUPTS OCCUR                 *
0238  C *                                                 *
0239  C ***************************************************
0240  C
```

```
      SUBROUTINE WASTE
      COMMON/A2/ITIM
      WRITE(3,600)
  600 FORMAT(/' THE DATA COLLECTION PROCEDURE HAS BEGUN ')
C
C          ****************************************
C          *     ENABLE THE INTERRUPTS            *
C          ****************************************
C
      CALL ENABLE
C
C          ****************************************
C          *           WASTE TIME                 *
C          ****************************************
C
      DO 700 JX=1,ITIM
      DO 650 J=1,1425
  650 CONTINUE
  700 CONTINUE
C
C          ****************************************
C          *         DISABLE THE INTERRUPTS       *
C          ****************************************
C
      RETURN
      END
C
C
C ************************************************************
C *                                                          *
C *                  INTERRUPT SERVICE                       *
C *         ROUTINE FOR PI CONTROL ALGORITHM                 *
C *                                                          *
C ************************************************************
C
C
      SUBROUTINE ISR
      DIMENSION ADCVAL(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER ADCVAL,D,C0,C1,MK,EK
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,D,C0,C1
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A1/ADCVAL,IU,ISEC
      COMMON/OUTBLK/MK,EK
      EK(1)=IU
      MK(1)=2047
      ADCVAL(1)=0
C
C          ****************************************
C          *  GET THE VALUE FROM THE TACHOMETER   *
C          ****************************************
C
      I=I+1
      IM1=I-1
      CALL ADC(ADCVAL(I))
C
C          ****************************************
C          *        CALCULATE THE ERROR           *
C          ****************************************
```

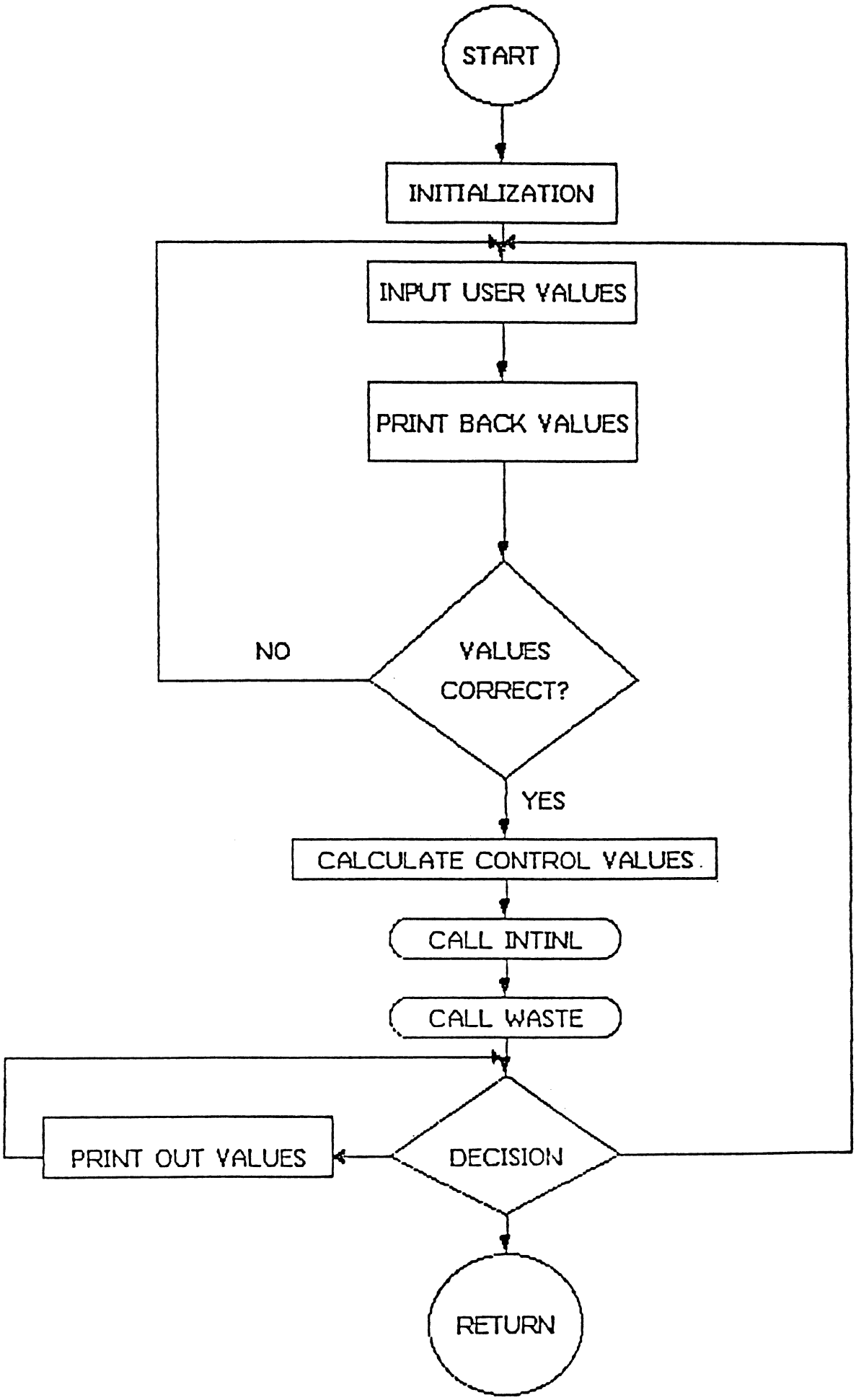
