27
1/11/79
25 up to NTIS

MASTER

BDX-613-1980

DIGITAL CONTROLLER DESIGN, ANALYSIS, AND
IMPLEMENTATION FOR THE MOLD HEATING
AND COOLING SYSTEM

By R. H. Floersch

Published December 1978

Topical Report

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Prepared for the United States Department of Energy
Under Contract Number DE-AC04-76-DP00613.

**Kansas City
Division**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Printed in the United States of America

Available From the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, Virginia 22161.

Price:	Microfiche	\$3.00
	Paper Copy	\$8.00

BDX-613-1980
Distribution Category UC-38

DIGITAL CONTROLLER DESIGN, ANALYSIS, AND
IMPLEMENTATION FOR THE MOLD HEATING
AND COOLING SYSTEM

By R. H. Floersch

Published December 1978

Topical Report
R. H. Floersch, Project Leader

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Communications Services

**Kansas City
Division**

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *98*

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

DIGITAL CONTROLLER DESIGN, ANALYSIS, AND IMPLEMENTATION FOR THE
MOLD HEATING AND COOLING SYSTEM

BDX-613-1980, Topical Report, Published December 1978

Prepared by R. H. Floersch

Mold heating and cooling control using metal temperature feedback was unsuccessful when conventional control strategies were used. Special digital control algorithms, developed to effect mold temperature control, were formulated by using empirical mold and fluid loop dynamic models in conjunction with Z transforms to arrive at a controller configuration.

WPC-drb

This report was prepared as an account of work sponsored by the United States Government. Neither the United States, nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, expressed or implied or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

The Bendix Corporation
Kansas City Division
P. O. Box 1159
Kansas City, Missouri 64141

A prime contractor with the United States
Department of Energy under Contract Number
DE-AC04-76-DP00613

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CONTENTS

Section	Page
SUMMARY.	11
DISCUSSION	15
SCOPE AND PURPOSE.	15
PRIOR WORK	15
ACTIVITY	15
<u>Controller Design.</u>	15
<u>Controller Analysis and Simulation</u>	20
<u>Controller Implementation and Testing.</u>	39
ACCOMPLISHMENTS.	65
FUTURE WORK.	65
REFERENCES	71
APPENDICES	
A. DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, FIRST ORDER PLANT.	73
B. DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, SECOND ORDER PLANT	81
C. DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, THIRD ORDER PLANT.	91
D. DIGITAL CONTROLLER DERIVATION: CASCADED FLUID AND MOLD TEMPERATURE FEEDBACK.	101
E. DIGITAL CONTROLLER DESIGN, CASCADE CONFIGURATION: INNER LOOP PI, OUTER LOOP DIGITAL DESIGN, SECOND ORDER PLANT.	113
F. SOURCE LISTING OF ROOT FINDING PROGRAM FOR FIRST ORDER PLANT DIGITAL CONTROLLER	125
G. SOURCE LISTING OF ROOT FINDING PROGRAM FOR CASCADED FLUID AND MOLD TEMPERATURE DIGITAL CONTROLLERS.	133

H.	SOURCE LISTING OF ROOT FINDING PROGRAM FOR CASCADED INNER LOOP PI AND OUTER LOOP DIGITAL CONTROLLERS.	141
I.	SOURCE LISTING OF SIMULATION PROGRAM FOR FIRST ORDER PLANT DIGITAL CONTROLLER	147
J.	SOURCE LISTING OF SIMULATION PROGRAM FOR CASCADED DIGITAL CONTROLLERS	179
K.	CLOSED LOOP ERROR ANALYSES FOR STEP AND RAMP SET POINT CHANGES, CASCADE SYSTEM.	213
L.	DIGITAL CONTROLLER DESIGN, CASCADE CONFIGURATIONS: INNER LOOP PI, OUTER LOOP DIGITAL DESIGN, FIRST ORDER PLANT.	219
	DISTRIBUTION	225

ILLUSTRATIONS

Figure		Page
1	Typical Digital Closed Loop Block Diagram. . .	18
2	Typical Digital Control Loop After Conversion to Pulse Transfer Form	18
3	Unit Circle in Z Plane, Indicating Stable Region	21
4	Block Diagram of Derived Controller for a First Order Plant	25
5	Mold Temperature Feedback, Digital Controller Simulation, First Order Plant, First Order System	26
6	Digital Controller, First Order System, Run 1.	30
7	Digital Controller, First Order System, Run 2.	31
8	Digital Controller, First Order System, Run 3.	32
9	Digital Controller, First Order System, Run 4.	33
10	Cascade Digital Control System	37
11	Cascade Simulated Digital Controller (LMJ=100, LMN=10), Run 1	40
12	Cascade Simulated Digital Controller (LMJ=200, LMN=10), Run 4	41
13	Cascade Simulated Digital Controller (LMJ=300, LMN=10), Run 7	42
14	Cascade Simulated Digital Controller (LMJ=400, LMN=10), Run 10.	43
15	Cascade Block Diagram of Inner Loop PID, Second Order Plant.	50
16	Heat Transfer Configuration With Air Gap Between Mold Inserts	62
17	Cavity Response, First Order Algorithm	67

18	Cavity Response, Second Order Algorithm. . . .	68
19	Punch Response, First Order Algorithm. . . .	69
20	Punch Response, Second Order Algorithm	70
A-1	Block Diagram, Mold Temperature Feedback, First Order Plant.	76
A-2	Equivalent Block Diagram, First Order Plant. .	78
B-1	Block Diagram, Mold Temperature Feedback, Second Order Plant	84
B-2	Equivalent Block Diagram, Plant Dynamics in Pulse Form, Second Order Plant	88
C-1	Digital Controller Design, Third Order Plant.	94
D-1	Cascaded Control Block Diagram	104
D-2	Equivalent Cascaded Control Block Diagram. . .	109
E-1	Cascade Control Configuration.	116
E-2	Reconfigured Cascade System, Inner Loop Approximated by a First Order Lag.	119
L-1	Actual Cascade Control Loop Configuration. . .	222
L-2	Design Configuration for Mold Loop	222

TABLES

Number		Page
1	Common Z-Transformations	17
2	Mold Temperature Feedback, Digital Controller Simulation: First Order Plant, First Order System Parameter List.	28
3	Cascaded Digital Control System Parameter List	38
4	Initial Algorithm Checkout, Mold Open Dynamics, MD-252503-700 Mold Base.	51
5	Recipe 62: Step Tests	54
6	Recipe 63: Ramp Tests	55
7	Ramp Test Results, New Algorithm	56
8	Mid-Test Recheck of Mold Open Dynamics for MD-252503-700 Mold Base.	57
9	Step Test Results, New Algorithm	58
10	Variable Time Constant Test Results, IAE . . .	59
11	Mold Base 6984967-005 With Inserts 4280-002 Dynamic Data for Algorithm Evaluation. . . .	65
12	First Order Versus Second Order Algorithm Test Results	66

THIS PAGE
WAS THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

SUMMARY

The mold heating and cooling project began in 1969 in an effort to develop direct metal temperature control for injection and compression molds used to make plastic parts. In 1971, a direct digital control (DDC) computer system was installed to directly control the heat transfer fluid temperature which indirectly controls the mold temperature response.

The mold heating and cooling equipment is used to control temperature profiles of the mold and the part during manufacturing. The metal temperature is indirectly controlled by controlling the heat transfer fluid temperature in each zone. Prior attempts to use metal temperature as the process variable in feedback loops have resulted in excessive overshoots and oscillations during steady-state operation. Both of those faults led to using the fluid temperature as the process variable and cycling the fluid temperature set point to cycle the mold temperature. The prime objective of this endeavor was to provide direct metal temperature control using the metal temperature as one of the process variables in any control scheme. Metal temperature set points instead of fluid temperature set points could then be prescribed to produce the desired time versus temperature profile for the mold. If the objective could be accomplished, approximately 90 percent of the time required to develop an acceptable set point recipe would be eliminated.

Conventional strategies provided only marginal success as general purpose control schemes, because of the extreme range of dynamic responses demonstrated by the various molds. Slow mold temperature transient responses coupled with the discontinuities caused by the air gap formation were the primary reasons for failure to meet the mold temperature response criteria. The conventional strategies succeeded when the dynamic temperature response was fast, but still required an extensive amount of time to determine the optimum tuning parameters. The conventional strategies did not provide a general purpose scheme to be used on all molds.

Development of a new control algorithm based on the mold temperature and fluid temperature dynamic responses was started in late 1974. The technique used was to develop a math model of the control loop responses, convert the model to a pulse transfer function by using Z transforms, specify in pulse transfer function form the desired closed loop response, and solve for the controller. The closed loop response will respond as defined if the math model of the process is reasonably accurate in terms of the type of response and if the transient response is reasonably linear. If the process exhibits dead time, then the controller will be automatically dead time compensated because the dead time enters the controller

from the pulse transfer function of the process and the specified closed loop response.

Early attempts were made to analyze the control loop for stability using Z-transforms to develop a pulse transfer function for the process using conventional positional and velocity algorithms. The Routh-Hurwitz stability criteria as modified for digital loops was used to determine stability. The initial models provided minimal information.

Beginning in early 1976, new control algorithms were designed using the procedure outlined above. The closed loop response was designed to be a first order lag plus dead time to set point changes. Algorithms were designed using the metal temperature as the only feedback variable and using a first through third order process model for controller design. Cascaded fluid and metal temperature control schemes were also designed.

Computer simulation programs were written to analyze the control algorithms off-line before implementation on the real-time computer. Computer programs were also written to determine the pole and zero locations of the controllers in the Z-plane for stability constraint checks. The control algorithms as simulated indicated that both the metal feedback only and cascaded metal and fluid schemes worked. Root location analysis indicated that the algorithms were stable.

The control strategy was then implemented on the real-time computer. The initial test runs of the metal feedback only control algorithms were failures. Investigations revealed that two model errors were made, yielding a nonrepresentative controller. First, the incremental memory module was modeled as a zero order hold, but was later determined to be a pure integrating amplifier. Second, the fluid loop was modeled as a first order lag, but in reality it is so nonlinear that the linear model was not satisfactory as a representation of its actual dynamics.

The incremental memory module error could be corrected in the derivation readily; however, the result is a differential controller which yields unsatisfactory results. Although the nonlinearity of the fluid loop is not correctable by itself, the closed loop fluid response under proportional-integral-derivative (PID) velocity control was nearly linear. If the fluid loop is linearized by using a PID controller, then the two problems are automatically solved. The control configuration would be cascade, with the outer loop the designed control loop and the inner loop a PID loop. That configuration was implemented with success.

A testing program was initiated to determine the most suitable sample period for the outer loop and to determine the gain values

to be used. The testing program also allows a means of determining the effect of parameter changes on the control loop performance.

The control algorithm has already been used to make off-schedule order parts and has performed satisfactorily. The testing program will be continued to learn as much about the algorithm as possible before implementation as a standard control scheme for compression/transfer molds.

THIS PAGE
WAS THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

DISCUSSION

SCOPE AND PURPOSE

Mold heating and cooling equipment is used by Bendix Kansas City to control the temperature of the mold and part during manufacturing. That control has been accomplished by temperature cycling the heat transfer fluid in the control loop. The major objective of this endeavor was to provide temperature control using the mold temperature as the prime feedback variable. The development of the new control algorithm provides the vehicle to satisfy that goal.

PRIOR WORK

All prior work was reported in five interim reports.¹⁻⁵

ACTIVITY

Controller Design

Work began in late 1974 to develop a new control algorithm to directly control mold temperature. That new control algorithm would use the actual open loop dynamics of the mold and fluid loop transient responses to calculate its parameters.

Control of the mold temperature to date has been achieved by controlling the fluid loop temperature.¹⁻⁵ Hot or cold central system fluid is allowed to enter the control loop through control valves to increase, decrease, or maintain the fluid loop temperature. The fluid temperature is sensed using a thermocouple whose output is converted to a milliamperage signal and sent to the computer analog-to-digital (A/D) converter. The converted signal is compared with the desired set point, and the difference is used by the velocity proportional-integral-derivative (PID) controller to compute a change in valve position. The change in valve position is sent to the incremental memory module (IMM) where it is converted to a change in memory output. The change in IMM output causes the control valves to change position, allowing more or less hot or cold fluid into the control loop.⁵

The objective of this endeavor is to close the loop around the mold metal temperature. With closed loop metal temperature control, recipe development time will be reduced by approximately 90 percent. Metal temperature set points will then be specified in the recipe instead of fluid temperature set points.

Conventional control strategies using PID controllers have been investigated and for general purpose application have been found to be unacceptable.⁵

The first steps in developing a new metal temperature controller were to learn to use and apply Z-transforms in pulse transfer function block diagrams. A great deal of information already existed concerning the fluid loop under computer control; therefore, the first analysis attempts were concerned with the fluid loop under PID control. The process dynamics are first converted from the continuous domain (S-plane) to the pulse domain (Z-plane) by use of the Z-transform. Table 1 lists some of the common transforms used in conversion from the S to Z domain.

In this procedure, the plant and hold continuous transfer functions are multiplied together to obtain an overall continuous process transfer function, which then is transformed into a pulse transfer function by use of the Z-transform tables or partial fraction expansion. The closed loop pulse transfer function is then derived in general block terms. If the controller is a PID type, the characteristic equation is analyzed for root location if the PID constants are known, or the Routh-Hurwitz array is constructed if the PID constants are unknown; if the controller is unknown, the closed loop response is specified and the controller pulse transfer function is derived.

To illustrate the procedure, the closed loop system of Figure 1 will be assumed.

The various components of Figure 1 are:

R(Z)	Set point,
E(Z)	Error,
D(Z)	Controller,
M(Z)	Controller output,
H(S)	Hold device,
G(S)	Plant dynamics, and
T	Computer sample period.

The first step is to combine the hold and plant dynamics and then convert the product to the Z domain.

$$HG(Z) = \mathcal{Z} \{H(S)G(S)\} \quad (1)$$

HG(Z) in Equation 1 is the pulse transfer function of the process. Because the process plant and the rest of the control system are all represented by pulse transfer functions, the block diagram of Figure 1 becomes that of Figure 2. Using block diagram algebra, the following relationships can be expressed.

Table 1. Common Z-Transformations

Time Function	Laplace Transform	Z-Transform
$U(t)$	$1/S$	$1/(1-Z^{-1})$
t	$1/S^2$	$TZ^{-1}/(1-Z^{-1})^2$
e^{-at}	$1/(S+a)$	$1/(1-e^{-aT}Z^{-1})$
$\frac{1}{b-a} (e^{-at} - e^{-bt})$	$\frac{1}{(S+a)(S+b)}$	$\frac{1}{b-a} \frac{1}{(1-e^{-aT}Z^{-1})} - \frac{1}{(1-e^{-bT}Z^{-1})}$

$$E(Z) = R(Z) - C(Z) \quad (2)$$

and

$$\frac{C(Z)}{E(Z)} = D(Z)HG(Z) \quad (3)$$

Substituting Equation 2 into 3 and rearranging gives

$$\frac{C(Z)}{R(Z)} = \frac{D(Z)HG(Z)}{1 + D(Z)HG(Z)} \quad (4)$$

Equation 4 is the closed loop pulse transfer function. The denominator of Equation 4 when equated to zero is the characteristic equation.

$$1 + D(Z)HG(Z) = 0 \quad (5)$$

In Equation 5, $D(Z)$ is the loop controller. If the loop controller is a PID velocity controller, then its pulse transfer function is

$$D(Z) = \frac{M(Z)}{E(Z)} = K_P \left[1 + \frac{K_I T}{60(1-Z^{-1})} + \frac{K_D 60}{T}(1-Z^{-1}) \right] \quad (6)$$

where

K_P = Proportional gain (percent/percent),

Figure 1. Typical Digital Closed Loop Block Diagram

Figure 2. Typical Digital Control Loop After Conversion to Pulse Transfer Form

K_I = Reset Rate (min),

K_D = Rate (min), and

T = Sample period (s).

Assuming that the fluid loop dynamics can be approximated by a first order lag and the hold device is a pure integrator, the process pulse transfer function is

$$HG(Z) = \mathcal{Z}\{H(S)G(S)\} = \mathcal{Z}\left\{\frac{1}{S} \frac{1/\tau}{S+1/\tau}\right\} \quad (7)$$

where τ is the fluid time constant. From Table 1, the Z transform of Equation 7 is

$$HG(Z) = \frac{1/\tau}{1/\tau} \left[\frac{1}{(1-Z^{-1})} - \frac{1}{(1-e^{-T/\tau}Z^{-1})} \right] \quad (8)$$

where $a = 0$ and $b = 1/\tau$.

Substituting Equations 6 and 8 into 5 and rearranging:

$$1 + \frac{Z^{-1}(1-e^{-T/\tau})}{(1-Z^{-1})(1-e^{-T/\tau}Z^{-1})} - \frac{3600\frac{K_D}{T}(1-Z^{-1})^2 + K_P 60(1-Z^{-1}) + K_I T}{60(1-Z^{-1})} = 0 \quad (9)$$

Letting

$$A = 3600\frac{K_D}{T} (1-e^{-T/\tau}) \quad (10)$$

$$B = 60K_P (1-e^{-T/\tau}) \quad (11)$$

$$C = K_I T (1-e^{-T/\tau}) \quad (12)$$

$$D = e^{-T/\tau} \quad (13)$$

then Equation 9 after rearrangement is

$$60Z^3 - [120+60D-A-B-C]Z^2 + [60+120D-B+2A]Z - [60D-A] = 0 \quad (14)$$

The roots of Equation 14 can be determined if the PID constants are known. The Routh array can be constructed if the constants are unknown. Before the array is constructed, the ratio of $(r+1)/(r-1)$ must be substituted for each Z term, the resulting equation expanded, and like terms collected. The stable ranges for the PID constants can then be determined by solving the resulting equations generated from constructing the Routh-Hurwitz array.

The above procedure was used to evaluate the gain ranges for fluid loop feedback using a velocity PID controller. The results for the fluid loop coincided with field experience. No new information was gained from the analyses. The Routh-Hurwitz procedure becomes very involved for complex control configurations, resulting in limited usefulness as an analytical tool.

It was decided that a new controller would be designed when the cascade tests using PID controllers proved to be only marginally successful.⁵ The new controller would replace the conventional PID controllers. The procedure used was first suggested by Dahlin⁷ and is known as "Dahlin's Method." It proceeds as follows. Equation 4 is rearranged, solving for the controller $D(Z)$.

$$D(Z) = \frac{1}{HG(Z)} \cdot \frac{C(Z)/R(Z)}{1-C(Z)/R(Z)} \quad (15)$$

HG(Z) in Equation 5 contains the process dynamics which are known or can be empirically determined. The unknown in Equation 15 is the closed loop response C(Z)/R(Z). If the desired closed loop response is specified for C(Z)/R(Z), then the derived controller will force the control system to respond as desired, assuming that the process is linear. The desired closed loop response for product considerations is a fast rise time with overshoot limited to a maximum of 1.5 percent (2°C). Overshoot is the limiting criteria for selection of the desired closed loop response. A first order lag plus dead time exhibits fast response with no overshoot to step changes in the forcing function (set point).

Because the closed loop response is a first order lag plus dead time, it remains only to determine the order of the process plant to most closely represent the actual open loop response. The digital controller derivations for systems having a zero order hold (ZOH) in conjunction with a first, second, and third order plus dead time process response are presented in Appendices A, B, and C respectively.

Two cascade configurations were also derived. Appendix D presents a dual digital controller derivation. Appendix E presents the cascaded configuration currently in use; that configuration uses a PID inner loop and a digitally derived outer loop.

The two most important points concerning the derived controllers are that the parameters for the controller are calculated from the actual process dynamics and that the control algorithms are automatically dead time compensated because the original process contained dead time.

Controller Analysis and Simulation

To analyze digital control loops for stability, the root locations of the characteristic equation or the pole-zero location of the controller algorithm must be examined. In the S-plane, the poles and zeroes had to be on the left hand side of the imaginary axis for the response to be stable. Poles and zeroes on the right hand side of the S-plane result in an unstable, ever increasing response. In the Z-plane, the poles and zeroes of the left hand side S-plane map into a unit circle except for the area on the negative real axis.^{7,8} The shaded area of Figure 3 illustrates the stable region in the Z-plane. The right hand S-plane roots map outside the unit circle in the Z-plane, indicating that any root outside the unit circle is unstable. If the control loop utilizes a conventional PID controller, then the roots of the characteristic equation must be examined for known values of the PID constants. As an example, the roots of the third order polynomial of Equation 14 could be determined if the values of A, B, C, and D are known.

Figure 3. Unit Circle in Z Plane, Indicating Stable Region

In the case of the derived controller, however, the closed loop response is already defined. For the first order lag (Appendix A, Equation A-27), the closed loop response is Equation 16.

$$\frac{C(Z)}{R(Z)} = Z^{-N-1} \left[\frac{(1-e^{-mT/\lambda}) + (e^{-mT/\lambda} - e^{-T/\lambda})Z^{-1}}{1-e^{-T/\lambda}Z^{-1}} \right] \quad (16)$$

The characteristic equation for Equation 16 is:

$$Z - e^{-T/\lambda} = 0. \quad (17)$$

In Equation 17, the only root is located at $Z = e^{-T/\lambda}$; therefore, by definition, it is a stable closed loop response. In the case of the derived controllers, it is presumed that the process model is linear or nearly linear and that the model closely approximates the actual response. If the above constraints are true, then for positive values of sample period (T) and closed loop time constant

(λ), the root of Equation 17 will always be less than one. That implies a stable response, which would be expected if the closed loop really responds as a first order lag. Looking at the roots of the characteristic equation for a derived controller system is, therefore, meaningless.

The pole and zero locations of the controller are not meaningless, however. If a pole or zero lies on the negative real axis between 0 and -1, then ringing will result.⁷ Excessive oscillation of the controller output results from ringing poles or zeroes which causes wear to the manipulated variable. In the case of direct metal temperature feedback, the manipulated variable is the control valve. Excessive movement of the valve could lead to premature failure of the valve diaphragms and reduce the valve seat life expectancy. Smith⁷ suggests that when ringing poles are encountered, simply let $Z=1$, add that to the root, and multiply or divide the gain by the result. As an example, suppose the controller pulse transfer function is as described by Equation 18.

$$D(Z) = \frac{1.995(1+0.631Z^{-1})(1-0.922Z^{-1})}{(1+0.228jZ^{-2}+0.084Z^{-1})(1-0.228jZ^{-1})(1-Z^{-1})(1+0.657Z^{-1})} \quad (18)$$

One pole ($Z=0.657$) and one zero ($Z=0.631$) display the ringing characteristics. Letting $Z = 1$ in both cases results in the following.

$$D(Z) = \frac{1.964(1-0.922Z^{-1})}{(1+0.228jZ^{-1}+0.084Z^{-1})(1-0.228jZ^{-1}+0.084Z^{-1})(1-Z^{-1})} \quad (19)$$

One other important reason for examining the controller poles and zeroes is to insure an excess pole at $Z=1$. Smith points out that for a control loop to exhibit no offset (contains the integral mode) that the control algorithm must contain a factor $(1-Z^{-1})$ in the denominator that does not cancel with a like term in the numerator. The PID controller illustrates the principle of the excess pole. Examination of Equation 6 shows that the integral mode contains the factor $(1-Z^{-1})$ in the denominator which does not cancel with the other terms. The integral mode of the PID controller eliminates steady-state offset. The poles and zeroes of any derived controller must be examined to ensure the excess pole at $Z=1$.

The two important points each derived controller must display are that the roots of the controller must lie within the Z circle of radius 1, and that the denominator must contain at least one excess pole at $Z=1$ for zero steady-state offset.

To study the pole-zero location of the derived controllers, computer programs were developed to solve for the roots of the controller. Appendices F, G, and H contain the source listings for the controllers derived in Appendices A, D, and E. A typical derived controller (from Appendix A, Equation A-37) is as follows.

$$D_P(Z) = \frac{K_{CS} (1 - Z^{-1}D_1 - Z^{-2}D_2)}{1 - Z^{-1}F_1 - Z^{-2}F_2 - Z^{-N-1}F_3 - Z^{-N-2}F_4 - Z^{-N-3}F_5} \quad (20)$$

The number of roots must equal the order of the system; consequently, in Equation 20 there will always be 2 zeroes and $N + 3$ poles, where N is the integer number of sample periods (T) in the dead time (θ).

$$N = \theta/T \quad (21)$$

The order of the denominator not only depends on the dead time of the process but also on the sample period. The order of the system is 27 if the mold dead time is 45 seconds and the sample period is 2 seconds. A polynomial with an order of 27 is not readily reducible by anything other than a digital computer. The polynomial order is reduced only by increasing the sample period because the dead time is a physical constant.

The root finding computer programs allowed examination of the controller poles and zeroes for a set of mold dynamics (time constant, gain, and dead time) while automatically incrementing the closed loop time constants (λ_1 and λ_2) for a fixed sample period (T). The controller roots were examined at varying sample periods (2, 4, 8, 10, 16, 20, and 30 seconds) for various mold dynamics and plant configurations. Sample periods of 8, 16, and 30 seconds consistently produced roots outside the Z circle of radius 1. Roots outside the stable Z circle were obtained more frequently as the order of the process plant was increased. Roots lying on the negative real axis between 0 and -1 were commonplace. One cascade control option which produced consistent roots outside the Z circle was eliminated from further consideration by using the root finding programs.

Simulations of the derived controllers were desired before implementation on the real time computer. The controllers were first converted to the time domain. Conversion from the Z domain to the time domain proceeds as follows. Restating Equation 20,

$$D_P(Z) = \frac{M(Z)}{E(Z)} = \frac{K_{CS} (1 - D_1 Z^{-1} - D_2 Z^{-2})}{1 - F_1 Z^{-1} - F_2 Z^{-2} - F_3 Z^{-N-1} - F_4 Z^{-N-2} - F_5 Z^{-N-3}} \quad (22)$$

Cross multiplying Equation 22,

$$M(Z)(1-F_1Z^{-1}-F_2Z^{-2}-F_3Z^{-N-1}-F_4Z^{-N-2}-F_5Z^{-N-3}) = K_{CS}E(Z)(1-D_1Z^{-1}-D_2Z^{-2}) \quad (23)$$

In Equation 23, each Z term simply represents the time function delayed by the magnitude of the exponent; consequently, the time function of 23 is

$$M_n - F_1M_{n-1} - F_2M_{n-2} - F_3M_{n-N-1} - F_4M_{n-N-2} - F_5M_{n-N-3} = K_{CS}(e_n - D_1e_{n-1} - D_2e_{n-2}) \quad (24)$$

where n is the integer number of the current sample period. Solving Equation 24 for the controller output M_n :

$$M_n = K_{CS}(e_n - D_1e_{n-1} - D_2e_{n-2}) + F_1M_{n-1} + F_2M_{n-2} + F_3M_{n-N-1} + F_4M_{n-N-2} + F_5M_{n-N-3} \quad (25)$$

In Equation 25, the term e_{n-2} represents the magnitude of the error two sample periods ago. The term M_{n-N-1} represents the magnitude of the controller output N + 1 sample periods previously. N is the integer number of sample periods in the dead time; consequently, the terms $n-N-i$, where $i = 1, 2$, or 3 , are the dead time compensation terms. Figure 4 illustrates Equation 25 in block diagram form. The blocks containing the term e^{-Ts} indicate a pure delay of one sample period.

Simulation programs similar to the conventional controller simulations⁵ were developed for each of the derived controller configurations. Appendices I and J contain the source listings of the first order plant digital controller and the cascaded digital controllers simulation programs. One major simplification was made to the previous simulation packages which resulted in significant impact. The fluid loop equations were eliminated, and a first order lag model was used in their place.

Figure 5 illustrates in block diagram form the simulation configuration. The first block past the sampler containing the terms K_O and τ_O is the first order lag representation of the fluid loop dynamics. The controller does not contain any term representing that fluid response because it was combined with the metal dynamics during the derivation. The ability of the controller to handle higher order systems that were approximated by lower order models was one of the items under investigation.

Figure 4. Block Diagram of Derived Controller for a First Order Plant

Figure 5. Mold Temperature Feedback, Digital Controller Simulation, First Order Plant, First Order System

The configuration of the controller is the velocity form of Equation 25. The extra gain term, K_c , was an added variable to allow manipulation of the calculated gain term, K_{cs} , to account for the nonlinearities and modeling inaccuracies. Appendices B and E show the extra gain term in the derivation. Table 2 defines the parameters in Figure 5. The simulation program was structured to allow nonlinear responses of the process plant. Parameters P_{30} and P_{31} are the parameters used by the program to compute the mold temperature response. Those parameters are redefined based on the direction of the set point change in subroutine SPADJ. The heating constants (P_{26} and P_{27}) are used for positive changes in set point, and cooling constants (P_{24} and P_{25}) are used for negative changes in set point. Parameters P_{12} through P_{14} are used to calculate the controller constants P_1 through P_9 .

The controller constants can be calculated from the slowest, fastest, or average mold time constants and can be totally independent of the actual response dynamics used in the simulation. The program was structured in that manner because the actual responses were known to be nonlinear but were modeled as linear responses. The simulation program would allow assessment of the controller's capability to respond to nonlinear feedback information.

Computer runs were made varying the heating and cooling time constants (P_{27} and P_{25}) and at the same time varying the mold time constant (P_{12}) used to calculate the controller parameters for various closed loop time constants (P_{18}). Figures 6 through 9 illustrate the type of response obtained from the simulation. The following parameters were held constant during the simulations depicted in Figures 6 through 9:

$$K_o = P_{22} = 1.0^\circ\text{F}/^\circ\text{F} \text{ } (^\circ\text{C}/^\circ\text{C}),$$

$$\tau_o = P_{23} = 30.0 \text{ s},$$

$$K_{m,c} = P_{24} = 1.07^\circ\text{F}/^\circ\text{F} \text{ } (^\circ\text{C}/^\circ\text{C}),$$

$$K_{m,h} = P_{26} = 0.963^\circ\text{F}/^\circ\text{F} \text{ } (^\circ\text{C}/^\circ\text{C}),$$

$$\tau_{m,c} = P_{25} = 95.0 \text{ s},$$

$$\tau_{m,h} = P_{27} = 168.0 \text{ s},$$

$$K_{m,D} = P_{12} = 1.0^\circ\text{F}/^\circ\text{F} \text{ } (^\circ\text{C}/^\circ\text{C}),$$

$$\tau_{m,D} = P_{13} = 95 \text{ s}, \text{ and}$$

$$\theta_m = P_{14} = 10 \text{ s}.$$

Table 2. Mold Temperature Feedback, Digital Controller
Simulation: First Order Plant, First Order
System Parameter List

P_1	Controller Constant, D_1
P_2	Controller Constant, D_2
P_3	Variable Gain, K_c
P_4	Fixed Closed Loop Gain, K_{cs}
P_5	Controller Constant, F_1
P_6	Controller Constant, F_2
P_7	Controller Constant, F_3
P_8	Controller Constant, F_4
P_9	Controller Constant, F_5
P_{10}	Not Used
P_{11}	5.5°F/Percent
P_{12}	Mold Gain for Controller, K_m
P_{13}	Mold Time Constant for Controller, λ_m
P_{14}	Mold Dead Time for Controller, θ_m
P_{15}	System Sample Period, T
P_{16}	0.18181818
P_{17}	-59.090909
P_{18}	Closed Loop Time Constant, λ
P_{19}	Ratio $\theta_m/T = P_{14}/P_{15}$
P_{20}	90°F
P_{21}	550°F

Table 2 Continued. Mold Temperature Feedback, Digital
Controller Simulation: First Order
Plant, First Order System Parameter
List

P ₂₂	Oil Gain, K_O (Normally = 1)
P ₂₃	Oil Time Constant, λ_O (Normally = 0.0 for theoretical simulation)
P ₂₄	Cooling Gain, K_M
P ₂₅	Cooling Time Constant, λ_M
P ₂₆	Heating Gain, K_M
P ₂₇	Heating Time Constant, λ_M
P ₃₀	Gain Used in DIFFEA, K
P ₃₁	Time Constant Used in DIFFEQ, λ

The closed loop time constant ($\lambda = P_{18}$) was varied from 100 to 400 seconds in increments of 100 seconds. The time constant used for the response calculation in DIFFEQ would be the heat-up time constant ($T_{m1,h}$) because the only set point change was positive. The parameters were calculated based on the cool-down time constant ($T_{m1,c}$). Figure 6 shows a slight overshoot of the metal temperature when the closed loop time constant was almost equal to the design time constant ($\tau_{m,D} = 95$ s, $\lambda = 100$ s). The successive plots indicate that as the closed loop time constant is increased the rise time of the closed loop response decreases, overshoot disappears, and the settling time increases. Before simulation, all the configurations were analyzed for pole and zero locations to ensure that the roots were inside the Z-circle. Unstable responses as indicated by the pole-zero location were also simulated and resulted in the mold temperature going to one temperature extreme or the other (hot or cold system temperature).

The simulation results were highly encouraging from the design stand-point. The closed loop response did in fact appear to respond as a first order lag plus dead time even though

The simulation model was a higher order response than the controller design,

Figure 6. Digital Controller, First Order System, Run 1

Figure 7. Digital Controller, First Order System, Run 2

Figure 8. Digital Controller, First Order System, Run 3

Figure 9. Digital Controller, First Order System, Run 4

The time constant used for controller parameter calculation was different than the simulation model, and

The simulation mold temperature model was nonlinear while the controller was designed from a linear model.

Based on the simulation results, it was decided to implement the control algorithm of Equation 25 in its velocity form on the real-time computer. The velocity form of Equation 25 (Appendix A, Equation A-39) is

$$\begin{aligned}\Delta M_n = & K_{cs}(\Delta e_n - D_1 \Delta e_{n-1} - D_2 \Delta e_{n-2}) + F_1 \Delta M_{n-1} + F_2 \Delta M_{n-2} + F_3 \Delta M_{n-3} \\ & + F_4 \Delta M_{n-4} + F_5 \Delta M_{n-5}.\end{aligned}\quad (26)$$

The constants for Equation 26 are as follows.

$$K_{cs} = \frac{1 - e^{-mT/\lambda}}{K_p (1 - e^{-mT/\tau_m})} \quad (27)$$

$$D_1 = e^{-T/\tau_m} - \frac{e^{-mT/\lambda} - e^{-T/\lambda}}{1 - e^{-mT/\lambda}} \quad (28)$$

$$D_2 = \frac{e^{-T/\tau_m} (e^{-mT/\lambda} - e^{-T/\lambda})}{1 - e^{-mT/\lambda}} \quad (29)$$

$$F_1 = e^{-T/\lambda} - \frac{e^{-mT/\tau_m} - e^{-T/\tau_m}}{1 - e^{-mT/\tau_m}} \quad (30)$$

$$F_2 = \frac{e^{-T/\lambda} (e^{-mT/\tau_m} - e^{-T/\tau_m})}{1 - e^{-mT/\tau_m}} \quad (31)$$

$$F_3 = 1 - e^{-mT/\lambda} \quad (32)$$

$$F_4 = (e^{-mT/\lambda} - e^{-T/\lambda}) - \frac{(e^{-mT/\tau_m} - e^{-T/\tau_m})(1 - e^{-mT/\lambda})}{(1 - e^{-mT/\tau_m})} \quad (33)$$

$$F_5 = \frac{(e^{-mT/\lambda} - e^{-T/\lambda})(e^{-mT/\tau_m} - e^{-T/\tau_m})}{(1 - e^{-mT/\tau_m})} \quad (34)$$

The parameters of Equations 27 through 34 are

τ_m = Open loop mold time constant (s),

λ = Closed loop time constant specified by user (s),

T = Computer sample period (s),

m = Fractional position of the dead time between sample periods, and

K_p = Mold temperature steady state gain °C/°C (°F/°F).

