27
4-17-78
250 NTS

MASTER

BDX-613-1764 (Rev.)

COMPUTER SIMULATION OF THE MOLD
HEATING AND COOLING CONTROL LOOP,
USING CONVENTIONAL CONTROL STRATEGIES

Topical Report

R. H. Floersch, Project Leader

Published February 1978

Prepared for the United States Department of Energy
Under Contract Number EY-76-C-04-0613.

**Kansas City
Division**

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

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-613-1764 (Rev.)
Distribution Category UC-38

COMPUTER SIMULATION OF THE MOLD
HEATING AND COOLING CONTROL LOOP,
USING CONVENTIONAL CONTROL STRATEGIES

BDX-613-1764 (Rev.)

Published February 1978

Project Leader:
R. H. Floersch
Department 861

Topical Report

NOTICE

**PORTIONS OF THIS REPORT ARE ILLEGIBLE. IT
HAS BEEN REDUCED FROM THE BEST AVAILABLE
COPY TO PERMIT THE BROADEST POSSIBLE AVAIL-
ABILITY.**

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

fly

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

COMPUTER SIMULATION OF THE MOLD HEATING AND COOLING CONTROL LOOP,
USING CONVENTIONAL CONTROL STRATEGIES

BDX-613-1764 (Rev.), UNCLASSIFIED Topical Report, Published
February 1978

Prepared by R. H. Floersch, D/861

Mathematical models of the mold and fluid temperature responses were developed experimentally. Those models were then used with a digital computer simulation program in an effort to duplicate mathematically the physical plant dynamic response. The simulation program underwent a series of evolutionary steps until satisfactory duplication of the processes existed. Various control strategies were then simulated. The evolution of the simulation program and the simulation results are presented.

WPC-dkvh

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.

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 EY-76-C-04-0613

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

CONTENTS

Section	Page
SUMMARY.	11
DISCUSSION	13
SCOPE AND PURPOSE.	13
PRIOR WORK	13
ACTIVITY	13
<u>General Model of the Control Loop.</u>	13
<u>Simulation Program Structure</u>	18
<u>Simulation of the Analog and Analog</u> <u>Equivalent Loops</u>	20
<u>Simulation of the Direct Digital Control Loop.</u>	37
<u>Simulation of the Cascade Control Loop</u>	53
ACCOMPLISHMENTS.	63
FUTURE WORK.	64
REFERENCES	65
APPENDICES	
A. EQUIVALENT VALVE FLOW ANALYSIS	67
B. COMPUTER PROGRAM FOR AN ANALOG CONTROL LOOP SIMULATION	73
C. COMPUTER PROGRAM FOR AN ANALOG EQUIVALENT TO A DIRECT DIGITAL CONTROL LOOP.	91
D. COMPUTER PROGRAM FOR A DIRECT DIGITAL CONTROL LOOP	111
E. COMPUTER PROGRAM FOR CASCADE CONTROL PID ALGORITHMS	131
DISTRIBUTION	157

ILLUSTRATIONS

Figure		Page
1	Single-Zone Schematic of a Four-Zone System in Automatic Control Mode.	14
2	Block Diagram of Individual Recirculating Loop for Analog Control, Revision 4, August 4, 1970	21
3	Block Diagram Descriptive Example.	23
4	Sample and Hold, Illustrating Effective Delay by Sampling.	28
5	Analog Equivalent to a DDC Loop, Revision 1, October 5, 1970.	29
6	Closed Loop Block Diagram, Root Locus Example.	36
7	Signal Flow of One Loop Under Computer Control.	39
8	DDC Simulation, Revision 3, July 27, 1972. . .	45
9	Conventional Cascade Control Scheme Block Diagram.	54
10	Cascade Simulation, Revision 1, March 4, 1974	55
11	Set Point Adjusting Technique to Achieve a 1.5-Percent Maximum Overshoot.	62
A-1	Effective Flow Network Supply and Return Valves for Mold Heating and Cooling Control Loop	70
A-2	Equivalent Flow Network.	71