```
0301  C
0302        EK( I )=IU-ADCVAL( I )
0303  C
0304  C          ****************************************
0305  C          *          CALCULATE THE CONTROL       *
0306  C          ****************************************
0307  C
0308        MK( I )=( MK( IM1 )*D+CO*EK( I )-C1*EK( IM1 ))
0309        IF ( MK( I ) .GT. 2047 ) MK( I )=2047
0310        IF ( MK( I ) .LT. -2048 ) MK( I )=-2048
0311        CALL DAC( MK( I ))
0312        RETURN
0313        END
0314
EOF
```

APPENDIX D

SELF-TUNING ADAPTIVE CONTROLLER FLOWCHARTS AND SOURCE CODE

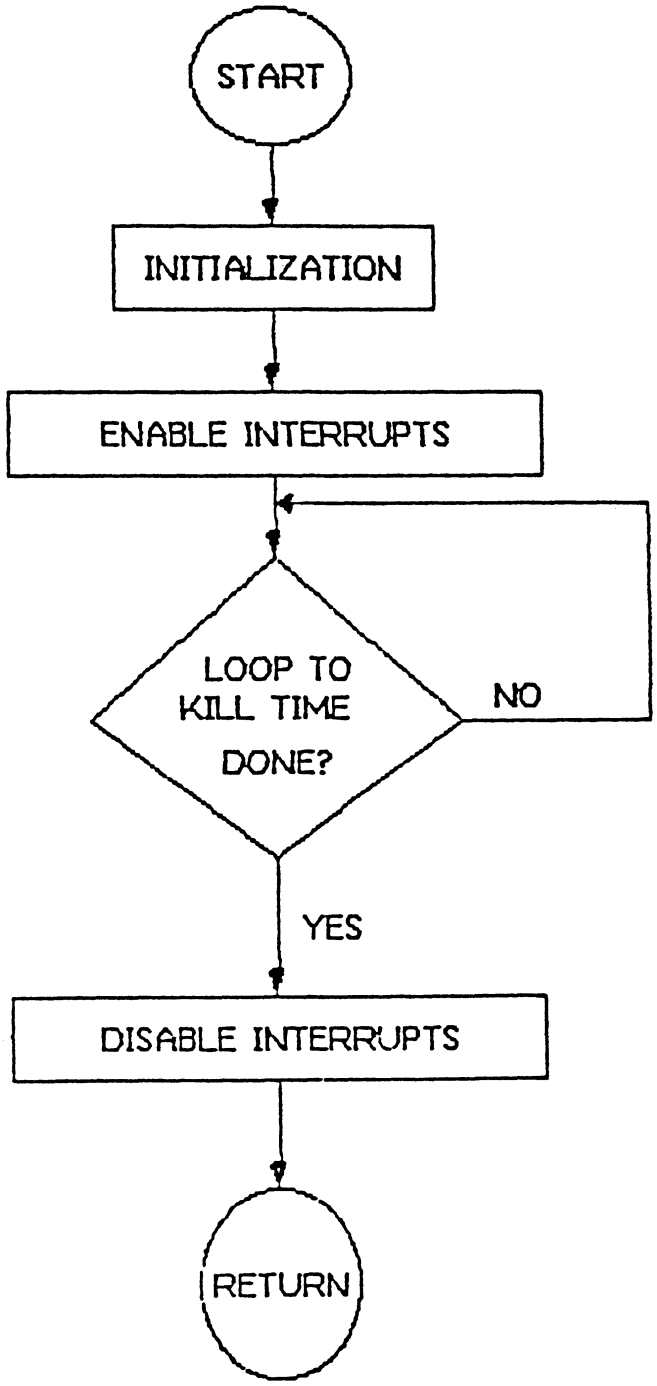
CONTROLLER DESIGN PHASE
LEVEL 1 FLOWCHART

MAIN PROGRAM
SELF-TUNING ADAPTIVE CONTROLLER



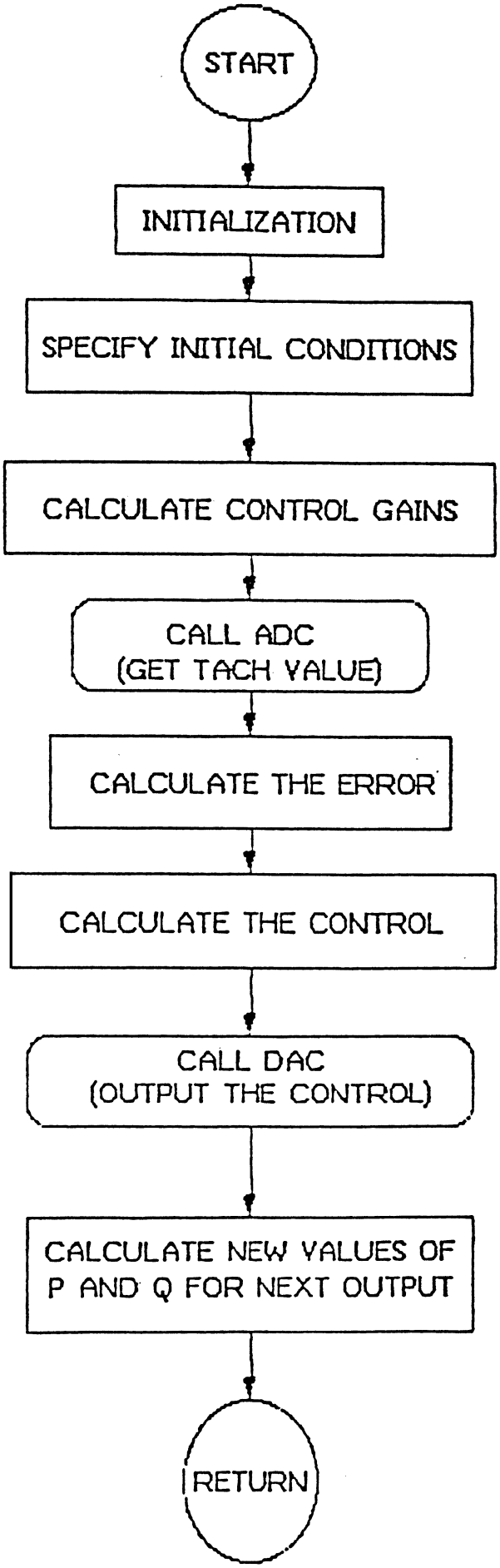
CONTROLLER DESIGN PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO WASTE TIME
SELF-TUNING ADAPTIVE CONTROLLER



CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

INTERRUPT SERVICE ROUTINE
SELF-TUNING ADAPTIVE CONTROLLER



```
C
C
C       ******************************************
C       *                                        *
C       *           ME600 PROJECT                *
C       * COMPARISON OF AUTOMATIC CONTROL SCHEMES *
C       *                                        *
C       *      CONTROLLER DESIGN PHASE           *
C       *                                        *
C       *     SELF-TUNING ADAPTIVE CONTROLLER    *
C       *                                        *
C       *  RESEARCH ASSISTANT: DAN LYMBURNER     *
C       *  RESEARCH ADVISOR: PROF. G. ULSOY      *
C       *                                        *
C       ******************************************
C
C      THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A
C      DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600
C      PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR
C      CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL
C      MICROCOMPUTER".  THE ALGORITHM USED IN THIS PARTICULAR
C      PROGRAM IS A SELF-TUNING ADAPTIVE CONTROLLER.