The value of m in the above equations is determined as follows. Suppose the mold dead time (θ_m) is 53 seconds, and the computer sample period (T) is 10 seconds. There are 5 integer sample periods (N) in the dead time:

$$N = \theta_m / T = 53 / 10 = 5 \quad (35)$$

which leaves 3 seconds of the dead time unaccounted. To account for the 3 seconds, the fractional position equation is

$$N = \theta_m / T = 53 / 10 = 5$$

$$m = 1 - \frac{\theta_m - NT}{T} \quad (36)$$

Solving Equation 36 for m :

$$m = 1 - \frac{53 - (5)(10)}{10} = 0.7. \quad (37)$$

The dead time can be expressed as

$$\begin{aligned} \theta_m &= NT + T - mT \\ &= (5)(10) + 10 - (0.7)(10) \\ &= 53 \text{ s.} \end{aligned} \quad (38)$$

While the software was being developed for the real time computer, cascade configurations of the control system were derived. The first cascade configuration derived resulted in totally unstable pole-zero root locations. It was determined that the block diagram representation of the fluid loop had to be restated in the time domain after a digital algorithm for the fluid loop was derived. Using the derived response for the fluid loop as specified in the Z-domain violated the plant pulse transfer function rules. That

is, the fluid and mold response must be combined in the time or Laplace domain and then converted to the pulse domain. The fluid loop was redefined as a first order lag in the Laplace domain after the inner loop controller was derived. The closed loop fluid response was defined as a first order lag and, from previous simulations, the closed loop appeared to react as a first order lag; consequently, the approximation was valid. The outer loop controller could then be derived after the fluid loop and metal dynamics were combined and a pulse transfer function formulated for the combined response (see Appendix D for derivation). The plant pulse transfer function is second order containing the fluid and metal temperature dynamics, both of which are first order approximations. The major loop closed loop response was designed to be a first order lag plus dead time. The outer loop controller equation (Appendix D, Equation D-64) is:

$$m_n = K_{cs2} (e_n^{-H_1} e_{n-1}^{+H_2} e_{n-2}^{+H_3} e_{n-3}) + P_1 m_{n-1} + P_2 m_{n-2} + P_3 m_{n-N} + P_4 m_{n-N'-1} + P_5 m_{n-N'-2} \quad (39)$$

The inner loop controller equation (Appendix D, Equation D-32) is:

$$\Delta m_{2,n} = K_{cs1} (\Delta e_{2,n-D_1} \Delta e_{2,n-1-D_2} \Delta e_{2,n-2}) + F_1 \Delta m_{2,n-1} + F_2 \Delta m_{2,n-2} + F_3 \Delta m_{2,n-N-1} + F_4 \Delta m_{2,n-N-2} + F_5 \Delta m_{2,n-N-3} \quad (40)$$

The constants in Equation 39 (K_{cs2} , H_1 through H_3 , and P_1 through P_5) are functions of the mold time constant (τ_m), mold gain (K_m), mold dead time (θ_2), inner loop closed loop time constant (λ_1), and outer loop closed loop time constant (λ_2). The constants in Equation 40 (K_{cs1} , D_1 and D_2 , F_1 through F_5) are functions of the fluid loop time constant (τ_ϕ), the fluid loop dead time (θ_1), and the inner closed loop time constant (λ).

The pole-zero computer program (Appendix G) indicated stable root locations for the derived controllers of Equation 39 and 40. The stable roots were a function of sample period. Sample periods of 8, 16, and 30 seconds produced some unstable roots, while sample periods of 2, 4, 5, 10, and 20 produced stable roots.

A computer program (Appendix J) was written to simulate the cascade configuration. Figure 10 illustrates in block diagram form the simulation configuration. Table 3 defines the parameters in Figure 10. The fluid loop was approximated by a first order lag plus dead time response as in the previous simulation.

Table 3. Cascaded Digital Control System Parameter List

P_1 through P_3	Calculated Controller Constants, Major Loop
P_4	Inputted Controller Gain, Major Loop
P_5	Calculated Controller Gain, Major Loop
P_6 through P_{10}	Calculated Controller Constants, Major Loop
P_{11} and P_{12}	Calculated
P_{13}	Inputted Controller Gain, Minor Loop
P_{14}	Calculated Controller Gain, Minor Loop
P_{15} through P_{19}	Calculated Controller Constants, Minor Loop
P_{20}	5.5°F/Percent (Percent to °F Constant)
P_{21}	Lower Temperature Limit (~80°F)
P_{22}	Upper Temperature Limit (~575°F)
P_{23}	Fluid Loop Gain, K_o
P_{24}	Fluid Loop Time Constant, λ_o
P_{25}	Mold Loop Gain, K_m
P_{26}	Mold Loop Time Constant, λ_m
P_{27}	0.1818182 Percent/°F (°F to Percent Constant)
P_{28}	-59.09091 Percent (°F to Percent Constant)
P_{29}	Sample Period, T (s)
P_{30}	Minor Loop Closed Loop Time Constant, λ_1
P_{31}	Major Loop Closed Loop Time Constant, λ_2
P_{32}	Mold Heat Up Time Constant, $\lambda_{m,H}$
P_{33}	Mold Cool Down Time Constant, $\lambda_{m,c}$
P_{34}	Mold Heat Up Gain, $K_{m,H}$
P_{35}	Mold Cool Down Gain, $K_{m,c}$
P_{36}	325°F (Percent to °F Constant)

The simulation results were favorable when comparing the expected response of the closed loop to the actual simulated response. Figures 11 through 14 graphically illustrate typical simulations. The inner loop closed loop time constant (λ_1) was held constant at 10 seconds while the outer loop closed loop time constant (λ_2) was varied from 100 to 400 seconds in 100 second increments. The plots show the set point, outer loop controller output, inner loop controller output, fluid temperature, and metal temperature. The plant configuration in the outer loop controller design was second order. From the second order plant simulations (Appendix B design), the metal temperature response was expected to perform more as second order than as first order to set point changes, even though the closed loop was specified as first order in the derivation. The metal responses shown in Figures 11 through 14 illustrate that phenomenon. The metal temperature response tends to damp out as the outer loop closed loop time constant is increased.

The same nonlinear options and alternate controller design specifications were carried over in the cascade simulation. Those options allowed investigating the effects that the nonlinear response would have on the linearly designed controllers.

The results of the simulations only enhanced the expectations for the success of the approach to meeting the endeavor goal of closed loop metal temperature control through digitally derived controllers.

The major impact of the stability analyses and the simulation effort was to start the software development for implementation of the derived digital algorithms. The analyses indicated that software development would not be a wasted effort.

Controller Implementation and Testing

Subroutines DCON1 and DLY from the simulation program of Appendix I were converted to assembly language routines and implemented on the real time computer. The software was installed on the computer in April 1977.

Prior to the first test of the controller, the mold dynamics for the control loops under test were empirically determined. The mold time constants, dead times, and gains were fed to the pole-zero batch computer program where the controller constants and root locations were calculated for varying closed loop time constants and sample periods. The roots as calculated by the program were stable with an occasional ringing pole.

The controller parameters were passed to the control algorithm, and the controller was activated. Initialization of the algorithm was carried out incorrectly by the software, but after $N + 1$ sample periods the effects of the initialization problems vanished. The control system did not respond. The set point was changed to

Figure 11. Cascade Simulated Digital Controller (LMJ=100, LMN=10), Run 1

Figure 12. Cascade Simulated Digital Controller (LMJ=200, LMN=10), Run 4

Figure 13. Cascade Simulated Digital Controller (LMJ=300, LMN=10), Run 7

Figure 14. Cascade Simulated Digital Controller (LMJ=400, LMN=10), Run 10

cause a large error perturbation, and the control system still did not respond. Increasing the artificial gain term caused the controller to respond, but it did not control correctly. The controller was deactivated. A mold response was assumed, and the algorithm output was hand calculated. The hand calculations confirmed that the computer calculations were too small in magnitude to properly adjust the control valves. The positional form of the algorithm was then installed. The output was of the correct magnitude, but resulted in system limit cycling. Limit cycling resulted because the algorithm form was positional, which saturated the hardware, causing either full hot or full cold fluid to enter the fluid control loop. It was at that point that modeling errors were suspected, because the algorithm was not performing as predicted by simulation.

The configuration of the incremental memory module (IMM) was the first suspect area. The hardware schematic revealed that the IMM was not equivalent to a zero order hold (ZOH) as was assumed in the algorithm derivation. The IMM, unlike the zero order hold which uses positional input information, uses the derivative of the position. When the transfer function for an integrator is used in place of the ZOH, the resulting controller equation for a first order plant becomes

$$D(Z) = \frac{M(Z)}{E(Z)} = \frac{K_{cs} (1-Z^{-1}D'_1 - Z^{-2}D'_2 + Z^{-3}D'_3)}{1-Z^{-1}F_1 - Z^{-2}F_2 - Z^{-N-1}F_3 - Z^{-N-2}F_4 - Z^{-N-3}F_5} \quad (41)$$

The denominator of Equation 41 is the same as Equation 22, but the numerator is modified by the factor $(1-Z^{-1})$. The pole-zero root locations are the same for the two equations, except for the extra zero at $Z=1$ for Equation 41. The extra zero at $Z=1$ cancels the only pole at $Z=1$, which means there is no integral mode for the algorithm of Equation 41. In the PID velocity algorithm, the integral mode provides the only constant source for set point referencing. That same referencing exists in Equation 39, because

$$1 - D_1 - D_2 \neq 0 \quad (42)$$

while in Equation 41,

$$1 - D'_1 - D'_2 + D'_3 = 0. \quad (43)$$

The significance of Equation 42 is that as long as an error exists the output of the controller will always change trying to reduce the error to zero, and an error will always exist as long as the process variable does not equal the set point. The significance

of Equation 43 is that, from the third sample period on after a set point change, the set point vanishes from the equation. To illustrate, the following substitution will be made for the error term (e):

$$e_i = SP_i - PV_i \quad (44)$$

where

SP_i = Set point at the i^{th} sample period, and

PV_i = Process variable at the i^{th} sample period.

After cross multiplying and converting to the time domain, Equation 41 becomes:

$$m_n = K_{cs} (e_n - D'_1 e_{n-1} - D'_2 e_{n-2} + D'_3 e_{n-3}) + F_1 m_{n-1} + F_2 m_{n-2} + F_3 m_{n-3} + F_4 m_{n-4} + F_5 m_{n-5} \quad (45)$$

Using only the error terms of Equation 45 and substituting in Equation 44, the equation becomes:

$$m_n = K_{cs} [(SP_n - PV_n) - D'_1 (SP_{n-1} - PV_{n-1}) - D'_2 (SP_{n-2} - PV_{n-2}) + D'_3 (SP_{n-3} - PV_{n-3})] + \dots \quad (46)$$

After the third sample period, the magnitude of the set point will be constant ($SP = SP_n = SP_{n-1} = SP_{n-2} = SP_{n-3}$); therefore, Equation 46 becomes:

$$m_n = K_{cs} [-PV_n + D'_1 PV_{n-1} + D'_2 PV_{n-2} - D'_3 PV_{n-3} + SP (1 - D'_1 - D'_2 + D'_3)] + \dots \quad (47)$$

Because the term $(1 - D'_1 - D'_2 + D'_3)$ is the relationship of Equation 43 and is zero, Equation 47 reduces to:

$$m_n = K_{cs} [-PV_n + D'_1 PV_{n-1} + D'_2 PV_{n-2} - D'_3 PV_{n-3}] + \dots \quad (48)$$

Equation 48 contains no reference to set point and, therefore, the output of the controller will wander without regard to the reference signal. The result of the above is that, with a pure integrating device as the output hardware, the design techniques used to derive

the controller will result in a derivative output that will allow steady-state offset. For most applications, including mold temperature control, steady-state offset is not allowed. The PID control algorithm, which is not derived from the plant dynamics, does contain an integral mode with constant reference to set point, resulting in zero steady-state offset when used with IMM type hardware.

The simulation program did not point out that discrepancy because of the model error and the use of the velocity form of the position controller. That is, the derivative was taken after the controller was designed which adds the term $(1-Z^{-1})$ to both the numerator and denominator. Had the oil loop not been approximated by a first order lag, the discrepancy would probably have been detected because of the fluid loop nonlinearity.

The problem at that point became one of how to overcome the hardware constraints. The fluid loop under PID control can be approximated by a second order underdamped model.

A cascaded control system with a PID inner loop and a digitally designed outer loop would overcome the hardware problems because the PID controller functions quite satisfactorily with the IMM hardware. The digital controller output would be the set point for the inner PID control loop. A further simplifying assumption was made: that the fluid loop could be approximated by a first order lag whose time constant can be determined from the closed loop natural frequency (ω_n) and damping ratio⁷ (ξ).

$$\tau = 1/\xi\omega_n. \quad (49)$$

The natural frequency and damping ratio of the closed loop can be determined experimentally by measuring the time to peak and the peak magnitude of the first overshoot for step changes in the fluid temperature set point. The damping coefficient can be determined from the percent overshoot equations:⁷

$$\xi = \sqrt{\frac{\{\ln[(PV-SP)/SP]\}^2}{\{\ln[(PV-SP)/SP]\}^2 + \pi^2}} \quad (50)$$

where

PV = Peak value of the fluid temperature, and

SP = Set point.

The natural frequency (ω_n) in rad per unit time is ⁷

$$\omega_n = \frac{\pi}{t_p \sqrt{1 - \xi^2}} \quad (51)$$

where

t_p = the time difference between the start of the response and the peak magnitude (PV) of the first cycle.

With the closed loop fluid response approximated by a first order lag, the controller derivation for the outer loop proceeds as outlined in the second half of the cascade derivation of Appendix D. The entire derivation is presented in Appendix E, but it is essentially the same as the second order plant of Appendix B. The outer loop controller (Appendix E, Equation E-49) is

$$m_n = K'K_{cs} (e_n - H_1 e_{n-1} - H_2 e_{n-2} + H_3 e_{n-3}) + P_1 m_{n-1} - P_2 m_{n-2} + P_3 m_{n-N} + P_4 m_{n-N-1} - P_5 m_{n-N-2} \quad (52)$$

where

$$\delta\tau = \tau_m - \lambda \quad (53)$$

$$X_1 = e^{-MT/\tau_m} \quad (54)$$

$$X_2 = e^{-MT/\lambda_1} \quad (55)$$

$$X_3 = e^{-MT/\lambda_2} \quad (56)$$

$$Y_1 = e^{-T/\tau_m} \quad (57)$$

$$Y_2 = e^{-T/\lambda_1} \quad (58)$$

$$Y_3 = e^{-T/\lambda_2} \quad (59)$$

$$C_1 = X_1 - X_2 \quad (60)$$

$$C_2 = Y_2 X_1 - Y_1 X_2 \quad (61)$$

$$C_3 = 1 - X_3 \quad (62)$$

$$C_4 = X_3 - Y_3 \quad (63)$$

$$K_{cs} = C_2 \delta \tau / C_1 K_m \quad (64)$$

$$H_1 = Y_1 + Y_2 - (C_4/C_3) \quad (65)$$

$$H_2 = (Y_1 + Y_2) \frac{C_4}{C_3} - Y_1 Y_2 \quad (66)$$

$$H_3 = Y_1 Y_2 C_4 / C_3 \quad (67)$$

$$P_1 = Y_3 + C_2 / C_1 \quad (68)$$

$$P_2 = Y_3 (C_2 / C_1) \quad (69)$$

$$P_3 = C_3 \quad (70)$$

$$P_4 = C_4 - (C_3 C_2 / C_1) \quad (71)$$

$$P_5 = C_4 C_2 / C_1 \quad (72)$$

τ_m = Mold time constant (empirically determined),

λ_1 = Fluid loop time constant (empirically determined),

λ_2 = Closed loop time constant of outer loop (specified),

K_m = Steady-state mold gain (empirically determined),

M = Fractional position of fluid and mold dead times, and

N = Integer number of sample periods in the fluid and metal dead times plus two.

In Equation 50

$$1 - H_1 - H_2 + H_3 \neq 0 \quad (73)$$

and

$$P_1 - P_2 + P_3 + P_4 - P_5 = 1. \quad (74)$$

The significance of Equation 74 is that, when the error goes to zero, the output of the controller will be maintained at a constant level. That is the equivalent of the integral mode of a PID positional controller.

Because of the modeling errors, the simulation activity associated with the dual digital cascaded controllers was directly applicable to the new concept. The modeling errors essentially treated the inner loop as a first order lag which was the same as the new concept. The important point was how the outer loop performed, and the outer loop simulation indicated that the new algorithm should succeed. The controller software was modified to the configuration of Equation 52 and is illustrated in block diagram form in Figure 15.

The first runs of the modified software were encouraging but still displayed problems. The software uses integer arithmetic; consequently, truncation errors can be significant. A batch computer program was written which used both integer and floating point arithmetic. The integer arithmetic of the simulator was written as close to the actual real time software as possible. An actual metal temperature response was then input to the simulator and both output types were computed. The integer output varied significantly from the floating point output, which indicated that truncation errors were causing the problems. The simulator also indicated that noise on the input signal had a detrimental effect on the controller output. The simulator confirmed that the controller should work if the truncation errors could be eliminated.

An extra significant digit was added to the algorithm which corrected the truncation problems. The noise on the analog-to-digital (A/D) converter still impeded the performance of the algorithm. If the sample period were increased, then the effects of noise would be minimized, because the majority of the change in input signal would be from mold temperature changes. That is, the signal-to-noise ratio would be increased. Increasing the sample period acts like addition of a high frequency noise filter without distorting the actual signal. When the sample period of the outer loop was increased from 2 to 10 seconds, the algorithm began to perform as expected and as simulated.

The mold base used in the tests was MD-252503-700 with a pair of inserts added. The mold-open dynamics of the punch and cavity are listed in Table 4. The mold-open dynamics were determined as follows. The set point of the fluid loop was changed from 65 to 148°C. Prior to that, the temperature at one mold time constant was calculated:

$$T(\tau) = T_i + 0.632(T_F - T_i) \quad (75)$$

Figure 15. Cascade Block Diagram of Inner Loop PID, Second Order Plant

Table 4. Initial Algorithm Checkout, Mold Designing Time, 10/1/70, Open Dynamics, MD-252503-700, Mold Base

	Punch	Cavity
Heat Up		
Fluid Time Constant, τ_o (s)	18	16
Fluid Dead Time, θ_o (s)	15	10
Mold Time Constant, τ_m (s)	392	654
Mold Dead Time, θ_m (s)	43	35
Mold Gain ($^{\circ}\text{C}/^{\circ}\text{C}$)	0.98	0.98
Cool Down		
Fluid Time Constant, τ_o (s)	48	36
Fluid Dead Time, θ_m (s)	12	10
Mold Time Constant, τ_m (s)	400	465
Mold Dead Time, θ_m (s)	40	40
Mold Gain ($^{\circ}\text{C}/^{\circ}\text{C}$)	0.98	0.98
Inner Loop Controller Constants		
K_p	7.0	7.0
K_I	8.0	8.0
K_D	0.0	0.0

where

$T(\tau)$ = Mold temperature at one time constant ($^{\circ}\text{C}$),

T_i = Initial mold temperature ($^{\circ}\text{C}$), and

T_F = Final mold temperature ($^{\circ}\text{C}$).

The elapsed time (t_m) from the set point change to the calculated temperature is recorded. The metal and fluid dead times are measured and recorded. Finally, the time to the first fluid temperature

peak (t_p) and peak magnitude (PV) are noted. The time to peak (t_p) is adjusted by the fluid dead time, and the fluid loop time constant is then determined from Equations 49, 50, and 51. The measured elapsed time (t_m) is adjusted by the mold dead time to yield the mold time constant. The process is then reversed to determine the cool down dynamics. That is, the set point is changed from 148°C back to 65°C, and the same parameters are recorded.

The nonlinearity⁴ of the heat-up and cool-down responses is again illustrated by the variation in cavity constants listed in Table 4.

A testing program was initiated to determine the set of time constants that should be used for parameter calculations, the sample period that provides optimum response, the effect that the ringing poles have on algorithm performance, and the effect that varying the gain has on closed loop performance.

Before the testing program began, an error analysis was performed to determine if steady-state offset for step changes in set point exist. The analysis also was made for ramp changes. Both analyses are presented in Appendix K. The results indicated that for step changes, zero steady-state offset should exist. For ramp changes in set point, a constant steady-state error will exist that is proportional to the ramp rate magnitude and computer sample period and inversely proportional to the added gain term.

To allow evaluation of the control system performance, the integral of the absolute error (IAE) was added to the software. The implemented form is:

$$IAE = T \sum_{i=1}^n |e_i| \quad (76)$$

where

T = computer sample period, and

e_i = error at the i^{th} sample period.

The IAE was kept for each zone. The IAE indices are zeroed and initiated when the production portion of the set point recipe is activated. Updating of the IAE is terminated when the last segment of the set point recipe times out.

The set point recipe is a set of segments that contain a set point, a ramp rate, and a segment duration time. The first two segments of the recipe are for mold warm up, and the remaining segments are for production cycling.

The recipe for compression molds is divided into two zones, each of which may have up to 61 production segments. Normally, the recipe would have supplied the fluid loop set points—which are now automatically calculated by the new algorithm. Thus, with the new algorithm, the set point recipe is the set points defining the mold metal temperature trajectory.

Two set point recipes were constructed for evaluation of the algorithm. Recipe 62 consists of two step changes, while recipe 63 consists of a dwell, a ramp, a dwell, and a ramp (Tables 5 and 6). Both zones of each recipe are identical; consequently, only zone 1 is listed in Tables 5 and 6. The first series of tests used the ramp recipe to manipulate the mold temperature set point.

Table 7 lists the results of the first series of tests. The optimum response would be the response with the lowest IAE and be non-oscillatory. The same recipe was run at varying sample periods, added gain magnitudes, and with and without the ringing poles or zeroes. The closed loop constants used were inner loop (λ_1) 20 seconds, outer loop (λ_2) 600 seconds and were held constant throughout the step and ramp tests. The heat-up mold dynamics were used for parameter calculations, and they also were held constant throughout the step and ramp tests.

The results obtained at sample periods of 5, 10, and 15 seconds gave the most satisfactory results. Elimination of the ringing poles or zeroes did not necessarily improve the closed loop response. In a great number of cases, eliminating the ringing pole/zero degraded the response, indicating the elimination effort was not worthwhile. The 15-second sample period produced the lowest IAEs, but were in general more oscillatory. The 5-second sample period gave more overdamped closed loop responses, which lengthens cycle times and is costly in production. The 10-second sample period produced the best consistent results.

The closed loop response dampened as the magnitude of the added gain term (henceforth called the gain) was decreased from 1.0 to 0.5. The damping effect is desirable if the response tends to be oscillatory, which appeared to be the case for gains of 1.0. Reducing the magnitude of the gain term appears to have the effect of increasing the closed loop damping factor.

The closed loop steady-state error during the ramp segments followed the trend predicted by the error analysis except for the effect of the sample period. The steady state error increased when the ramp rate (k) was increased or when the gain term (K') was decreased, as predicted by the error equation.

Table 5. Recipe 62: Step Tests

Segment	Ramp Rate (°C/min)	Set Point (°C)	Dwell (min)
Warm Up			
1	0.0	93	2.0
2	0.0	93	10.0
Production			
3	0.0	93	2.0
4	0.0	149	20.0
5	0.0	93	20.0

$$E(\infty) = \frac{kT}{K} \frac{f(1)}{(1-Y_3)} \quad (77)$$

In Equation 77, $f(1)$ is the factored numerator evaluated at $Z = 1$; therefore, the ratio of $f(1)/(1-Y_3)$ is a constant which is based on the mold dynamics but which will change with the sample period. The steady-state error for sample periods of 10 and 15 seconds during heat-up were 5.5 and 3.6°C respectively for a gain of 1.0. Evaluation of Equation 77 for the two sample periods indicate that the steady-state error at a 15-second sample period should be 1.007 times higher than at a 10-second sample period for identical ramp rates. In fact, the steady-state error decreased by a factor of 0.65. No explanation is offered for the discrepancy except that the actual process system is nonlinear. The steady-state offset decreased for increasing sample periods through 15 seconds during the tests.

The next series of tests were the step tests. The same algorithm parameters were used as during the ramp tests for the various sample periods to allow correlation with the first series of tests. Recipe 62 (Table 5) was used to manipulate the set point.

A recheck of the mold dynamics was made before the step tests were initiated because the responses appeared to be different as the ramp tests progressed. The mold dynamics were obtained in the same manner as previously outlined. The results are listed in Table 8.

Table 6. Recipe 63: Ramp Tests

Segment	Ramp Rate (°C/min)	Set Point (°C)	Dwell (min)
Warm Up			
1	0.0	93	1.00
2	0.0	93	0.05
Production			
3	0.0	93	2.00
4	5.6	148	10.00
5	0.0	148	10.00
6	-5.6	93	10.00
7	0.0	93	10.00

There is a significant difference between the dynamics listed in Table 4 and those listed in Table 8. The mold was not moved or in any way altered between the two test times. The only thing that had occurred was that the ramp tests had been carried out; therefore, the mold had been repeatedly heated and cooled. The air gap formation^{4,5} was the reason for the shift in dynamics. The punch dynamics increased across the board, while the cavity dynamics decreased. The dynamics used to calculate the parameters for the algorithm were those of Table 4, which were no longer representative of the actual response. The closed loop response for some of the test conditions was still quite satisfactory in spite of the deviation in dynamics, indicating that, as long as the gain term is added, the dynamics used for parameter calculation need only be close to the actual dynamics for satisfactory closed loop response. The insensitivity of the algorithm to changes in dynamics was encouraging from a general use stand point.

The step tests were carried out in the same manner as the ramp tests. That is, the sample period and gain were varied, and the ringing pole/zero were eliminated. Table 9 lists the results of the step tests.

The best results were again obtained at a sample period of 10 seconds. If the mold temperature overshoot more than 2°C, the response was considered oscillatory. In the case of the cavity, that was the

Table 7. Ramp Test Results, New Algorithm

Sample Period (s)	Added Gain Magnitude	Ringing Pole/Zero	Punch		Cavity	
			IAE (Percent- Second)	Oscillatory	IAE (Percent- Second)	Oscillatory
2	1.0	in	15129	No	12040	No
5	1.0	in	5850	No	5401	No
5	0.5	in	10511	No	9412	No
10	1.0	in	2989	No	2954	Yes
10	1.0	out	4984	Yes	4511	Yes
10	0.5	in	5385	No	5236	No
10	0.5	out	8775	No	7940	Yes
15	1.0	in	2540	Yes	2577	Yes
15	1.0	out	2242	Yes	2344	Yes
15	0.5	in	3792	No	3733	Yes
15	0.5	out	3868	No	2995	Yes
20	1.0	in	2371	Yes	2676	Yes
20	1.0	out	2234	Yes	2886	Yes
20	0.5	in	4001	No	3081	Yes
20	0.5	out	3240	No	4135	Yes

Table 8. Mid-Test Recheck of Mold Open Dynamics
for MD-252503-700 Mold Base

	Punch		Cavity	
	Heat Up	Cool Down	Heat Up	Cool Down
Fluid Time Constant (s)	21	28	15	27
Fluid Dead Time (s)	10	13	12	12
Mold Time Constant (s)	440	530	600	315
Mold Dead Time (s)	45	42	48	48
Mold Gain ($^{\circ}\text{C}/^{\circ}\text{C}$)	0.98	0.98	0.98	0.98

reason nearly all of the response categories were judged oscillatory. Highly oscillatory responses will have higher IAEs than slightly underdamped responses. The best results were achieved with 5 and 10 second sample periods.

Combining the two tests, a sample period of 10 seconds was judged to yield the best overall response. It was decided for compression mold temperature control that a 10-second outer loop sample period would be used. The inner loop sample period was maintained at 2 seconds. Tests conducted in 1973 indicated that a 2-second sample period was optimum for the PID controlled fluid loop.

With the outer loop sample period fixed at 10 seconds and the effort to eliminate the ringing pole/zero judged unnecessary, it remained only to determine the set of dynamic data that should be used to obtain the best closed loop response. The heat-up and cool-down mold dynamics are significantly different, especially on the cavity. It was decided to find out which set, when used for parameter calculation, yielded the optimum closed loop response. Eighteen tests were run using the fastest dynamics (minimum time constant), the slowest dynamics (maximum time constant), and the average of the two. The mold was left open on some tests and closed on a neoprene part on others. The gain was changed and both step and ramp recipes were run. Table 10 lists the results of those tests. Recipes 62 and 63 were used to manipulate the set point.

The integer number of sample periods in the dead time (N) was reduced by one in runs 4, 5, and 6 to check the software. It was suspected that the delayed controller output (M_{n-N}) was coming out

Table 9. Step Test Results, New Algorithm

Sample Period (S)	Added Gain Magnitude	Ringing Pole/Zero	Punch		Cavity	
			IAE (Percent- Second)	Oscillatory	IAE (Percent- Second)	Oscillatory
2	1.0	in	12915	No	12805	No
2	0.5	in	21211	No	19062	No
5	1.0	in	8660	No	10576	Yes
5	0.5	in	11282	No	11930	No
10	1.0	in	9861	No	13087	Yes
10	1.0	out	10184	Yes	12679	Yes
10	0.5	in	9284	No	11498	Yes
10	0.5	out	10330	Yes	10938	Yes
15	1.0	in	10601	No	13227	Yes
15	1.0	out	11690	Yes	12018	Yes
15	0.5	in	11047	No	12163	Yes
15	0.5	out	10650	No	12072	Yes
20	1.0	in	12465	Yes	12214	Yes
20	0.5	in	11340	No	12405	Yes

Table 10. Variable Time Constant Test Results, IAE

	Run 1	2	3	4*	5*	6*
Recipe**	S	S	S	S	S	S
Gain	1.0	0.3	0.3	0.3	0.3	0.5
Mold***	0	0	C	C	0	C
Time Constants (percent-second)						
Minimum (Fastest)						
Punch						
= 440S	10499†	9962	9509	9300	9851	5190
Cavity						
= 315S	12800†	11866†	8868	8702	11654	4947
Maximum (Slowest)						
Punch						
= 530S	11048†	9475	8823	8529	9862	4542
Cavity						
= 600S	11750†	10222†	7919	7728	10178	3948
Average						
Punch						
= 485S	9917†	9628	8937	8819	9406	4873
Cavity						
= 458S	11060†	10523†	8177	8022	10454	4349

*N reduced by 1 to check dead time software routine.

**S is step; R is Ramp.

***0 is open; C is closed.

†Overshoot greater than 2°C.

of the delay buffer at $n-N-1$ instead of $n-N$. The integer number of sample periods is manually fed to the algorithm, so simply decrementing its value by 1 allowed investigation to determine if the suspicion was true. Comparing runs 3 and 4 indicates that with the dead time integer decremented by 1, the IAE is lower for all three runs on both the punch and cavity. The same is also true when comparing runs 2 and 5, except for the punch on the maximum time constant run. The conclusion drawn from the runs is that the delay buffer contains one extra delay period. The impact of the extra delay period is negligible when comparing the average absolute errors. The average difference in the errors is only 0.16°C . The fact that an error of 10 seconds in the dead time buffer has such little effect on the closed loop response means that a 10-second model error after the software is fixed will also have minimal effect on the algorithm performance.

The other major point of interest from the tests is that using the slowest time constants for parameter calculations produces the best results when comparing average IAEs. The next best results were obtained from the average time constants, and the worst results were obtained from the fastest time constants. The results are also improved by reducing the gain term as previously indicated, which tends to dampen the closed loop response.

At the end of the variable time constant tests, the mold was disconnected from the pump package and sent to the tool room to have the cavity insert recut to a new configuration. The punch was not altered nor was the mating surface of the cavity insert to the mold base. The insert was reassembled on the mold base and was then reconnected to the pump package. The dynamics should not have changed since the insert to mold base surface was not altered. When a recheck was made, the cavity dynamics had changed drastically. The heat-up time constant changed from 600 seconds to 363 seconds while the cool-down time constant changed from 315 seconds to 270 seconds. The change was attributed solely to the contact resistance between the insert to mold base mating surfaces which causes the air gap formation. The cavity algorithm parameters were recalculated with the new dynamics. The mold using the new control algorithm for temperature control was used to meet an order commitment for 10 parts. The algorithm performed with good repeatability for all 10 parts. That run was considered the pilot production run for the algorithm.

The problem of the air gap formation and the change in dynamic response of the mold again became a major subject of concern. Mr. Bruce Hefner, Plastic Products Engineering, suggested using the same type of product used with cartridge heaters to improve the contact resistance by filling the gap with a conductive paste.

The following heat transfer example will illustrate the problem and its solution. Figure 16 will be assumed as the configuration of the inside of a mold punch. If the mold is assumed spherical in the region in question, then the overall heat transfer coefficient (U) is:

$$UA = \frac{1}{\frac{r_2 - r_1}{4\pi r_1 r_2 k_s} + \frac{r_3 - r_2}{4\pi r_2 r_3 k_g} + \frac{r_4 - r_3}{4\pi r_3 r_4 k_s}} \quad (78)$$

The following values for the various parameters will be assumed:

$$\begin{aligned} r_1 &= 76.2 \text{ mm} & k_s &= 0.156 \text{ W/m}\cdot\text{K} \\ r_2 &= 101.6 \text{ mm} & k_g &= 0.006 \text{ W/m}\cdot\text{K (mat'l)} \\ r_3 &= 101.727 \text{ mm} & k_g &= 0.0002316 \text{ W/m}\cdot\text{K (air)} \\ r_4 &= 127.127 \text{ mm} \end{aligned}$$

The thermal conductivity of the material filling the gap was assumed to be 0.006 W/m·K while for air it is 0.0002316 W/m·K. Equation 78 evaluated with no gap is:

$$UA = \frac{4\pi r_1 r_4 k_s}{r_4 - r_1} = \frac{4(3.14)(127)(76.2)(0.156)}{100(127 - 76.2)} = 3.73 \text{ W/K} \quad (79)$$

Equation 78 evaluated with air in the gap is

$$UA = 2.368 \text{ W/K} \quad (80)$$

and Equation 78 evaluated with a semi-liquid material in the gap is

$$UA = 3.52 \text{ W/K.} \quad (81)$$

The air gap represents a 37 percent reduction in the heat transfer capability from the no gap configuration. The material filled gap represents only a 6 percent reduction from the no gap configuration.

Adding a filler significantly increases the overall heat transfer coefficient which will help minimize the cause of the gap formation by reducing the temperature gradient. Initial tests using the new filler material indicate success in improving the heating and cooling response. The heat-up to cool-down time constant ratio is nearly 1.0 for both the punch and the cavity. To date, no before

Figure 16. Heat Transfer Configuration
With Air Gap Between Mold
Inserts

and after tests have been conducted to determine the percentage improvement; however, any before data would be questionable because of the variability of the contact resistance normally encountered.