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

TABLES

Number		Page
1	Block Definitions for Analog Control Simulation, Revision 4, August 4, 1970	24
2	Parameter List for Analog Control Simulation, Revision 4, August 4, 1970	25
3	Parameter List for Analog Equivalent to DDC, Revision 1, October 6, 1970.	31
4	Parameter List for DDC Simulation, Revision 3, July 28, 1972.	42
5	Performance Results of DDC Simulation, August 17, 1972, General K-2810 Cavity Dynamics	50
6	Controller Gain Product Versus IAE and ISE With No Options.	52
7	Random Sample of PID Velocity Algorithm Constants in Use October 1, 1976	52
8	Parameter List for Cascade Simulation, Revision 1, March 4, 1974.	57
9	Cascade Test Results of Modeling the Closed Loop Responses as Second Order Underdamped.	60

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

SUMMARY

The mold heating and cooling project was started 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 control two- and four-zone fluid temperature loops which control the mold temperature response. Investigation of various ways of accomplishing direct metal temperature control was reported in May 1970. The control system operation and theory were discussed in reports issued in June 1971¹ and in September 1972,² and experimental model results were presented in a report issued in August 1974.³

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 at using 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 is to provide direct metal temperature control, using the metal temperature as one of the process variables in any control scheme. Cycling the metal temperature set point could then be used instead of the fluid temperature, thus eliminating much of the time required to develop the proper oil time-temperature profile *recipe*.

An expeditious means of evaluating the various control schemes without developing real-time software for each scheme was necessary. A computer program which simulates the real mold and fluid temperature responses provided the means for evaluating any control scheme.

The first simulation programs were developed before hardware was available to determine mold responses experimentally. The results of the early programs were marginal because some of the assumptions made for simplification were invalid, and approximations made at various response characteristics were at best only guesses. The early simulations pointed out that more information was necessary concerning the loop flow rates and the valve flow characteristics to evaluate the impact those variables had on stability. Those areas were investigated extensively, and the results were published in an interim report.²

The simulation package evolved as follows:

- Fluid loop feedback-analog controller,
- Fluid loop feedback-analog equivalent,
- Fluid loop feedback-direct digital control, using proportional integral derivative (PID) algorithm, and
- Fluid and metal feedback-cascade control, using PI and PID algorithms.

Progress was made through each evolution toward a more accurate representation of the fluid control loop. The direct digital control simulation produced responses closest to the observed system responses.

Cascade control, using conventional control algorithms, was simulated and correlated to experimental testing. Simulation showed that gain settings of the major and minor loop controller were critical to the performance of the control scheme. Process dynamics played a large role in the overall stability of the control configuration and dictated the major loop controller constants. The longer the response time of the mold, the smaller the controller constants became. The simulation indicated that while cascade control would work satisfactorily with fast responding systems, it would not work within specifications for slow responding systems. One criterion for general purpose metal temperature control is that it must work satisfactorily for all types of mold responses. Cascade control did not fulfill that criterion.

A program has been started, to develop a control algorithm specially tailored to the desired mold temperature response, using Z-transforms and process dynamics.

DISCUSSION

SCOPE AND PURPOSE

The purpose of the mold heating and cooling equipment used by Bendix Kansas City Division is to control the temperatures of the mold and the part during manufacturing. That control is currently accomplished by temperature-cycling the heat transfer fluid in the control loop. The major objective of this endeavor is to provide temperature control by using the metal temperature as a process variable. Evaluation of the control schemes proposed to accomplish direct metal temperature control is an important part of accomplishing that goal.

PRIOR WORK

All prior work was reported in four interim reports, issued in May 1970, June 1971,¹ September 1971,² and August 1974.³

ACTIVITY

General Model of the Control Loop

The fluid temperature control loop, illustrated schematically in Figure 1, uses a Proportional-Integral-Derivative (PID) controller to manipulate the control valves when the system is in the *automatic* mode. The fluid temperature is sensed just downstream of the centrifugal pump by a thermocouple in a well in the fluid stream. The millivolt signal of the thermocouple is converted to a milliamp signal by the millivolt-to-current converter. The output signal range of the millivolt-to-current converter is 4 to 20 milliamps, which represents a temperature range of 10 to 316°C (50 to 600°F). The current signal is converted to a voltage signal by dropping across a 250-ohm resistor. The voltage signal range is 1 to 5 volts dc. That voltage is compared with the set point voltage. The difference is used by the PID controller to compute a valve position which will force the process variable to the set point. The 4- to 20-milliamp output of the controller is converted to a 21- to 103-kPa (3 to 15 psi) pneumatic signal. The pneumatic signal is used by the valve positioners to position the valve stems. The hot supply and return valves will open proportionally to the pneumatic signal between 62 and 103 kPa and will remain closed below 62 kPa. The cold supply and return valves will open proportionally to the pneumatic signal between 62 and 21 kPa (9 to 3 psi) and will remain closed above 62 kPa.