C
C      THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
C      READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.
C
C
C ***************************************************************
C *                                                             *
C *                      MAIN PROGRAM                           *
C *                                                             *
C ***************************************************************
C
C
C       ****************************************
C       *          INITIALIZATION              *
C       ****************************************
C
      DIMENSION Y(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER Y,MK,EK
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,B,C,P,Q,F
      COMMON/A1/Y,IU,ISEC
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A2/ITIM
      COMMON/OUTBLK/MK,EK
      ADCHN(1)=2
      DACHN(1)=0
      NADC=1
      NDAC=1
C
C       ****************************************
C       *  SEND A ZERO TO THE MOTOR            *
C       ****************************************
C
      CALL DAC(0)
C
C       ****************************************
C       *        PROGRAM EXPLANATION           *
```

```
0061  C          ****************************************
0062  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 SELF TUNING
0068        C' ADAPTIVE 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(I6)
0083  C
0084  C          ************************************
0085  C          *     INPUT THE USER VALUES          *
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          READ(2,64) Z1R
0100       64 FORMAT(F10.0)
0101          WRITE(3,66)
0102       66 FORMAT(/' INPUT THE IMAGINARY PART OF Z1 & Z2. INCLUDE A PERIOI
0103          READ(2,64) ZI
0104          WRITE(3,67)
0105       67 FORMAT(/' INPUT THE REAL PART OF Z2. INCLUDE A PERIOD ')
0106          READ(2,64) Z2R
0107          WRITE(3,68)
0108       68 FORMAT(/' INPUT THE INTERRUPT PERIOD IN SEC. INCLUDE A PERIOD.
0109          READ(2,64) Z
0110          WRITE(3,200)
0111      200 FORMAT(/' INPUT THE INITIAL VALUE OF PH. INCLUDE A PERIOD.  ')
0112          READ(2,64) P
0113          WRITE(3,210)
0114      210 FORMAT(/' INPUT THE INITIAL VALUE OF QH. INCLUDE A PERIOD.  ')
0115          READ(2,64) Q
0116          WRITE(3,220)
0117      220 FORMAT(/' INPUT THE INITIAL VALUE OF THE MATRIX F. INCLUDE '/
0118        C' A PERIOD   ')
0119          READ(2,64) F
0120  C
```

```
C          ************************************
C          *       PRINT BACK THE VALUES      *
C          ************************************
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 )
C
C          ************************************
C          *     ARE THE VALUES CORRECT?      *
C          ************************************
C
      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 AND THE     *
C          *     CONTROL PARAMETERS           *
C          ************************************
C
      ITIM=530.*ISEC/15.
      IOT=ISEC*10.
      I=1
      B=-(Z1R+Z2R)
      IF (ZI .EQ. 0.) GO TO 95
      C=((2.*ZI)**2.+B**2.)/4.
      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)

C
C          ************************************
C          *  CALL SUBROUTINE TO INITIALIZE   *
C          *        THE INTERRUPTS            *
C          ************************************
C
      CALL INTINL
C
C          ************************************
C          *    START THE CONTROL             *
C          ************************************
```

```
C
      WRITE(3,100)
  100 FORMAT(/' PRESS RETURN TO BEGIN MOTOR CONTROL  ')
      READ(2,15) KA
C
C          ************************************
C          *    CALL SUBROUTINE TO WASTE TIME *
C          ************************************
C
      CALL WASTE
C
C          ****************************************
C          * SEND A ZERO TO SHUT OFF THE MOTOR *
C          ****************************************
C
      CALL DISABL
      CALL DAC(0)