After the order commitment was satisfied, mold base 6984967-005 with 4280-002 inserts was put on the pump package. A new closed loop algorithm was derived at the same time. The new algorithm was derived using a first order lag to describe the plant dynamics. That is, the incremental memory module, the fluid loop, and mold dynamics were combined and modeled as a first order lag. Appendix L contains the derivation.

The controller algorithm (Appendix L, Equation L-17) is:

$$m_n = K'K_{cs}(e_n - P_0 e_{n-1} - P_1 e_{n-2}) + P_5 m_{n-1} + P_8 m_{n-N-1} + P_9 m_{n-N-2} \quad (82)$$

where

$$K_{cs} = \tau_p / K_p e^{-mT/\tau_p} \quad (83)$$

$$P_0 = e^{-T/\tau_p} - \frac{e^{-mT/\lambda} - e^{-T/\lambda}}{1 - e^{-mT/\lambda}} \quad (84)$$

$$P_1 = \frac{e^{-T/\tau_p} (e^{-mT/\lambda} - e^{-T/\lambda})}{1 - e^{-mT/\lambda}} \quad (85)$$

$$P_5 = e^{-T/\lambda} \quad (86)$$

$$P_8 = 1 - e^{-mT/\lambda} \quad (87)$$

$$P_9 = e^{-mT/\lambda} - e^{-T/\lambda} \quad (88)$$

T = Computer sample period (s),

λ = Specified closed loop time constant (s),

τ_p = Plant time constants (s),

N = Integer number of sample periods in the plant dead time, and

m = Fractional position of the plant dead time between sample periods.

The reason for the unusual parameter subscripts is to allow utilization of the software designed for Equation 52, as implemented in Figure 15.

It was decided that a comparison test would be made to evaluate the new first order algorithm versus the second order algorithm that had previously been tested. The advantages of using the first order algorithm over the second order algorithm are:

Fewer algorithm parameters would require smaller amounts of core storage,

Plant dynamics can be easily and accurately determined by empirical measurements,

Software could be easily set up to determine plant dynamics automatically, and

Simulations of algorithms (Appendices A and B) indicate that first order plant design is less oscillatory than the second order plant design.

The dynamics of the mold used to analyze the two algorithms had not been determined. In 2 hours the plant dynamics were determined, the algorithm parameters calculated, a test recipe constructed, and the mold cycle started. The 2-hour elapsed time is approximately one-tenth the time required to develop a fluid temperature recipe to accomplish the same mold temperature trajectory. Table 11 lists the dynamics of the mold used.

The punch dynamics and the cavity dynamics are radically different. The cavity responds approximately 3.5 times faster than the punch. The difference in dynamics would normally be expected to cause problems in matching transient responses. No attempt was made to match transient responses because the punch closed loop time constant (λ) was specified as 700 seconds while the cavity was specified to be 400 seconds. The closed loop time constants were held constant between the two runs. Table 12 lists the steady-state offset during the ramp segments, the overshoot during the dwell segments, and the IAE for the cycle. The steady-state offset for the two algorithms is comparable, while the first order algorithm is better in overshoot. The IAEs are comparable, indicating very little difference between the two runs. From a simplicity standpoint, the first order algorithm performs slightly better.

The mold temperature responses of the punch and cavity for the two runs were digitized using computer graphics. Figures 17 through 20 illustrate the digitized responses.

The first order algorithm as indicated by simulation was less oscillatory than the second order algorithm. The figures indicate that during the dwell after heat-up, the second order algorithm was more oscillatory on both the punch and the cavity.

The order of the process plant does affect the closed loop behavior of the algorithm. The simulation programs and actual tests both indicate that as the order of the process plant is increased the oscillatory tendency of the closed loop increases even though the closed loop response is specified as a first order response in the derivation.

The algorithms are general in nature; therefore, conclusions can be drawn from the tests. The first order algorithm is more desirable because the performance is equal to or better than the second order version and because it is simpler to use.

Table 11. Mold Base 6984967-005 With Inserts
4280-002 Dynamic Data for Algorithm
Evaluation

	Punch		Cavity	
	Heat Up	Cool Down	Heat Up	Cool Down
Fluid Time Constant (s)	14	24	9	14
Fluid Dead Time (s)	10	15	13	10
Mold Time Constant (s)	674	594	185	184
Mold Dead Time (s)	16	22	27	16
Mold Gain ($^{\circ}\text{C}/^{\circ}\text{C}$)	0.96	1.0	1.0	1.0

ACCOMPLISHMENTS

The goal of accomplishing closed loop mold temperature control has been realized by designing and implementing a digital control algorithm specifically tailored to the design constraints of fast rise time and no overshoot. The algorithm uses empirically determined mold dynamic data in the calculation of its parameters which makes it a general purpose control scheme. The algorithm was tested in a pilot production environment and yielded the designed responses.

FUTURE WORK

The testing program will be continued to gain as much information concerning the first order algorithm as possible. An extension of this development endeavor or the initiation of a new endeavor will be sought to implement the new control algorithm for compression mold applications. A testing program shall be initiated to determine if the control algorithm is suitable for injection molding applications.

Table 12. First Order Versus Second Order Algorithm
Test Results

	Punch		Cavity	
	Heat Up	Cool Down	Heat Up	Cool Down
Steady State Offset (°C)				
First Order	9.7	9.9	7.9	8.8
Second Order	9.1	10.1	7.5	8.4
Overshoot*(°C)				
First Order	-1.6	-0.7	+0.2	+0.2
Second Order	-1.4	-2.3	+1.7	+0.4
IAE				
	Punch		Cavity	
First Order	5644		3710	
Second Order	5774		3640	
*Negative sign indicates undershoot.

Figure 17. Cavity Response, First Order Algorithm

Figure 18. Cavity Response, Second Order Algorithm

Figure 19. Punch Response, First Order Algorithm

Figure 20. Punch Response, Second Order Algorithm

REFERENCES

- ¹R. H. Floersch, Mold Heating and Cooling Systems (Interim Report). Bendix Kansas City: BDX-613-1990 (Rev.), September 1978 (Available from NTIS).
- ²R. H. Floersch, Theory and Operation of the Bendix Mold Heating and Cooling Equipment (Interim Report). Bendix Kansas City: BDX-613-369, June 1971 (Available from NTIS).
- ³R. H. Floersch, Fluid Flow Analyses in the Mold Heating and Cooling Control Loop (Interim Report). Bendix Kansas City: BDX-613-742, September 1972 (Available from NTIS).
- ⁴R. H. Floersch, Experimental Testing for Mold Temperature Responses (Interim Report). Bendix Kansas City: BDX-613-1111 (Rev.), May 1975 (Available from NTIS).
- ⁵R. H. Floersch, Computer Simulation of the Mold Heating and Cooling Control Loop Using Conventional Control Strategies (Interim Report). Bendix Kansas City: BDX-613-1764 (Rev.), February 1978 (Available from NTIS).
- ⁶R. C. Dorf, Modern Control Systems. Reading, MA: Addison-Wesley, 1967.
- ⁷C. L. Smith, Digital Computer Process Control. New York: International Textbook, Intext Educational Publishers, 1972.
- ⁸B. C. Kud, Discrete-Data Control Systems. Englewood Cliffs, NJ: Prentice-Hall Electrical Engineering Series, 1970.

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix A

DIGITAL CONTROLLER DERIVATION:
MOLD TEMPERATURE FEEDBACK,
FIRST ORDER PLANT

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Appendix A

DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, FIRST ORDER PLANT

The following is the derivation of a digital control algorithm designed to perform as a first order lag plus dead time. The fluid and mold temperature response are combined into one differential equation that is a first order lag plus dead time.

Figure A-1 illustrates the closed loop in block diagram form. The pulse transfer function for the time domain part is determined as follows:

$$HG(Z) = \mathcal{Z} \left\{ \frac{K_I (1 - e^{-ST}) K_m e^{-\theta S}}{S(\tau_m S + 1)} \right\} \quad (A-1)$$

where

K_I = Incremental memory module gain,

K_m = Plant gain,

τ_m = Plant time constant,

T = Computer sample period, and

θ = Plant dead time.

The plant dead time can be defined in terms of the computer sample period and a remainder.

$$N = \frac{\theta}{T}, \quad (A-2)$$

and

$$\theta^1 = \theta - NT = T - mT, \quad (A-3)$$

where

N = Integer number of evenly divisible sample periods in the plant dead time,

θ^1 = The difference between the actual dead time and evenly divisible dead time, and

Figure A-1. Block Diagram, Mold Temperature Feedback, First Order Plant

m = Decimal value of the position of the remainder between sample periods.

Inserting Equations A-2 and A-3 into Equation A-1 and expanding, but first defining, a system gain gives

$$K_p = K_I K_m. \quad (A-4)$$

$$\begin{aligned} HG(Z) &= \mathcal{Z} \left\{ \frac{K_p (1-e^{-ST}) e^{-(N+1)TS} e^{mTS/\tau_m}}{S(S+1/\tau_m)} \right\} \\ &= K_p (1-Z^{-1}) Z^{-N-1} \mathcal{Z} \left\{ \frac{e^{mTS/\tau_m}}{S(S+1/\tau_m)} \right\} \\ &= K_p (1-Z^{-1}) Z^{-N-1} \mathcal{Z} \left\{ \frac{K_1}{S} + \frac{K_2}{S+1/\tau_m} \right\}. \end{aligned} \quad (A-5)$$

Determining K_1 and K_2 ,

$$K_1 = SF(S) \bigg|_{S=0} = \frac{e^{mTS/\tau_m}}{S+1/\tau_m} \bigg|_{S=0} \quad (A-6)$$

$$K_2 = \left(S + \frac{1}{\tau_m} \right) F(S) \bigg|_{S=-1/\tau_m} = \frac{e^{mTS/\tau_m}}{S} \bigg|_{S=-1/\tau_m} = -e^{-mT/\tau_m}. \quad (A-7)$$

Substituting Equations A-6 and A-7 into A-5 and taking the Z transform gives:

$$HG(Z) = K_p (1-Z^{-1}) Z^{-N-1} \left[\frac{(1-e^{-mT/\tau_m}) + Z^{-1}(e^{-mT/\tau_m} - e^{-T/\tau_m})}{(1-Z^{-1})(1-e^{-T/\tau_m}Z^{-1})} \right] \quad (A-8)$$

Defining the following constants, Equation A-8 reduces to

$$X = e^{-T/\tau_m} \quad (A-9)$$

$$Y = e^{-mT/\tau_m} \quad (A-10)$$

$$C_1 = 1 - Y \quad (A-11)$$

$$C_2 = Y - X \quad (A-12)$$

and

$$HG(Z) = K_p Z^{-n-1} \left[\frac{C_1 + Z^{-1}C_2}{1-XZ^{-1}} \right] = \frac{C(Z)}{M(Z)} \quad (A-13)$$

Figure A-1 can now be redrawn as illustrated in Figure A-2. The closed loop transfer function of Figure A-2 is

$$\frac{C(Z)}{R(Z)} = \frac{D(Z) HG(Z)}{1 + D(Z) HG(Z)} \quad (A-14)$$

Equation A-14 is solved for D(Z) to determine the controller algorithm

$$D(Z) = \frac{C(Z)/R(Z)}{1 - C(Z)/R(Z)} \cdot \frac{1}{HG(Z)} \quad (A-15)$$

Specifying that the closed loop respond as a first order lag plus dead time to changes in set point is known as Dahlin's method.⁷ That method is executed as follows. C(Z) is defined as

$$C(Z) = \frac{1}{S} \cdot \frac{e^{-\theta S}}{(\lambda S + 1)} \quad (A-16)$$

where

$$\theta = NT + \theta' \quad (A-17)$$

$$\theta' = T - mT \quad (A-18)$$

Figure A-2. Equivalent Block Diagram, First Order Plant

and λ is the closed loop time constant. Expanding Equation A-16 after substituting in Equations A-18 and A-19,

$$\begin{aligned} C(Z) &= Z^{-N-1} \mathcal{Z} \left\{ \frac{e^{mTS/\lambda}}{S(S+1/\lambda)} \right\} \\ &= Z^{-N-1} \mathcal{Z} \left\{ \frac{K_1}{S} + \frac{K_2}{(S+1/\lambda)} \right\} \end{aligned} \quad (A-19)$$

Solving Equation A-19 for K_1 and K_2

$$K_1 = S F(S) \Big|_{S=0} = \frac{e^{mTS/\lambda}}{(S+1/\lambda)} \Big|_{S=0} = 1 \quad (A-20)$$

$$K_2 = (S+1/\lambda)F(S) \Big|_{S=-1/\lambda} = \frac{e^{mTS/\lambda}}{S} \Big|_{S=-1/\lambda} = -e^{-mT/\lambda} \quad (A-21)$$

Substituting Equations A-20 and A-21 into A 19, taking the Z transform, and defining following constants results in:

$$X' = e^{-T/\lambda} \quad (A-22)$$

$$Y' = e^{-mT/\lambda} \quad (A-23)$$

$$C_3 = 1 - Y' \quad (A-24)$$

$$C_4 = Y' - X' \quad (A-25)$$

$$C(Z) = Z^{-N-1} \left[\frac{C_3 + Z^{-1}C_4}{(1-Z^{-1})(1-Z^{-1}X')} \right] \quad (A-26)$$

Noting that the term $(1-Z^{-1})$ in the denominator of Equation A-26 is the same as a unit step, A-26 can be redefined as:

$$\frac{C(Z)}{R(Z)} = Z^{-N-1} \left[\frac{C_3 + Z^{-1}C_4}{1 - Z^{-1}X'} \right] \quad (A-27)$$

Substituting Equations A-27 and A-13 into A-15,

$$\begin{aligned} D_p(Z) &= \left[\frac{1-Z^{-1}X}{Z^{-N-1}K_p C_1 + Z^{-1}C_2} \right] \left\{ \frac{Z^{-N-1}(C_3 + Z^{-1}C_4) (1-Z^{-1}X')}{[1-Z^{-1}X' - Z^{-N-1}(C_3 + Z^{-1}C_4)] (1-Z^{-1}X')} \right\} \\ &= \frac{C_3}{K_p C_1} \left\{ \frac{1-Z^{-1}\left(X - \frac{C_4}{C_3}\right) - Z^{-2}\frac{C_4}{C_3}X}{\left[1-Z^{-1}\left(X' - \frac{C_2}{C_1}\right) - Z^{-2}\frac{C_2}{C_1}X' - Z^{-N-1}C_3 - Z^{-N-2}\left(\frac{C_4 C_1 + C_2 C_3}{C_1}\right) - Z^{-N-3}\frac{C_2 C_4}{C_1}\right]} \right\} \end{aligned} \quad (A-28)$$

Defining the following constants, Equation A-28 becomes:

$$K_{CS} = C_3 K_p C_1 \quad (A-29)$$

$$D_1 = (XC_3 - C_4)/C_3 \quad (A-30)$$

$$D_2 = XC_4/C_3 \quad (A-31)$$

$$F_1 = (X'C_1 - C_2)/C_1 \quad (A-32)$$

$$F_2 = X'C_2/C_1 \quad (A-33)$$

$$F_3 = C_3 \quad (A-34)$$

$$F_4 = (C_4 C_1 + C_2 C_3)/C_1 \quad (A-35)$$

$$F_5 = C_2 C_4/C_1 \quad (A-36)$$

and

$$D_p(Z) = \frac{K_{CS}(1-Z^{-1}D_1 - Z^{-2}D_2)}{(1-Z^{-1}F_1 - Z^{-2}F_2 - Z^{-N-1}F_3 - Z^{-N-2}F_4 - Z^{-N-3}F_5)} = \frac{M(Z)}{E(Z)} \quad (A-37)$$

Cross multiplying and converting Equation A-37 to the time domain:

$$M_n = K_{CS} (e_n^{-D_1} e_{n-1}^{-D_2} e_{n-2}) + F_1 M_{n-1} + F_2 M_{n-2} + F_3 M_{n-N-1} + F_4 M_{n-N-2} + F_5 M_{n-N-3} \quad (A-38)$$

Equation A-38 is the positional form of the desired algorithm. Because the computer currently in use uses the velocity form, the $N^{th}-1$ value is subtracted from the N^{th} value, yielding the following equation:

$$\Delta M_n = K_{CS} (\Delta e_n^{-D_1} \Delta e_{n-1}^{-D_2} \Delta e_{n-2}) + F_1 \Delta M_{n-1} + F_2 \Delta M_{n-2} + F_3 \Delta M_{n-N-1} + F_4 \Delta M_{n-N-2} + F_5 \Delta M_{n-N-3} \quad (A-39)$$

By using the derived digital controller, the closed loop transfer function will have the form of Equation A-27 which will always appear stable upon examination of its roots. Therefore, to determine if the control system is stable, the poles and zeroes of the controller transfer function must be examined. If the poles and zeroes lie with the Z circle of radius one, then the system is stable. Putting Equation A-39 back into pulse transfer form yields

$$D_v(Z) = \frac{K_{CS} [1-Z^{-1}(D_1+1)+Z^{-2}(D_1-D_2)+Z^{-3}D_3]}{[1-Z^{-1}(F_1+1)+Z^{-2}(F_1-F_2)+Z^{-3}F_2-Z^{-N-1}F_3+Z^{-N-2}(F_3-F_4)+Z^{-N-3}(F_4-F_5)+Z^{-N-4}F_5]} \quad (A-40)$$

Appendix B

DIGITAL CONTROLLER DERIVATION:
MOLD TEMPERATURE FEEDBACK,
SECOND ORDER PLANT

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix B

DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, SECOND ORDER PLANT

Figure B-1 illustrates in block diagram from the closed loop control system for mold temperature feedback. The transfer functions $H(S)$, $G_\phi(S)$ and $G_M(S)$ represent the dynamic responses of the incremental memory module, fluid, and mold dynamics. The mold response is assumed to be a first order lag plus dead time, and the fluid dynamics are broken out of the overall response and allowed to stand on their own. The transfer functions are:

$$H(S) = \frac{K_I (1 - e^{-ST})}{S} = \frac{T_I(S)}{M^*(S)} \quad (B-1)$$

$$G_\phi(S) = \frac{K_\phi}{(\tau_\phi S + 1)} = \frac{T_\phi(S)}{T_I(S)} \quad (B-2)$$

$$G_M(S) = \frac{K_m e^{-\theta S}}{(\tau_m S + 1)} = \frac{C(S)}{T_\phi(S)} \quad (B-3)$$

where

$$\theta = NT + \theta' \quad (B-4)$$

and

$$\theta' = T - MT = \frac{\theta}{T} - N. \quad (B-5)$$

Equations B-1 through B-3 are the Laplace notations of the differential equations relating each response. The constants in Equations B-1 through B-5 are

K_I = Incremental memory module gain,

T = Computer sample period,

K_ϕ = Open loop fluid gain,

τ_ϕ = Open loop fluid time constant,

K_m = Open loop mold gain,

Figure B-1. Block Diagram, Mold Temperature Feedback, Second Order Plant

τ_m = Open loop mold time constant,

θ_m = Mold dead time,

N = Number of evenly divisible sample periods of mold dead time,

θ' = Decimal remainder of the dead time after division by the sample period and having subtracted the number of evenly divisible sample periods (N), and

M = Fractional value of where the dead time ends between sample periods.

Substituting Equations B-4 and B-5 into B-3 yields:

$$G_M(S) = \frac{\frac{K_m e^{-(N+1-M)TS}}{\tau_m}}{\left(S + \frac{1}{\tau_m}\right)} \quad (B-6)$$

The pulse transfer function for the incremental memory module, fluid, and mold dynamics is:

$$\begin{aligned} HG(Z) &= \mathcal{Z} \left[\frac{K_I (1 - e^{-st})}{S} \cdot \frac{K_\phi / \tau_\phi}{(S + 1/\tau_\phi)} \cdot \frac{\frac{K_m e^{-(N+1)TS} e^{MTS}}{\tau_m}}{(S + 1/\tau_m)} \right] \\ &= \frac{K_I K_\phi K_m}{\tau_\phi \tau_m} (1 - Z^{-1}) 1 - Z^{-(N+1)} \mathcal{Z} \left[\frac{e^{MTS}}{S(S + 1/\tau_\phi)(S + 1/\tau_m)} \right] \end{aligned} \quad (B-7)$$

Letting $K_p = K_I K_\phi K_m$ and taking the partial fraction expansion of Equation B-7 leads to:

$$HG(Z) = \frac{K_p}{\tau_\phi \tau_m} Z^{-N-1} (1 - Z^{-1}) \mathcal{Z} \left[\frac{K_1}{S} + \frac{K_2}{S + 1/\tau_\phi} + \frac{K_3}{S + 1/\tau_m} \right] \quad (B-8)$$

Solving Equation B-8 for K_1 through K_3 ,

$$K_1 = \left. SF(S) \right|_{S=0} = \left. \frac{e^{MTS}}{(S + 1/\tau_\phi)(S + 1/\tau_m)} \right|_{S=0} = \tau_\phi \tau_m \quad (B-9)$$

$$K_2 = (S+1/\tau_\phi)F(S) \left| \begin{array}{c} = \frac{e^{MTS}}{S(S+1/\tau_m)} \\ S=1/\tau_\phi \end{array} \right|_{S=-1/\tau_\phi} = \frac{\tau_\phi^2 \tau_m e^{-MT/\tau_\phi}}{(\tau_m - \tau_\phi)} \quad (B-10)$$

$$K_3 = (S+1/\tau_m)F(S) \left| \begin{array}{c} = \frac{e^{MTS}}{S(S+1/\tau_\phi)} \\ S=-1/\tau_m \end{array} \right|_{S=-1/\tau_m} = - \frac{\tau_\phi^2 \tau_m e^{-MT/\tau_m}}{(\tau_m - \tau_\phi)} \quad (B-11)$$

Inserting Equations B-9 through B-11 back into B-8 and taking the Z transform of each,

$$HG(Z) = \frac{K_P}{\tau_\phi \tau_m} Z^{-N-1} (1-Z^{-1}) \left[\frac{\tau_\phi \tau_m}{(1-Z^{-1})} + \frac{\tau_\phi^2 \tau_m e^{-MT/\tau_\phi}}{(\tau_m \tau_\phi) (1-e^{-T/\tau_\phi} Z^{-1})} + \frac{\tau_\phi \tau_m^2 e^{-MT/\tau_m}}{(\tau_m - \tau_\phi) (1-e^{-T/\tau_m} Z^{-1})} \right] \quad (B-12)$$

Defining the following constants, Equation B-12 can be rearranged into the following form.

$$A = \frac{\tau_\phi e^{-MT/\tau_\phi}}{(\tau_m - \tau_\phi)} \quad (B-13)$$

$$B = \frac{\tau_m e^{-MT/\tau_m}}{(\tau_m - \tau_\phi)} \quad (B-14)$$

$$X_1 = e^{-T/\tau_\phi} \quad (B-15)$$

$$Y_1 = e^{-T/\tau_m} \quad (B-16)$$

$$C_1 = 1 + A - B \quad (B-17)$$

$$C_2 = X_1 + Y_1 + A(Y_1+1) - B(X_1+1) \quad (B-18)$$

$$C_3 = X_1 Y_1 + A Y_1 - B X_1 \quad (B-19)$$

$$D_1 = X_1 + Y_1 \quad (B-20)$$

$$D_2 = X_1 Y_1 \quad (B-21)$$

$$HG(Z) = K_P C_1 Z^{-N-1} \left[\frac{1-Z^{-1} \frac{C_2}{C_1} + Z^{-2} \frac{C_3}{C_1}}{1-Z^{-1} D_1 + Z^{-2} D_2} \right] \quad (B-22)$$

The inverse pulse transfer function is

$$HG(Z)^{-1} = \frac{(1-Z^{-1} D_1 + Z^{-2} D_2)}{K_P C_1 Z^{-N-1} \left(1-Z^{-1} \frac{C_2}{C_1} + Z^{-2} \frac{C_3}{C_1} \right)} \quad (B-23)$$

As the pulse transfer function is known, the block diagram of Figure B-1 reduces to that shown in Figure B-2. The closed loop pulse transfer function from Figure B-2 is:

$$\frac{C(Z)}{R(Z)} = \frac{K_C D(Z) HG(Z)}{1 + K_C D(Z) HG(Z)} \quad (B-24)$$

Solving Equation B-24 for $D(Z)$:

$$D(Z) = \frac{1}{K_C HG(Z)} \frac{C(Z)/R(Z)}{1 - C(Z)/R(Z)} \quad (B-25)$$

By specifying a closed loop response, Equation B-25 can be solved to determine the digital controller required to achieve the specified response. The following technique is attributed to Dahlin.⁷

If the closed loop response $C(Z)$ is forced to respond as a first order lag plus dead time to step changes in setpoint, $C(Z)$ is specified as follows:

$$C(Z) = \mathcal{Z} \left\{ \frac{e^{-\theta S}}{S(\lambda S + 1)} \right\} \quad (B-26)$$

where

Figure B-2. Equivalent Block Diagram, Plant Dynamics in Pulse Form, Second Order Plant

$$\theta = NT + \theta', \quad (B-27)$$

$$\theta' = T - MT, \quad (B-28)$$

and

λ = closed loop time constant.

Equations B-27 and B-28 are identical to Equations B-4 and B-5.

Expanding Equation B-26,

$$\begin{aligned} C(Z) &= \mathcal{Z} \left\{ \frac{e^{(-N-1+M)TS/\lambda}}{S(S+1/\lambda)} \right\} \\ &= Z^{-N-1} \mathcal{Z} \left\{ \frac{e^{-MTS/\lambda}}{S(S+1/\lambda)} \right\} \\ &= Z^{-N-1} \mathcal{Z} \left\{ \frac{K_1}{S} + \frac{K_2}{S+1/\lambda} \right\}. \end{aligned} \quad (B-29)$$

Solving the K_1 and K_2

$$K_1 = SF(S) \Big|_{S=0} = \frac{e^{MTS/\lambda}}{S+1/\lambda} \Big|_{S=0} = 1 \quad (B-30)$$

$$K_2 = (S+1/\lambda)F(S) \Big|_{S=-1/\lambda} = \frac{e^{MTS/\lambda}}{S} \Big|_{S=-1/\lambda} = -e^{-MT/\lambda}. \quad (B-31)$$

Substituting Equations B-30 and B-31 into B-29 and taking the Z transform:

$$C(Z) = Z^{-N-1} \left[\frac{(1-e^{-MT/\lambda}) + Z^{-1}(e^{-MT/\lambda} - e^{-T/\lambda})}{(1-Z^{-1})(1-e^{-T/\lambda}Z^{-1})} \right]. \quad (B-32)$$

Defining the following constants, B-32 reduces to:

$$X_2 = e^{-T/\lambda} \quad (B-33)$$

$$Y_2 = e^{-MT/\lambda} \quad (B-34)$$

$$C_4 = 1 - Y_2 \quad (B-35)$$

$$C_5 = Y_2 - X_2 \quad (B-36)$$

$$C(Z) = Z^{-N-1} \frac{C_4 + Z^{-1}C_5}{(1 - Z^{-1}X_2)(1 - Z^{-1})} \quad (B-37)$$

The term $(1 - Z^{-1})$ in the denominator is a unit step; therefore, Equation B-37 may be rewritten as

$$\frac{C(Z)}{R(Z)} = Z^{-N-1} \frac{C_4 + Z^{-1}C_5}{(1 - X_2 Z^{-1})} \quad (B-38)$$

Substituting Equations B-23 and B-38 into B-25 after rearrangement yields the controller transfer function.

$$D(Z) = \frac{C_4}{K_c K_p C_1} \left[\frac{1 + Z^{-1} \left(\frac{C_5}{C_4} - D_1 \right) + Z^{-2} \left(D_2 - D_1 \frac{C_5}{C_4} \right) + Z^{-3} D_2 \frac{C_5}{C_4}}{1 - Z^{-1} \left(X_2 + \frac{C_2}{C_1} \right) + Z^{-2} \left(\frac{C_2 X_2 + C_3}{C_1} \right) - Z^{-3} \left(\frac{C_3 X_2}{C_1} \right) - Z^{-N-1} C_4} \right] \quad (B-39)$$

$$+ Z^{-N-2} \frac{C_2 C_4 - C_1 C_5}{C_1} + Z^{-N-3} \frac{C_2 C_5 - C_3 C_4}{C_1} - Z^{-N-4} \frac{C_3 C_5}{C_1}$$

By defining the following constants,

$$F_1 = X_2 + \frac{C_2}{C_1} \quad (B-40)$$

$$F_2 = (C_2 X_2 + C_3) / C_1 \quad (B-41)$$

$$F_3 = C_3 X_2 / C_1 \quad (B-42)$$

$$F_4 = C_4 \quad (B-43)$$

$$F_5 = (C_2 C_4 - C_5 C_1) / C_1 \quad (B-44)$$

$$F_6 = (C_2 C_5 - C_3 C_4) / C_1 \quad (B-45)$$

$$F_7 = C_3 C_5 / C_1 \quad (B-46)$$

$$H_1 = (C_5 - D_1 C_4) / C_4 \quad (B-47)$$

$$H_2 = D_2 C_4 - D_1 C_5 / C_4 \quad (B-48)$$

$$H_3 = D_2 C_5 / C_4 \quad (B-49)$$

$$K_{cs} = C_4 / K_c K_p C_1 \quad (B-50)$$

and after converting to the time domain, Equation B-39 becomes:

$$\begin{aligned} M_n = & K_{cs} (e_n + H_1 e_{n-1} + H_2 e_{n-2} + H_3 e_{n-3}) + F_1 M_{n-1} + F_2 M_{n-2} \\ & + F_3 M_{n-3} + F_4 M_{n-N-1} - F_5 M_{n-N-2} - F_6 M_{n-N-3} \\ & - F_7 M_{n-N-4} \end{aligned} \quad (B-51)$$

Appendix C

DIGITAL CONTROLLER DERIVATION:
MOLD TEMPERATURE FEEDBACK,
THIRD ORDER PLANT

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Appendix C

DIGITAL CONTROLLER DERIVATION: MOLD TEMPERATURE FEEDBACK, THIRD ORDER PLANT

Figure C-1 illustrates, in block diagram form, the closed loop control system. The transfer functions $H(S)$, $G_\phi(S)$, and $G_M(S)$ represent the dynamic responses of the incremental memory module, the fluid loop, and the mold loop, respectively.

$$H(S) = \frac{K_I(1-e^{-ST})}{S} = \frac{T_I(S)}{M^*(S)} \quad (C-1)$$

$$G_\phi(S) = \frac{K_\phi}{(\tau_\phi S + 1)} = \frac{T_\phi(S)}{T_I(S)} \quad (C-2)$$

$$G_M(S) = \frac{K_m e^{-\theta_m S}}{(\tau_1 S + 1)(\tau_2 S + 1)} = \frac{C(S)}{T_\phi(S)} \quad (C-3)$$

Equations C-1, C-2, and C-3 are the Laplace notations of the differential equations relating each response. The constants are:

K_I = Incremental memory module gain,

K_ϕ = Fluid loop open loop gain,

K_m = Mold loop open loop gain,

T = Computer sample period,

τ_ϕ = Fluid loop open loop time constant,

τ_1 = First mold loop open loop time constant,

τ_2 = Second mold loop open loop time constant, and

θ_m = Mold loop dead time

The transfer function $C(S)/M^*(S)$ is the multiple of the three transfer functions:

Figure C-1. Digital Controller Design, Third Order Plant

$$\frac{C(S)}{M^*(S)} = H(S) G_{\phi}(S) G_m(S). \quad (C-4)$$

The pulse transfer function of Equation C-4 is

$$\frac{C(Z)}{M(Z)} = \mathcal{Z} [H(S) G_{\phi}(S) G_m(S)] = HG(Z). \quad (C-5)$$

Using block diagram algebra, the output $C(Z)$ is related to the setpoint $R(Z)$ as follows:

$$\frac{C(Z)}{E(Z)} = D(Z) HG(Z) \quad (C-6)$$

and

$$E(Z) = R(Z) - C(Z). \quad (C-7)$$

Solving Equations C-6 and C-7 simultaneously,

$$\frac{C(Z)}{R(Z)} = \frac{D(Z)HG(Z)}{1+D(Z)HG(Z)}. \quad (C-8)$$

Equation C-8 can be solved for $D(Z)$.

$$D(Z) = \frac{C(Z)/R(Z)}{1-C(Z)/R(Z)} \frac{1}{HG(Z)}. \quad (C-9)$$

If the closed loop response $C(Z)/R(Z)$ is specified and the process dynamics $HG(Z)$ are known, then the controller required to force the closed loop response to that specified is obtained by solving Equation C-9. That procedure is executed as follows.

The pulse transfer function for the plant must first be determined. Inserting Equations C-1, C-2, and C-3 into C-5,

$$HG(Z) = \mathcal{Z} \left[\frac{K_I (1-e^{-ST})}{S} \cdot \frac{K_\phi}{\tau_\phi S+1} \cdot \frac{K e^{-\theta_m S}}{(\tau_1 S+1)(\tau_2 S+1)} \right]. \quad (C-10)$$

If θ_m is evenly divisible by T , then θ_m may be represented as NT .

$$\begin{aligned} HG(Z) &= \mathcal{Z} \left[\frac{\frac{K_I K_\phi K (1-e^{-ST}) e^{-NTS}}{\tau_\phi \tau_1 \tau_2}}{S(S+1/\tau_\phi)(S+1/\tau_1)(S+1/\tau_2)} \right] \\ &= K_I K_\phi K (1-Z^{-1}) Z^{-N} \mathcal{Z} \left[\frac{1/\tau_\phi \tau_1 \tau_2}{S(S+1/\tau_\phi)(S+1/\tau_1)(S+1/\tau_2)} \right] \end{aligned} \quad (C-11)$$

Partial fraction expanding Equation C-11,

$$\begin{aligned} HG(Z) &= K_I K_\phi K (1-Z^{-1}) Z^{-N} \mathcal{Z} \left[\frac{K_1}{S} + \frac{K_2}{S+a} + \frac{K_3}{S+b} + \frac{K_4}{S+c} \right] \\ &= K_A (1-Z^{-1}) Z^{-N} [T(S)]. \end{aligned} \quad (C-12)$$

where $a = 1/\tau_\phi$, $b = 1/\tau_1$, $c = 1/\tau_2$, and $K_A = K_I K_\phi K_m$

Solving Equation C-12 for K_1 through K_4 :

$$K_1 = ST(S) \Big|_{S=0} = \frac{abc}{(S+a)(S+b)(S+c)} \Big|_{S=0} = 1 \quad (C-13)$$

$$K_2 = (S+a)T(S) \Big|_{S=-a} = \frac{abc}{S(S+b)(S+c)} \Big|_{S=-a} = -\frac{bc}{(b-a)(c-a)} \quad (C-14)$$

$$K_3 = (S+b)T(S) \Big|_{S=-b} = \frac{abc}{S(S+a)(S+c)} \Big|_{S=-b} = -\frac{ac}{(a-b)(c-b)} \quad (C-15)$$

$$K_4 = (S+c)T(S) \Big|_{S=-c} = \frac{abc}{S(S+a)(S+b)} \Big|_{S=-c} = -\frac{ab}{(a-c)(b-c)} \quad (C-16)$$

Letting $A = K_2$, $B = K_3$, and $C = K_4$, Equation C-12 becomes, after transforming to the Z-domain,

$$HG(Z) = K_A (1-Z^{-1}) Z^{-N} \left[\frac{1}{(1-Z^{-1})} - \frac{A}{(1-e^{-aT}Z^{-1})} - \frac{B}{(1-e^{-bT}Z^{-1})} - \frac{C}{(1-e^{-cT}Z^{-1})} \right]. \quad (C-17)$$

Getting a common denominator, Equation C-17 becomes, after collecting like terms,

$$HG(Z) = K_A Z^{-N} \left[\frac{C_1 - Z^{-1}C_2 + Z^{-2}C_3 - Z^{-3}C_4}{1 - Z^{-1}D_1 + Z^{-2}D_2 - Z^{-3}D_3} \right] \quad (C-18)$$

where

$$C_1 = 1 - A - B - C \quad (C-19)$$

$$C_2 = e^{-aT}(1-B-C) + e^{-bT}(1-A-C) + e^{-cT}(1-A-B) - A - B - C \quad (C-20)$$

$$C_3 = e^{-(a+b)T}(1-C) + e^{-(a+c)T}(1-B) + e^{-(b+c)T}(1-A) - e^{-aT}(B+C) - e^{-bT}(A+C) - e^{-cT}(A+B) \quad (C-21)$$

$$C_4 = e^{-(a+b+c)T} - Ae^{-(b+c)T} - Be^{-(a+c)T} - Ce^{-(a+b)T} \quad (C-22)$$

$$D_1 = e^{-aT} + e^{-bT} + e^{-cT} \quad (C-23)$$

$$D_2 = e^{-(a+b)T} + e^{-(b+c)T} + e^{-(a+c)T} \quad (C-24)$$

$$D_3 = e^{-(a+b+c)T} \quad (C-25)$$

Expansion of Equation C-19 reveals that C_1 is always zero; consequently, Equation C-18 becomes:

$$HG(Z) = -K_A C_2 Z^{-N-1} \left[\frac{1-Z^{-1} \frac{C_3}{C_2} + Z^{-2} \frac{C_4}{C_2}}{1-Z^{-1} D_1 + Z^{-2} D_2 - Z^{-3} D_3} \right] \quad (C-26)$$

Using Dahlin's method of forcing the closed loop response to respond as a first order lag plus dead time to step changes, $C(Z)$ is specified as

$$C(Z) = \mathcal{Z} \left\{ \frac{1}{S} \cdot \frac{e^{-\theta S}}{\lambda S + 1} \right\} \quad (C-27)$$

where

θ = System dead time = mold dead time = NT , and

λ = System closed loop time constant.