Figure 1. Single Zone Schematic of a Four-Zone System in Automatic Control Mode

The model used by the simulation programs is developed as follows. The loop temperature can be determined by taking a mass and energy balance on the circulating loop,¹ with the following result:

$$T_L(t) = T_R(t) + \frac{Q_S(t)}{Q_L} [T_S - T_R(t)], \quad (1)$$

where

$T_L(t)$ = Loop temperature ($^{\circ}\text{C}$) ($^{\circ}\text{F}$),

$T_R(t)$ = Return temperature ($^{\circ}\text{C}$) ($^{\circ}\text{F}$),

$Q_S(t)$ = Flow rate into the loop (m^3/s) (gal/min),

Q_L = Loop flow rate (m^3/s) (gal/min), and

T_S = Central system supply temperature ($^{\circ}\text{C}$) ($^{\circ}\text{F}$).

The supply temperature (T_S) in Equation 1 will be either the hot system temperature (287°C) or the cold system temperature (32°C), depending on the pneumatic signal magnitude.

The loop flow rate (Q_L) in Equation 1 is determined by the fluid channel design² in the mold. That loop flow rate will be the value at the intersection of the loop loss curve and the pump gain curve.

The amount of hot or cold fluid entering the control loop [$Q_S(t)$] is dependent on the valve stem travel, the equivalent valve flow coefficient (Appendix A), and the central system differential pressure.² The relationship is

$$Q_S(t) = C_{vcq} \sqrt{\frac{\Delta P}{\rho}} \left(e^{-4[ST(t) - 1.0]} \right), \quad (2)$$

where

$Q_S(t)$ = Flow rate into the loop (m^3/s) (gal/min),

C_{veq} = Equivalent valve flow coefficient for the supply-return valve combination, (Appendix A for derivation), ($\text{m}^3/\text{s}-\text{kPa}^{-1/2}$) (gal/min-psi $^{-1/2}$),

ΔP = Central system differential pressure (kPa) (psi),

ρ = Fluid specific gravity, and

ST(t) = Valve stem travel as decimal fraction of range.

The central system differential pressure in Equation 2 is held constant by the central system differential pressure valve. The equivalent valve coefficient, a function of the valve hardware, is constant. The flow rate into and out of the control loop is, therefore, a function of the valve stem travel only. The loop temperature is controlled by controlling the flow rate of central system fluid into the loop.

The last variable in Equation 1, the return fluid temperature $[T_R(t)]$, is the most difficult to predict since a number of variables affect the transfer of heat from the fluid to the metal and vice versa. Analysis has shown that the inability of the metal to quickly conduct the heat away from the channel area is the greatest resistance to heat transfer. Other variables such as the configuration, diameter, length, and roughness of the mold channels, amount of exposed surface area, ambient conditions, fluid characteristics, and scale buildup affect the amount of heat transferred.

It was decided that the most acceptable way of predicting the return fluid temperature was to use a combination of empirically obtained mold response information, plus a lumped mass model to depict the mold temperature response. The lumped mass model is derived from an energy balance. The amount of heat given up by the fluid must equal the amount of heat absorbed by the metal, plus the heat lost to the surroundings.

$$\sum_{j=1}^n m_j (h_{i,j} - h_{o,j}) = MC_p \frac{dT_m}{dt} + \dot{Q}_{LS} \quad (3)$$

where the subscripts i, o, and j represent the inlet and outlet conditions of the jth mold channel, and

m = Mass flow rate (kg/hr) (lbm/hr),

h = Fluid enthalpy (kJ/kg) (Btu/lbm),

M = Mold mass (kg) (lbm)

C_p = Metal specific heat (kJ/kg-°C) (Btu/lbm-°F),

$\dot{Q}_{LS}$ = Heat losses (kJ/hr) (Btu/hr),

dT_m/dt = Time rate of change of mold temperature (°C/hr) (°F/hr),
and

n = Number of mold zones.