C
C
C          ************************************
C          *              TRY AGAIN?          *
C          ************************************
C
  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
  144 FORMAT(13X,'SPEED=',I5,'    REAL Z=',F5.3,2X,F5.3,'    IM Z=',F5.
      WRITE(1,145)
  145 FORMAT(5X,'TIME',5X,'CONTROL EFFORT',5X,'ACTUAL SPEED',5X,'ERRO
      DO 147 L=1,IOT
      WRITE(1,146) L,MK(L),Y(L),EK(L)
  146 FORMAT(6X,I3,9X,I5,11X,I5,7X,I8)
  147 CONTINUE
      GO TO 125
  150 STOP
      END
C
C
C ***************************************************************
C *                                                             *
C *          SUBROUTINE TO WASTE TIME WHILE                     *
C *                  INTERRUPTS OCCUR                           *
C *                                                             *
C ***************************************************************
C
      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          ***************************************
C          *          WASTE TIME                 *
C          ***************************************
C
      DO 700 JX=1,ITIM
      DO 650 J=1,1425
  650 CONTINUE
  700 CONTINUE
      RETURN
      END
C
C
C ************************************************************
C *                                                          *
C *                  INTERRUPT SERVICE                       *
C *   ROUTINE FOR SELF TUNING ADAPTIVE CONTROLLER            *
C *                                                          *
C ************************************************************
C
C
      SUBROUTINE ISR
      DIMENSION Y(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER Y,MK,EK
      REAL KPN, KIN
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,B,C,P,Q,F
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A1/Y,IU,ISEC
      COMMON/OUTBLK/MK,EK
C
C          ***************************************
C          *     SPECIFY INITIAL VALUES          *
C          ***************************************
C
      IF (I .NE. 1) GO TO 500
      MK(1)=2047
      YHP=0.
      QHP=Q
      PHP=P
      F11P=F
      F22P=F
      CALL ADC(Y(1))
      CALL DAC(MK(1))
      EK(1)=IU-Y(1)
  500 I=I+1
      J=I-1
      KPN=(PHP-C)/QHP
      KIN=(B+1.+PHP-QHP*KPN)/(.1*QHP)
C
C          ***************************************
```