From the tables, Equation C-27 transforms to:

$$C(Z) = \frac{(1-e^{-T/\lambda}) Z^{-N-1}}{(1-Z^{-1})(1-e^{-T/\lambda} Z^{-1})}. \quad (C-28)$$

The term $(1-Z^{-1})$ in the denominator of Equation C-29 is a unit step. If $R(Z)$ is a unit step C-28 becomes:

$$\frac{C(Z)}{R(Z)} = \frac{(1-e^{-T/\lambda}) Z^{-N-1}}{(1-e^{-T/\lambda} Z^{-1})}. \quad (C-29)$$

Letting $x=e^{-T/\lambda}$ and $K_{c1}=1-x$, Equation C-29 may be redefined as:

$$\frac{C(Z)}{R(Z)} = \frac{K_{c1} Z^{-N-1}}{1-xZ^{-1}}. \quad (C-30)$$

Equations C-26 and C-30 can now be inserted into Equation C-9:

$$D(Z) = \frac{(1-Z^{-1} D_1 + Z^{-2} D_2 - Z^{-3} D_3)}{-K_A C_2 Z^{-N-1} \left(1-Z^{-1} \frac{C_3}{C_2} + Z^{-2} \frac{C_4}{C_2} \right)} \cdot \frac{\frac{K_{c1} Z^{-N-1}}{(1-xZ^{-1})}}{\frac{(1-xZ^{-1} - K_{c1} Z^{-N-1})}{(1-xZ^{-1})}} \quad (C-31)$$

Equation C-31 after expansion and collection of terms is:

$$D(Z) = \frac{M(Z)}{E(Z)} = \frac{K_{cs} (1-Z^{-1}D_1 + Z^{-2}D_2 - Z^{-3}D_3)}{1-Z^{-1}F_1 + Z^{-2}F_2 - Z^{-3}F_3 - Z^{-N-1}F_4 + Z^{-N-2}F_5 - Z^{-N-3}F_6} \quad (C-32)$$

where

$$K_{cs} = -K_{c1}/K_A C_2, \quad (C-33)$$

$$F_1 = C_3/C_2 + X, \quad (C-34)$$

$$F_2 = C_4/C_2 + C_3 X/C_2, \quad (C-35)$$

$$F_3 = C_4 X/C_2, \quad (C-36)$$

$$F_4 = K_{c1}, \quad (C-37)$$

$$F_5 = K_{c1} C_3/C_2, \text{ and} \quad (C-38)$$

$$F_6 = K_{c1} C_4/C_2. \quad (C-39)$$

Cross-multiplying and converting to the time domain pulse form

$$\begin{aligned} M_n = & K_{cs} (e_n^{-D_1} e_{n-1}^{+D_2} e_{n-2}^{-D_3} e_{n-3}) + F_1 M_{n-1} - F_2 M_{n-2} + F_3 M_{n-3} \\ & + F_4 M_{n-N-1} - F_5 M_{n-N-2} + F_6 M_{n-N-3}. \end{aligned} \quad (C-40)$$

The velocity form of Equation C-40 is obtained by subtracting the (n-1)th value from the nth value of C-40.

$$\begin{aligned} \Delta M_n = & K_{cs} (e_n^{-D_1} e_{n-1}^{+D_2} e_{n-2}^{-D_3} e_{n-3}) - D_1 (e_{n-1}^{-D_1} e_{n-2}^{+D_2} e_{n-3}^{-D_3} e_{n-4}) - D_3 (e_{n-3}^{-D_1} e_{n-4}^{+D_2} e_{n-5}^{-D_3} e_{n-6}) \\ & + F_1 (M_{n-1} - M_{n-2}) - F_2 (M_{n-2} - M_{n-3}) + F_3 (M_{n-3} - M_{n-4}) \\ & + F_4 (M_{n-N-1} - M_{n-N-2}) - F_5 (M_{n-N-2} - M_{n-N-3}) + F_6 (M_{n-N-3} - M_{n-N-4}) \\ = & K_{cs} (\Delta e_n^{-D_1} \Delta e_{n-1}^{+D_2} \Delta e_{n-2}^{-D_3} \Delta e_{n-3}) + F_1 \Delta M_{n-1} - F_2 \Delta M_{n-2} \\ & + F_3 \Delta M_{n-3} + F_4 \Delta M_{n-N-1} - F_5 \Delta M_{n-N-2} + F_6 \Delta M_{n-N-3} \end{aligned} \quad (C-41)$$

Converting Equation C-41 back to pulse transfer form,

$$D_v(Z) = \frac{K_{cs} \left[1 - Z^{-1}(D_1 + 1) + Z^{-2}(D_1 + D_2) - Z^{-3}(D_2 + D_3) + Z^{-4}D_3 \right]}{1 - Z^{-1}(F_1 + 1) + Z^{-2}(F_1 + F_2) - Z^{-3}(F_2 + F_3) + Z^{-4}F_3 - Z^{-N-1}F_4} \cdot \frac{+Z^{-N-2}(F_4 + F_5) - Z^{-N-3}(F_5 + F_6) + Z^{-N-4}F_6}{\quad} \quad (C-42)$$

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Appendix D

DIGITAL CONTROLLER DERIVATION: CASCADED FLUID AND MOLD TEMPERATURE FEEDBACK

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix D

DIGITAL CONTROLLER DERIVATION: CASCADED FLUID AND MOLD TEMPERATURE FEEDBACK

The following is the derivation for a cascaded control system with both metal temperature and mold temperature as process variables.

Figure D-1 illustrates the control system configuration in block diagram form. In Figure D-1, the known elements are:

$$H(S) = \frac{K_I(1-e^{-ST})}{S}$$

$$G_\phi(S) = \frac{K_\phi e^{-\theta_1 S}}{\tau_\phi S + 1}$$

$$G_m(S) = \frac{K_m e^{-\theta_2 S}}{\tau_m S + 1}$$

where

K_I = Incremental memory module gain,

K_ϕ = Fluid loop gain,

K_m = Mold loop gain,

θ_1 = Fluid loop dead time,

τ_ϕ = Fluid loop time constant,

θ_2 = Mold loop dead time, and

τ_m = Mold time constant.

Figure D-1. Cascaded Control Block Diagram

The inner closed loop pulse transfer function is:

$$\frac{C_1(Z)}{M_1(Z)} = \frac{D'_1(Z)HG_\phi(Z)}{1 + D'_1(Z)HG_\phi(Z)} \quad (D-1)$$

Solving Equation D-1 for the controller function:

$$D'_1(Z) = \frac{1}{HG_\phi(Z)} \frac{C_1(Z)/M_1(Z)}{1 - C_1(Z)/M_1(Z)} \quad (D-2)$$

The pulse transfer function for the inner plant $HG_\phi(Z)$ is determined as follows:

$$HG_\phi(Z) = \mathcal{Z} \left\{ \frac{K_I(1 - e^{-ST})}{S} \cdot \frac{K_\phi e^{-\theta_1 S}}{(\tau_\phi S + 1)} \right\} \quad (D-3)$$

where

$$\theta_1 = N_1 T + \theta'_1 \quad (D-4)$$

and N_1 is the integer number of sample periods in θ_1 . The remainder θ'_1 is the time between sample periods when the dead time ends and is defined as

$$m_1 = 1 - \frac{\theta'_1}{T} \quad (D-5)$$

$$Y_1 = e^{-m_1 T / \tau_\phi} \quad (D-12)$$

$$C_1 = 1 - Y_1 \quad (D-13)$$

$$C_2 = Y_1 - X_1 \quad (D-14)$$

and

$$HG(Z) = K_p Z^{-N_1-1} \left[\frac{C_1 + Z^{-1} C_2}{1 - Z^{-1} X_1} \right] \quad (D-15)$$

It is desired that the inner closed loop perform as a first order lag plus dead time to setpoint changes. To force the output variable $C_1(Z)$ to perform as a first order lag, it is so defined:

$$C_1(Z) = \frac{1}{S} \cdot \frac{e^{-\theta_1 S}}{\lambda_1 S + 1} \quad (D-16)$$

where

$$\theta_1 = N_1 T + T - m_1 T, \text{ and}$$

λ_1 = closed loop time constant of the inner loop. Comparison of Equations D-16 and D-3 leads to a direct transformation.

$$C_1(Z) = Z^{-N_1-1} \left[\frac{(1 - e^{-m_1 T / \lambda_1}) + Z^{-1} (e^{-m_1 T / \lambda_1} e^{-T / \lambda_1})}{(1 - Z^{-1})(1 - Z^{-1} e^{-T / \lambda_1})} \right] \quad (D-17)$$

Defining the following constants and recognizing that the term $(1 - Z^{-1})$ in the denominator of Equation D-17 is equivalent to a unit step, Equation D-17 becomes:

$$\frac{C_1(Z)}{M_1(Z)} = Z^{-N_1-1} \left[\frac{C_3 + Z^{-1} C_4}{1 - Z^{-1} X_2} \right] \quad (D-18)$$

where

$$X_2 = e^{-T / \lambda_1} \quad (D-19)$$

Substituting Equations D-4 and D-5 into D-3 and partial fraction expanding,

$$\begin{aligned}
 HG(Z) &= K_I K_\phi (1-Z^{-1}) \mathcal{Z} \left[\frac{e^{-(N_1+1)TS} e^{m_1 TS}}{\tau_\phi S(S+1/\tau_\phi)} \right] \\
 &= K_p (1-Z^{-1}) Z^{-N_1-1} \mathcal{Z} \left[\frac{e^{m_1 TS}}{\tau_\phi S(S+1/\tau_\phi)} \right] \\
 &= K_p (1-Z^{-1}) Z^{-N_1-1} \mathcal{Z} \left[\frac{K_1}{S} + \frac{K_\phi}{S+1/\tau_\phi} \right] \quad (D-6)
 \end{aligned}$$

and

$$K_p = K_I K_\phi. \quad (D-7)$$

Solving Equation D-6 for K_1 and K_2 :

$$K_1 = SF(S) \bigg|_{S=0} = \frac{e^{m_1 TS}}{\tau_\phi S(S+1/\tau_\phi)} \bigg|_{S=0} = 1 \quad (D-8)$$

and

$$K_2 = (S+1/\tau_\phi) F(S) \bigg|_{S=-1/\tau_\phi} = \frac{e^{m_1 TS}}{\tau_\phi S} \bigg|_{S=-1/\tau_\phi} = -e^{-m_1 T/\tau_\phi} \quad (D-9)$$

Inserting Equations D-8 and D-9 into D-6 and taking the Z transform:

$$HG_\phi(Z) = K_p (1-Z^{-1}) A^{-N_1-1} \left[\frac{(1-e^{-m_1 T/\tau_\phi})_+ Z^{-1} (e^{-m_1 T/\tau_\phi} - e^{-T/\tau_\phi})}{(1-Z^{-1})(1-e^{-T/\tau_\phi} Z^{-1})} \right] \quad (D-10)$$

By defining the following constants, Equation D-10 reduces to:

$$X_1 = e^{-T/\tau_\phi} \quad (D-11)$$

$$Y_2 = e^{-m_1 T / \lambda_1} \quad (D-20)$$

$$C_3 = 1 - Y_2 \quad (D-21)$$

$$C_4 = Y_2 - X_2 \quad (D-22)$$

Substituting Equations D-18 and D-15 into D-2, the inner loop controller can be determined:

$$D_1'(Z) = \frac{(1-Z^{-1}X_1)}{K_p Z^{-N_1-1} (C_1+Z^{-1}C_2)} \frac{\frac{Z^{-N_1-1} (C_3+Z^{-1}C_4)}{(1-Z^{-1}X_2)}}{\left[\frac{1-Z^{-1}X_2-Z^{-N_1-1} (C_3+Z^{-1}C_4)}{(1-Z^{-1}X_2)} \right]}$$

$$= \frac{K_{CS1} (1-Z^{-1}D_1-Z^{-2}D_2)}{1-Z^{-1}F_1-Z^{-2}F_2-Z^{-N_1-1}F_3-Z^{-N_1-2}F_4-Z^{-N_1-3}F_5} = \frac{M_2(Z)}{E_2(Z)} \quad (D-23)$$

where

$$D_1 = (X_1 C_3 - C_4) / C_3 \quad (D-24)$$

$$D_2 = C_4 X_1 / C_3 \quad (D-25)$$

$$F_1 = (X_2 C_1 - C_2) / C_1 \quad (D-26)$$

$$F_2 = C_2 X_2 / C_1 \quad (D-27)$$

$$F_3 = C_3 \quad (D-28)$$

$$F_4 = (C_4 C_1 + C_2 C_3) / C_1 \quad (D-29)$$

$$F_5 = C_2 C_4 / C_1 \quad (D-30)$$

$$K_{CS1} = C_3 / K_p C_1 \quad (D-31)$$

Cross multiplying Equation D-23, converting to the time domain, and putting into velocity form, the result is:

$$\Delta M_{2,n} = K_{CS1}(\Delta e_{2,n-D_1} \Delta e_{2,n-1-D_2} \Delta e_{2,n-2}) + F_1 \Delta M_{2,n-1} + F_2 \Delta M_{2,n-2} + F_3 \Delta M_{2,n-N-1} + F_4 \Delta M_{2,n-N-2} + F_5 \Delta M_{2,n-N-3} \quad (D-32)$$

The pulse transfer form of Equation D-32 is

$$D_{2,V}(Z) = \frac{K_{CS1}(1-Z^{-1}(D_1+1)+Z^{-2}(D_1-D_2)+Z^{-3}D_2)}{1-Z^{-1}(F_1+1)+Z^{-2}(F_1-F_2)+Z^{-3}F_2-Z^{-N_1-1}F_3+Z^{-N_1-2}(F_3-F_4)} \cdot \frac{1}{+Z^{-N_1-3}(F_4-F_5)+Z^{-N_1-4}F_5} \quad (D-33)$$

The poles and zeroes of Equation D-33 can be examined to ensure they fall within the Z circle of radius one for stability determination.

Since the inner loop is now defined and is designed to perform as a first order lag, Figure D-1 can be redrawn as illustrated by Figure D-2. Before the outer loop controller ($D_2'(Z)$) can be defined, the outer loop pulse transfer function must be determined, and the closed loop response ($C_2(Z)/R(Z)$) must be defined. First, the plant pulse transfer function is determined:

$$HG_m(Z) = \mathcal{Z}\{G_1(S)G_M(S)\} = \mathcal{Z}\left\{\frac{e^{-\theta_1 S}}{\lambda_1 S+1} \cdot \frac{K_m e^{-\theta_2 S}}{\tau_m S+1}\right\} \quad (D-34)$$

where

$$\theta_1 = \text{Inner loop dead time} = N_1 T + T - m_1 T, \text{ and} \quad (D-35)$$

$$\theta_2 = \text{Outer loop dead time} = N_2 T + T - m_2 T. \quad (D-36)$$

Inserting Equations D-35 and D-36 into D-34 and partial fraction expanding:

Figure D-2. Equivalent Cascaded Control Block Diagram

$$\begin{aligned}
 HG_M(Z) &= K_m \mathcal{Z} \left[\frac{\frac{e^{-(N_1+1)TS} e^{-(N_2+1)TS} e^{(m_1+m_2)TS}}{\lambda_1 \tau_m}}{\left(S + \frac{1}{\lambda_1}\right) \left(S + \frac{1}{\tau_m}\right)} \right] \\
 &= K_m Z^{-N_1-N_2-2} \mathcal{Z} \left[\frac{\frac{e^{(m_1+m_2)TS}}{\lambda_1 \tau_m}}{\left(S + \frac{1}{\lambda_1}\right) \left(S + \frac{1}{\tau_m}\right)} \right] \\
 &= K_m Z^{-N_2'} \mathcal{Z} \left[\frac{K_3}{S + 1/\lambda_1} + \frac{K_4}{S + 1/\tau_m} \right] \quad (D-37)
 \end{aligned}$$

Letting $M = m_1 + m_2$ and solving Equation D-37 for K_3 and K_4 :

$$K_3 = (S + 1/\lambda_1) F(S) \bigg|_{S=-1/\lambda_1} = \frac{e^{MTS/\lambda_1 \tau_m}}{S + 1/\tau_m} \bigg|_{S=-1/\lambda_1} = \frac{e^{-MT/\lambda_1}}{\tau_m - \lambda_1} \quad (D-38)$$

$$K_4 = (S + 1/\tau_m) F(S) \bigg|_{S=-1/\tau_m} = \frac{e^{MTS/\lambda_1 \tau_m}}{S + 1/\lambda_1} \bigg|_{S=-1/\tau_m} = \frac{e^{-MT/\tau_m}}{\tau_m - \lambda_1} \quad (D-39)$$

Inserting Equations D-38 and D-39 into D-37, taking the Z transform, and rearranging results in:

$$HG_M(Z) = \frac{K_m Z^{-N_2'}}{\Delta \tau} \left[\frac{C_5 + Z^{-1} C_6}{(1 - Z^{-1} X_2)(1 - Z^{-1} X_3)} \right] \quad (D-40)$$

where

$$\Delta\tau = \tau_m - \lambda_1 \quad (D-41)$$

$$X_3 = e^{-T/\tau_m} \quad (D-42)$$

$$C_5 = e^{-MT/\tau_m} - e^{-MT/\lambda_1} \quad (D-43)$$

$$C_6 = e^{-MT/\lambda_1} X_3 - e^{-MT/\tau_m} X_2 \quad (D-44)$$

Now defining the outer closed loop response to be a first order lab plus dead time to a unit step forcing:

$$C_2(Z) = \mathcal{Z} \left\{ \frac{1}{S} \cdot \frac{e^{-(\theta_1 + \theta_2)S}}{\lambda_2 S + 1} \right\} \quad (D-45)$$

where

$$\begin{aligned} \theta_1 + \theta_2 &= (N_1 + N_2 + 2)T - (m_1 + m_2)T \\ &= N_2' T - MT. \end{aligned} \quad (D-46)$$

Substituting Equation D-46 into D-45,

$$C_2(Z) = Z^{-N_2'} \mathcal{Z} \left[\frac{e^{MTS/\lambda_2}}{S(S+1/\lambda_2)} \right]. \quad (D-47)$$

Comparing Equation D-47 and D-16 leads to a direct conversion

$$\frac{C_2(Z)}{R(Z)} = Z^{-N_2'} \left[\frac{C_7 + Z^{-1} C_8}{(1 - Z^{-1} X_4)} \right] \quad (D-48)$$

where

$$Y_4 = e^{-MT/\lambda_2} \quad (D-49)$$

$$X_4 = e^{-T/\lambda_2} \quad (D-50)$$

$$C_7 = 1 - Y_4 \quad (D-51)$$

$$C_8 = Y_4 - X_4. \quad (D-52)$$

The outer loop controller using Figure D-2 and block diagram algebra is:

$$D_2'(Z) = \frac{1}{HG_M(Z)} \frac{C_2(Z)/R(Z)}{1-C_2(Z)/R(Z)} \quad (D-53)$$

Substituting Equations D-40 and D-48 into D-53:

$$D_2'(Z) = \frac{\frac{\Delta\tau}{K_m Z^{-N_2'} C_5} \left[\frac{(1-Z^{-1}X_2)(1-Z^{-1}X_3)}{1+Z^{-1}\frac{C_6}{C_5}} \right]}{\left[\frac{Z^{-N_2'} C_7 \left(1+Z^{-1}\frac{C_8}{C_7}\right)}{1-Z^{-1}X_4} \right] \frac{1-Z^{-1}X_4 - Z^{-N_2'} C_7 \left(1+Z^{-1}\frac{C_8}{C_7}\right)}{1-Z^{-1}X_4}}$$

$$= \frac{\Delta\tau C_7}{K_m C_5} \left[\frac{1-Z^{-1} \left(X_2+X_3-\frac{C_8}{C_7}\right) + Z^{-2} \left[X_2X_3 - \frac{C_8}{C_7}(X_2+X_3)\right] + Z^{-3} \frac{C_8}{C_7} X_2X_3}{1-Z^{-1} \left(X_4 - \frac{C_6}{C_5}\right) - Z^{-2} \left(\frac{C_6}{C_5} X_4\right) - Z^{-N_2'} C_7 - Z^{-N_2'-1} \left(\frac{C_5 C_8 + C_6 C_7}{C_5}\right) - Z^{-N_2'-2} \left(\frac{C_6 C_8}{C_5}\right)} \right]$$

$$= K_{CS2} \left[\frac{1-Z^{-1}H_1 + Z^{-2}H_2 + Z^{-3}H_3}{1-Z^{-1}P_1 - Z^{-2}P_2 - Z^{-N_2'}P_3 - Z^{-N_2'-1}P_4 - Z^{-N_2'-2}P_5} \right] = \frac{M_1(Z)}{E_1(Z)} \quad (D-54)$$

where

$$H_1 = X_2 + X_3 - \frac{C_8}{C_7} \quad (D-55)$$

$$H_2 = X_2X_3 - \frac{C_8}{C_7}(X_2+X_3) \quad (D-56)$$

$$H_3 = \frac{C_8}{C_7} X_2X_3 \quad (D-57)$$

$$P_1 = (X_4 C_5 - C_6)/C_5 \quad (D-58)$$

$$P_2 = C_6 X_4 / C_5 \quad (D-59)$$

$$P_3 = C_7 \quad (D-60)$$

$$P_4 = (C_8 C_5 + C_6 C_7) / C_5 \quad (D-61)$$

$$P_5 = C_6 C_8 / C_5 \quad (D-62)$$

$$K_{CS2} = \Delta \tau C_7 / K_m C_5 \quad (D-63)$$

Cross-multiplying Equation D-54 and converting to the time domain,

$$\begin{aligned} M_{1,n} = & K_{CS2} (e_{1,n} - H_1 e_{1,n-1} + H_2 e_{1,n-2} + H_3 e_{1,n-3}) + P_1 M_{1,n-1} \\ & + P_2 M_{1,n-2} + P_3 M_{1,n-N'_2} + P_4 M_{1,n-N'_2-1} + P_5 M_{1,n-N'_2-2} \end{aligned} \quad (D-64)$$

Equations D-32 and D-64 are the controller equations for the cascade configuration.

Appendix E

DIGITAL CONTROLLER DESIGN, CASCADE
CONFIGURATION: INNER LOOP PI, OUTER
LOOP DIGITAL DESIGN, SECOND ORDER PLANT

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix E

DIGITAL CONTROLLER DESIGN, CASCADE CONFIGURATION: INNER LOOP PI, OUTER LOOP DIGITAL DESIGN, SECOND ORDER PLANT

Figure E-1 illustrates the cascade control configuration in block diagram form. The variables are defined as follows:

$T_M(S)$ = Mold metal temperature,

$T_L(S)$ = Heat transfer fluid temperature,

$G_2(S)$ = Mold metal dynamics transfer function,

$HG_1(S)$ = Fluid loop dynamics transfer function,

$R(Z)$ = Metal temperature set point,

$E_1(Z)$ = Metal temperature error,

$M_1(Z)$ = Metal loop controller output,

$E_2(Z)$ = Fluid temperature error,

$M_2(Z)$ = Fluid loop controller output,

$D_1(Z)$ = Metal loop controller pulse transfer function,

$D_2(Z)$ = Fluid loop controller pulse transfer function, and

K = Variable major loop gain.

The fluid closed loop pulse transfer function from Figure E-1 is:

$$\frac{T_L(Z)}{M_1(Z)} = \frac{D_2(Z)HG_1(Z)}{1+D_2(Z)HG_1(Z)} \quad (E-1)$$

The fluid loop controller $D_2(Z)$ is currently a proportional-integral-derivative (PID) velocity algorithm. The inner closed loop responds as a second order underdamped response. The degree of damping is dependent on the gain settings of the PID controller, which are in turn dependent on the mold fluid channel design. The closed loop system as depicted by Equation E-1 can be modeled as second order underdamped response:

Figure E-1. Cascade Control Configuration

$$\frac{T_L(S)}{M_1(Z)} = \frac{\omega_n^2}{S^2 + 2\xi\omega_n S + \omega_n^2} = \frac{D_2(Z)HG_1(Z)}{1 + D_2(Z)HG_1(Z)} \quad (E-2)$$

where

ω_n is the natural frequency and

ξ is the damping coefficient.

From the peak response (T_p) percent overshoot (PO), and settling time (T_s) the closed loop response can be characterized. The equations relating the variables are

$$T_s = 4\lambda_1 = \frac{4}{\xi\omega_n} \quad (E-3)$$

$$T_p = \frac{\pi}{\omega_n \sqrt{1-\xi^2}} \quad (E-4)$$

$$PO = 100e^{-\xi\pi/\sqrt{1-\xi^2}} \quad (E-5)$$

where

λ is the time constant of the fluid loop.

The second order model can be further simplified by approximating it with a first order lag. The prime concern of the inner fluid loop is that it reaches set point and settles out in a timely manner. Since the fluid loop is from 10 to 30 times faster in response than the metal loop, a second order or first order fluid response as seen by the mold will be close to the same. The first order approximation will induce very little error in predicting the overall response. The closed loop response as approximated by a first order lag is:

$$\frac{T_L(S)}{M_1(Z)} = \frac{e^{-\theta_1 S}}{\lambda_1 S + 1} \quad (E-6)$$

The hold device is included in the approximation; consequently, the term $M_1(Z)$ is the denominator of the transfer function. The dead time of the inner loop is θ_1 , and the time constant is λ_1 as defined by Equation E-3.

Figure E-1 may be redrawn as illustrated by Figure E-2 using the inner loop definition of Equation E-6.

In Figure E-2, the following terms are defined as:

K_m - Steady-state mold gain ($^{\circ}\text{F}/^{\circ}\text{F}$) ($^{\circ}\text{C}/^{\circ}\text{C}$),

θ_m - Mold dead time (s), and

τ_m - Mold time constant (s).

The pulse transfer function for the major loop controller can now be derived. The pulse transfer function for the plant must be determined first. The transfer function for the plant is:

$$\frac{T_M(S)}{M_1(Z)} = \frac{K_m e^{-(\theta_1 + \theta_m)S}}{(\lambda_1 S + 1)(\tau_m S + 1)} \quad (\text{E-7})$$

The derived controller algorithm will be automatically dead time compensated because dead time is included in the plant definition. The two dead times can further be defined as

$$\theta_1 = n_1 T + T - m_1 T = T(n_1 + 1 - m_1) \quad (\text{E-8})$$

and

$$\theta_m = n_2 T + T - m_2 T = T(n_2 + 1 - m_2). \quad (\text{E-9})$$

where n_1 and n_2 are the integer number of sample periods in the respective dead times. The remainder of the two equations $T - m_1 T$ and $T - m_2 T$ account for the fractional part of the sample period that dead time overlaps and are determined as follows:

$$\theta'_i = \theta_i - n_i T \quad (\text{E-10})$$

and

$$m_i = 1 - \theta'_i / T. \quad (\text{E-11})$$

Figure E-2. Reconfigured Cascade System, Inner Loop Approximated by a First Order Lag

Substituting Equations E-8 and E-9 into E-7 results in:

$$\frac{T_M(S)}{M_1(Z)} = \frac{K_m e^{-(n_1+n_2+2)TS} e^{(m_1+m_2)TS}}{\lambda_1 \tau_m (S+1/\lambda_1) (S+1/\tau_m)} \quad (E-12)$$

The Z transformation of Equation E-12 proceeds as follows:

$$HG_2(Z) = \mathcal{Z} \left[\frac{T_M(S)}{M_1(Z)} \right] = \mathcal{Z} \left[\frac{K_m e^{-(n_1+n_2+2)TS} e^{(m_1+m_2)TS}}{\lambda_1 \tau_m (S+1/\lambda_1) (S+1/\tau_m)} \right] \quad (E-13)$$

Defining the constants $N = n_1+n_2+2$ and $M = m_1+m_2$ and partial fraction expanding Equation E-13:

$$\begin{aligned} HG_2(Z) &= \frac{K_m Z^{-N}}{\lambda_1 \tau} \mathcal{Z} \left[\frac{e^{MTS}}{(S+1/\lambda_1) (S+1/\tau_m)} \right] \\ &= \frac{KZ^{-N}}{\lambda_1 \tau} \mathcal{Z} \left[\frac{K_1}{S+1/\lambda_1} + \frac{K_2}{S+1/\tau_m} \right] \end{aligned} \quad (E-14)$$

Solving Equation E-14 for K_1 and K_2 :

$$K_1 = (S+1/\lambda_1) F(S) \Big|_{S=-1/\lambda_1} = \frac{e^{MTS}}{S+1/\tau_m} \Big|_{S=-1/\lambda_1} = - \frac{\lambda_1 \tau_m e^{-MT/\lambda_1}}{\tau_m - \lambda_1} \quad (E-15)$$

and

$$K_2 = (S+1/\tau_m)F(S) \bigg|_{S=-1/\tau_m} = \frac{e^{MTS}}{S+1/\lambda_1} \bigg|_{S=-1/\tau_m} = + \frac{\lambda_1 \tau_m e^{-MT/\tau_m}}{\tau_m - \lambda_1 Z} \quad (E-16)$$

Substituting Equations E-15 and E-16 into E-14 and transforming:

$$\begin{aligned} HG_2(Z) &= \frac{K_m Z^{-N}}{\tau_m - \lambda_1} \left[\frac{e^{-MT/\tau_m}}{S+1/\tau_m} - \frac{e^{-MT/\lambda_1}}{S+1/\lambda_1} \right] \\ &= \frac{K_m Z^{-N}}{\tau_m - \lambda_1} \left[\frac{e^{-MT/\tau_m}}{1 - e^{-T/\tau_m} Z^{-1}} - \frac{e^{-MT/\lambda_1}}{1 - e^{-T/\lambda_1} Z^{-1}} \right] \end{aligned} \quad (E-17)$$

Letting

$$\Delta\tau = \tau_m - \lambda_1 \quad (E-18)$$

$$X_1 = e^{-MT/\tau_m} \quad (E-19)$$

$$X_2 = e^{-MT/\lambda_1} \quad (E-20)$$

$$Y_1 = e^{-T/\tau_m} \quad (E-21)$$

$$Y_2 = e^{-T/\lambda_1} \quad (E-22)$$

Substituting into Equation E-17 and getting a common denominator,

$$\frac{T_M(Z)}{M_1(Z)} = HG_2(Z) = \frac{KZ^{-N}}{\Delta\tau} \left[\frac{(X_1 - X_2) - Z^{-1}(Y_2 X_1 - Y_1 X_2)}{(1 - Y_1 Z^{-1})(1 - Y_2 Z^{-1})} \right] \quad (E-23)$$

The closed loop transfer function from Figure E-2 and using Equation E-23 is:

$$\frac{T_M(Z)}{R(Z)} = \frac{KD_1(Z)HG_2(Z)}{1 + KD_1(Z)HG_2(Z)} \quad (E-24)$$

Solving Equation E-24 for the controller $D_1(Z)$:

$$D_1(Z) = \frac{1}{K} \frac{1}{HG_2(Z)} \left[\frac{T_M(Z)/R(Z)}{1 - T_M(Z)/R(Z)} \right] \quad (E-25)$$

Before Equation E-25 can be solved, the closed loop response $T_M(Z)/R(Z)$ must be defined. One of the design criteria is that the mold temperature not overshoot set point by more than 2°C (5°F) nor must it oscillate out of a $\pm 2^\circ\text{C}$ band during cure. To minimize cycle time, the response speed should be as fast as possible without overshooting. The response best suited for that is a critically damped second order or a first order response. Because a first order response is easier to work with and does not require knowledge of the roots, that is the response that will be used.⁷

To force the output variable ($T_M(S)$) to perform as a first order lag plus dead time to step changes in set point, it is so defined:

$$T_M(Z) = \mathcal{Z} \left[\frac{1}{S} \cdot \frac{e^{-\theta' S}}{\lambda_2 S + 1} \right] \quad (E-26)$$

where

λ_2 = Closed loop time constant and

θ' = Closed loop dead time.