In the steady state, Equation 3 reduces to

$$\sum_{j=1}^n m_j (h_{i,j} - h_{o,j}) = \dot{Q}_{LS} \quad (4)$$

If the empirical mold temperature response is modeled as a first order lag plus dead time,^{1,3} then the mold temperature differential equation is

$$\tau_m \frac{dT_m}{dt} + T_m = K T_f(t - \theta) \quad (5)$$

where

τ_m = Mold time constant (hr),

T_m = Empirical mold temperature (°C) (°F),

K = Steady state gain (°C/°C) (°F/°F),

$T_f(t)$ = Fluid temperature forcing (°C) (°F), and

θ = Dead time.

The steady state gain in Equation 5 adjusts the inlet fluid temperature by the amount of the losses to obtain the mold temperature. Since the losses are included in the metal temperature response, the loss term $\dot{Q}_{LS}$ in Equation 3 can be neglected. Assuming a single zone, Equation 3 can be solved for the outlet enthalpy.

$$h_o = h_i - \frac{MC_p}{m} \frac{dT_m}{dt} \quad (6)$$

Solving Equation 5 for the derivative and substituting into Equation 6 yields the outlet enthalpy.

$$h_o = h_i - \frac{MC_p}{m} \left[\frac{K T_f(t) - T_m}{\tau_m} \right] \quad (7)$$

Equation 5 can be solved by numerical integration to obtain the mold temperature for use in Equation 7. Both Equations 6 and 7 are used to obtain the outlet fluid enthalpy. Equation 5 is used to obtain the mold response.

The relationship between enthalpy (h) and temperature (T) is

$$h = C_1 T^2 + C_2 T + C_3, \quad (8)$$

where

C_1 through C_3 are constants and have units $\text{kJ/kg-}^\circ\text{C}^2$, $\text{kJ/kg-}^\circ\text{C}$, and kJ/kg . Once the outlet enthalpy is known, by solving either Equation 6 or 7, the outlet fluid temperature can be found by solving Equation 8 as a quadratic.

$$T = \frac{-C_2 + \sqrt{C_2^2 - 4C_1(C_3 - h)}}{2C_1}. \quad (9)$$

The above model for determining the return fluid temperature is used in all four simulations described in this report.

The model described in this section is the most current evolution; consequently, the early simulations diverge in some respects to the one described above.

Simulation Program Structure

The basic structure of the simulation program consists of a main program and related subprograms. The main program controls the execution of the simulation. The basic information required by the simulation program is read in either in the main program or by a subroutine called INPUT. The input variables that are read are as follows:

Parameter list (P's),

Initial conditions of the state variables (Y's),

Time delays (H's),

Integration and timing controls (DELT, BEGTIM, ENDTIM, AND NOUTPT),

Print and execution option variables (ND's),

Mold identifiers (MMFR, MSFR, MTEST, MDES, IDMO), and

X and Y variable numbers to be printed or plotted (NTY, NGY, NTX, NGX).

The following variable labels are maintained throughout the simulations.

X's = Intermediate variables,
P's = Parameters or constants in equations,
Y's = State variables,
F's = Derivatives of state variables, and
H's = Time delays.

The program calls the initialize function (INTFUN) subroutine after all the input information has been read. The intermediate variables that require an initial value other than zero, and the delay blocks, are initialized in INTFUN. The next routine called is the intermediate calculation (INTCAL) subprogram where all intermediate values are calculated. The following routine called is the differential equation routine (DIFFEQ or DERFUN) where all the derivatives of the state variables are calculated.

Before the first integration step is made, all the parameters and identifiers are printed either from the main program or routine ALIST. Next, the total number of iterations (NSTEP) and the output interval (NW) are computed from the integration step size (DELT) and the timing parameters (ENDTIM, BEGTIM, and PTIME).