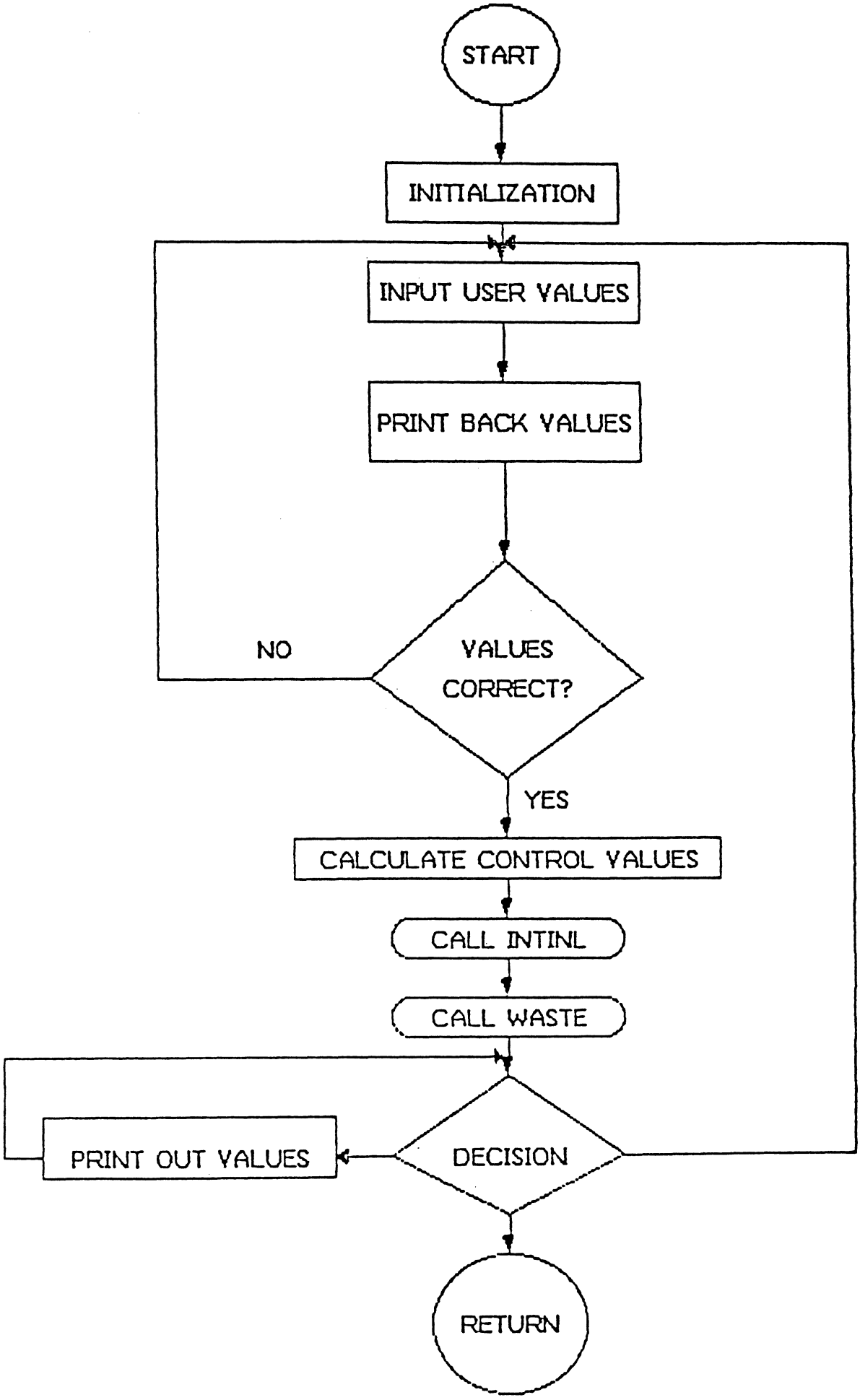
```
0301  C          *  GET THE VALUE FROM THE TACHOMETER  *
0302  C          ***************************************
0303  C
0304         CALL ADC(Y(I))
0305         EK(I)=IU-Y(I)
0306  C
0307  C          ***************************************
0308  C          *          CALCULATE THE CONTROL      *
0309  C          ***************************************
0310  C
0311         MK(I)=(MK(J)+(KPN+KIN/10.)*EK(I)-KPN*EK(J))
0312         IF (MK(I) .GT. 2047) MK(I)=2047
0313         IF (MK(I) .LT. -2048) MK(I)=-2048
0314         CALL DAC(MK(I))
0315  C
0316  C          ***************************************
0317  C          *    CALCULATE PH AND QH              *
0318  C          ***************************************
0319  C
0320  C
0321  C        *    CALCULATE EN          *
0322  C
0323         C11P=(F11P*(-YHP))*(-YHP)
0324         C22P=(F22P*MK(J))*MK(J)
0325         EN=(Y(I)-PHP*YHP-QHP*MK(J))/(C11P+C22P+1.)
0326  C
0327  C        *    CALCULATE FN      *
0328  C
0329         DFP=C11P+C22P+2.
0330         F11N=(F11P-(C11P*F11P)/DFP)
0331         F22N=(F22P-(C22P*F22P)/DFP)
0332  C
0333  C          *    CALCULATE NEW VALUES OF PH AND QH     *
0334  C
0335         PHN=PHP-(F11P*(-YHP))*EN
0336         QHN=QHP+(F22P*MK(J))*EN
0337         YHN=PHN*YHP+QHN*MK(J)
0338         PHP=PHN
0339         QHP=QHN
0340         F11P=F11N
0341         F22P=F22N
0342         YHP=YHN
0343         RETURN
0344         END
0345
EOF
```

APPENDIX E

ROBUST FEEDBACK CONTROLLER FLOWCHARTS AND SOURCE CODE

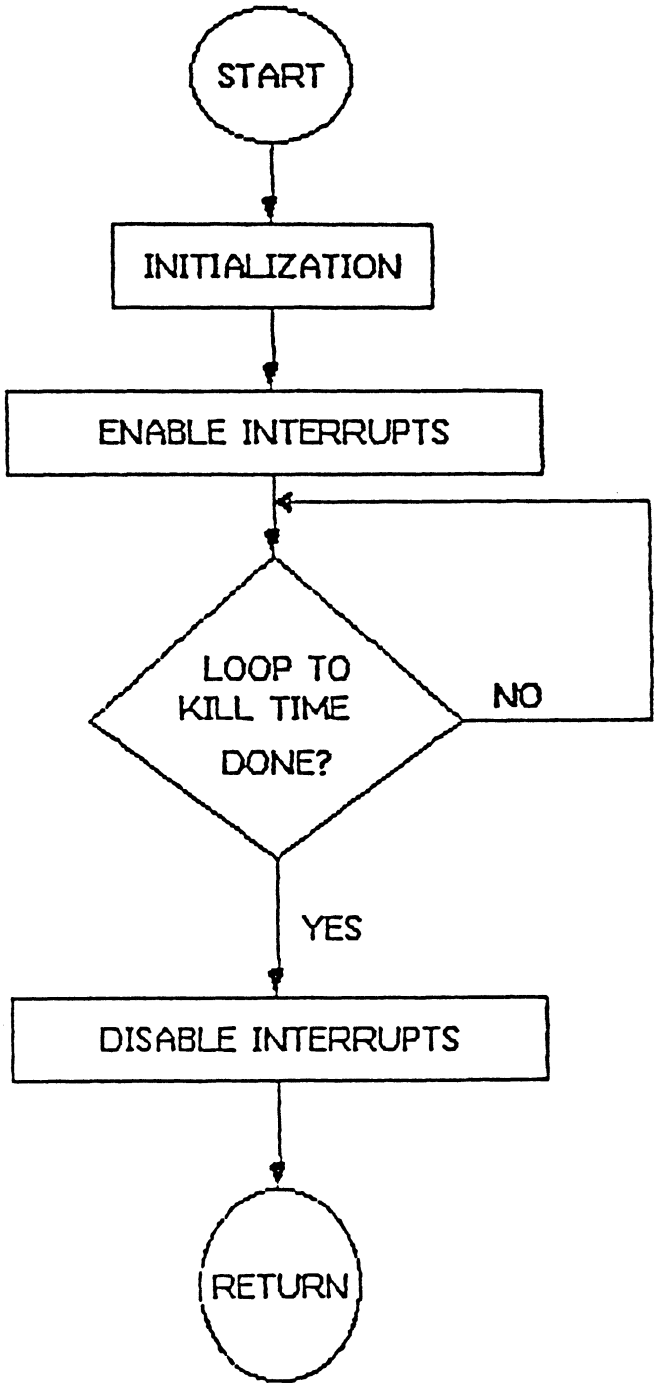
CONTROLLER DESIGN PHASE
LEVEL 1 FLOWCHART

MAIN PROGRAM
ROBUST FEEDBACK CONTROLLER



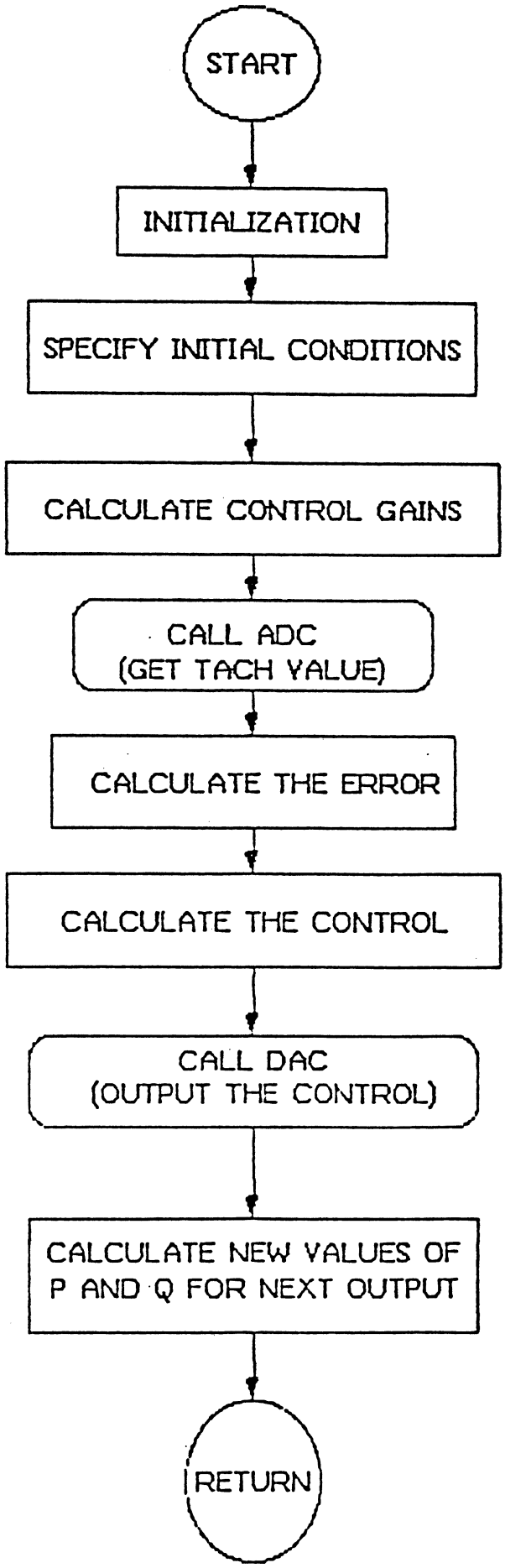
CONTROLLER DESIGN PHASE
LEVEL 2 FLOWCHART

SUBROUTINE TO WASTE TIME
ROBUST FEEDBACK CONTROLLER



CONTROLLER DESIGN PHASE
LEVEL 3 FLOWCHART

INTERRUPT SERVICE ROUTINE
ROBUST FEEDBACK CONTROLLER



```
C
C     ***********************************************
C     *                                             *
C     *            ME600 PROJECT                    *
C     * COMPARISON OF AUTOMATIC CONTROL SCHEMES     *
C     *                                             *
C     *       CONTROLLER DESIGN PHASE               *
C     *                                             *
C     *      ROBUST FEEDBACK CONTROLLER             *
C     *                                             *
C     *  RESEARCH ASSISTANT: DAN LYMBURNER          *
C     *  RESEARCH ADVISOR: PROF. G. ULSOY           *
C     *                                             *
C     ***********************************************
C
C       THIS PROGRAM IS USED TO CONTROL THE MOTOR SPEED OF A
C       DC MOTOR DURING THE CONTROLLER DESIGN PHASE OF THE ME600