The closed loop dead time defined in Equation E-26 must equal the plant dead time.

$$\theta' = (N-M)T \quad (E-27)$$

where

$$N = n_1 + n_2 + 2 \quad (E-28)$$

$$M = m_1 + m_2 \quad (E-29)$$

Using tables to determine the Z-transform of Equation E-26:

$$T_M(Z) = Z^{-N} \left[\frac{(1-X_3) + Z^{-1}(X_3 - Y_3)}{(1-Z^{-1})(1-Y_3 Z^{-1})} \right] \quad (E-30)$$

where

$$X_3 = e^{-MT/\lambda_2} \quad (E-31)$$

$$Y_3 = e^{-T/\lambda_2} \quad (E-32)$$

In the denominator of Equation E-30, the term $(1-Z^{-1})$ is a unit step in the Z domain. Extracting that term from the denominator and defining the constants

$$C_3 = (1-X_3) \quad (E-33)$$

$$C_4 = (X_3-Y_3) \quad (E-34)$$

then

$$\frac{T_M(Z)}{R(Z)} = Z^{-N} \left[\frac{C_3 + Z^{-1}C_4}{1 - Z^{-1}Y_3} \right] \quad (E-35)$$

Substituting Equations E-23 and E-35 into E-25 will yield the desired controller definition. First defining the following constants:

$$C_1 = X_1 - X_2 \text{ and} \quad (E-36)$$

$$C_2 = Y_2X_1 - Y_1X_2 ; \quad (E-37)$$

then

$$D_1(Z) = \frac{1}{K} \cdot \frac{\Delta\tau}{K_M Z^{-N}} \cdot \frac{(1-Z^{-1}Y_1)(1-Y_2Z^{-1})}{(C_1-C_2Z^{-1})} \cdot \frac{\frac{Z^{-N}(C_3+C_4)}{1-Y_3Z^{-1}}}{\frac{1-Y_3Z^{-1}-Z^{-N}(C_3+C_4Z^{-1})}{1-Y_3Z^{-1}}}$$

$$= \frac{C_3\Delta\tau}{KC_1K_M} \left[\frac{\left(1+Z^{-1}\frac{C_4}{C_3}\right)\left(1-Z^{-1}Y_1\right)\left(1-Z^{-1}Y_2\right)}{\left(1-Z^{-1}\frac{C_2}{C_1}\right)\left(1-Z^{-1}Y_3-Z^{-N}C_3-Z^{-N-1}C_4\right)} \right]$$

$$= \frac{C_3 \Delta \tau}{K C_1 K_M}$$

$$\left\{ \frac{1-Z^{-1} \left(Y_1 + Y_2 - \frac{C_4}{C_3} \right) - Z^{-2} \left[\left(Y_1 + Y_2 \right) \frac{C_4}{C_3} - Y_1 Y_2 \right] + Z^{-3} Y_1 Y_2 \frac{C_4}{C_3}}{1-Z^{-1} \left(\frac{C_2}{C_1} + Y_3 \right) + Z^{-2} \left(\frac{C_2}{C_1} Y_3 \right) - Z^{-N} C_3 - Z^{-N-1} \left(C_4 - \frac{C_3 C_2}{C_1} \right) + Z^{-N-2} \frac{C_4 C_2}{C_1}} \right\} \quad (E-38)$$

Defining the following constants,

$$K_{CS} = C_2 \Delta \tau / C_1 K_M \quad (E-39)$$

$$H_1 = Y_1 + Y_2 - C_4 / C_3 \quad (E-40)$$

$$H_2 = (Y_1 + Y_2) C_4 / C_3 - Y_1 Y_2 \quad (E-41)$$

$$H_3 = Y_1 Y_2 C_4 / C_3 \quad (E-42)$$

$$P_1 = (C_2 / C_1 + Y_3) \quad (E-43)$$

$$P_2 = (C_2 / C_1 Y_3) \quad (E-44)$$

$$P_3 = C_3 \quad (E-45)$$

$$P_4 = C_4 - \frac{C_3 C_2}{C_1} \quad (E-46)$$

$$P_5 = C_4 C_2 / C_1 \quad (E-47)$$

Equation E-38 becomes

$$D_1(Z) = \frac{M_1(Z)}{E(Z)} = \frac{K_{CS}}{K} \left[\frac{1-Z^{-1} H_1 - Z^{-2} H_2 + Z^{-3} H_3}{1-Z^{-1} P_1 + Z^{-2} P_2 - Z^{-N} P_3 - Z^{-N-1} P_4 + Z^{-N-2} P_5} \right] \quad (E-48)$$

Cross multiplying and converting to the time domain:

$$m_n = \frac{K_{CS}}{K} (e_n - H_1 e_{n-1} - H_2 e_{n-2} + H_3 e_{n-3}) + P_1 m_{n-1} - P_2 m_{n-2} + P_3 m_{n-N} + P_4 m_{n-N-1} - P_5 m_{n-N-2} \quad (E-49)$$

Equation E-49 is the positional controller.

$$D_p(Z) = K_{CS} \left[\frac{1+Z^{-1}H_1+Z^{-2}H_2+Z^{-3}H_3}{1-Z^{-1}F_1+Z^{-2}F_2-Z^{-3}F_3-Z^{-N-1}F_4+Z^{-N-2}F_5+Z^{-N-3}F_6+Z^{-N-4}F_7} \right]$$

$$= \frac{M(Z)}{E(Z)} \quad (E-50)$$

Cross multiplying and converting to the time domain:

$$K_{CS}E(Z) \left[1+Z^{-1}H_1+Z^{-2}H_2+Z^{-3}H_3 \right] = M(Z) \left[1-Z^{-1}F_1+Z^{-2}F_2-Z^{-3}F_3-Z^{-N-1}F_4 \right. \\ \left. +Z^{-N-2}F_5+Z^{-N-3}F_6+Z^{-N-4}F_7 \right] \quad (E-51)$$

and

$$m_n = K_{CS} (e_n + H_1 e_{n-1} + H_2 e_{n-2} + H_3 e_{n-3}) + F_1 m_{n-1} - F_2 m_{n-2} + F_3 m_{n-3} \\ + F_4 m_{n-N-1} - F_5 m_{n-N-2} - F_6 m_{n-N-3} - F_7 m_{n-N-4} \quad (E-52)$$

Equation E-52 is the positional form of the digital algorithm. To obtain the velocity form of the equation, the $n^{\text{th}}-1$ value is subtracted from the n^{th} value. The resulting equation is:

$$\Delta m_n = K_{CS} (\Delta e_n + H_1 \Delta e_{n-1} + H_2 \Delta e_{n-2} + H_3 \Delta e_{n-3}) + F_1 \Delta m_{n-1} - F_2 \Delta m_{n-2} \\ + F_3 \Delta m_{n-3} + F_4 \Delta m_{n-N-1} - F_5 \Delta m_{n-N-2} - F_6 \Delta m_{n-N-3} \\ - F_7 \Delta m_{n-N-4} \quad (E-53)$$

To determine if the algorithm is stable, the poles and zeroes of the pulse transfer function must be examined to determine whether they fall within the Z circle with radius one. Putting Equation E-53 into transfer function form,

$$D_V(Z) = \frac{K_{CS} (1-Z^{-1})(1-H_1)-Z^{-2}(H_1-H_2)-Z^{-3}(H_2-H_3)-Z^{-4}H_3}{1-Z^{-1}(F_1+1)+Z^{-2}(F_1+F_2)-Z^{-3}(F_2+F_3)+Z^{-4}F_3-Z^{-N-1}F_4} \\ \cdot \frac{1}{+Z^{-N-2}(F_4+F_5)-Z^{-N-3}(F_5+F_6)-Z^{-N-4}(F_6+F_7)-Z^{-N-5}F_7} \quad (E-54)$$

The poles and zeroes of Equation E-54 can be examined for stability criteria.

Appendix F

SOURCE LISTING OF ROOT FINDING
PROGRAM FOR FIRST ORDER PLANT
DIGITAL CONTROLLER

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```

PROGRAM POLE(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,TAPE3)
C
COMMON C(4),D(4),P(8),H(4),AKMO,T,C3,F(6),N,X,Y
DATA IY/1HY/
C
100 FORMAT(1H1,10H****TM****/)
101 FORMAT(1X,30H**THETA***---KM----- T****/)
102 FORMAT(3F10.0)
103 FORMAT(1H1,5X,5HTM = ,F6.1/)
104 FORMAT(6X,8HTHETA = ,F6.1/6X,5HKM = ,F6.4/6X,5HKP = ,
1F6.5/)
107 FORMAT(3X,8HC2/C1 = ,IPE13.6)
108 FORMAT(3X,24HTO STOP TYPE Y ELSE N = )
109 FORMAT(A1)
110 FORMAT(3X,4HN = ,I2,3X,4HM = ,F6.4)
111 FORMAT(3X,38HWANT CLOSED LOOP CONTROLLER CONSTANTS ,
116HTYPE Y ELSE N = )
112 FORMAT(6X,16HSAMPLE PERIOD = ,F6.1/)
C
1 WRITE(2,100)
READ(1,102)TM
WRITE(2,101)
READ(1,102)TH,AKM,T
AKMO = 0.16*AKM
N=TH/T
DF=TH-N*T
AM=1.-(DF/T)
B=1./TM
Y=EXP(-AM*B*T)
X=EXP(-B*T)
C(1) = 1. - Y
C(2) = Y - X
C3 = C(2)/C(1)
C
WRITE(3,103)TM
WRITE(3,112)T
WRITE(3,104)TH,AKM,AKMO
WRITE(3,110)N,AM
WRITE(3,107)C3
C
WRITE(2,111)
READ(1,109)IC
IF(IC.EQ.IY)CALL TIME(AM)
WRITE(2,108)
READ(1,109)IS
IF(IS.NE.IY)GO TO 1
STOP
END

```

```

SUBROUTINE TIME(AM)
C
C   CLOSED LOOP TIME CONSTANT NEEDED TO DETERMINE
C   CONTROLLER COEFFICIENTS.
C
COMMON C(4),D(4),P(8),H(4),AKMO,T,C3,F(6),N,X,Y
DIMENSION S(20),Z(19),SH(19),SA(19)
COMPLEX Z
C
100 FORMAT(1H-,2X,35HINCREMENT CLOSED LOOP TIME CONSTANT/1X
1,18H*FROM*--TO*--**BY**/)
101 FORMAT(3F6.0)
102 FORMAT(1H-,3X,6HKCS = ,1PE13.6)
103 FORMAT(6(2X,2HP(,11,4H) = ,1PE13.6))
104 FORMAT(6(2X,2HH(,11,4H) = ,1PE13.6))
105 FORMAT(3X,33H INPUT NEW TIME CONSTANT Y OR N -)
106 FORMAT(A1)
107 FORMAT(1H-,2X,26HNUMERATOR COEFFICIENTS ARE)
108 FORMAT(1H-,2X,28HDENOMINATOR COEFFICIENTS ARE)
109 FORMAT(6(2X,2HF(,11,4H) = ,1PE13.6))
110 FORMAT(1H-,2X,18HF COEFFICIENTS ARE)
112 FORMAT(1H-,2X,31HPOSITIONAL CONTROLLER POLES ARE)
113 FORMAT(1H-,2X,29HVELOCITY CONTROLLER POLES ARE)
114 FORMAT(5X,9HREAL ROOT,6X,14HIMAGINARY ROOT,3X,
19HMAGNITUDE,8X,5HANGLE)
115 FORMAT(4X,3(1PE13.6,3X),0P,F9.3)
116 FORMAT(1H-,2X,28HCLOSED LOOP TIME CONSTANT = ,F10.2)
117 FORMAT(6(2X,2HS(,11,4H) = ,1PE13.6))
118 FORMAT(6(2X,2HD(,11,4H) = ,1PE13.6))
119 FORMAT(6(2X,2HC(,11,4H) = ,1PE13.6))
120 FORMAT(3X,22HINPUT VARIABLE GAIN - )
121 FORMAT(F10.0)
122 FORMAT(3X,21HVARIABLE GAIN (KC) = ,F11.5/)
123 FORMAT(3X,9HKC*KCS = ,1PE13.6,0P)
124 FORMAT(1H-,2X,40HNUMERATOR ZEROES FOR POSITIONAL FORM ARE /)
125 FORMAT(8(/))
C
DATA IY/1HY/
C
1 WRITE(2,100)
READ(1,101)TCI,TCF,TIN
WRITE(2,120)
READ(1,121)VK
TC = TCI
50 IF(TC.GT.TCF) GO TO 12
XP=EXP(-T/TC)
YP=EXP(-T*AM/TC)
C(3) = 1. - YP
C(4) = YP - XP
AKCL = C(3)/(AKMO*C(1))
C4 = C(4)/C(3)

```

```

D(1) = X - C4
D(2) = X*C4
F(1) = XP - C3
F(2) = C3*XP
F(3) = C(3)
F(4) = C(4) + C3*C(3)
F(5) = C3*C(4)
H(1) = D(1) + 1.
H(2) = D(1) - D(2)
H(3) = D(2)
P(1) = F(1) + 1.
P(2) = F(1) - F(2)
P(3) = F(2)
P(4) = F(3)
P(5) = F(3) - F(4)
P(6) = F(4) - F(5)
P(7) = F(5)

```

C

```

WRITE(3,125)
WRITE(3,116) TC
WRITE(3,102) AKCL
WRITE(3,122) VK
VKP = VK*AKCL
WRITE(3,123) VKP
WRITE(3,119) (I,C(I),I=1,4)
WRITE(3,118) (I,D(I),I=1,2)
S(1) = VKP
S(2) = -VKP*D(1)
S(3) = -VKP*D(2)
NS = 2
CALL ZRPOLY(S,NS,Z,IER)
DO 30 I=1,NS
  ARL = REAL(Z(I))
  AIM = AIMAG(Z(I))
  SH(I) = CABS(Z(I))
  IF(SH(I).LE.1.E-5) SA(I) = 0.0
  IF(SH(I).LE.1.E-5) GO TO 30
  AGL = AIM/ARL
  SUM = 0.0
  IF(ARL.GT.0.0.AND.AIM.LT.0.0) SUM = 360.
  IF(ARL.LT.0.0) SUM = 180.
  SA(I) = ATAN(AGL)*57.3 + SUM
30 CONTINUE
WRITE(3,124)
WRITE(3,117) (I,S(I),I=1,3)
WRITE(3,114)
DO 35 I=1,NS

```

```

35 WRITE(3,115)Z(I),SH(I),SA(I)
   WRITE(3,110)
   WRITE(3,109)(J,F(J),J=1,5)
   NS = N+3
   NP = N+1
   NC = NS+1
   DO 2 I=1,NC
2  S(I) = 0.0
   S(1) = 1.
   II=3
   DO 3 I=1,2
   K=I+1
3  S(K) = -F(I)
   DO 4 I=1,3
   K=I+N+1
   S(K) = -F(II) + S(K)
4  II = II + 1
   CALL ZRPOLY(S,NS,Z,IER)
   DO 5 I=1,NS
   ARL = REAL(Z(I))
   AIM = AIMAG(Z(I))
   SH(I) = SQRT(ARL*ARL + AIM*AIM)
   IF(SH(I).LE.1.E-5) SA(I) = 0.0
   IF(SH(I).LE.1.E-5) GO TO 5
   AGL = AIM/ARL
   SUM = 0.0
   IF(ARL.GT.0.0.AND.AIM.LT.0.0) SUM = 360.
   IF(ARL.LT.0.0) SUM = 180.
   SA(I) = ATAN(AGL)*57.3 + SUM
5  CONTINUE
   WRITE(3,117)(I,S(I),I=1,NC)
   WRITE(3,112)
   WRITE(3,114)
   DO 6 I=1,NS
6  WRITE(3,115)Z(I),SH(I),SA(I)
   WRITE(3,107)
   WRITE(3,104)(I,H(I),I=1,3)
   WRITE(3,108)
   WRITE(3,103)(I,P(I),I=1,7)
   NS = N+4
   NP = N + 1
   NC = NS + 1
   DO 7 I=1,NC
7  S(I) = 0.0
   S(1) = 1.
   II = 5
   DO 8 I=2,3
   K=I+1

```

```

8 S(K) = P(I)
  S(2) = -P(1)
  DO 9 I=2,4
    K=I+N+1
    S(K) = P(II) + S(K)
9  II = II + 1
    K=N+2
    S(K) = -P(4) + S(K)
    CALL ZRPOLY(S,NS,Z,IER)
    DO 10 I=1,NS
      ARL = REAL(Z(I))
      AIM = AIMAG(Z(I))
      SH(I) = SQRT(ARL*ARL + AIM*AIM)
      IF(SH(I).LE.1.E-5) SA(I) = 0.0
      IF(SH(I).LE.1.E-5) GO TO 10
      AGL = AIM/ARL
      SUM = 0.0
      IF(ARL.GT.0.0.AND.AIM.LT.0.0) SUM = 360.
      IF(ARL.LT.0.0) SUM = 180.
      SA(I) = ATAN(AGL)*57.3 + SUM
10  CONTINUE
      WRITE(3,117)(I,S(I),I=1,NC)
      WRITE(3,113)
      WRITE(3,114)
      DO 11 I=1,NS
11  WRITE(3,115)Z(I),SH(I),SA(I)
      TC = TC + TIN
      GO TO 50
C
12 WRITE(2,105)
    READ(1,106) NTC
    IF(NTC.EQ.IY) GO TO 1
    RETURN
    END

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix G

SOURCE LISTING OF ROOT FINDING
PROGRAM FOR CASCADED FLUID AND
MOLD TEMPERATURE DIGITAL CONTROLLERS

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```

C      PROGRAM POL(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,TAPE3)
C
C      THIS ROUTINE FINDS THE POLES AND ZEROES OF THE MAJOR LOOP
C      AND MINOR LOOP CONTROLLERS.
C
COMMON/AA/C(10),Q(10),P(10),H(10),R(10)
COMMON/BB/S(20),SH(19),SA(19),Z(19)
COMPLEX Z
REAL M1,M2
DATA IY/IHY/

C
100 FORMAT(1H1,30H****TM****---KM-----*THETAM**/)
101 FORMAT(1X,30H**** T*****--TO-----*THETA0**/)
102 FORMAT(3F10.0)
103 FORMAT(1H1,5X,35HCASCADE CONTROLLER POLES AND ZEROES)
104 FORMAT(1H-,5X,22HMETAL TIME CONSTANT = ,F10.3,4H SEC)
105 FORMAT(1H-,5X,22HFLUID TIME CONSTANT = ,F10.3,4H SEC)
106 FORMAT(6X,13HMETAL GAIN = ,F10.6)
107 FORMAT(6X,13HFLUID GAIN = ,F10.6)
108 FORMAT(6X,18HMETAL DEAD TIME = ,F5.1,4H SEC)
109 FORMAT(6X,18HFLUID DEAD TIME = ,F5.1,4H SEC)
110 FORMAT(1H-,5X,16HSAMPLE PERIOD = ,F4.0,4H SEC)
111 FORMAT(1H-,5X,31HMAJOR LOOP CONTROLLER CONSTANTS/)
112 FORMAT(1H-,5X,19HNUMERATOR CONSTANTS/)
113 FORMAT(1H-,5X,21HDENOMINATOR CONSTANTS/)
114 FORMAT(6(2X,2HH(,11,4H) = ,1PE13.6)0P)
115 FORMAT(6(2X,2HP(,11,4H) = ,1PE13.6)0P)
116 FORMAT(1H-,5X,31HMINOR LOOP CONTROLLER CONSTANTS/)
117 FORMAT(6(2X,2HQ(,11,4H) = ,1PE13.6)0P)
118 FORMAT(6(2X,2HR(,11,4H) = ,1PE13.6)0P)
119 FORMAT(1H-,5X,15HC CONSTANTS ARE/)
120 FORMAT(6(2X,2HC(,11,4H) = ,1PE13.6)0P)
121 FORMAT(1H-,5X,10HZEROES ARE/)
122 FORMAT(1H-,5X,9HPOLES ARE/)
123 FORMAT(5X,9HREAL ROOT,6X,14HIMAGINARY ROOT,3X,
*9HMAGNITUDE,8X,5HANGLE)
124 FORMAT(4X,3(1PE13.6,3X),0P,F9.3)
125 FORMAT(1H-/1H-,5X,32HCLOSED LOOP TIME CONSTANTS ---- ,
*8HMAJOR = ,F6.0,4H SEC,3X,8HMINOR = ,F6.0,4H SEC/)
126 FORMAT(6X,31HCLOSED LOOP GAINS ---- MAJOR = ,F12.6,3X,
*8HMINOR = ,F12.6/)
127 FORMAT(5X,35HINCREMENT MINOR LOOP TIME CONSTANTS/)
128 FORMAT(5X,35HINCREMENT MAJOR LOOP TIME CONSTANTS/)
129 FORMAT(1X,18H*FROM---TO---*BY**/)
130 FORMAT(3F6.0)
131 FORMAT(5X,24HTO STOP TYPE Y ELSE N = )
132 FORMAT(A1)
133 FORMAT(5X,5HN1 = ,I3,3X,5HN2 = ,I3,3X,5HM1 = ,F7.5,3X,
*5HM2 = ,F7.5/)

```

C

```

1 WRITE(2,100)
  READ(1,102)TM,AKM,THM
  WRITE(2,101)
  READ(1,102)T,TO,THO
  AKO = 1.
  N1 = THO/T
  N2 = THM/T
  DF1 = THO - N1*T
  DF2 = THM - N2*T
  M1 = 1. - (DF1/T)
  M2 = 1. - (DF2/T)
  SM = M1 + M2

```

C

```

  WRITE(3,103)
  WRITE(3,104)TM
  WRITE(3,106)AKM
  WRITE(3,108)THM
  WRITE(3,105)TO
  WRITE(3,107)AKO
  WRITE(3,109)THO
  WRITE(3,133)N1,N2,M1,M2
  WRITE(3,110)T

```

C

```

  WRITE(2,127)
  WRITE(2,129)
  READ(1,130)G1S,G1E,G1B
  WRITE(2,128)
  WRITE(2,129)
  READ(1,130)G2S,G2E,G2B
  G2 = G2S

```

C

```

2 X3 = EXP(-T/TM)
  X4 = EXP(-T/G2)
  Y3 = EXP(-SM*T/TM)
  Y4 = EXP(-T*(M1+M2)/G2)
  C(7) = 1. - Y4
  C(8) = Y4 - X4
  G1=G1S
3 X1 = EXP(-T/TO)
  X2 = EXP(-T/G1)
  Y1 = EXP(-T*M1/TO)
  Y2 = EXP(-T*M1/G1)
  Y3P = EXP(-SM*T/G1)

```

C

```

  C(1) = 1. - Y1
  C(2) = Y1 - X1
  C(3) = 1. - Y2
  C(4) = Y2 - X2
  C(5) = Y3 - Y3P
  C(6) = Y3P*X3 - Y3*X2
  DT = TM - G1

```

C

```

D1 = X1 - (C(4)/C(3))
D2 = C(4)*X1/C(3)
F1 = X2 - (C(2)/C(1))
F2 = X2*C(2)/C(1)
F3 = C(3)
F4 = C(4) + (C(2)*C(3)/C(1))
F5 = C(2)*C(4)/C(1)

```

C

```

Q(1) = D1 + 1.
Q(2) = D1 - D2
Q(3) = D2

```

C

```

R(1) = F1 + 1.
R(2) = F1 - F2
R(3) = F2
R(4) = F3
R(5) = F3 - F4
R(6) = F4 - F5
R(7) = F5

```

C

```

AKCS1 = C(3)/(0.16*C(1))
AKCS2 = C(7)*DT/(AKM*C(5))

```

C

```

H(1) = X2 + X3 - (C(8)/C(7))
H(2) = X2*X3 - (C(8)*(X2+X3)/C(7))
H(3) = C(8)*X2*X3/C(7)

```

C

```

P(1) = X4 - (C(6)/C(5))
P(2) = C(6)*X4/C(5)
P(3) = C(7)
P(4) = C(8) + (C(6)*C(7)/C(5))
P(5) = C(6)*C(8)/C(5)

```

C

```

WRITE(3,125) G2,G1
WRITE(3,126) AKCS2,AKCS1

```

C

```

WRITE(3,111)
WRITE(3,112)
WRITE(3,114) (I,H(I),I=1,3)
WRITE(3,121)
S(1) = 1.0
S(2) = -H(1)
S(3) = H(2)
S(4) = H(3)
NS = 3
CALL ZRPOLY(S,NS,Z,IER)
CALL ANGL(NS)

```

C

```

WRITE(3,123)
DO 4 I = 1,NS
4 WRITE(3,124) Z(I),SH(I),SA(I)

```

C

```

WRITE(3,113)
WRITE(3,115)(I,P(I),I=1,5)
WRITE(3,122)
N2P = N1 + N2 + 2
NS = N2P + 2
NC = NS + 1
DO 5 I = 1,NC
5 S(I) = 0.0
S(1) = 1.0
S(2) = -P(1)
S(3) = -P(2)
II=3
DO 6 I=1,3
K=I+N2P
S(K) = -P(II) + S(K)
6 II=II+1
CALL ZRPOLY(S,NS,Z,IER)
CALL ANGL(NS)

```

C

```

WRITE(3,123)
DO 7 I=1,NS
7 WRITE(3,124)Z(I),SH(I),SA(I)

```

C

```

WRITE(3,116)
WRITE(3,112)
WRITE(3,117)(I,Q(I),I=1,3)
WRITE(3,121)
S(1) = 1.
S(2) = -Q(1)
S(3) = Q(2)
S(4) = Q(3)
NS = 3

```

C

```

CALL ZRPOLY(S,NS,Z,IER)
CALL ANGL(NS)

```

C

```

WRITE(3,123)
DO 8 I=1,NS
8 WRITE(3,124)Z(I),SH(I),SA(I)

```

C

```

WRITE(3,113)
WRITE(3,118)(I,R(I),I=1,7)
WRITE(3,122)
NS = N1+4
NC = NS + 1
DO 9 I=1,NC
9 S(I) = 0.0
S(1) = 1.
II=5

```

```

      DO 10 I=2,3
      K = I + 1
10  S(K) = R(I)
      S(2) = - R(1)
C
      DO 11 I=2,4
      K=I+N1+1
      S(K) = R(II) + S(K)
11  II=II+1
      K=N1+2
      S(K) = -R(4) + S(K)
C
      CALL ZRPOLY(S,NS,Z,IER)
      CALL ANGL(NS)
C
      WRITE(3,123)
      DO 12 I=1,NS
12  WRITE(3,124)Z(I),SH(I),SA(I)
C
      WRITE(3,119)
      WRITE(3,120)(I,C(I),I=1,8)
C
      G1 = G1 + G1B
      IF(G1.LE.G1E) GO TO 3
      G2 = G2 + G2B
      IF(G2.LE.G2E) GO TO 2
C
      WRITE(2,131)
      READ(1,132)IE
      IF(IE.NE.IY) GO TO 1
      STOP
      END
      SUBROUTINE ANGL(NS)
C
      COMMON/BB/S(20),SH(19),SA(19),Z(19)
      COMPLEX Z
C
      DO 1 I=1,NS
      AR = REAL(Z(I))
      AM = AIMAG(Z(I))
      SH(I) = CABS(Z(I))
      IF(SH(I).LT.1.E-5) SA(I) = 0.0
      IF(SH(I).LT.1.E-5) GO TO 1
      AG = AM/AR
      SUM = 0.0
      IF(AR.GT.0.0.AND.AM.LT.0.0) SUM = 360.
      IF(AR.LT.0.0) SUM = 180.
      SA(I) = ATAN(AG)*57.3 + SUM
1  CONTINUE
      RETURN
      END

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix H

SOURCE LISTING OF ROOT FINDING
PROGRAM FOR CASCADED INNER LOOP
PI AND OUTER LOOP DIGITAL CONTROLLERS

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

```

C      PROGRAM CPOL(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,TAPE3)
C
C      THIS ROUTINE FINDS THE POLES AND ZEROES OF THE MAJOR LOOP
C      AND MINOR LOOP CONTROLLERS.
C
COMMON/AA/C(10),Q(10),P(10),H(10),R(10)
COMMON/BB/S(100),SH(99),SA(99),Z(99)
COMPLEX Z
REAL M1,M2
DATA IY/1HY/

C
100 FORMAT(1H1,30H****TM*****--KM---**THETAM**/)
101 FORMAT(1X,30H**** T*****--TO---**THETAU**/)
102 FORMAT(3F10.0)
103 FORMAT(1H1,5X,35HCASCADE CONTROLLER POLES AND ZEROES)
104 FORMAT(1H-,5X,22HMETAL TIME CONSTANT = ,F10.3,4H SEC)
105 FORMAT(1H-,5X,22HFLUID TIME CONSTANT = ,F10.3,4H SEC)
106 FORMAT(6X,13HMETAL GAIN = ,F10.6)
107 FORMAT(6X,13HFLUID GAIN = ,F10.6)
108 FORMAT(6X,18HMETAL DEAD TIME = ,F5.1,4H SEC)
109 FORMAT(6X,18HFLUID DEAD TIME = ,F5.1,4H SEC)
110 FORMAT(1H-,5X,16HSAMPLE PERIOD = ,F4.0,4H SEC)
111 FORMAT(1H-,5X,31HMAJOR LOOP CONTROLLER CONSTANTS/)
112 FORMAT(1H-,5X,19HNUMERATOR CONSTANTS/)
113 FORMAT(1H-,5X,21HDENOMINATOR CONSTANTS/)
114 FORMAT(6(2X,2HH(,11,4H) = ,1PE13.6)0P)
115 FORMAT(6(2X,2HP(,11,4H) = ,1PE13.6)0P)
116 FORMAT(1H-,5X,31HMINOR LOOP CONTROLLER CONSTANTS/)
117 FORMAT(6(2X,2HQ(,11,4H) = ,1PE13.6)0P)
118 FORMAT(6(2X,2HR(,11,4H) = ,1PE13.6)0P)
119 FORMAT(1H-,5X,15HC CONSTANTS ARE/)
120 FORMAT(6(2X,2HC(,11,4H) = ,1PE13.6)0P)
121 FORMAT(1H-,5X,10HZEROES ARE/)
122 FORMAT(1H-,5X,9HPOLES ARE/)
123 FORMAT(5X,9HREAL ROOT,6X,14HIMAGINARY ROOT,3X,
*9HMAGNITUDE,8X,5HANGLE)
124 FORMAT(4X,3(1PE13.6,3X),0P,F9.3)
125 FORMAT(1H-/1H-,5X,32HCLOSED LOOP TIME CONSTANTS ---- ,
*8HMAJOR = ,F6.0,4H SEC,3X,8HMINOR = ,F6.0,4H SEC/)
126 FORMAT(6X,31HCLOSED LOOP GAINS ---- MAJOR = ,F12.6,3X,
*8HMINOR = ,F12.6/)
127 FORMAT(5X,35HINCREMENT MINOR LOOP TIME CONSTANTS/)
128 FORMAT(5X,35HINCREMENT MAJOR LOOP TIME CONSTANTS/)
129 FORMAT(1X,18H*FROM---TO---**BY**/)
130 FORMAT(3F6.0)
131 FORMAT(5X,24HTO STOP TYPE Y ELSE N = )
132 FORMAT(A1)
133 FORMAT(5X,5HN1 = ,I3,3X,5HN2 = ,I3,3X,5HM1 = ,F7.5,3X,
*5HM2 = ,F7.5/)

```

C

```

1 WRITE(2,100)
  READ(1,102)TM,AKM,THM
  WRITE(2,101)
  READ(1,102)T,TO,THO
  AKO = 1.
  N1 = THO/T
  N2 = THM/T
  DF1 = THO - N1*T
  DF2 = THM - N2*T
  M1 = 1. - (DF1/T)
  M2 = 1. - (DF2/T)
  SM = M1 + M2

```

C

```

  WRITE(3,103)
  WRITE(3,104)TM
  WRITE(3,106)AKM
  WRITE(3,108)THM
  WRITE(3,105)TO
  WRITE(3,107)AKO
  WRITE(3,109)THO
  WRITE(3,133)N1,N2,M1,M2
  WRITE(3,110)T

```

C

```

  WRITE(2,127)
  WRITE(2,129)
  READ(1,130)G1S,G1E,G1B
  WRITE(2,128)
  WRITE(2,129)
  READ(1,130)G2S,G2E,G2B
  G2 = G2S

```

C

```

  X1 = EXP(-SM*T/TM)
  Y1 = EXP(-T/TM)
  X3 = EXP(-SM*T/G2)
  Y3 = EXP(-T/G2)
  C(3) = 1. - X3
  C(4) = X3 - Y3
  CR2 = C(4)/C(3)
  G1 = G1S

```

2

```

  X2 = EXP(-SM*T/G1)
  Y2 = EXP(-T/G1)
  C(1) = X1 - X2
  C(2) = Y2*X1 - Y1*X2
  CR1 = C(2)/C(1)
  DT = TM - G1

```

3

C

```

  AKCS2 = C(3)*DT/(C(1)*AKM)

```

C

```

  H(1) = (Y1+Y2) - CR2
  H(2) = (Y1+Y2)*CR2 - Y1*Y2
  H(3) = Y1*Y2*CR2

```

C

```

P(1) = Y3 + CR1
P(2) = Y3*CR1
P(3) = C(3)
P(4) = C(4) -C(3)*CR1
P(5) = CR1*C(4)

```

C

```

WRITE(3,125) G2,G1
WRITE(3,126) AKCS2

```

C

```

WRITE(3,111)
WRITE(3,112)
WRITE(3,114) (I,H(I),I=1,3)
WRITE(3,121)
S(1) = 1.0
S(2) = -H(1)
S(3) = -H(2)
S(4) = H(3)
NS = 3
CALL ZRPOLY(S,NS,Z,IER)
CALL ANGL(NS)

```

C

```

WRITE(3,123)
DO 4 I = 1,NS

```

```

4 WRITE(3,124) Z(I),SH(I),SA(I)

```

C

```

WRITE(3,113)
WRITE(3,115) (I,P(I),I=1,5)
WRITE(3,122)
P(5) = -P(5)
N2P = N1 + N2 + 2
NS = N2P + 2
NC = NS + 1
DO 5 I = 1,NC

```

```

5 S(I) = 0.0

```

```

S(1) = 1.0
S(2) = -P(1)
S(3) = P(2)

```

```

II=3

```

```

DO 6 I=1,3

```

```

K=I+N2P

```

```

S(K) = -P(II) + S(K)

```

```

6 II=II+1

```

```

CALL ZRPOLY(S,NS,Z,IER)
CALL ANGL(NS)

```

C

```

WRITE(3,123)

```

```

DO 7 I=1,NS

```

```

7 WRITE(3,124) Z(I),SH(I),SA(I)

```

```

C
C
WRITE(3,119)
WRITE(3,120) (I,C(I),I=1,4)
C
G1 = G1 + G1B
IF(G1.LE.G1E) GO TO 3
G2 = G2 + G2B
IF(G2.LE.G2E) GO TO 2
C
WRITE(2,131)
READ(1,132) IE
IF(IE.NE.IY) GO TO 1
STOP
END
SUBROUTINE ANGL(NS)
C
COMMON/BB/S(100),SH(99),SA(99),Z(99)
COMPLEX Z
C
DO 1 I=1,NS
AR = REAL(Z(I))
AM = AIMAG(Z(I))
SH(I) = CABS(Z(I))
IF(SH(I).LT.1.E-5) SA(I) = 0.0
IF(SH(I).LT.1.E-5) GO TO 1
AG = AM/AR
SUM = 0.0
IF(AR.GT.0.0.AND.AM.LT.0.0) SUM = 360.
IF(AR.LT.0.0) SUM = 180.
SA(I) = ATAN(AG)*57.3 + SUM
1 CONTINUE
RETURN
END

```

Appendix I

SOURCE LISTING OF SIMULATION
PROGRAM FOR FIRST ORDER PLANT
DIGITAL CONTROLLER

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK


```

OVERLAY(CDIG1,0,0)
PROGRAM CONTRL(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,TAPE6)

```

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NM,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/DD/ITY,ITX,IGY,IGX,NG,SR(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON/II/ISAMP,NTIME,NDY,IDY,CDI(20),CDO

```