The program is now ready to go through the iterative, DO LOOP, NSTEP times. The first subroutine called inside the loop is the integration routine (RKINT). The integration routine uses the fourth order Runge Kutta technique for integrating the state variables. When control passes back to the loop from RKINT, the intermediate calculation routine is called to recalculate the intermediate variables with the new state variable values. Each time the DO LOOP is executed, an output index (NI) is incremented by one. When the output index (NI) equals the output interval (NW), the X and Y variables to be printed and/or plotted are stored in a storage matrix, and the output index is reset to zero. After NSTEP iterations, the X and Y values that were stored are printed or plotted by routines PRTTBL and PLOT. The routines not mentioned above, but encountered in the various programs, are defined below.

SPADJ	Adjusts set point as a function of time,
VELCON	Velocity controller algorithm,
POSCON	Positional controller algorithm,
DELAY	Keeps track of time delayed variables,
SCALE	Used by PLOT to scale data,
RECIPE	Reads in the set point recipe,
PARADJ	Reads in new parameter values for next run,

POT   Used by PLOT, and
 PRTPLT   Sets up plot header information.

Simulation of the Analog and Analog Equivalent Loops

The early simulation attempts were primarily made in an effort to determine the stable ranges of the proportional-integral-derivative (PID) controller constants. The equation that most closely describes the output of a PID controller is

$$m(t) = K_p \left[e(t) + \frac{1}{\tau_I} \int e(t) dt + \tau_D \frac{de(t)}{dt} \right] + M_i, \quad (10)$$

where

$m(t)$ = Output of the controller (mA),

K_p = Proportional gain constant (mA/V),

$e(t)$ = Error (difference between set point and process variable) (V),

τ_I = Integration time constant (min),

τ_D = Derivative time constant (rate) (min), and

m_i = Initial controller output at the time the controller is turned on (mA).

The constants under investigation were K_p , τ_I , and τ_D . Figure 2 illustrates the control loop under analog control in block diagram form. Each block indicates a calculation, time delay, or limit check. The block is an intermediate calculation when the output of the block is designated by an X variable. The block is a state variable and requires an integration when the output is designated by a Y variable. State variables are represented in transfer function form with S being the Laplace operator. The parameters (P's) are multiplied, divided, added, or subtracted from or to the intermediate or state variables and are listed in close proximity to the block where they are used. Figure 3 illustrates the technique used. The equations represented by the two blocks might be

$$X_2 = P_1 X_1 + P_2, \quad (11)$$

and

Figure 2. Block Diagram of an Individual Recirculating Loop for Analog Control,
Revision 1, August 4, 1970.

Figure 3. Block Diagram Descriptive Example

$$P_3 \frac{dY_1}{dt} + Y_1 = P_4 X_2, \quad (12)$$

where

P_1 = Slope of current-to-temperature conversion ($^{\circ}\text{C}/\text{mA}$),

P_2 = Intercept of current-to-temperature conversion ($^{\circ}\text{C}$),

P_3 = Time constant of state variable Y_1 (s), and

P_4 = Steady state gain of state variable Y_1 ($^{\circ}\text{C}/^{\circ}\text{C}$).

The transfer function notation of the differential equation of Equation 12 is

$$\frac{Y_1}{X_2} = \frac{P_4}{P_3 S + 1}, \quad (13)$$

where the initial condition of the state variable is assumed to be zero.

The various block definitions are listed in Table 1, and the parameter definitions are listed in Table 2 for the analog simulations. The time constant, dead time, and gain parameters used in the analog simulation were obtained from experimental data collected, using equipment similar to the computer controlled equipment. The open loop dynamic data obtained from those tests were expected to be different when obtained on the new computer controlled pump package loops. None of the new equipment had been installed at the time of the early simulations; therefore, empirical response data could not be obtained. As an example, linear valve flow up to the maximum and nominal design pump flow rates had to be assumed for the new equipment, because no better information existed. It was also believed at the time that the pneumatic range of the control valves was zero to 103 kPa (zero to 15 psi) instead of the actual 21 to 103 kPa (3 to 15 psi).