C       PROJECT "A COMPARISON OF AUTOMATIC CONTROL SCHEMES FOR
C       CONTROL OF THE SPEED OF A DC MOTOR USING A DIGITAL
C       MICROCOMPUTER".  THE ALGORITHM USED IN THIS PARTICULAR
C       PROGRAM IS A ROBUST FEEDBACK CONTROLLER.
C
C       THE PROGRAM SENDS A SIGNAL OUT ON DAC CHANNEL ZERO AND
C       READS THE VALUE FROM THE TACHOMETER ON ADC CHANNEL TWO.
C
C
C ***************************************************************
C *                                                             *
C *                       MAIN PROGRAM                          *
C *                                                             *
C ***************************************************************
C
C
C           ****************************************
C           *          INITIALIZATION              *
C           ****************************************
C
      DIMENSION Y(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER Y,MK,EK
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,B,C,P,Q,F,AK2
      COMMON/A1/Y,IU,ISEC
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A2/ITIM
      COMMON/OUTBLK/MK,EK
      ADCHN(1)=2
      DACHN(1)=0
      NADC=1
      NDAC=1
C
C           ****************************************
C           *  SEND A ZERO TO THE MOTOR            *
C           ****************************************
C
      CALL DAC(0)
C
C           ****************************************
C           *        PROGRAM EXPLANATION           *
```

```
C        ****************************************
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        ****************************************
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 PERIOI
      READ(2,64) ZI
      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.
      READ(2,64) Z
      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 QH. INCLUDE A PERIOD.  ')
      READ(2,64) Q
      WRITE(3,220)
  220 FORMAT(/' INPUT THE INITIAL VALUE OF THE MATRIX F. INCLUDE '/
```

```
      C' A PERIOD   ' )
       READ(2,64) F
C
C          ************************************
C          *        PRINT BACK THE VALUES     *
C          ************************************
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(3,245) 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 )
C
C          ************************************
C          *     ARE THE VALUES CORRECT?      *
C          ************************************
C
       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 AND THE     *
C          *     CONTROL PARAMETERS           *
C          ************************************
C
       ITIM=530.*ISEC/15.
       IOT=ISEC*10.
       I=1
       B=-(Z1R+Z2R)
       IF (ZI .EQ. 0.) GO TO 95
       C=((2.*ZI)**2.+B**2.)/4.