```

ND(1)  = 1  READS IDENTIFER TITLES AND INTEGRATION CONTROLS.
ND(2)  = 1  READS INITIAL CONDITIONS OF STATE VARIABLES.- Y'S
ND(3)  = 1  READS MAGNITUDES OF DELAYS. H'S
ND(4)  = 1  READS THE PARAMETER LIST. P'S
ND(5)  = 1  READS X AND Y SUBSCRIPTS TO BE PRINTED AND PLOTTED.
ND(6)  = 0  SETS INITIAL CONDITIONS OF Y'S EQUAL TO THEIR FINAL
            VALUE FROM THE PREVIOUS RUN.
ND(6)  = 1  SETS INITIAL CONDITIONS OF Y'S EQUAL TO THE VALUE
            READ IN FROM DATA CARDS.
ND(7)  = 1  SETS IBUG
ND(8)  = 1  SETS IADJ CAUSING PARADJ TO BE CALLED.
ND(16) = 0  CAUSES SCALE TO SET SCALE LIMITS FOR EACH PLOT.
ND(16) = 1  CAUSES SCALE TO READ IN UPPER AND LOWER LIMITS WHICH
            ARE USED ON ALL PLOTS.
ND(16) = 2  CAUSES SCALE TO SCAN ALL PLOT DATA TO FIND LIMITS
            WHICH ARE THEN USED FOR ALL PLOTS.
ND(16) = 3  CAUSES SCALE TO READ PLOT LIMITS FOR EACH PLOT.
ND(16) = 4  CAUSES SCALE TO USE PREVIOUSLY DEFINED LIMITS.
ND(17) = 1  CAUSES A NEW SETPOINT RECIPE TO BE READ IN.
ND(18) = 1  CAUSES STORED X DATA TO BE WRITTEN TO TAPE6.

```

```

C
C   P'S ARE PARAMETER VARIABLES READ IN OR DEFINED IN INIFUN.
C   X'S ARE INTERMEDIATE VARIABLES.
C   Y'S ARE STATE VARIABLES.
C   F'S ARE STATE DERIVATIVES.
C   H'S ARE TIME DELAY MAGNITUDES.
C
C
C   X(1) AND Y(NSYS+1) ARE RESERVED FOR TIME.
C
C   BTIME = BEGINNING TIME
C   ETIME = END TIME
C   PTIME = PRINT OUT INTERVAL
C   DELT  = INTEGRATION STEP SIZE  MUST BE .1,.2, .5 OR A MULTIPLE OF
C           10 TIMES THOSE STEP SIZE VALUES.
C
C
C   101 FORMAT(1H1,10X,46HTHE OUTPUT INTERVAL IS TOO SMALL - OUTPUT ARRA,
C       *30HY EXCEEDS 800  PTIME RESET TO ,F7.2//)
C   102 FORMAT(//,10X,29HTHE ISE OF THE MAJOR LOOP IS ,1PE14.7/10X,
C       *29HTHE IAE OF THE MAJOR LOOP IS ,1PE14.7)
C   103 FORMAT(20(1X,I1))
C   104 FORMAT(4(5X,2HY(,I2,4H) = ,F12.5,2X))
C   105 FORMAT(4(5X,2HX(,I2,4H) = ,F12.5,2X))
C   150 FORMAT(8(F9.2,1H ))
C   160 FORMAT(8(I2,1X))
C   161 FORMAT(I2,1X,I2,1X,I3)
C   162 FORMAT(2X,36H1ST ORDER SYSTEM DIGITAL CONTROLLER.)
C   163 FORMAT(5(F12.6,2X))
C
C
C   IRUN = 0
C   1 IPAGE = 0
C     IRUN = IRUN + 1
C     READ(1,103)(ND(I),I=1,20)
C     IF(EOF(1))999,2
C   2 IBUG = ND(7)
C     JSTOP = 0

```

```

NSUM = 0
DO 3 I=1,20
3 NSUM = NSUM + ND(I)
  IF(NSUM.EQ.0) JSTOP = 1
  IF(JSTOP.EQ.1.AND.IADJ.EQ.0) GO TO 999
  IADJ = ND(8)
C
C   ABOVE READS IN NDATA AND TERMINATES RUN IF NO CHANGES IN ND
C
  CALL INPUT
  CALL INIFUN
  IF(NX.EQ.0) GO TO 4
  CALL INTCAL
4 CALL DIFFEQ
  CALL ALIST
C
C   ABOVE INITIALIZES PROGRAM AND PRINTS OUT HEADER INFO.
C
  NSTEP = ((ETIME - BTIME)/DELT) + 0.5
  NW = ((PTIME/DELT) + 0.5)
  NMOG = NSTEP/NW
  IF(NMOG.GT.800) GO TO 6
  GO TO 7
6 NPT = NSTEP/800
  NW = NPT + (1./DELT)
  PTIME = NW*DELT
  WRITE(2,101)PTIME
C
C   ABOVE ESTABLISHES PRINT INTERVAL
C
7 JI=1
  DO 8 I=1,MX
8 XS(JI,I) = X(JX(I))
  DO 9 I=1,MY
9 YS(JI,I) = Y(JY(I))
C
C   STORE IN PRINT MATRIX INITIAL X AND Y PRINT/PLOT VARIABLES.
C
C   PROCEED WITH PROGRAM EXECUTION BELOW.

```

C

```

NI = 0
DO 15 KK=1,NSTEP
NI = NI + 1
CALL RKINT(KK)
IF(IRUG.NE.1) GO TO 10
N1 = NY + 1
WRITE(2,104)N1,Y(N1)
WRITE(2,104)(I,Y(I),I=1,NY)
WRITE(2,105)(I,X(I),I=1,NX)
10 IF(NX.EQ.0) GO TO 11
CALL INTCAL
11 IF(NI.NE.NW) GO TO 15
NI = 0
IF(IRUG.NE.1) GO TO 12
WRITE(2,104)N1,Y(N1)
WRITE(2,104)(I,Y(I),I=1,NY)
WRITE(2,105)(I,X(I),I=1,NX)
12 JI = JI + 1

```

C

C

C

STORE X AND Y DATA IN PRINT/PLOT MATRICES BELOW.

```

IF(MX.EQ.0) GO TO 50
DO 13 I=1,MX
XS(JI,I) = X(JX(I))
IF(IGX.EQ.0) GO TO 13
IF(XS(JI,I).GE.XG(I,2)) XG(I,2) = XS(JI,I)
IF(XS(JI,I).LE.XG(I,1)) XG(I,1) = XS(JI,I)
13 CONTINUE

```

C

```

50 IF(MY.EQ.0) GO TO 15
DO 14 I=1,MY
YS(JI,I) = Y(JY(I))
IF(IGY.EQ.0) GO TO 14
IF(YS(JI,I).GE.YG(I,2)) YG(I,2) = YS(JI,I)
IF(YS(JI,I).LE.YG(I,1)) YG(I,1) = YS(JI,I)
14 CONTINUE
15 CONTINUE

```

C

C NOW THAT ITERATIONS COMPLETE PRINT AND PLOT RESULTS.
C

IF(ND(18).NE.1) GO TO 201
WRITE(6,162)
WRITE(6,161)MX,NP,JI
WRITE(6,160)(JX(I),I=1,MX)
WRITE(6,163)(P(I),I=1,NP)
DO 200 I=1,JI
200 WRITE(6,150)(XS(I,JJK),JJK=1,MX)
ENDFILE 6
201 CONTINUE
NOUT = JI
IF(ITY.EQ.0) GO TO 16

C
C PRINT Y TABLES.
C

CALL PRTTBL(JI,0)
16 IF(ITX.EQ.0) GO TO 17

C
C PRINT X TABLES.
C

CALL PRTTBL(JI,1)
17 IF(IGY.EQ.0) GO TO 18

C
C PLOT Y DATA.
C

NG = IGY
CALL PLOT(YS,1,YG)
IPAGE = IPAGE + 1
18 IF(IGX.EQ.0) GO TO 19

C
C PLOT X DATA.
C

NG = IGX
CALL PLOT(XS,0,XG)
IPAGE = IPAGE + 1

C
C ADJUST THE PARAMETER (P'S) LIST IF IADJ = 1
C

```

19 IF (IADJ.NE.0) CALL PARADJ
   WRITE(2,102)Y(3),Y(4)
   IF (JSTOP.NE.0) GO TO 999
   GO TO 1
999 STOP
   END
   SUBROUTINE INPUT

```

C
C

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSG,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

```

C
C

```

100 FORMAT(8F10.0)
101 FORMAT(10A8)
103 FORMAT(4(I2,1X))
104 FORMAT(8(1X,I2))
105 FORMAT(3(3A8,1X))

```

C
C
C

ZERO OUT ARRAYS

```

DO 1 I=1,8
DO 1 J=1,2
XG(I,J) = 0.0
1 YG(I,J) = 0.0
IF (ND(5).EQ.0) GO TO 3
DO 2 I=1,8
NGY(I) = 0
NGX(I) = 0
NTY(I) = 0
NTX(I) = 0
JX(I) = 0

```

```

2 JY(I) = 0
  ITY=ITX=IGY=IGX=MY=MX=0
3 IF (ND(1).EQ.0) GO TO 4

```

C
C
C
C
C
C

```

  READ IN TWO 80 CHARACTER TITLES, MOLD TITLE(MT), MOLD NUMBER(MN),
  AND MOLD SECTION(MS). THEN READ DATE(JD), NUMBER OF Y'S, X'S,
  P'S, AND H'S AS WELL AS BEGIN AND END TIMES, INTEGRATION SIZE,
  AND PRINT INTERVAL.

```

```

  READ(1,101)(IT1(I),I=1,10)
  READ(1,101)(IT2(J),J=1,10)
  READ(1,105)(MT(K),K=1,3),(MN(L),L=1,3),(MS(M),M=1,3)
  READ(1,105)(JD(K),K=1,3)
  READ(1,103)NY,NX,NP,NH
  READ(1,100)BTIME,ETIME,DELT,PTIME
  Y(NY+1) = BTIME
  Y(NY+2) = DELT
  N2 = NY + 2
  NMAX = 2*NY + 2

```

```

4 IF (ND(2).EQ.0) GO TO 5
  READ(1,100)(Y(I),I=1,NY)
  GO TO 7
5 IF (ND(6).NE.0) GO TO 7
  DO 6 I=1,N2
6 Y(I) = YT(I)
7 DO 8 I=1,N2
8 YT(I) = Y(I)

```

C
C
C
C
C
C

```

  INTEGRATION CONTROLS READ IF ND(1) = 1 ABOVE. ALSO, THE INITIAL
  CONDITIONS OF THE STATE VARIABLES ARE SET.

```

```

  IF (ND(3).EQ.0) GO TO 9
  READ(1,100)(H(I),I=1,NH)
9 IF (ND(4).EQ.0) GO TO 10
  READ(1,100)(P(I),I=1,NP)

```

C
C
C
C

```

  THE ABOVE READS THE TIME DELAY AND PARAMETER MAGNITUDE ARRAYS.

```

```

C      BELOW READ IN Y AND X INDEX NUMBERS FOR PRINT AND PLOT I/O.
C
10  IF(ND(5).EQ.0) GO TO 18
    READ(1,104)(NTY(I),I=1,8)
    READ(1,104)(NGY(I),I=1,8)
    READ(1,104)(NTX(I),I=1,8)
    READ(1,104)(NGX(I),I=1,8)
C
C      DETERMINE HOW MANY OF EACH FUNCTION ARE TO BE PERFORMED.
C
    DO 11 J=1,8
      IF(NTX(J).NE.0) ITX = ITX + 1
      IF(NTY(J).NE.0) ITY = ITY + 1
      IF(NGX(J).NE.0) IGX = IGX + 1
      IF(NGY(J).NE.0) IGY = IGY + 1
11  CONTINUE
C
C      STORE PRINT X INDICES IN ARRAY JX , Y INDICES IN JY.BELOW
C
    DO 13 I=1,8
      IF(NTX(I).EQ.0) GO TO 12
      MX = MX + 1
      JX(MX) = NTX(I)
12  IF(NTY(I).EQ.0) GO TO 13
      MY = MY + 1
      JY(MY) = NTY(I)
13  CONTINUE
C
C      CHECK TO SEE IF PLOT X AND Y INDICES ARE THE SAME AS PRINT INDICES
C      IF SO DO NOT STORE INDEX. IF NOT SAME STORE INDEX IN JX OR JY.
C
    DO 17 I=1,8
      IF(NGX(I).EQ.0) GO TO 15
      DO 14 J=1,MX
        IF(NGX(I).EQ.JX(J)) GO TO 15
14  CONTINUE
      MX = MX + 1
      IF(MX.GT.8) MX = 8
      JX(MX) = NGX(I)
15
17

```



```

15 IF(NGY(I).EQ.0) GO TO 17
   DO 16 J=1,MY
   IF(NGY(I).EQ.JY(J)) GO TO 17
16 CONTINUE
   MY = MY + 1
   IF(MY.GT.8) MY = 8
   JY(MY) = NGY(I)
17 CONTINUE

```

C
C
C

```

18 IF(ND(17).NE.0) CALL RECIPE

```

C

```

RETURN
END
SUBROUTINE RECIPE

```

C
C
C

```

READS IN THE MAJOR LOOP SET POINT RECIPE.

```

C

```

COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1

```

C

```

1 FORMAT(3F10.0)

```

```

2 FORMAT(I2)

```

C

```

READ(1,2)NSEG

```

C

```

DO 3 I=1,NSEG

```

```

3 READ(1,1)RAMP(I),STPT(I),TIM(I)

```

C

```

JFLAG = 0

```

```

RETURN

```

```

END

```

```

SUBROUTINE ALIST

```

C
C
C

```

HEADER OUTPUT ROUTINE.

```

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP

```

```

COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)

```

```

COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1

```

COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
 COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
 COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

C
C
C

```

100 FORMAT(1H1,6X,48HMOLD FEEDBACK ONLY DIGITAL CONTROLLER SIMULATION,
113H ** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9HPAGE NO. ,I3/)
101 FORMAT(1X,120(1H*))
102 FORMAT(65X,10HDESCRIPTOR/)
103 FORMAT(20X,10A8/20X,10A8//)
104 FORMAT(5X,19H* PROGRAM CONTROLS,24X,1H*,16X,17H* MOLD IDENTIFIE,
*2HRS,33X,1H*)
105 FORMAT(5X,1H*,42X,1H*,16X,1H*,51X,1H*)
106 FORMAT(5X,24H* NUMBER OF Y EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
*10HMOLD TITLE,13X,3A8,2X,1H*)
107 FORMAT(//5X,44(1H*),16X,53(1H*))
108 FORMAT(5X,24H* NUMBER OF X EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
*11HMOLD NUMBER,12X,3A8,2X,1H*)
109 FORMAT(5X,28H* NUMBER OF P PARAMETERS ,I3,12X,1H*,16X,1H*,2X,
*12HMOLD SECTION,11X,3A8,2X,1H*)
110 FORMAT(5X,21H* NUMBER OF H DELAYS,7X,I3,12X,1H*,16X,1H*,2X,
*15HSIMULATION DATE,8X,3A8,2X,1H*)
111 FORMAT(5X,13H* START TIME,12X,F6.2,3X,10HSECONDS *,
116X,1H*,51X,1H*)
112 FORMAT(5X,11H* END TIME,11X,F9.2,3X,10HSECONDS *,
116X,1H*,51X,1H*)
113 FORMAT(5X,18H* OUTPUT INTERVAL,8X,F5.2,3X,10HSECONDS *,16X,
*53(1H*))
114 FORMAT(5X,25H* INTEGRATOR STEP SIZE ,F6.3,3X,10HSECONDS */5X,
*44(1H*)//)
115 FORMAT(//5X,44HTHE FOLLOWING ARE THE TIME DELAYS IN SECONDS/)
116 FORMAT(4(5X,2HH(,I2,4H) = ,F12.3,2X))
117 FORMAT(//5X,38HTHE FOLLOWING IS THE PARAMETER LISTING/)
118 FORMAT(4(5X,2HP(,I2,4H) = ,F12.6,2X))

```

```

119 FORMAT(/5X,48HENSUING ARE THE INITIAL CONDITIONS OF THE STATE ,
      *9H VARIABLES/)
120 FORMAT(4(5X,2HY(,I2,4H) = ,F12.5,2X))
121 FORMAT(/5X,49HENSUING ARE THE X VALUES AFTER INIFUN AND INTCAL ,
      *33HBUT BEFORE THE FIRST INTEGRATION./)
122 FORMAT(4(5X,2HX(,I2,4H) = ,F12.4,2X))
123 FORMAT(8(/),50X,20HCONTROLLER CONSTANTS//)
124 FORMAT(2X,2(2HD(,I2,4H) = ,1PE13.6,5X),0P)
125 FORMAT(2X,5(2HF(,I2,4H) = ,1PE13.6,5X),0P)
126 FORMAT(5X,25H* METAL TIME CONSTANT ,F6.1,3X,10HSECONDS */
      15X,25H* METAL PROCESS GAIN ,F6.4,3X,10HDEG/DEG *)
127 FORMAT(5X,25H* FLUID TIME CONSTANT ,F6.1,3X,10HSECONDS */
      15X,25H* FLUID PROCESS GAIN ,F6.4,3X,10HDEG/DEG *)
128 FORMAT(5X,25H* CLOSED LOOP TIME CONST,F6.1,3X,10HSECONDS */
      15X,25H* CLOSED LOOP GAIN KCS ,1PE13.6,0P)
129 FORMAT(2X,4(2HC(,I1,4H) = ,1PE13.6,6X),0P)
130 FORMAT(5X,8HC3/C2 = ,1PE13.6,5X,7HBETA = ,1PE13.6,5X,6HKCL = ,
      11PE13.6,0P)
131 FORMAT(39X,42H* THE METAL LOOP SET POINT RECIPE IS *)
132 FORMAT(39X,42(1H*))
133 FORMAT(39X,42H* RAMP * SETPOINT * DWELL * SEGMENT *)
134 FORMAT(39X,42H* F/MIN * DEGREE F * MIN * NO. *)
135 FORMAT(39X,2H* ,F5.1,2H *,3X,F5.1,2X,3H* ,F5.2,3H *,4X,12,5X,
      *1H*)
136 FORMAT(///)
137 FORMAT(5X,20H* MOLD DEAD TIME = ,5X,F6.1,3X,10HSECONDS *)

```

C

```

IPAGE = IPAGE + 1
WRITE(2,100) (JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,102)
WRITE(2,103) (IT1(I),I=1,10), (IT2(J),J=1,10)
WRITE(2,101)
WRITE(2,107)
WRITE(2,104)
WRITE(2,105)
WRITE(2,106) NY, (MT(J),J=1,3)
WRITE(2,108) NX, (MN(J),J=1,3)
WRITE(2,109) NP, (MS(J),J=1,3)

```

```

WRITE(2,110)NH,(JD(J),J=1,3)
WRITE(2,111)BTIME
WRITE(2,112)ETIME
WRITE(2,113)PTIME
WRITE(2,114)DELT
WRITE(2,101)
WRITE(2,115)
WRITE(2,101)
WRITE(2,116)(I,H(I),I=1,NH)
WRITE(2,101)
WRITE(2,117)
WRITE(2,101)
WRITE(2,118)(I,P(I),I=1,NP)
WRITE(2,101)
IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,119)
WRITE(2,101)
WRITE(2,120)(I,Y(I),I=1,NY)
WRITE(2,101)
WRITE(2,121)
WRITE(2,101)
WRITE(2,122)(I,X(I),I=1,NX)
WRITE(2,101)
WRITE(2,101)
IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,123)
WRITE(2,124)(I,P(I),I=1,2)
WRITE(2,125)(I,P(I+4),I=1,5)
WRITE(2,129)(I,C(I),I=1,4)
WRITE(2,127)P(23),P(22)
WRITE(2,126)P(13),P(12)
WRITE(2,128)P(18),P(4)
WRITE(2,137)P(14)
WRITE(2,136)
WRITE(2,101)

```

```

WRITE(2,136)
WRITE(2,132)
WRITE(2,131)
WRITE(2,132)
WRITE(2,133)
WRITE(2,134)
WRITE(2,132)
WRITE(2,135) (RAMP(I),STPT(I),TIM(I),I,I=1,NSEG)
WRITE(2,132)
IPAGE = IPAGE + 1
RETURN
END
SUBROUTINE INIFUN

```

```

C
C
C  INITILIZE FUNCTION SUBROUTINE.

```

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON/II/ISAMP,NTIME,NDY,IDY,CDI(20),CDO

```

```

C
C
100 FORMAT(1H1,5X,23H** TIME DELAY VARIABLE ,I1,18H EXCEEDS THE MAXIM,
*46HUM ARRAY SIZE WHEN DIVIDED BY DELT RGM STOP **/)

```

```

C
C
IMAX = 0
DO 1 I=1,NH
IH(I) = H(I)/DELT + 1
IF(IH(I).GE.IMAX) IMAX = IH(I)
IF(IMAX.GT.500) GO TO 2
1 CONTINUE
GO TO 3
2 WRITE(2,100)I
STOP
3 DO 4 I=1,NX
4 X(I) = 0.0

```

```

ADY = (P(14)/P(15) + 0.05)
NDY = ADY
P(19) = NDY
IDY = NDY + 1
DO 5 I=1,IDY
5 CDI(I) = 0.0
X(1) = BTIME
SAM = P(15)/DELT + 0.05
ISAMP = SAM
NTIME = 0

```

C

```

X(32) = Y(2)
X(32) = ALIM(X(32),50.,600.)
X(35) = Y(2)
X(34) = Y(2)
X(36) = Y(2)
X(2) = Y(2)
X(3) = X(2)*P(16) + P(17)
DO 6 I=1,IMAX
6 XSTR(1,I) = Y(2)
XTH(1) = XSTR(1,IMAX)

```

C

```

DF=P(14) - P(15)*NDY
AM = 1. - (DF/P(15))
AP = 1./P(13)
YA = EXP(-AM*AP*P(15))
XA = EXP(-AP*P(15))
BP = 1./P(18)
YB = EXP(-AM*BP*P(15))
XB = EXP(-BP*P(15))
C(1) = 1. - YA
C(2) = YA - XA
C(3) = 1.- YB
C(4) = YB - XB
C3 = C(2)/C(1)
C4 = C(4)/C(3)
AK = C(3)/(0.16*P(12)*C(1))
P(1) = XA - C4
P(2) = XA*C4

```

```

P(4) = AK
P(5) = XB - C3
P(6) = XB*C3
P(7) = C(3)
P(8) = C(4) + C3*C(3)
P(9) = C3*C(4)

```

C

```

RETURN
END
SUBROUTINE INTCAL

```

C

C

C

THIS IS THE INTERMEDIATE CALCULATION SUBROUTINE.

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON/II/ISAMP,NTIME,NDY,IDY,CDI(20),CDO

```

C

C

C

```

X(1) = Y(NY+1)
IF(X(1).LE.BTIME) GO TO 1
IF(NTIME.EQ.ISAMP) CALL DCON1
NTIME = NTIME + 1
1 XT(1) = Y(2)

```

C

CALL DELAY

C

```

X(36) = XTH(1)
X(34) = Y(1)
X(35) = Y(2)

```

C

```

RETURN
END
FUNCTION ALIM(V,B,T)
ALIM = V
IF(V.GT.T) ALIM = T
IF(V.LT.B) ALIM = B

```

RETURN
END
SUBROUTINE DCON1

```

C
C THIS ROUTINE IS THE DIGITAL CONTROLLER DESIGNED FROM A FIRST
C ORDER SYSTEM. THE PLANT IS FIRST ORDER.
C SEE DERIVATION DATED 5/19/76.
C
COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/II/ISAMP,NTIME,NDY,IDY,CDI(20),CDO

C CALL SPADJ

C
X(5) = X(4)
X(8) = X(7)
X(7) = X(6)

C
X(22) = X(21)
X(21) = X(20)
X(20) = CDO
X(27) = X(26)
X(26) = X(18)

C
X(37) = X(36)*P(16) + P(17)
X(4) = X(3) - X(37)
X(6) = X(4) - X(5)
X(9) = X(7)*P(1)
X(10) = X(8)*P(2)

C
X(11) = X(6) - X(9)
X(12) = X(11) - X(10)
X(13) = X(12)*P(3)*P(4)

C
X(28) = X(26)*P(5)
X(29) = X(27)*P(6)
X(23) = X(20)*P(7)
X(24) = X(21)*P(8)

```


X(25) = X(22)*P(9)

C

X(14) = X(13) + X(28)

X(15) = X(14) + X(29)

X(16) = X(15) + X(23)

X(17) = X(16) + X(24)

X(18) = X(17) + X(25)

X(19) = ALIM(X(18),-50.,50.)

C

IF(NDY.EQ.0) GO TO 1

CALL DLY

1 CDI(1) = X(18)

IF(NDY.EQ.0) CDO = CDI(1)

X(33) = X(32)

X(30) = X(19)*P(11)

X(31) = 0.16*X(30)

X(32) = X(31) + X(33)

X(32) = ALIM(X(32),P(20),P(21))

C

NTIME = 0

RETURN

END

SUBROUTINE DLY

C

COMMON/II/ISAMP,NTIME,NDY,IDY,CDI(20),CDO

C

DO 1 I= 1,NDY

K = IDY + 1 - I

J = IDY - I

1 CDI(K) = CDI(J)

CDO = CDI(IDY)

C

RETURN

END

SUBROUTINE SPADJ

C

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP

COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1

C

C

THIS IS THE SET POINT ADJUSTMENT ROUTINE. THE RECIPE IS READ IN

```

C      FROM SUBROUTINE RECIPE.  SOME OF THE FEATURES OF THE ROUTINE ARE%
C
C      ... 1) NO RAMP MAY BE SPECIFIED IN SEGMENT 1.
C      ... 2) DWELL PERIODS ARE IN MINUTES AND APPLY TO THE SPECIFIED
C      ...      SEGMENT ONLY.
C      ... 3) THE SET POINT IN ANY SEGMENT EXCEPT 1 MAY BE SPECIFIED AS ZERO
C      ...      AND THAT SEGMENT WILL USE THE PRECEDING SEGMENTS SET POINT.
C      ... 4) TWO CONSECUTIVE SEGMENTS MUST HAVE AT LEAST ONE SET POINT
C      ...      SPECIFIED.
C      ... 5) THE DWELL IN ANY ONE SEGMENT MAY BE SPECIFIED AS ZERO AND THAT
C      ...      SEGMENT WILL BE PASSED OVER.  THIS OPTION IS USUALLY USED TO
C      ...      SPECIFY A DIFFERENT SET POINT AS A START POINT FOR A RAMP.
C      ... 6) A DWELL MUST BE SPECIFIED IN A RAMP SEGMENT.
C      ... 7) NO TWO CONSECUTIVE SEGMENTS MAY BE SPECIFIED WITH A ZERO DWELL
C      ... 8) POSITIVE RAMPS ARE DEFINED BY A PLUS OR NO SIGN BEFORE THE
C      ...      RAMP RATE MAGNITUDE.
C      ... 9) COOL DOWN RAMPS ARE DEFINED BY A NEGATIVE SIGN BEFORE THE RAMP
C      ...      RATE MAGNITUDE
C      ... 10) A RAMP SEGMENT MUST HAVE SPECIFIED A RAMP RATE, A TERMINATION
C      ...      SET POINT, AND A TERMINATION TIME LIMIT.  THAT TIME LIMIT MAY
C      ...      BE SPECIFIED GREATER THAN THE ACTUAL RAMP TIME.
C
C
C
C

```

```

      IF(JFLAG.EQ.1) GO TO 2
      ST = 0.0
      DWELL(1) = TIM(1)*60.
      JFLAG = 1
      DO 1 I=2,NSEG
      K=I-1
      1 DWELL(I) = DWELL(K) + TIM(I)*60.
      2 IF(X(1).GE.DWELL(1)) GO TO 3
      SP = STPT(1)
      I=1
      GO TO 14
      3 DO 4 I=2,NSEG
      IF(X(1).GE.DWELL(I-1).AND.X(1).LT.DWELL(I)) GO TO 5
      4 CONTINUE
      RETURN
      5 IRAMP = RAMP(I)
      IF(IRAMP.NE.0) GO TO 6

```

```

GO TO 7
6 BT = DWELL(I-1)
  SLP = RAMP(I)/60.
  SP = SLP*(X(I)-BT) + STPT(I-1)
  IF(STPT(I-1).LT.50.) SP = SLP*(X(I)-BT) + STPT(I-2)
  IF(ABS(SP-STPT(I)).LT.0.5.AND.X(I).LT.DWELL(I)) GO TO 12
  IF(RAMP(I).LT.0.0) GO TO 11
  GO TO 10
7 SP = STPT(I)
  IF(SP.GE.50.0) GO TO 9
  SP = STPT(I-1)
  IF(I.GT.2) GO TO 8
  GO TO 14
8 CH = SP - STPT(I-2)
  IF(CH.LT.0.0) GO TO 11
  GO TO 10
9 CH = SP - STPT(I-1)
  IF(CH.LT.0.0) GO TO 11
10 P1 = P(12)
  P2 = P(13)
  P3 = 0.0
  GO TO 14
11 P1 = P(12)
  P2 = P(13)
  P3 = 0.0
  GO TO 14
12 ADT = DWELL(I) - X(I)
  SP = STPT(I+1)
  IF(SP.LT.50.) SP=STPT(I)
  DWELL(I) = DWELL(I) - ADT
  IF(SP.EQ.STPT(I-1)) CH = SP-STPT(I-2)
  IF(CH.LT.0.0) GO TO 11
  CH = SP - STPT(I-1)
  I=I+1
  IF(I.GT.NSEG) GO TO 14
  DO 13 K=I,NSEG
13 DWELL(K) = DWELL(K-1) + TIM(K)*60.
  GO TO 10
14 X(2) = SP

```

```
X(3) = SP*P(16) + P(17)
```

```
C
```

```
MMSP = I
```

```
RETURN
```

```
END
```

```
SUBROUTINE DELAY
```

```
C
```

```
C
```

```
THIS SUBROUTINE SETS UP THE DELAY PIPELINES.
```

```
C
```

```
COMMON/BB/XSTR(1,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
```

```
C
```

```
ISTOP = IMAX - 1
```

```
DO 1 KK=1,ISTOP
```

```
IS = IMAX - KK
```

```
JS = IS + 1
```

```
DO 1 II = 1,NH
```

```
1 XSTR(II,JS) = XSTR(II,IS)
```

```
C
```

```
DO 2 II = 1,NH
```

```
JJ = IH(II)
```

```
XTH(II) = XSTR(II,JJ)
```

```
2 XSTR(II,1) = XT(II)
```

```
C
```

```
RETURN
```

```
END
```

```
SUBROUTINE DIFFEQ
```

```
C
```

```
C
```

```
DIFFERENTIAL EQUATION ROUTINE.
```

```
C
```

```
COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
```

```
C
```

```
F(1) = FOL(P(22),X(32),Y(1),P(23))
```

```
F(2) = FOL(P(12),Y(1),Y(2),P(13))
```

```
F(3) = X(4)*X(4)
```

```
F(4) = ABS(X(4))
```

```
C
```

```
C
```

```
C
```

```
DO 1 LK = 1,NY
```

```

      IQ = NY + 2 + LK
1 Y(IQ) = F(LK)
C
      RETURN
      END
      FUNCTION FOL(GA,XIN,YOUT,TAU)
      IF(TAU.NE.0.0) GO TO 1
      YOUT = GA*XIN
      FOL = 0.0
      RETURN
1 FOL = (GA*XIN - YOUT)/TAU
      RETURN
      END
      SUBROUTINE RKINT(LL)
C
C      RUNGE-KUTTA NUMERICAL INTEGRATION ROUTINE FOR SOLVING DIFFERENTIAL
C      EQUATIONS.
C
      COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
C
      DIMENSION YU(42),BET(3),DY(4,20)
      DOUBLE PRECISION YU
C
      IF(LL.NE.1) GO TO 1
      BET(1) = 0.5
      BET(2) = 0.5
      BET(3) = 1.0
      N2 = NY + 2
      N1 = NY + 1
      XV = Y(N1)
      CALL DIFFEQ
1 DO 2 I=1,NY
2 YU(I) = Y(I)
      DO 6 K=1,4
      IF(K.EQ.1) GO TO 3
      CALL DIFFEQ
3 DO 4 I=1,NY
      IJ = I + N2
4 DY(K,I) = Y(N2)*Y(IJ)

```

```

      IF(K.EQ.4) GO TO 6
      DO 5 I=1,NY
5     Y(I) = YU(I) + BET(K)*DY(K,I)
      Y(N1) = XV + BET(K)*Y(N2)
6     CONTINUE
      DO 7 I=1,NY
      DEL = (DY(1,I)+2.*DY(2,I)+2.*DY(3,I)+DY(4,I))/6.
      YU(I) = YU(I) + DEL
      Y(I) = YU(I)
7     CONTINUE
      Y(N1) = XV + Y(N2)
      CALL DIFFEQ
      XV = Y(N1)
      RETURN
      END
      SUBROUTINE PRTTBL(MAXLIN,IV)

```

C
C
C

THIS ROUTINE PRINTS THE SPECIFIED TABLES.

```

COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/HH/BTIME,ETDME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON ARY(800,8),NPRT(8)

```

C
C

```

100 FORMAT(1H1,6X,48HMOLD FEEDBACK ONLY DIGITAL CONTROLLER SIMULATION,
113H ** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9HPAGE NO. ,I3/)
101 FORMAT(8(7X,A1,A1,I2,A1,2X))
102 FORMAT(8(2X,F12.4))
103 FORMAT(1X,120(1H*)/)

```

C
C

```

      IF(IV.EQ.0)GO TO 3
      IB = IX
      NCOL = ITX
      DO 1 I=1,MAXLIN
      DO 1 J=1,NCOL
1     ARY(I,J) = XS(I,J)

```

```

DO 2 I=1,NCOL
2 NPRT(I) = NTX(I)
GO TO 6
3 IB = IY
  NCOL = IY
  DO 4 I=1,MAXLIN
    DO 4 J=1,NCOL
4 ARY(I,J) = YS(I,J)
  DO 5 I=1,NCOL
5 NPRT(I) = NTY(I)

```

C
C
C

STORE X OR Y DATA IN ARY MATRIX ABOVE.

```

6 LINE = 1
  WRITE(2,100) (JD(I),I=1,3),IRUN,IPAGE
  WRITE(2,103)
  WRITE(2,101) (IB,PL,NPRT(I),PR,I=1,NCOL)

```

C

```

DO 7 KK = 1,MAXLIN
  WRITE(2,102) (ARY(KK,I),I=1,NCOL)
  LINE = LINE + 1
  IF (LINE.LT.55) GO TO 7
  LINE = 1
  IPAGE = IPAGE + 1
  WRITE(2,103)
  WRITE(2,100) (JD(I),I=1,3),IRUN,IPAGE
  WRITE(2,103)
  WRITE(2,101) (IB,PL,NPRT(I),PR,I=1,NCOL)
7 CONTINUE

```

C
C
C

ABOVE PRINTS THE DATA IN MATRIX ARY IN TABULAR FORM.

```

WRITE(2,103)
RETURN
END
SUBROUTINE PLOT(SAVE,NSS,GR)

```

C
C
C

THIS ROUTINE PLOTS THE DATA STORED IN THE SAVE MATRIX.