Table 1. Block Definitions for Analog Control Simulation, Revision 4, August 4, 1970*

Block	Definition
T/I	Temperature to current
LIM	High-low limit check
I/P	Current to pressure
P/%L	Pressure to percent lift
%L/Q _f	Percent lift to fraction flow
Q _f /Q _s	Fraction flow to supply flow
Q _i /M _i	Flow to mass flow; i = s for supply; i = r for return
T _{hs} /h _s	Hot system temperature to enthalpy
T _{cs} /h _s	Cold system temperature to enthalpy
h _i /T _i	Inlet enthalpy to inlet temperature
T _m /h _r	Metal temperature to return enthalpy
h _r /T _r	Return enthalpy to return temperature

*See Figure 2.

The objective of the simulation was to determine the stable ranges for the PID controller setting, and to establish starting points for tuning those parameters. The simulations indicated that the proportional gain (K_p) should be approximately 1.5; the reset rate (inverse of the integration time constant) should be approximately 2.5 min⁻¹; and the rate (derivative time constant) should be approximately 0.01. The integral of the squared error was used as the performance index to be minimized.

$$\text{Index} = \text{ISE} = \int_0^t [e(t)]^2 dt . \quad (14)$$

The simulations indicated that the fluid loop was generally stable, and that the reset rate should be one and one-half to two times greater than the proportional gain for the best results. Simulations indicated that large derivative gains (rate) caused instability, and that little or no derivative gain produced the

Table 2. Parameter List for Analog Control Simulation, Revision 4, August 4, 1970

Parameter	Value	Description	Units
18	K_p	Controller Proportional Gain	%/%
19	$1/\tau_I$	Integral Gain	min^{-1}
20	τ_D	Derivative Gain	min
21	5.175 (0.75)	Current-to-Pressure Conversion	kPa/mA (psi/mA)
22	0.0	---	---
23	62.1 (9.0)	{ Pressure-to-Percent- Lift Conversion	kPa (psi)
24	41.4 (6.0)		kPa (psi)
25	100.0 (100.0)		Percent
26	3.5	{ Percent Lift to Percent Flow	Percent
27	0.033543		Percent
28	Q_T	Total Loop Flow Rate	m^3/s (gal/min)
29	-0.895 (-0.00461)	{ Temperature- to- Density Conversion	$\text{kg}/\text{m}^3\text{-}^\circ\text{C}$ (lbm/gal- $^\circ\text{F}$)
30	1377.5 (11.917)		kg/m^3 (lbm/gal)
31	-0.895 (-0.00461)		$\text{kg}/\text{m}^3\text{-}^\circ\text{C}$ (lbm/gal- $^\circ\text{F}$)
32	1377.5 (11.917)		kg/m^3 (lbm/gal)
33	7.5×10^{-4} (1.03×10^{-4})	{ Temperature-to- Enthalpy Conversion	$\text{kJ}/\text{kg-}^\circ\text{C}^2$ (Btu/lbm- $^\circ\text{F}^2$)
34	1.134 (0.26237)		$\text{kJ}/\text{kg-}^\circ\text{C}$ (Btu/lbm- $^\circ\text{F}$)
35	-0.284 (-8.686)		kJ/kg (Btu/lbm)

Table 2 Continued. Parameter List for Analog Control Simulation, Revision 4, August 4, 1970

Parameter	Value	Description	Units
36	-752.7 (-1276.0)	{ Enthalpy- to- Temperature Conversion	$^{\circ}\text{C}$ ($^{\circ}\text{F}$)
37	1.28498 (0.0691)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
38	-0.00302 (-0.000412)		$\text{kJ/kg-}^{\circ}\text{C}$ ($\text{Btu/lbm-}^{\circ}\text{F}$)
39	0.2804 (8.686)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
40	0.001506 (0.000206)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
41	K_3	Mold Gain	$^{\circ}\text{C}/^{\circ}\text{C}$ ($^{\circ}\text{F}/^{\circ}\text{F}$)
42	τ_3	Mold Time Constant	s
43	M	Mold Metal Mass	kg (lbm)
44	0.46 (0.11)	Specific Heat of Steel	$\text{kJ/kg-}^{\circ}\text{C}$ ($\text{Btu/lbm-}^{\circ}\text{F}$)
45	K_2	Thermocouple Gain	$^{\circ}\text{C}/^{\circ}\text{C}$ ($^{\circ}\text{F}/^{\circ}\text{F}$)
46	τ_2	Thermocouple Time Constant	s
47	0.05229 (0.02909)	{ Temperature-to- Current Conversion	$\text{mA}/^{\circ}\text{C}$ ($\text{mA}/^{\circ}\text{F}$)
48	3.4771 (2.5455)		mA
49	τ_v	Valve Time Constant	s