       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)

C
C          ************************************
C          *  CALL SUBROUTINE TO INITIALIZE   *
C          *         THE INTERRUPTS           *
C          ************************************
C
```

```
0181          CALL INTINL
0182    C
0183    C          ************************************
0184    C          *    START THE CONTROL             *
0185    C          ************************************
0186    C
0187          WRITE(3,100)
0188      100 FORMAT(/' PRESS RETURN TO BEGIN MOTOR CONTROL  ')
0189          READ(2,15) KA
0190    C
0191    C          ************************************
0192    C          *    CALL SUBROUTINE TO WASTE TIME *
0193    C          ************************************
0194    C
0195          CALL WASTE
0196    C
0197    C          ************************************
0198    C          * SEND A ZERO TO SHUT OFF THE MOTOR *
0199    C          ************************************
0200    C
0201          CALL DISABL
0202          CALL DAC(0)
0203    C
0204    C
0205    C          ************************************
0206    C          *             TRY AGAIN?           *
0207    C          ************************************
0208    C
0209      125 WRITE(3,130)
0210      130 FORMAT(/' THE MOTOR CONTROL IS COMPLETE.  YOU NOW HAVE'/
0211         C' THE OPTION OF: '/
0212         C'       1=PRINTING OUT THE CONTROL DATA  '/
0213         C'       2=REPEATING THE ROBUST FEEDBACK CONTROL PROCEDURE '/
0214         C'       3=QUITTING ')
0215      135 WRITE(3,140)
0216      140 FORMAT(/' ENTER 1,2, OR 3   ')
0217          READ(2,15) JD
0218          IF (JD .EQ. 1) GO TO 142
0219          IF (JD .EQ. 2) GO TO 40
0220          IF (JD .EQ. 3) GO TO 150
0221          GO TO 135
0222      142 WRITE(1,143)
0223      143 FORMAT(/,15X,'SYSTEM RESPONSE USING A ROBUST FEEDBACK CONTROLLEI
0224          WRITE(1,144) IU,Z1R,Z2R,ZI
0225      144 FORMAT(13X,'SPEED=',I5,'   REAL Z=',F5.3,2X,F5.3,'   IM Z=',F5.
0226          WRITE(1,145)
0227      145 FORMAT(5X,'TIME',5X,'CONTROL EFFORT',5X,'ACTUAL SPEED',5X,'ERRO
0228          DO 147 L=1,IOT
0229          WRITE(1,146) L,MK(L),Y(L),EK(L)
0230      146 FORMAT(6X,I3,9X,I5,11X,I5,7X,I8)
0231      147 CONTINUE
0232          GO TO 125
0233      150 STOP
0234          END
0235    C
0236    C
0237    C ***************************************************
0238    C *                                                 *
0239    C *         SUBROUTINE TO WASTE TIME WHILE          *
0240    C *                 INTERRUPTS OCCUR                *
```

```
C *                                                              *
C ****************************************************************
C
      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          ****************************************
C          *          WASTE TIME                  *
C          ****************************************
C
      DO 700 JX=1,ITIM
      DO 650 J=1,1425
  650 CONTINUE
  700 CONTINUE
      RETURN
      END
C
C
C ****************************************************************
C *                                                              *
C *                  INTERRUPT SERVICE                           *
C *   ROUTINE FOR SELF TUNING ADAPTIVE CONTROLLER                *
C *                                                              *
C ****************************************************************
C
C
      SUBROUTINE ISR
      DIMENSION Y(300), ADCHN(8), DACHN(4), EK(300), MK(300)
      INTEGER Y,MK,EK
      REAL KPN, K3
      LOGICAL*1 NADC,NDAC,ADCHN,DACHN
      COMMON/ISRBLK/I,B,C,P,Q,F,AK2
      COMMON/ADCBLK/NADC,ADCHN
      COMMON/DACBLK/NDAC,DACHN
      COMMON/A1/Y,IU,ISEC
      COMMON/OUTBLK/MK,EK
C
C          ****************************************
C          *     SPECIFY INITIAL VALUES           *
C          ****************************************
C
      IF (I .NE. 1) GO TO 500
      MK(1)=2047
      YHP=0.
      GHP=Q
      PHP=P
      F11P=F
      F22P=F
      CALL ADC(Y(1))
      CALL DAC(MK(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.)+1.+PHP-KPN*QHP)/(QHP)
C
C          ****************************************
C          *  GET THE VALUE FROM THE TACHOMETER   *
C          ****************************************
C
      CALL ADC(Y(I))
      EK(I)=IU-Y(I)
C
C          ****************************************
C          *          CALCULATE THE CONTROL       *
C          ****************************************
C
      S=AK2*(1-PHP)-KPN-K3
      T=KPN*Y(J)
      MK(I)=(MK(J)+S*Y(I)+K3*IU+T)/(1.+AK2*QHP)
      IF (MK(I) .GT. 2047) MK(I)=2047
      IF (MK(I) .LT. -2048) MK(I)=-2048
      CALL DAC(MK(I))
C
C          ****************************************
C          *     CALCULATE PH AND QH              *
C          ****************************************
C
C
C         *     CALCULATE EN          *
C
      C11P=(F11P*(-YHP))*(-YHP)
      C22P=(F22P*MK(J))*MK(J)
      EN=(Y(I)-PHP*YHP-QHP*MK(J))/(C11P+C22P+1.)
C
C         *     CALCULATE FN       *
C
      DFP=C11P+C22P+2.
      F11N=(F11P-(C11P*F11P)/DFP)
      F22N=(F22P-(C22P*F22P)/DFP)
C
C          *     CALCULATE NEW VALUES OF PH AND QH
C
      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

EOF
```

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
                                ;(EVERTHING BUT CLOCK)
EOI     EQU     020H            ;END OF INTERRUPT
;
CLKEOI  MACRO
;
;       This macro issues the interrupt acknowledge
;       command to the real time clock, and is desisned 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
                                ;BOUNARY. 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,DISABL
        EXT ISR
;
```

```
;
;
INTINL: 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 SO 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 tickins from this point on
;       .
;       .