```

REAL LINE(101),DEL(8)
DIMENSION SAVE(800,8),IN(8)
DIMENSION GR(8,2),STEPS(10),NL(8)

```

C
C

```

COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

```

C

```

DATA PLUS,BLANK/1H+,1H /

```

C
C

```

100 FORMAT(1H1,6X,48HMOLD FEEDBACK ONLY DIGITAL CONTROLLER SIMULATION,
113H.** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9HPAGE NO. ,I3/)
101 FORMAT(1H+,50X,9HSYMBOL = ,A1,26H, FOR COINCIDENT POINTS)
102 FORMAT(1H )
103 FORMAT(1H ,20X,A1,A1,I2,A1,2H =,F11.2,12X,9HSYMBOL = ,A1,1H,,4X,
*15HSCALE FACTOR = ,F11.3,12X,A1,A1,I2,A1,2H =,F11.2)
104 FORMAT(1H ,12X,15HTIME IN SECONDS)
105 FORMAT(///31X,20H** OUTPUT WARNING **,I5,2X,17HPOINTS WERE NOT W,
*22HITHIN THE GRAPH RANGE.)
106 FORMAT(1H ,20X,101A1)
107 FORMAT(1H+,7X,F11.2)

```

C
C
C
C
C
C

```

IN(1) IS RESERVED FOR TIME.

```

C

```

IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,102)

```

```

MSS = NSS
NN = 2
IF(MSS.EQ.1) GO TO 2
DO 1 I=1,IGX
DO 1 J=1,MX
IF(NGX(I).EQ.JX(J)) IN(I) = J

```



```

1 CONTINUE
  GO TO 4
2 DO 3 I=1,IGY
  DO 3 J=1,MY
    IF(NGY(I).EQ.JY(J)) IN(I) = J
3 CONTINUE
C
4 CALL SCALE(INN,GR,IN)
C
  DO 6 I=2,NG
    IF(GR(IN(I),1).NE.0.0.OR.GR(IN(I),2).NE.0.)GO TO 5
    GR(IN(I),1) = -0.1
    GR(IN(I),2) = 0.1
    GO TO 6
5 IF(GR(IN(I),1).NE.GR(IN(I),2)) GO TO 6
  GR(IN(I),1) = GR(IN(I),1) - (0.1*FLOAT(I)*ABS(GR(IN(I),1)))
  GR(IN(I),2) = GR(IN(I),2) + (0.1*ABS(GR(IN(I),2)))
6 DEL(I) = (GR(IN(I),2) - GR(IN(I),1))/10.
  IF(MSS.EQ.1) GO TO 7
  WRITE(2,103)(IX,PL,NGX(I),PR,GR(IN(I),1),SB(I),DEL(I),IX,PL,NGX(I)
    *,PR,GR(IN(I),2),I=2,NG)
  GO TO 8
7 WRITE(2,103)(IY,PL,NGY(I),PR,GR(IN(I),1),SB(I),DEL(I),IY,PL,NGY(I)
    *,PR,GR(IN(I),2),I=2,NG)
8 WRITE(2,104)
  IF((NG-1).EQ.1) GO TO 9
  WRITE(2,101) SB(9)
9 DO 10 I=2,NG
10 GR(IN(I),2) = 100./(GR(IN(I),2) + GR(IN(I),1))

```

C ABOVE SETS AND WRITES OUT PLOT INFO AND CONSTANTS.

C BELOW PLOT THE DATA IN SAVE MATRIX.

C
 NPR=0
 NO = 0
 DO 20 K=1,NOUT
 NPR = NPR + 1
 IF(K.NE.1.AND.K.NE.NOUT) GO TO 11

```

      NPR = 5
11  IF(NPR.NE.5) GO TO 13
      DO 12 I=1,101
12  LINE(I) = PLUS
      GO TO 16
13  DO 14 I=2,100
14  LINE(I) = BLANK
      DO 15 I=1,101,10
15  LINE(I) = PLUS
16  DO 19 I=2,NG
      N = IFIX((SAVE(K,IN(I))-GR(IN(I),1))*GR(IN(I),2)+1.5)
      IF(N.GE.1.AND.N.LE.101) GO TO 17
      NO = NO + 1
      GO TO 19
17  IF(LINE(N).EQ.BLANK.OR.LINE(N).EQ.PLUS) GO TO 18
      LINE(N) = SB(9)
      GO TO 19
18  LINE(N) = SB(I)
19  CONTINUE
      WRITE(2,106) LINE
      IF(NPR.NE.5) GO TO 20
      WRITE(2,107) SAVE(K,IN(1))
      NPR = 0
20  CONTINUE
      IF(NO.NE.0) WRITE(2,105) NO
21  RETURN
      END
      SUBROUTINE SCALE(JJ,GR,JG)

```

C

C

C

THIS ROUTINE SCALES THE DATA FOR PLOT LIMITS.

```

COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME

```

C

```

DIMENSION GR(8,2),STEPS(10),JG(8)
DIMENSION SGR(8,2)

```

C

```

DATA STEPS/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./

```

C

100 FORMAT(2F10.0)

C

IF(ND(16).EQ.4) GO TO 15
IF(ND(16).EQ.0) GO TO 7
IF(ND(16).EQ.2) GO TO 3
IF(ND(16).EQ.3) GO TO 1
READ(1,100)YMIN,YMAX
GO TO 5

1 DO 2 I=2,NG
READ(1,100)YMIN,YMAX
SGR(JG(I),1) = YMIN
SGR(JG(I),2) = YMAX
GR(JG(I),1) = YMIN
2 GR(JG(I),2) = YMAX
RETURN

C

3 YMIN = GR(JG(2),1)
YMAX = GR(JG(2),2)
IF((NG-1).EQ.1) GO TO 7

C

DO 4 I=JJ,NG
IF(YMIN.GE.GR(JG(I),1)) YMIN = GR(JG(I),1)
IF(YMAX.LE.GR(JG(I),2)) YMAX = GR(JG(I),2)
4 CONTINUE
5 DO 6 I=JJ,NG
SGR(JG(I),1) = YMIN
SGR(JG(I),2) = YMAX
GR(JG(I),1) = YMIN
6 GR(JG(I),2) = YMAX

C

IF(ND(16).EQ.1.OR.ND(16).EQ.2) RETURN

C

7 DO 14 K=JJ,NG
YMIN = GR(JG(K),1)
YMAX = GR(JG(K),2)
DY = (YMAX - YMIN)/10.
IF(DY.LE.0.0) GO TO 14
TA = ALOG10(DY)
IR = TA

```

      IF(TA.LT.0.0) IR = TA - 1.
      DY = DY*10.**(-IR) - 0.005
      DO 8 J=1,10
      IF(STEPS(J).LT.DY) GO TO 8
      I=J
      GO TO 9
8     CONTINUE
      I=10
9     CONTINUE
      IF(I.LE.10) GO TO 10
      I=I-10
      IR=IR+1
10    CONTINUE
      COR = STEPS(I)*10.**IR
      IF(I.EQ.10) IR=IR+1
      IF(YMIN.GE.0.0) GO TO 11
      MIN = YMIN/COR - .995
      GO TO 12
11    CONTINUE
      MIN = YMIN/COR + .005
12    CONTINUE
      TA = MIN*COR
      TB = TA + 10.*COR
      IF(TB.GE.YMAX) GO TO 13
      I=I+1
      GO TO 9
13    GR(JG(K),1) = TA
      GR(JG(K),2) = TB
      SGR(JG(K),1) = TA
      SGR(JG(K),2) = TB
14    CONTINUE
      RETURN
15    DO 16 I=JJ,NG
      GR(JG(I),1) = SGR(JG(I),1)
16    GR(JG(I),2) = SGR(JG(I),2)
      RETURN
      END
      SUBROUTINE PARADJ

```

```

COMMON/AA/X(50),P(40),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
C
  DIMENSION NPA(20)
C
  1 FORMAT(20(I2,1X))
  2 FORMAT(I2)
  3 FORMAT(8F10.0)
C
  READ(1,2)JP
  READ(1,1)(NPA(J),J=1,JP)
C
  READ(1,3)(P(NPA(J)),J=1,JP)
C
  RETURN
  END
  BLOCK DATA
C
  COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
C
  DATA SB/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H*,1H8,1H+,1H-/
  DATA PL,PR,IX,IY/1H(,1H),1HX,1HY/
C
  END

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix J

SOURCE LISTING OF SIMULATION PROGRAM FOR CASCADED DIGITAL CONTROLLERS

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

OVERLAY(MDCAS,0,0)
PROGRAM CONTRL(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,TAPE6)

COMMON/AA/X(80),P(80),Y(22),Z(22),F(10),C(9),NY,NX,NP,MMS
COMMON/BB/XSTR(2,500),XT(9),XTH(9),IH(9),NM,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/DD/ITY,ITX,IGY,IGX,NG,S8(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NXX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/GG/ND(20),NPT,NSP,NM,PTIME
COMMON/HH/BTIME,ETIME,DELTA,I1(10),I2(10),MT(3),MN(3),JD(3),MS(3)
COMMON/II/ISAMP,NTIME,NDP,IDP,CDD(2),ND2,ID2,ND1,ID1

ND(1) = 1 READS IDENTIFIER TITLES AND INTEGRATION CONTROLS.
ND(2) = 1 READS INITIAL CONDITIONS OF STATE VARIABLES.- Y'S
ND(3) = 1 READS MAGNITUDES OF DELAYS. H'S
ND(4) = 1 READS THE PARAMETER LIST. P'S
ND(5) = 1 READS X AND Y SUBSCRIPTS TO BE PRINTED AND PLOTTED.
ND(6) = 0 SETS INITIAL CONDITIONS OF Y'S EQUAL TO THEIR FINAL
VALUE FROM THE PREVIOUS RUN.
ND(6) = 1 SETS INITIAL CONDITIONS OF Y'S EQUAL TO THE VALUE
READ IN FROM DATA CARDS.
ND(7) = 1 SETS IBUG
ND(8) = 1 SETS IADJ CAUSING PARADJ TO BE CALLED.
ND(16) = 0 CAUSES SCALE TO SET SCALE LIMITS FOR EACH PLOT.
ND(16) = 1 CAUSES SCALE TO READ IN UPPER AND LOWER LIMITS WHICH
ARE USED ON ALL PLOTS.
ND(16) = 2 CAUSES SCALE TO SCAN ALL PLOT DATA TO FIND LIMITS
WHICH ARE THEN USED FOR ALL PLOTS.
ND(16) = 3 CAUSES SCALE TO READ PLOT LIMITS FOR EACH PLOT.
ND(16) = 4 CAUSES SCALE TO USE PREVIOUSLY DEFINED LIMITS.
ND(17) = 1 CAUSES A NEW SETPOINT RECIPE TO BE READ IN.
ND(18) = 1 CAUSES STORED X DATA TO BE WRITTEN TO TAPE6.

X(1) AND Y(NSYS+1) ARE RESERVED FOR TIME.

```

BTIME = BEGINNING TIME
ETIME = END TIME
PTIME = PRINT OUT INTERVAL
DELT  = INTEGRATION STEP SIZE  MUST BE .1,.2, .5 OR A MULTIPLE OF
      10 TIMES THOSE STEP SIZE VALUES.

```

```

101 FORMAT(1H1,10X,46HTHE OUTPUT INTERVAL IS TOO SMALL - OUTPUT ARRA,
    *30HY EXCEEDS 800 PTIME RESET TO ,F7.2//)
102 FORMAT(//,10X,29HTHE ISE OF THE MAJOR LOOP IS ,1PE14.7/10X,
    *29HTHE IAE OF THE MAJOR LOOP IS ,1PE14.7)
103 FORMAT(20(1X,I1))
104 FORMAT(4(5X,2HY(,I2,4H) = ,F12.5,2X))
105 FORMAT(4(5X,2HX(,I2,4H) = ,F12.5,2X))
150 FORMAT(8(F9.2,1H ))
160 FORMAT(8(I2,1X))
161 FORMAT(I2,1X,I2,1X,I3)
162 FORMAT(2X,36HCASCADE 1ST ORDER DIG. CONTROLLERS. )
163 FORMAT(5(F12.6,2X))

```

C
C

```

      IRUN = 0
1  IPAGE = 0
      IRUN = IRUN + 1
      READ(1,103)(ND(I),I=1,20)
      IF(EOF(1))999,2
2  IBUG = ND(7)
      JSTOP = 0

```

```

      NSUM = 0
      DO 3 I=1,20
3     NSUM = NSUM + ND(I)
      IF(NSUM.EQ.0) JSTOP = 1
      IF(JSTOP.EQ.1.AND.IADJ.EQ.0) GO TO 999
      IADJ = ND(8)

C
C     ABOVE READS IN NDATA AND TERMINATES RUN IF NO CHANGES IN ND
C
      CALL INPUT
      CALL INIFUN
      IF(NX.EQ.0) GO TO 4
      CALL INTCAL
4     CALL DIFFEQ
      CALL ALIST

C
C     ABOVE INITIALIZES PROGRAM AND PRINTS OUT HEADER INFO.
C
      NSTEP = ((ETIME - BTIME)/DELT) + 0.5
      NW = ((PTIME/DELT) + 0.5)
      NMOG = NSTEP/NW
      IF(NMOG.GT.800) GO TO 6
      GO TO 7
6     NPT = NSTEP/800
      NW = NPT + (1./DELT)
      PTIME = NW*DELT
      WRITE(2,101)PTIME

C
C     ABOVE ESTABLISHES PRINT INTERVAL
C
      7  JI=1
         DO 8 I=1,MX
8        XS(JI,I) = X(JX(I))
         DO 9 I=1,MY
9        YS(JI,I) = Y(JY(I))

C
C     STORE IN PRINT MATRIX INITIAL X AND Y PRINT/PLOT VARIABLES.
C
C     PROCEED WITH PROGRAM EXECUTION BELOW.

```

C

```

      NI = 0
      DO 15 KK=1,NSTEP
      NI = NI + 1
      CALL RKINT(KK)
      IF(IBUG.NE.1) GO TO 10
      N1 = NY + 1
      WRITE(2,104)N1,Y(N1)
      WRITE(2,104)(I,Y(I),I=1,NY)
      WRITE(2,105)(I,X(I),I=1,NX)
10  IF(NX.EQ.0) GO TO 11
      CALL INTCAL
11  IF(NI.NE.NW) GO TO 15
      NI = 0
      IF(IBUG.NE.1) GO TO 12
      WRITE(2,104)N1,Y(N1)
      WRITE(2,104)(I,Y(I),I=1,NY)
      WRITE(2,105)(I,X(I),I=1,NX)
12  JI = JI + 1

```

C
C
C

STORE X AND Y DATA IN PRINT/PLOT MATRICES BELOW.

```

      IF(MX.EQ.0) GO TO 50
      DO 13 I=1,MX
      XS(JI,I) = X(JX(I))
      IF(IGX.EQ.0) GO TO 13
      IF(XS(JI,I).GE.XG(I,2)) XG(I,2) = XS(JI,I)
      IF(XS(JI,I).LE.XG(I,1)) XG(I,1) = XS(JI,I)
13  CONTINUE

```

C

```

50  IF(MY.EQ.0) GO TO 15
      DO 14 I=1,MY
      YS(JI,I) = Y(JY(I))
      IF(IGY.EQ.0) GO TO 14
      IF(YS(JI,I).GE.YG(I,2)) YG(I,2) = YS(JI,I)
      IF(YS(JI,I).LE.YG(I,1)) YG(I,1) = YS(JI,I)
14  CONTINUE
15  CONTINUE

```

C

```

C   NOW THAT ITERATIONS COMPLETE PRINT AND PLOT RESULTS.
C
    IF(ND(18).NE.1) GO TO 201
    WRITE(6,162)
    WRITE(6,161)MX,NP,JI
    WRITE(6,160)(JX(I),I=1,MX)
    WRITE(6,163)(P(I),I=1,NP)
    DO 200 I=1,JI
200  WRITE(6,150)(XS(I,JJK),JJK=1,MX)
    ENDFILE 6
201  CONTINUE
    NOUT = JI
    IF(ITY.EQ.0) GO TO 16

C   PRINT Y TABLES.
C
C   CALL PRTTBL(JI,0)
16  IF(ITX.EQ.0) GO TO 17

C   PRINT X TABLES.
C
C   CALL PRTTBL(JI,1)
17  IF(IGY.EQ.0) GO TO 18

C   PLOT Y DATA.
C
C   NG = IGY
    CALL PLOT(YS,1,YG)
    IPAGE = IPAGE + 1
18  IF(IGX.EQ.0) GO TO 19

C   PLOT X DATA.
C
C   NG = IGX
    CALL PLOT(XS,0,XG)
    IPAGE = IPAGE + 1

C   ADJUST THE PARAMETER (P'S) LIST IF IADJ = 1
C
C

```

```

19 IF(IADJ.NE.0) CALL PARADJ
   WRITE(2,102)Y(3),Y(4)
   IF(JSTOP.NE.0) GO TO 999
   GO TO 1
999 STOP
   END
   SUBROUTINE INPUT

```

C
C

```

COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/BB/XSTR(2,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

```

C
C
C
C

```

100 FORMAT(8F10.0)
101 FORMAT(10A8)
103 FORMAT(4(I2,1X))
104 FORMAT(8(1X,I2))
105 FORMAT(3(3A8,1X))

```

C
C
C

ZERO OUT ARRAYS

```

DO 1 I=1,8
DO 1 J=1,2
XG(I,J) = 0.0
1 YG(I,J) = 0.0
IF(ND(5).EQ.0) GO TO 3
DO 2 I=1,8
NGY(I) = 0
NGX(I) = 0
NTY(I) = 0
NTX(I) = 0

```

```

      JX(I) = 0
2     JY(I) = 0
      ITY=ITX=IGY=IGX=MY=MX=0
3     IF(ND(1).EQ.0) GO TO 4

C
C     READ IN TWO 80 CHARACTER TITLES, MOLD TITLE(MT), MOLD NUMBER(MN),
C     AND MOLD SECTION(MS). THEN READ DATE(JD), NUMBER OF Y'S, X'S,
C     P'S, AND H'S AS WELL AS BEGIN AND END TIMES, INTEGRATION SIZE,
C     AND PRINT INTERVAL.
C
      READ(1,101)(IT1(I),I=1,10)
      READ(1,101)(IT2(J),J=1,10)
      READ(1,105)(MT(K),K=1,3),(MN(L),L=1,3),(MS(M),M=1,3)
      READ(1,105)(JD(K),K=1,3)
      READ(1,103)NY,NX,NP,NH
      READ(1,100)BTIME,ETIME,DELT,PTIME
      Y(NY+1) = BTIME
      Y(NY+2) = DELT
      N2 = NY + 2
      NMAX = 2*NY + 2
4     IF(ND(2).EQ.0) GO TO 5
      READ(1,100)(Y(I),I=1,NY)
      GO TO 7
5     IF(ND(6).NE.0) GO TO 7
      DO 6 I=1,N2
6     Y(I) = YT(I)
7     DO 8 I=1,N2
8     YT(I) = Y(I)

C
C     INTEGRATION CONTROLS READ IF ND(1) = 1 ABOVE. ALSO, THE INITIAL
C     CONDITIONS OF THE STATE VARIABLES ARE SET.
C
      IF(ND(3).EQ.0) GO TO 9
      READ(1,100)(H(I),I=1,NH)
9     IF(ND(4).EQ.0) GO TO 10
      READ(1,100)(P(I),I=1,NP)

C
C     THE ABOVE READS THE TIME DELAY AND PARAMETER MAGNITUDE ARRAYS.

```

```

C
C   BELOW READ IN Y AND X INDEX NUMBERS FOR PRINT AND PLOT I/O.
C
10 IF (ND(5).EQ.0) GO TO 18
   READ(1,104) (NTY(I),I=1,8)
   READ(1,104) (NGY(I),I=1,8)
   READ(1,104) (NTX(I),I=1,8)
   READ(1,104) (NGX(I),I=1,8)
C
C   DETERMINE HOW MANY OF EACH FUNCTION ARE TO BE PERFORMED.
C
   DO 11 J=1,8
   IF (NTX(J).NE.0) ITX = ITX + 1
   IF (NTY(J).NE.0) ITY = ITY + 1
   IF (NGX(J).NE.0) IGX = IGX + 1
   IF (NGY(J).NE.0) IGY = IGY + 1
11 CONTINUE
C
C   STORE PRINT X INDICES IN ARRAY JX , Y INDICES IN JY.BELOW
C
   DO 13 I=1,8
   IF (NTX(I).EQ.0) GO TO 12
   MX = MX + 1
   JX(MX) = NTX(I)
12 IF (NTY(I).EQ.0) GO TO 13
   MY = MY + 1
   JY(MY) = NTY(I)
13 CONTINUE
C
C   CHECK TO SEE IF PLOT X AND Y INDICES ARE THE SAME AS PRINT INDICES
C   IF SO DO NOT STORE INDEX. IF NOT SAME STORE INDEX IN JX OR JY.
C
   DO 17 I=1,8
   IF (NGX(I).EQ.0) GO TO 15
   DO 14 J=1,MX
   IF (NGX(I).EQ.JX(J)) GO TO 15
14 CONTINUE
   MX = MX + 1
   IF (MX.GT.8) MX = 8

```



```

      JX(MX) = NGX(I)
15  IF(NGY(I).EQ.0) GO TO 17
      DO 16 J=1,MY
      IF(NGY(I).EQ.JY(J)) GO TO 17
16  CONTINUE
      MY = MY + 1
      IF(MY.GT.8) MY = 8
      JY(MY) = NGY(I)
17  CONTINUE

```

C
C
C

```

18  IF(ND(17).NE.0) CALL RECIPE

```

C

```

      RETURN
      END
      SUBROUTINE RECIPE

```

C

C

C

```

      READS IN THE MAJOR LOOP SET POINT RECIPE.

```

C

```

      COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1

```

C

```

      1 FORMAT(3F10.0)
      2 FORMAT(I2)

```

C

```

      READ(1,2)NSEG

```

C

C

C

```

      DO 3 I=1,NSEG
3  READ(1,1)RAMP(I),STPT(I),TIM(I)

```

C

```

      JFLAG = 0
      RETURN
      END
      SUBROUTINE ALIST

```

C

C

C

```

      HEADER OUTPUT ROUTINE.

```

```

      COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
      COMMON/BB/XSTR(2,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)

```

```

COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,ITI(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

```

C
C
C

```

100 FORMAT(1H1,6X,48HCASCADE SIMULATION--DESIGNED DIGITAL CONTROLLERS,
113H ** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9HPAGE NO. ,I3/)
101 FORMAT(1X,120(1H*))
102 FORMAT(65X,10HDESCRIPTOR/)
103 FORMAT(20X,10A8/20X,10A8//)
104 FORMAT(5X,19H* PROGRAM CONTROLS,24X,1H*,16X,17H* MOLD IDENTIFIE,
*2HRS,33X,1H*)
105 FORMAT(5X,1H*,42X,1H*,16X,1H*,51X,1H*)
106 FORMAT(5X,24H* NUMBER OF Y EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
*10HMOLD TITLE,13X,3A8,2X,1H*)
107 FORMAT(//5X,44(1H*),16X,53(1H*))
108 FORMAT(5X,24H* NUMBER OF X EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
*11HMOLD NUMBER,12X,3A8,2X,1H*)
109 FORMAT(5X,28H* NUMBER OF P PARAMETERS ,I3,12X,1H*,16X,1H*,2X,
*12HMOLD SECTION,11X,3A8,2X,1H*)
110 FORMAT(5X,21H* NUMBER OF H DELAYS,7X,I3,12X,1H*,16X,1H*,2X,
*15HSIMULATION DATE,8X,3A8,2X,1H*)
111 FORMAT(5X,13H* START TIME,12X,F6.2,3X,10HSECONDS *,
116X,1H*,51X,1H*)
112 FORMAT(5X,11H* END TIME,11X,F9.2,3X,10HSECONDS *,
116X,1H*,51X,1H*)
113 FORMAT(5X,18H* OUTPUT INTERVAL,8X,F5.2,3X,10HSECONDS *,16X,
*53(1H*))
114 FORMAT(5X,25H* INTEGRATOR STEP SIZE ,F6.3,3X,10HSECONDS */5X,
*44(1H*))
115 FORMAT(//5X,44HTHE FOLLOWING ARE THE TIME DELAYS IN SECONDS/)
116 FORMAT(4(5X,2HH(,I2,4H) = ,F12.3,2X))
117 FORMAT(//5X,38HTHE FOLLOWING IS THE PARAMETER LISTING/)
118 FORMAT(4(5X,2HP(,I2,4H) = ,F12.6,2X))
119 FORMAT(//5X,48HENSUING ARE THE INITIAL CONDITIONS OF THE STATE ,
*9HVARIABLES/)
120 FORMAT(4(5X,2HY(,I2,4H) = ,F12.5,2X))

```

```

121 FORMAT(//5X,49HENSUING ARE THE X VALUES AFTER INIFUN AND INTCAL ,
*33HBUT BEFORE THE FIRST INTEGRATION./)
122 FORMAT(4(5X,2HX(,I2,4H) = ,F12.4,2X))
123 FORMAT(50X,20HCONTROLLER CONSTANTS//)
124 FORMAT(2X,2(2HD(,I2,4H) = ,1PE13.6,5X),0P)
125 FORMAT(2X,5(2HF(,I2,4H) = ,1PE13.6,5X),0P)
126 FORMAT(5X,25H* METAL TIME CONSTANT ,F6.1,3X,10HSECONDS */
15X,25H* METAL PROCESS GAIN ,F6.4,3X,10HDEG/DEG *)
127 FORMAT(5X,25H* FLUID TIME CONSTANT ,F6.1,3X,10HSECONDS */
15X,25H* FLUID PROCESS GAIN ,F6.4,3X,10HDEG/DEG *)
128 FORMAT(5X,48H* CLOSED LOOP TIME CONSTANTS * -- MAJOR LOOP = ,F6.1
*,3X,13HMINOR LOOP = ,F6.1,8H SECONDS)
129 FORMAT(5(2X,2HC(,I1,4H) = ,1PE13.6,4X),0P)
130 FORMAT(5X,39H* CLOSED LOOP GAINS * -- MAJOR LOOP = ,F10.6,3X,
*13HMINOR LOOP = ,F10.6)
131 FORMAT(39X,42H* THE METAL LOOP SET POINT RECIPE IS *)
132 FORMAT(39X,42(1H*))
133 FORMAT(39X,42H* RAMP * SETPOINT * DWELL * SEGMENT *)
134 FORMAT(39X,42H* F/MIN * DEGREE F * MIN * NO. *)
135 FORMAT(39X,2H* ,F5.1,2H *,3X,F5.1,2X,3H* ,F5.2,3H *,4X,12,5X,
*1H*)
136 FORMAT(///)
137 FORMAT(5X,20H* MOLD DEAD TIME = ,5X,F6.1,3X,10HSECONDS *,3X,
*18HFLUID DEAD TIME = ,F6.1,8H SECONDS)
138 FORMAT(60X,10HMAJOR LOOP/)
139 FORMAT(60X,10HMINOR LOOP/)
140 FORMAT(5(2X,2HH(,I2,4H) = ,1PE13.6,3X),0P)
141 FORMAT(5(2X,2HP(,I2,4H) = ,1PE13.6,3X),0P)

```

C

```

IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,102)
WRITE(2,103)(IT1(I),I=1,10),(IT2(J),J=1,10)
WRITE(2,101)
WRITE(2,107)
WRITE(2,104)
WRITE(2,105)
WRITE(2,106)NY,(MT(J),J=1,3)

```

```
WRITE(2,108)NX,(MN(J),J=1,3)
WRITE(2,109)NP,(MS(J),J=1,3)
WRITE(2,110)NH,(JD(J),J=1,3)
WRITE(2,111)BTIME
WRITE(2,112)ETIME
WRITE(2,113)PTIME
WRITE(2,114)DELT
WRITE(2,101)
WRITE(2,115)
WRITE(2,101)
WRITE(2,116)(I,H(I),I=1,NH)
WRITE(2,101)
WRITE(2,117)
WRITE(2,101)
WRITE(2,118)(I,P(I),I=1,NP)
WRITE(2,101)
IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,119)
WRITE(2,101)
WRITE(2,120)(I,Y(I),I=1,NY)
WRITE(2,101)
WRITE(2,121)
WRITE(2,101)
WRITE(2,122)(I,X(I),I=1,NX)
WRITE(2,101)
WRITE(2,101)
IPAGE = IPAGE + 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,123)
WRITE(2,138)
WRITE(2,140)(I,P(I),I=1,3)
WRITE(2,141)(I,P(I+5),I=1,5)
WRITE(2,139)
WRITE(2,124)(I,P(I+10),I=1,2)
WRITE(2,125)(I,P(I+14),I=1,5)
WRITE(2,136)
WRITE(2,129)(I,C(I),I=1,8)
```

```

WRITE (2,136)
WRITE (2,127) P(24), P(23)
WRITE (2,126) P(26), P(25)
WRITE (2,128) P(31), P(30)
WRITE (2,130) P(5), P(14)
WRITE (2,137) H(2), H(1)
WRITE (2,136)
WRITE (2,101)
WRITE (2,136)
WRITE (2,132)
WRITE (2,131)
WRITE (2,132)
WRITE (2,133)
WRITE (2,134)
WRITE (2,132)
WRITE (2,135) (RAMP(I), STPT(I), TIM(I), I, I=1, NSEG)
WRITE (2,132)
IPAGE = IPAGE + 1
RETURN
END
SUBROUTINE INIFUN

```

```

C
C
C   INITILIZE FUNCTION SUBROUTINE.

```

```

COMMON/AA/X(80), P(80), Y(22), YT(22), F(10), C(9), NY, NX, NP, MMSP
COMMON/BB/XSTR(2,500), XT(9), XTH(9), IH(9), NH, IMAX, H(9)
COMMON/CC/RAMP(14), STPT(14), TIM(14), DWELL(14), JFLAG, NSEG, SP, SP1
COMMON/GG/ND(20), NPT, NSP, NTM, PTIME
COMMON/HH/BTIME, ETIME, DELT, IT1(10), IT2(10), MT(3), MN(3), JD(3), MS(3)
COMMON/II/ISAMP, NTIME, NDP, IDP, CDI(2,20), CDO(2), ND2, ID2, ND1, ID1

```

```

C
C   DIMENSION FT(10), HT(10), D(9)

```

```

C
C   100 FORMAT(1H1,5X,23H** TIME DELAY VARIABLE ,I1,18H EXCEEDS THE MAXIM,
*46HUM ARRAY SIZE WHEN DIVIDED BY DELT RGM STOP **/)

```

```

C
C   IMAX = 0
DO 1 I=1, NH

```

```

      IH(I) = H(I)/DELT + 1
      IF(IH(I).GE.IMAX) IMAX = IH(I)
      IF(IMAX.GT.500) GO TO 2
1    CONTINUE
      GO TO 3
2    WRITE(2,100)I
      STOP
3    DO 4 I=1,NX
4    X(I) = 0.0
      AD1 = (H(1)/P(29) + 0.05)
      ND1 = AD1
      AD2 = (H(2)/P(29) + 0.05)
      ND2 = AD2
      ID1 = ND1 + 1
      ID2 = ND2 + 1
      IDP = ID1 + ID2
      NDP = IDP - 1
      JMAX = ID1
      IF(IDP.GT.JMAX) JMAX = IDP
      DO 5 I= 1,2
      DO 5 J=1,JMAX
5    CDI(I,J) = 0.0
      P(25) = P(34)
      P(26) = P(32)
      X(64) = Y(2)
      X(63) = Y(2)
      X(62) = Y(2)/P(25)
      X(61) = X(62)
      Y(1) = X(61)
      X(19) = X(62)*P(27) + P(28)
      X(60) = X(62)
      X(59) = X(62)
      DO 6 I = 1,IDP
6    CDI(2,I) = X(19)

```

C

```

      X(1) = BTIME
      SAM = P(29)/DELT + 0.05
      ISAMP = SAM
      NTIME = 0

```

C

$DF1 = H(1) - ND1 * P(29)$
 $DF2 = H(2) - ND2 * P(29)$
 $AM1 = 1. - (DF1 / P(29))$
 $AM2 = 1. - (DF2 / P(29))$
 $SM = AM1 + AM2$
 $DTU = P(26) - P(30)$
 $T = P(29)$
 $X1 = EXP(-T / P(24))$
 $X2 = EXP(-T / P(30))$
 $X3 = EXP(-T / P(26))$
 $X4 = EXP(-T / P(31))$
 $Y1 = EXP(-T * AM1 / P(24))$
 $Y2 = EXP(-T * AM1 / P(30))$
 $Y3 = EXP(-T * SM / P(26))$

$Y3P = EXP(-T * SM / P(30))$
 $Y4 = EXP(-T * SM / P(31))$
 $C(1) = 1. - Y1$
 $C(2) = Y1 - X1$
 $C(3) = 1. - Y2$
 $C(4) = Y2 - X2$
 $C(5) = Y3 - Y3P$
 $C(6) = Y3P * X3 - Y3 * X2$
 $C(7) = 1. - Y4$
 $C(8) = Y4 - X4$

C

$R1 = C(2) / C(1)$
 $R2 = C(4) / C(3)$
 $R3 = C(6) / C(5)$
 $R4 = C(8) / C(7)$

C

$D(1) = X1 - R2$
 $D(2) = R2 * X1$

C

$FT(1) = X2 - R1$
 $FT(2) = X2 * R1$
 $FT(3) = C(3)$
 $FT(4) = C(4) + R1 * C(3)$
 $FT(5) = C(4) * R1$

C

HT(1) = X2 + X3 - R4
 HT(2) = X2*X3 - R4*(X2+X3)
 HT(3) = R4*X2*X3

C

P(6) = X4 - R3
 P(7) = X4*R3
 P(8) = C(7)
 P(9) = C(8) + C(7)*R3
 P(10) = C(8)*R3

C

P(5) = C(7)*DTU/(P(25)*C(5))
 P(14) = C(3)/(0.16*C(1))
 DO 7 I=1,3
 7 P(I) = HT(I)
 P(11) = D(1)
 P(12) = D(2)
 DO 8 I=1,5
 K=I+14
 8 P(K) = FT(I)
 DO 9 I=20,24
 9 X(I) = X(19)
 DO 10 I=6,10
 10 X(I+19) = X(I+14)*P(I)
 DO 11 I=1,IMAX
 XSTR(1,I) = Y(1)
 11 XSTR(2,I) = Y(2)
 DO 12 I=1,NH
 12 XTH(I) = XSTR(I,IMAX)

C

RETURN
 END
 SUBROUTINE INTCAL

C

C

C

THIS IS THE INTERMEDIATE CALCULATION SUBROUTINE.

COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
 COMMON/BB/XSTR(2,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
 COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1


```

COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON/II/ISAMP,NTIME,NDP,IDP,CDI(2,20),CDO(2),ND2,ID2,ND1,ID1

```

C
C
C

```

X(1) = Y(NY+1)
IF(X(1).LE.BTIME) GO TO 1
IF(NTIME.EQ.ISAMP) CALL CMAJOR
NTIME = NTIME + 1
1 XT(1) = Y(1)
  XT(2) = Y(2)

```

C

CALL DELAY

C

```

X(61) = Y(1)
X(62) = XTH(1)
X(62) = ALIM(X(62),P(21),P(22))
X(63) = Y(2)
X(64) = XTH(2)

```

C

```

RETURN
END
FUNCTION ALIM(V,B,T)
ALIM = V
IF(V.GT.T) ALIM = T
IF(V.LT.B) ALIM = B
RETURN
END
SUBROUTINE CMAJOR

```

C

C

C

C

C

THIS CONTROLLER IS A SPECIALLY DESIGNED POSITIONAL CONTROLLER
FOR THE MAJOR LOOP. DESIGNED TO PERFORM AS A 1 ST ORDER LAG.

```

COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/II/ISAMP,NTIME,NDP,IDP,CDI(2,20),CDO(2),ND2,ID2,ND1,ID1

```

```

C      CALL SPADJ
C
X(24) = X(23)
X(23) = X(22)
X(22) = CD0(2)
X(21) = X(20)
X(20) = X(19)
X(7) = X(6)
X(6) = X(5)
X(5) = X(4)
C
X(65) = X(64)*P(27) + P(28)
X(4) = X(3) - X(65)
DO 1 I=1,3
1 X(I+7) = X(I+4)*P(I)
DO 2 I=6,10
2 X(I+19) = X(I+14)*P(I)
C
X(11) = X(4) - X(8)
X(12) = X(11) + X(9)
X(13) = X(12) + X(10)
C
X(14) = X(13)*P(4)*P(5)
C
DO 3 I=15,19
K=I+10
3 X(I) = X(I-1) + X(K)
X(67) = X(19)*P(20) + P(36)
C
CALL DLY(2,NDP,IDP)
C
CDI(2,1) = X(19)
C
CALL CMINOR
C
NTIME = 0
RETURN
END

```

```

SUBROUTINE CMINOR
C
C MINOR LOOP VELOCITY CONTROLLER.  SPECIALLY DESIGNED.
C
COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFALG,NSEG,SP,SP1
COMMON/GG/ND(20),NPT,NSP,NTM,PTIME
COMMON/II/ISAMP,NTIME,NDP,IOP,CDI(2,20),CDO(2),ND2,ID2,ND1,ID1
C
X(66) = X(62)*P(27) + P(28)
C
X(34) = X(33)
X(33) = X(32)
X(31) = X(30)
X(49) = X(48)
X(48) = X(47)
X(47) = CDO(1)
X(46) = X(45)
X(45) = X(44)
C
X(30) = X(19) - X(66)
X(32) = X(30) - X(31)
X(35) = X(33)*P(11)
X(36) = X(34)*P(12)
X(37) = X(32) - X(35)
X(38) = X(37) - X(36)
X(39) = X(38)*P(14)*P(13)
C
DO 1 I=15,19
  J = I + 35
  K=I+30
1 X(J) = X(K)*P(I)
C
DO 2 I=40,44
2 X(I) = X(I-1) + X(I+10)
C
IF(ND1.EQ.0) GO TO 3
C
CALL DLY(1,ND1,ID1)

```

```

C
3 CDI(1,1) = X(44)
  IF(ND1.EQ.0) CDO(1) = X(44)
C
  X(55) = ALIM(X(44),-50.,50.)
  X(56) = X(55)*P(20)
  X(57) = 0.16*X(56)
  X(60) = X(59)
  X(58) = X(60) + X(57)
  X(59) = ALIM(X(58),P(21),P(22))
C
  RETURN
  END
  SUBROUTINE DLY(L,ND,ID)
C
  COMMON/II/ISAMP,NTIME,NDP,IDP,CDI(2,20),CDO(2),ND2,ID2,ND1,ID1
C
  DO 1 I= 1,ND
    K = ID + 1 - I
    J = ID - I
1  CDI(L,K) = CDI(L,J)
    CDO(L) = CDI(L,ID)
C
  RETURN
  END
  SUBROUTINE SPADJ
C
  COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
  COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
C
  THIS IS THE SET POINT ADJUSTMENT ROUTINE. THE RECIPE IS READ IN
  FROM SUBROUTINE RECIPE. SOME OF THE FEATURES OF THE ROUTINE ARE%
C
C ... 1) NO RAMP MAY BE SPECIFIED IN SEGMENT 1.
C ... 2) DWELL PERIODS ARE IN MINUTES AND APPLY TO THE SPECIFIED
C      SEGMENT ONLY.
C ... 3) THE SET POINT IN ANY SEGMENT EXCEPT 1 MAY BE SPECIFIED AS ZERO
C      AND THAT SEGMENT WILL USE THE PRECEDING SEGMENTS SET POINT.

```

C ... 4) TWO CONSECUTIVE SEGMENTS MUST HAVE AT LEAST ONE SET POINT
 C SPECIFIED.
 C ... 5) THE DWELL IN ANY ONE SEGMENT MAY BE SPECIFIED AS ZERO AND THAT
 C SEGMENT WILL BE PASSED OVER. THIS OPTION IS USUALLY USED TO
 C SPECIFY A DIFFERENT SET POINT AS A START POINT FOR A RAMP.
 C ... 6) A DWELL MUST BE SPECIFIED IN A RAMP SEGMENT.
 C ... 7) NO TWO CONSECUTIVE SEGMENTS MAY BE SPECIFIED WITH A ZERO DWELL
 C ... 8) POSITIVE RAMPS ARE DEFINED BY A PLUS OR NO SIGN BEFORE THE
 C RAMP RATE MAGNITUDE.
 C ... 9) COOL DOWN RAMPS ARE DEFINED BY A NEGATIVE SIGN BEFORE THE RAMP
 C RATE MAGNITUDE
 C ... 10) A RAMP SEGMENT MUST HAVE SPECIFIED A RAMP RATE, A TERMINATION
 C SET POINT, AND A TERMINATION TIME LIMIT. THAT TIME LIMIT MAY
 C BE SPECIFIED GREATER THAN THE ACTUAL RAMP TIME.
 C

```

IF(JFLAG.EQ.1) GO TO 2
ST = 0.0
DWELL(1) = TIM(1)*60.
JFLAG = 1
DO 1 I=2,NSEG
K=I-1
1 DWELL(I) = DWELL(K) + TIM(I)*60.
2 IF(X(1).GE.DWELL(1)) GO TO 3
SP = STPT(1)
I=1
GO TO 14
3 DO 4 I=2,NSEG
IF(X(1).GE.DWELL(I-1).AND.X(1).LT.DWELL(I)) GO TO 5
4 CONTINUE
RETURN
5 IRAMP = RAMP(I)
IF(IRAMP.NE.0) GO TO 6
GO TO 7
6 BT = DWELL(I-1)
SLP = RAMP(I)/60.
SP = SLP*(X(1)-BT) + STPT(I-1)
IF(STPT(I-1).LT.50.) SP = SLP*(X(1)-BT) + STPT(I-2)
IF(ABS(SP-STPT(I)).LT.0.5.AND.X(1).LT.DWELL(I)) GO TO 12
IF(RAMP(I).LT.0.0) GO TO 11
  
```

```

      GO TO 10
7  SP = STPT(I)
   IF (SP.GE.50.0) GO TO 9
   SP = STPT(I-1)
   IF (I.GT.2) GO TO 8
   GO TO 14
8  CH = SP - STPT(I-2)
   IF (CH.LT.0.0) GO TO 11
   GO TO 10
9  CH = SP - STPT(I-1)
   IF (CH.LT.0.0) GO TO 11
10 P(25) = P(34)
   P(26) = P(32)
   P3 = 0.0
   GO TO 14
11 P(25) = P(35)
   P(26) = P(33)
   P3 = 0.0
   GO TO 14
12 ADT = DWELL(I) - X(1)
   SP = STPT(I+1)
   IF (SP.LT.50.) SP=STPT(I)
   DWELL(I) = DWELL(I) - ADT
   IF (SP.EQ.STPT(I-1)) CH = SP-STPT(I-2)
   IF (CH.LT.0.0) GO TO 11
   CH = SP - STPT(I-1)
   I=I+1
   IF (I.GT.NSEG) GO TO 14
   DO 13 K=I,NSEG
13 DWELL(K) = DWELL(K-1) + TIM(K)*60.
   GO TO 10
14 X(2) = SP
   X(3) = SP*P(27) + P(28)

```

C

```

MMSP = I
RETURN
END

```

```

SUBROUTINE DELAY
C
C THIS SUBROUTINE SETS UP THE DELAY PIPELINES.
C
COMMON/BB/XSTR(2,500),XT(9),XTH(9),IH(9),NH,IMAX,H(9)
C
  ISTOP = IMAX - 1
  DO 1 KK=1,ISTOP
    IS = IMAX - KK
    JS = IS + 1
    DO 1 II = 1,NH
1 XSTR(II,JS) = XSTR(II,IS)
C
    DO 2 II = 1,NH
      JJ = IH(II)
      XTH(II) = XSTR(II,JJ)
2 XSTR(II,1) = XT(II)
C
  RETURN
END
SUBROUTINE DIFFEQ
C
C DIFFERENTIAL EQUATION ROUTINE.
C
COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
C
  F(1) = FOL(P(23),X(59),Y(1),P(24))
  F(2) = FOL(P(25),X(62),Y(2),P(26))
  F(3) = X(4)*X(4)
  F(4) = ABS(X(4))
C
C
C
  DO 1 LK = 1,NY
    IQ = NY + 2 + LK
1 Y(IQ) = F(LK)
C
  RETURN
END

```

```

FUNCTION FOL(GA,XIN,YOUT,TAU)
  IF(TAU.NE.0.0) GO TO 1
  YOUT = GA*XIN
  FOL = 0.0
  RETURN
1 FOL = (GA*XIN - YOUT)/TAU
  RETURN
END
SUBROUTINE RKINT(LL)
C
C  RUNGE-KUTTA NUMERICAL INTEGRATION ROUTINE FOR SOLVING DIFFERENTIAL
C  EQUATIONS.
C
COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
C
DIMENSION YU(42),RET(3),DY(4,20)
DOUBLE PRECISION YU
C
  IF(LL.NE.1) GO TO 1
  BET(1) = 0.5
  BET(2) = 0.5
  BET(3) = 1.0
  N2 = NY + 2
  N1 = NY + 1
  XV = Y(N1)
  CALL DIFFEQ
1 DO 2 I=1,NY
2 YU(I) = Y(I)
  DO 6 K=1,4
  IF(K.EQ.1) GO TO 3
  CALL DIFFEQ
3 DO 4 I=1,NY
  IJ = I + N2
4 DY(K,I) = Y(N2)*Y(IJ)
  IF(K.EQ.4) GO TO 6
  DO 5 I=1,NY
5 Y(I) = YU(I) + BET(K)*DY(K,I)
  Y(N1) = XV + BET(K)*Y(N2)
6 CONTINUE

```



```

DO 7 I=1,NY
DEL = (DY(1,I)+2.*DY(2,I)+2.*DY(3,I)+DY(4,I))/6.
YU(I) = YU(I) + DEL
Y(I) = YU(I)
7 CONTINUE
Y(N1) = XV + Y(N2)
CALL DIFFEQ
XV = Y(N1)
RETURN
END
SUBROUTINE PRTTBL(MAXLIN,IV)

```

C
C
C

THIS ROUTINE PRINTS THE SPECIFIED TABLES.

```

COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IRAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NOX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/HH/BTIME,ETDME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
COMMON ARY(800,8),NPRT(8)

```

C
C

```

100 FORMAT(1H1,6X,48HCASCADE SIMULATION--DESIGNED DIGITAL CONTROLLERS,
113H ** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9MPAGE NO. ,I3/)
101 FORMAT(8(7X,A1,A1,I2,A1,2X))
102 FORMAT(8(2X,F12.4))
103 FORMAT(1X,120(1H*))//

```

C
C

```

IF(IV.EQ.0)GO TO 3
IB = IX
NCOL = ITX
DO 1 I=1,MAXLIN
DO 1 J=1,NCOL
1 ARY(I,J) = XS(I,J)
DO 2 I=1,NCOL
2 NPRT(I) = NTX(I)
GO TO 6
3 IB = IY
NCOL = ITY

```

```

DO 4 I=1,MAXLIN
DO 4 J=1,NCOL
4 ARY(I,J) = YS(I,J)
DO 5 I=1,NCOL
5 NPRT(I) = NTY(I)

```

C
C
C

STORE X OR Y DATA IN ARY MATRIX ABOVE.

```

6 LINE = 1
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,103)
WRITE(2,101)(IB,PL,NPRT(I),PR,I=1,NCOL)

```

C

```

DO 7 KK = 1,MAXLIN
WRITE(2,102)(ARY(KK,I),I=1,NCOL)
LINE = LINE + 1
IF(LINE.LT.55) GO TO 7
LINE = 1
IPAGE = IPAGE + 1
WRITE(2,103)
WRITE(2,100)(JD(I),I=1,3),IRUN,IPAGE
WRITE(2,103)
WRITE(2,101)(IB,PL,NPRT(I),PR,I=1,NCOL)

```

```

7 CONTINUE

```

C
C
C

ABOVE PRINTS THE DATA IN MATRIX ARY IN TABULAR FORM.

```

WRITE(2,103)
RETURN
END
SUBROUTINE PLOT(SAVE,NSS,GR)

```

C
C
C

THIS ROUTINE PLOTS THE DATA STORED IN THE SAVE MATRIX.

```

REAL LINE(101),DEL(8)
DIMENSION SAVE(800,8),IN(8)
DIMENSION GR(8,2),STEPS(10),NL(8)

```

C

```

C      COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)

```

```

C      DATA PLUS,BLANK/1H+,1H /

```

```

C
C
C      100 FORMAT(1H1,6X,4BHCASCADE SIMULATION--DESIGNED DIGITAL CONTROLLERS,
113H ** RUN DATE ,3A8,2X,8HRUN NO. ,I2,2X,9HPAGE NO. ,I3/)
101 FORMAT(1H+,50X,9HSYMBOL = ,A1,26H,      FOR COINCIDENT POINTS)
102 FORMAT(1H )
103 FORMAT(1H ,20X,A1,A1,I2,A1,2H =,F11.2,12X,9HSYMBOL = ,A1,1H,,4X,
*15HSCALE FACTOR = ,F11.3,12X,A1,A1,I2,A1,2H =,F11.2)
104 FORMAT(1H ,12X,15HTIME IN SECONDS)
105 FORMAT(///31X,20H** OUTPUT WARNING **,I5,2X,17HPPOINTS WERE NOT W,
*22HITHIN THE GRAPH RANGE.)
106 FORMAT(1H ,20X,101A1)
107 FORMAT(1H+,7X,F11.2)

```

```

C      IN(1) IS RESERVED FOR TIME.

```

```

C
C
C
C
C      IPAGE = IPAGE + 1
WRITE(2,100)(JD(1),I=1,3),IRUN,IPAGE
WRITE(2,102)

```

```

C
      MSS = NSS
      NN = 2
      IF(MSS.EQ.1) GO TO 2
      DO 1 I=1,IGX
      DO 1 J=1,MX
      IF(NGX(I).EQ.JX(J)) IN(I) = J
1 CONTINUE
      GO TO 4
2 DO 3 I=1,IGY
      DO 3 J=1,MY

```

```
IF(NGY(I).EQ.JY(J)) IN(I) = J
```

```
3 CONTINUE
```

C

```
4 CALL SCALE(NN,GR,IN)
```

C

```
DO 6 I=2,NG
```

```
IF(GR(IN(I),1).NE.0.0.OR.GR(IN(I),2).NE.0.)GO TO 5
```

```
GR(IN(I),1) = -0.1
```

```
GR(IN(I),2) = 0.1
```

```
GO TO 6
```

```
5 IF(GR(IN(I),1).NE.GR(IN(I),2)) GO TO 6
```

```
GR(IN(I),1) = GR(IN(I),1) - (0.1*FLOAT(I)*ABS(GR(IN(I),1)))
```

```
GR(IN(I),2) = GR(IN(I),2) + (0.1*ABS(GR(IN(I),2)))
```

```
6 DEL(I) = (GR(IN(I),2) - GR(IN(I),1))/10.
```

```
IF(MSS.EQ.1) GO TO 7
```

```
WRITE(2,103)(IX,PL,NGX(I),PR,GR(IN(I),1),SB(I),DEL(I),IX,PL,NGX(I)
```

```
*,PR,GR(IN(I),2),I=2,NG)
```

```
GO TO 8
```

```
7 WRITE(2,103)(IY,PL,NGY(I),PR,GR(IN(I),1),SB(I),DEL(I),IY,PL,NGY(I)
```

```
*,PR,GR(IN(I),2),I=2,NG)
```

```
8 WRITE(2,104)
```

```
IF((NG-1).EQ.1) GO TO 9
```

```
WRITE(2,101) SB(9)
```

```
9 DO 10 I=2,NG
```

```
10 GR(IN(I),2) = 100./(GR(IN(I),2) + GR(IN(I),1))
```

C

C

ABOVE SETS AND WRITES OUT PLOT INFO AND CONSTANTS.

C

C

BELOW PLOT THE DATA IN SAVE MATRIX.

C

```
NPR=0
```

```
NO = 0
```

```
DO 20 K=1,NOUT
```

```
NPR = NPR + 1
```

```
IF(K.NE.1.AND.K.NE.NOUT) GO TO 11
```

```
NPR = 5
```

```
11 IF(NPR.NE.5) GO TO 13
```

```
DO 12 I=1,101
```

```
12 LINE(I) = PLUS
```

```

      GO TO 16
13 DO 14 I=2,100
14 LINE(I) = BLANK
   DO 15 I=1,101,10
15 LINE(I) = PLUS
16 DO 19 I=2,NG
   N = IFIX((SAVE(K,IN(I))-GR(IN(I),1))*GR(IN(I),2)+1.5)
   IF(N.GE.1.AND.N.LE.101) GO TO 17
   NO = NO + 1
   GO TO 19
17 IF(LINE(N).EQ.BLANK.OR.LINE(N).EQ.PLUS) GO TO 18
   LINE(N) = SB(9)
   GO TO 19
18 LINE(N) = SB(I)
19 CONTINUE
   WRITE(2,106) LINE
   IF(NPR.NE.5) GO TO 20
   WRITE(2,107) SAVE(K,IN(1))
   NPR = 0
20 CONTINUE
   IF(NO.NE.0) WRITE(2,105) NO
21 RETURN
END
SUBROUTINE SCALE(JJ,GR,JG)

```

C THIS ROUTINE SCALES THE DATA FOR PLOT LIMITS.

C COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
C COMMON/GG/ND(20),NPT,NSP,NTM,PTIME

C DIMENSION GR(8,2),STEPS(10),JG(8)
C DIMENSION SGR(8,2)

C DATA STEPS/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./

C 100 FORMAT(2F10.0)

C IF(ND(16).EQ.4) GO TO 15
C IF(ND(16).EQ.0) GO TO 7

```

      IF(ND(16).EQ.2) GO TO 3
      IF(ND(16).EQ.3) GO TO 1
      READ(1,100)YMIN,YMAX
      GO TO 5
1    DO 2 I=2,NG
      READ(1,100)YMIN,YMAX
      SGR(JG(I),1) = YMIN
      SGR(JG(I),2) = YMAX
      GR(JG(I),1) = YMIN
2    GR(JG(I),2) = YMAX
      RETURN
C
3    YMIN = GR(JG(2),1)
      YMAX = GR(JG(2),2)
      IF((NG-1).EQ.1) GO TO 7
C
      DO 4 I=JJ,NG
      IF(YMIN.GE.GR(JG(I),1)) YMIN = GR(JG(I),1)
      IF(YMAX.LE.GR(JG(I),2)) YMAX = GR(JG(I),2)
4    CONTINUE
5    DO 6 I=JJ,NG
      SGR(JG(I),1) = YMIN
      SGR(JG(I),2) = YMAX
      GR(JG(I),1) = YMIN
6    GR(JG(I),2) = YMAX
C
      IF(ND(16).EQ.1.OR.ND(16).EQ.2) RETURN
C
7    DO 14 K=JJ,NG
      YMIN = GR(JG(K),1)
      YMAX = GR(JG(K),2)
      DY = (YMAX - YMIN)/10.
      IF(DY.LE.0.0) GO TO 14
      TA = ALOG10(DY)
      IR = TA
      IF(TA.LT.0.0) IR = TA - 1.
      DY = DY*10.**(-IR) - 0.005
      DO 8 J=1,10
      IF(STEPS(J).LT.DY) GO TO 8

```

```

      I=J
      GO TO 9
8     CONTINUE
      I=10
9     CONTINUE
      IF(I.LE.10) GO TO 10
      I=I-10
      IR=IR+1
10    CONTINUE
      COR = STEPS(I)*10.**IR
      IF(I.EQ.10) IR=IR+1
      IF(YMIN.GE.0.0) GO TO 11
      MIN = YMIN/COR - .995
      GO TO 12
11    CONTINUE
      MIN = YMIN/COR + .005
12    CONTINUE
      TA = MIN*COR
      TB = TA + 10.*COR
      IF(TB.GE.YMAX) GO TO 13
      I=I+1
      GO TO 9
13    GR(JG(K),1) = TA
      GR(JG(K),2) = TB
      SGR(JG(K),1) = TA
      SGR(JG(K),2) = TB
14    CONTINUE
      RETURN
15    DO 16 I=JJ,NG
      GR(JG(I),1) = SGR(JG(I),1)
16    GR(JG(I),2) = SGR(JG(I),2)
      RETURN

```

```

      END
      SUBROUTINE PARADJ

```

```

C      COMMON/AA/X(80),P(80),Y(22),YT(22),F(10),C(9),NY,NX,NP,MMSP
C
      DIMENSION NPA(20)

```

```
C
1 FORMAT(20(I2,1X))
2 FORMAT(I2)
3 FORMAT(8F10.0)

C
  READ(1,2)JP
  READ(1,1)(NPA(J),J=1,JP)

C
  READ(1,3)(P(NPA(J)),J=1,JP)

C
  RETURN
  END
  BLOCK DATA

C
  COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT

C
  DATA SB/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H*,1H8,1H*,1H./
  DATA PL,PR,IX,IY/1H(.1H),1HX,1HY/

C
  END
```


Appendix K

CLOSED LOOP ERROR ANALYSES FOR
STEP AND RAMP SET POINT CHANGES,
CASCADE SYSTEM

... THIS PAGE
WAS THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix K

CLOSED LOOP ERROR ANALYSES FOR STEP AND RAMP SET POINT CHANGES, CASCADE SYSTEM

To determine whether offset exists for step changes in set point and what the offset should be for ramp changes, an error analysis must be performed and the limit taken. The analyses will be for the cascade closed loop system with the PID inner loop, digital outer loop design of Appendix E.

From Appendix E, the plant pulse transfer function and closed loop response are:

$$HG(Z) = \frac{C_1 K_m Z^{-N}}{\Delta \tau} \left[\frac{1 - Z^{-1} \frac{C_2}{C_1}}{(1 - Y_1 Z^{-1})(1 - Y_2 Z^{-1})} \right] \quad (K-1)$$

and

$$\frac{T_M(Z)}{R(Z)} = Z^{-N} C_3 \left[\frac{1 + Z^{-1} \frac{C_4}{C_3}}{1 - Y_3 Z^{-1}} \right] \quad (K-2)$$

where

$$C_1 = X_1 - X_2 \quad (K-3)$$

$$C_2 = Y_2 X_1 - Y_1 X_2 \quad (K-4)$$

$$C_3 = 1 - X_3 \quad (K-5)$$

$$C_4 = X_3 - Y_3 \quad (K-6)$$

$$X_1 = e^{-MT/\tau_m} \quad (K-7)$$

$$X_2 = e^{-MT/\lambda_1} \quad (K-8)$$

$$X_3 = e^{-MT/\lambda_2} \quad (K-9)$$

$$Y_1 = e^{-T/\tau_m} \quad (K-10)$$

$$Y_2 = e^{-T/\lambda_1} \quad (K-11)$$

and

$$Y_3 = e^{-T/\lambda_2} \quad (K-12)$$

The controller $D(Z)$ is

$$D(Z) = \frac{1}{K'} \frac{1}{HG(Z)} \frac{T_M(Z)/R(Z)}{1 - T_M(Z)/R(Z)} = \frac{M(Z)}{E(Z)}$$

$$= \frac{Z^{-N} C_3 \Delta \tau K'''}{C_1 K_m} \left[\frac{\left(1 + Z^{-1} \frac{C_4}{C_3}\right) (1 - Y_1 Z^{-1}) (1 - Y_2 Z^{-1})}{\left(1 - Z^{-1} \frac{C_2}{C_1}\right) (1 - Y_3 Z^{-1} - C_3 Z^{-N} - C_4 Z^{-N-1})} \right] \quad (K-13)$$

where

$$K''' = 1/K'.$$

From block diagram algebra,

$$\frac{T_M(Z)}{E(Z)} = D(Z)HG(Z). \quad (K-14)$$

Combining Equations K-1 and K-13, the result is:

$$\frac{T_M(Z)}{E(Z)} = Z^{-N} C_3 K''' \left[\frac{\left(1 + Z^{-1} \frac{C_4}{C_3}\right)}{\left(1 - Y_3 Z^{-1} - C_3 Z^{-N} - C_4 Z^{-N-1}\right)} \right] \quad (K-15)$$

The transfer functions for error in relation to the set point is obtained from Equations K-2 and K-15.

$$\frac{E(Z)}{R(Z)} = \frac{E(Z)}{T_M(Z)} \cdot \frac{T_M(Z)}{R(Z)} = \frac{1}{K'''} \left[\frac{(1-Y_3 Z^{-1} - C_3 Z^{-N} - C_4 Z^{-N-1})}{(1-Y_3 Z^{-1})} \right] \quad (K-16)$$

First, the steady-state error for a step change in set point will be determined. A step change results in the following equation.

$$E(Z) = \frac{1}{1-Z^{-1}} \cdot \frac{1}{K'''} \cdot \left[\frac{(1-Y_3 Z^{-1} - C_3 Z^{-N} - C_4 Z^{-N-1})}{(1-Y_3 Z^{-1})} \right] \quad (K-17)$$

The steady-state error is determined by taking the limit of $(1-Z^{-1}) E(Z)$ as Z^{-1} approaches 1.

$$E(\infty) = \lim_{Z^{-1} \rightarrow 1} (1-Z^{-1}) E(Z) \quad (K-18)$$

Applying Equation K-18 to K-17 and substituting the constants back into the result, yields:

$$\begin{aligned} E(\infty) &= \lim_{Z^{-1} \rightarrow 1} (1-Z^{-1}) E(Z) = \frac{1}{K'''} \left(\frac{1-Y_3-C_3-C_4}{1-Y_3} \right) \\ &= \frac{1}{K'''} \left[\frac{1-Y_3 - (1-X_3) - (X_3-Y_3)}{1-Y_3} \right] = 0. \end{aligned} \quad (K-19)$$

As was expected, the steady-state error would be zero as long as the fluid and mold dead times are not zero. Because the controller contains an excess pole at $Z = 1$, the algorithm contains the integral mode which forces the steady-state error to zero.

The same analyses will now be applied assuming a ramp input. The error becomes:

$$E(Z) = \frac{KTZ^{-1}}{(1-Z^{-1})^2} \cdot \frac{1}{K'''} \cdot \left[\frac{(1-Y_3 Z^{-1} - C_3 Z^{-N} - C_4 Z^{-N-1})}{(1-Y_3 Z^{-1})} \right] \quad (K-20)$$

The numerator of Equation K-20 contains a factor of $(1-Z^{-1})$ which is not obvious but exists. Rewriting Equation K-20 in terms of a modified numerator,

$$E(Z) = \frac{KTZ^{-1}}{(1-Z^{-1})^2} \cdot \frac{1}{K'''} \cdot \left[\frac{(1-Z^{-1})f(Z)}{(1-Y_3Z^{-1})} \right] \quad (K-21)$$

Taking the limit of Equation K-20 yields

$$E(\infty) = \lim_{Z^{-1} \rightarrow 1} (1-Z^{-1})E(Z) = \frac{KT}{K'''} \left[\frac{f(1)}{(1-Y_3)} \right] = \frac{KTC}{K'''} \quad (K-22)$$

where

$$C = \text{Constant} = F(1)/(1-Y_3).$$

Equation K-22 indicates that in the steady-state the error will approach a constant whose magnitude will be affected by the ramp rate magnitude (k), the sample period (T), the artificial gain (K'''), and a constant (C) that is a function of the sample period.

Appendix L

DIGITAL CONTROLLER DESIGN, CASCADE
CONFIGURATIONS: INNER LOOP PI, OUTER
LOOP DIGITAL DESIGN, FIRST ORDER PLANT

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix L

DIGITAL CONTROLLER DESIGN, CASCADE CONFIGURATIONS: INNER LOOP PI, OUTER LOOP DIGITAL DESIGN, FIRST ORDER PLANT

The following is a derivation of a digital control algorithm designed to force the closed loop to perform as first order lag. The actual configuration of the control loop is as illustrated in Figure L-1. In prior derivations, the inner fluid loop closed loop response was approximated by a first order lag. That first order lag when combined with the first order lag approximation of the mold temperature response gave the combined plant second order characteristics. If the mold and fluid loop responses are combined and modeled as a first order lag, the resulting configuration would be as illustrated in Figure L-2. The empirical techniques for determining the mold time constants are based on the plant response time to step changes in set point to the fluid loop. The measured plant dynamics are a combination of the fluid and mold dynamics. The plant dead time is the sum of the fluid and mold dead times, while the plant time constant is a function of both the fluid and mold time constants. In the second order plant, the mold dynamics had to be decoupled from the measured fluid and plant dynamics. In this derivation, the plant dynamics are the only dynamics of interest.

From Figure L-2, the plant model is

$$\frac{T_M(S)}{M^*(S)} = \frac{K_p e^{-\theta_p S}}{\tau_p S + 1} \quad (L-1)$$

letting the dead time be defined as

$$\theta_p = NT + (T - MT) \quad (L-2)$$

where

N = the integer number of sample periods in the dead time,

m = the fractional position of the dead time between sample periods,

T = the sample period,

K_p = the steady-state gain, and

Figure L-1. Actual Cascade Control Loop Configuration

Figure L-2. Design Configuration for Mold Loop

τ_p = the plant time constant.

Inserting Equation L-2 into L-1,

$$G_p(S) = \frac{T_M(S)}{M^*(S)} = \frac{K_p}{\tau_p} \left\{ \frac{e^{-(N+1)TS} e^{mTS}}{S+1/\tau_p} \right\} \quad (L-3)$$

From the tables, the pulse transfer function of Equation L-3 is:

$$G_p(Z) = \frac{T_M(Z)}{M(Z)} = \frac{K_p Z^{-N-1}}{\tau_p} \left\{ \frac{e^{-mT/\tau_m}}{1 - e^{-T/\tau_m} Z^{-1}} \right\} \quad (L-4)$$

Specifying the closed loop response as a first order lag plus dead time to step changes in set point:

$$T_M(Z) = \mathcal{Z} \left\{ \frac{1}{S} \cdot \frac{e^{-\theta S}}{\lambda S + 1} \right\} \quad (L-5)$$

where

$\theta = \theta_p$ = Plant dead time and

λ = Closed loop time constant specified by the user.

From the tables, Equation L-5, becomes:

$$T_M(Z) = \frac{(1 - e^{-mT/\lambda}) + Z^{-1}(e^{-mT/\lambda} - e^{-T/\lambda})}{(1 - Z^{-1})(1 - e^{-T/\lambda}Z^{-1})} \quad (L-6)$$

Because the term $(1 - Z^{-1})$ in the denominator is a unit step, Equation L-6 becomes

$$\frac{T_M(Z)}{R(Z)} = \frac{(1 - e^{-mT/\lambda}) + Z^{-1}(e^{-mT/\lambda} - e^{-T/\lambda})}{(1 - e^{-T/\lambda}Z^{-1})} \quad (L-7)$$

The controller, by solving the block diagram, is:

$$D(Z) = \frac{1}{K} \cdot \frac{1}{G_p(Z)} \cdot \frac{T_m(Z)/R(Z)}{1 - T_m(Z)/R(Z)} \quad (L-8)$$

Substituting Equations L-3 and L-7 into L-8 gives

$$D(Z) = K' K_{cs} \left\{ \frac{1 - p_0 Z^{-1} - p_1 Z^{-2}}{1 - p_5 Z^{-1} - p_8 Z^{-N-1} - p_9 Z^{-N-2}} \right\} = \frac{M(Z)}{E(Z)} \quad (L-9)$$

where

$$K' = 1/K \quad (L-10)$$

$$K_{cs} = \frac{\tau_p (1 - e^{-mT/\lambda})}{K_p e^{-mT/\tau_m}} \quad (L-11)$$

$$p_0 = e^{-T/\tau_m} - \frac{e^{-mT/\lambda} - e^{-T/\lambda}}{1 - e^{-mT/\lambda}} \quad (L-12)$$

$$P_1 = \frac{e^{-T/\tau_m}(e^{-mT/\lambda} - e^{-T/\lambda})}{1 - e^{-mT/\lambda}} \quad (L-13)$$

$$P_5 = e^{-T/\lambda} \quad (L-14)$$

$$P_8 = 1 - e^{-mT/\lambda} \quad (L-15)$$

and

$$P_9 = e^{-mT/\lambda} - e^{-T/\lambda} \quad (L-16)$$

The parameter subscripts were chosen to be equivalent to the subscripts of the second order algorithm. Cross multiplying and converting to the time domain, the control algorithm is Equation L-17.

$$M_n = K'K_{cs} [1 - P_0 e_n - P_1 e_{n-1} + P_5 M_{n-1} + P_8 M_{n-N-1} + P_9 M_{n-N-1}] \quad (L-17)$$

DISTRIBUTION

Copy

R. Bulcock, DOE-KCAO	1
R. Caudle, DOE-MA, Washington, DC	2
H. N. Edin, D/171, 1A31	3
J. A. Elleman, D/171, 1A31	4
J. E. Long, D/531, 1A46	5
H. T. Barnes, D/554, BD50	6-7
L. Stratton, D/554, 2C44	8-10
R. H. Poste, D/744, MC45	11-12
R. P. Frohmberg, D/800, 2A39	13
V. E. Alley, D/818, 2A31	14
R. H. Floersch, D/818, 2A31	15-22
D. R. Hahn, D/818, 2A31	23
B. A. Hefner, D/818, 2A31	24
G. E. Martinette, D/818, 2A31	25
R. E. Kessler, D/865, 2C40	26

BDX-613-1980

DIGITAL CONTROLLER DESIGN, ANALYSIS, AND IMPLEMENTATION FOR THE MOLD HEATING AND COOLING SYSTEM. R. H. Floersch, Topical, December 1978

Mold heating and cooling control using metal temperature feedback was unsuccessful when conventional control strategies were used. Special digital control algorithms, developed to effect mold temperature control, were formulated by using empirical mold and fluid loop dynamic models in conjunction with Z transforms to arrive at a controller configuration.

SPECIAL PROJECTS: Mold Heating and Cooling
DIGITAL CONTROLLER DESIGN, ANALYSIS, AND IMPLEMENTATION FOR THE MOLD HEATING AND COOLING SYSTEM. R. H. Floersch, Topical, BDX-613-1980, December 1978

Mold heating and cooling control using metal temperature feedback was unsuccessful when conventional control strategies were used. Special digital control algorithms, developed to effect mold temperature control, were formulated by using empirical mold and fluid loop dynamic models in conjunction with Z transforms to arrive at a controller configuration.

DIGITAL CONTROLLER DESIGN, ANALYSIS, AND IMPLEMENTATION FOR THE MOLD HEATING AND COOLING SYSTEM. R. H. Floersch, Topical, BDX-613-1980, December, 1978

Mold heating and cooling control using metal temperature feedback was unsuccessful when conventional control strategies were used. Special digital control algorithms, developed to effect mold temperature control, were formulated by using empirical mold and fluid loop dynamic models in conjunction with Z transforms to arrive at a controller configuration.