best results. When the proportional gain became as high as 5, the loop started oscillating. The simulations indicated that if the system functioned as nominally designed, it should have worked satisfactorily.

After the equipment had been installed and testing was started (January, 1971), the validity of the simulations was proved. The PID constants predicted by the program were very close to the optimum values obtained by actually tuning the analog control loops.

Before the control equipment was ready for testing, the analog equivalent to a direct digital control (DDC) loop simulation was formulated. This simulation was developed to find out if the additional delay caused by the sampling of the computer would degrade control. The additional delay concept⁴ is illustrated in Figure 4. The solid line represents the sampled signal, which is sampled every T units of time. The zero order hold unit will hold the sampled value until the next sample period when a new value is sampled. The output of the hold is the solid stair step curve in Figure 4.

The dashed line in Figure 4 is a curve constructed through the midpoints of the hold curve and appears as the original signal delayed in time by half the sample period. The net effect of the sample and hold components of a computer system is to introduce an additional time delay to the control loop equal to half the sample period. The effect of time delays on control systems is to add phase shift, which makes the control system less stable.⁵

The effect of the time delay can be studied by adding a time delay to the analog simulation. The analog equivalent simulation block diagram is presented in Figure 5, and the parameter list is defined in Table 3.

The basic differences between the analog equivalent to a DDC loop and the analog simulation occur in the controller, in the addition of time delays, and in the order of the mold response.

The analog controller was modified to allow gain changes when the absolute error exceeded a predetermined value. That option was added to the simulation to allow evaluation of the impact that the gain changing feature of the computer control algorithm would have on the performance of the fluid loop. The gain change option would most likely affect the control loop only on set point changes when the error would initially be large.

The time delays were added to more closely approximate the transport lags of the fluid flowing in the control loop. The transport lags H_7 , H_8 , and H_9 are the times required for the fluid to flow

Figure 4. Sample and Hold, Illustrating Effective Delay by Sampling

Figure 5. Analog Equivalent to a DDC Loop, Revision 1, October 5, 1970

Table 3. Parameter List for Analog Equivalent to DDC, Revision 1, October 6, 1970

Parameter	Value	Description	Units
18	K_p	Controller Proportional Gain	%/%
19	$1/\tau_I$	Reset (Integral Gain)	min^{-1}
20	τ_D	Rate (Derivative Gain)	min
21	5.175 (0.75)	Current-to-Pressure Conversion	kPa/mA (psi/mA)
22	0.0	---	---
23	62.1 (9.0)	{ Pressure-to- Percent-Lift Conversion	kPa (psi)
24	41.4 (6.0)		kPa (psi)
25	100.0 (100.0)		Percent
26	K_3	Mold Gain	$^{\circ}\text{C}/^{\circ}\text{C}$ ($^{\circ}\text{F}/^{\circ}\text{F}$)
27	$\tau_{3,1}$	First Mold Time Constant	s
28	Q_T	Total Loop Flow Rate	m^3/s (gal/min)
29	-0.895 (-0.00461)	{ Temperature-to- Density Conversion	$\text{kg}/\text{m}^3\text{-}^{\circ}\text{C}$ (lbm/gal- $^{\circ}\text{F}$)
30	1377.53 (11.917)		kg/m^3 (lbm/gal)
31	$\tau_{3,2}$	Second Mold Time Constant	s
32	---	Percent Lift to Fraction Flow	---
33	7.5×10^{-4} (1.03×10^{-4})	{ Temperature-to- Enthalpy Conversion	$\text{kJ}/\text{kg-}^{\circ}\text{C}^2$ (Btu/lbm- $^{\circ}\text{F}^2$)
34	1.134 (0.26287)		$\text{kJ}/\text{kg-}^{\circ}\text{C}$ (Btu/lbm- $^{\circ}\text{F}$)
35	-0.284 (-8.686)		kJ/kg (Btu/lbm)