;
        IM      0
;
;       RETURN FROM SUBROUTINE
;
        POP     AF
        POP     HL
        RET
;
;
;
;
;
;       SEVICE 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
        CLKEOI          ;ACKNOWLEDGE INTERRUPT
                        ;AND RETURN
;
;
;
```

```
0121  ENABLE: EI
0122          RET
0123  ;
0124  ;
0125  DISABL: DI
0126          RET
0127  ;
0128  ;
0129          TITLE   ADC SUBROUTINE
0130          SUBTTL  DOCUMENTATION
0131  ;
0132  ;
0133  ;               This routine is desisned to be called
0134  ;       from a FORTRAN prosram and will perform analos
0135  ;       to disital conversions usins the XYCOM 1850B
0136  ;       board. The mode of operation is random channel
0137  ;       addressins with interrupts disabled. The
0138  ;       number and order of addressins is specified by
0139  ;       the labeled common block /ADCBLK/. The sub-
0140  ;       routine returns values in an INTEGER*2 array
0141  ;       which is the arsument to the call.
0142  ;
0143  ;               A sample callins sequence is:
0144  ;
0145  ;       INTEGER*2       ADCVAL(27)
0146  ;       LOGICAL*1       NADC,ADCHN(8)
0147  ;               .
0148  ;               .
0149  ;       NADC    =3
0150  ;       ADCHN(1)=0
0151  ;       ADCHN(2)=3
0152  ;       ADCHN(3)=2
0153  ;
0154  ;               .
0155  ;               .
0156  ;       DO 100 I=1,27,3
0157  ;       CALL    ADC(ADCVAL(I))
0158  ;       CONTINUE
0159  ;
0160  ;
0161          SUBTTL  COMMON /ADCBLK/ NADC,ADCHN(8)
0162          COMMON  /ADCBLK/
0163  NADC:   DEFS    1       ;NUMBER OF CHANNELS
0164  ADCHN:  DEFS    8       ;CHANNELS SELECTED
0165          SUBTTL  ADC CODE
0166          CSEG
0167          PUBLIC  ADC
0168  ;-----------------------------------------------------
0169  ;
0170  ;       SYMBOLS
0171  ;
0172  ;-----------------------------------------------------
0173  ;
0174  ;
0175  SLOTNO  EQU     0FF70H  ;SLOT CODE WHERE 1850B IS
0176                          ;INSTALLED AND  IS THE BASE
0177                          ;FOR THE FOLLOWING:
0178  ;
0179  STATUS  EQU     SLOTNO  ;STATUS WORD
0180  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
;
ADC:    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,ADCHN        ;IX FOR MULTIPLEXING
        LD      IY,STATUS       ;IY TO CHECK STSTUS
;
;
;       THE FOLLOWING LOOP MULTIPLEXES, WAITS, & STORES
;
;
        LD      A,(IX)          ;ACCUMULATOR HAS CHANNEL
ADCLP:  LD      (MULPLX),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
                                ;(DATA2) -> D
;
;
        LD      (HL),D          ;STORE HIGH BYTE
        INC     HL              ;AND INCREMENT
;
;
        LD      (HL),E          ;STORE LOW BYTE
        INC     HL              ;AND INCREMENT HL
```

```
0241  ;
0242  ;
0243          DJNZ    AIOCLP          ;LOOP IF MORE CHANNELS
0244  ;
0245  ;
0246  ;       RESTOR REGISTERS
0247  ;
0248  ;
0249          POP     AF
0250          POP     BC
0251          POP     DE
0252          POP     HL
0253          POP     IY
0254          POP     IX
0255          RET
0256          TITLE   DAC SUBROUTINE
0257          SUBTTL  DOCUMENTATION
0258  ;
0259  ;
0260  ;               This routine is desisned to be called
0261  ;       from a FORTRAN prosram and will perform disital
0262  ;       to analos conversions usins the XYCOM 1850R
0263  ;       board. The mode of operation is random channel
0264  ;       addressins with interrupts disabled. The
0265  ;       number and order of addressins is specified by
0266  ;       the labeled common block /DACBLK/. The sub-
0267  ;       routine sends values in an INTEGER*2 array
0268  ;       which is the arsument to the call.
0269  ;
0270  ;               A samole callins sequence is:
0271  ;
0272  ;       INTEGER*2       DACVAL(27)
0273  ;       LOGICAL*1       NDAC,DACHN(4)
0274  ;       COMMON /DACBLK/ NDAC,DACHN
0275  ;               .
0276  ;               .
0277  ;       NDAC    =2
0278  ;       DACHN(1)=0
0279  ;       DACHN(2)=1
0280  ;
0281  ;               .
0282  ;               .
0283  ;       DO 100 I=1,27,3
0284  ;       CALL    DAC(DACVAL(I))
0285  ;       CONTINUE
0286  ;
0287  ;
0288          SUBTTL  COMMON /DACBLK/ NDAC,DACHN(4)
0289          COMMON  /DACBLK/
0290  NDAC:   DEFS    1               ;NUMBER OF CHANNELS
0291  DACHN:  DEFS    4               ;CHANNELS SELECTED
0292          SUBTTL  DAC CODE
0293          CSEG
0294          PUBLIC  DAC
0295  ;
0296  ;
0297  ;
0298  ;
0299  ;-----------------------------------------------------
0300  ;
```

```
;       CODE
;
;-------------------------------------------------------
;
;
;       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
;
;
DACLP:  LD      D,(HL)          ;GET LOW BYTE
        INC     HL              ;AND INCREMENT
        LD      A,0FH           ;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    DACLP           ;LOOP IF MORE CHANNELS
;
;
;       RESTOR REGISTERS
;
;
        POP     AF
        POP     BC
        POP     DE
        POP     HL
        POP     IX
        RET
        END
```