Table 3 Continued. Parameter List for Analog Equivalent to DDC, Revision 1, October 6, 1970

Parameter	Value	Description	Units
36	-752.7 (-1276.0)	{ Enthalpy- to- Temperature Conversion	$^{\circ}\text{C}$ ($^{\circ}\text{F}$)
37	1.28498 (0.0691)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
38	-0.00302 (-0.000412)		$\text{kJ/kg-}^{\circ}\text{C}$ ($\text{Btu/lbm-}^{\circ}\text{F}$)
39	0.2804 (8.686)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
40	0.001506 (0.000206)		$\text{kJ/kg-}^{\circ}\text{C}^2$ ($\text{Btu/lbm-}^{\circ}\text{F}^2$)
41	K_4	Thermocouple Gain	$^{\circ}\text{C}/^{\circ}\text{C}$ ($^{\circ}\text{F}/^{\circ}\text{F}$)
42	τ_4	Thermocouple Time Constant	s
43	M	Mold Metal Mass	kg (lbm)
44	0.46 (0.11)	Specific Heat of Steel	$\text{kJ/kg-}^{\circ}\text{C}$ ($\text{Btu/lbm-}^{\circ}\text{F}$)
45	K_2	Thermocouple Gain	$^{\circ}\text{C}/^{\circ}\text{C}$ ($^{\circ}\text{F}/^{\circ}\text{F}$)
46	τ_2	Thermocouple Time Constant	s
47	0.05229 (0.02909)	{ Temperature-to- Current Conversion	$\text{mA}/^{\circ}\text{C}$ ($\text{mA}/^{\circ}\text{F}$)
48	3.4771 (2.5454)		mA
49	τ_v	Valve Time Constant	s
50	---	Not Used	---
51	K_{AE}	Absolute Error Gain Multiplier	---
52	Max x (18)	Value Above Which Absolute Error Multiplier is Used	---
53	0.00217	{ Percent-Lift- to-Fraction-Flow Conversion	$\%^{-2}$
54	0.10367		$\%^{-1}$
55	0.50667		---

Table 3 Continued. Parameter List for Analog Equivalent to DDC, Revision 1,
October 6, 1970

Parameter	Value	Description	Units
56	---	New Set Point	---
57	---	New Gain	---
58	---	New Mold Time Constant $\tau_{3,1}$	---
59	---	Time From Zero When P56, P57, P58, and P60 Are Used	---
60	---	New Mold Time Constant $\tau_{3,2}$	---

from the sensing point, through the mold, and back to the mixing point. The effective time delay of the sample and hold device is H_6 .

The mold dynamics were changed to second order so that the calculated mold response would more closely depict the observed response for step changes in the forcing function. The differential equation describing a second order overdamped¹ response is

$$\tau_1 \tau_2 \frac{d^2 Y_1}{dt^2} + (\tau_1 + \tau_2) \frac{dY_1}{dt} + Y_1 = F(t) . \quad (15)$$

The numerical integration routine (RKINT) can only integrate a first order differential equation. Equation 15 must be broken up into two first order differential equations for the simulation as follows.

Letting

$$\frac{dY_1}{dt} = Y_2 , \quad (16)$$

then

$$\frac{d^2 Y_1}{dt^2} = \frac{dY_2}{dt} . \quad (17)$$

Using Equations 16 and 17, Equation 15 becomes

$$\tau_1 \tau_2 \frac{dY_2}{dt} + (\tau_1 + \tau_2) Y_2 + Y_1 = F(t) . \quad (18)$$

Solving Equation 18 for the derivative of Y_2 ,

$$\frac{dY_2}{dt} = \left[F(t) - Y_1 - (\tau_1 + \tau_2) Y_2 \right] / \tau_1 \tau_2 . \quad (19)$$

Equations 16 and 19 are the two first order equations used to solve Equation 15.

The analog equivalent simulation package was first used to determine the impact of the time delay of the sampler. The gain change option was not utilized during the time delay evaluation period. The results with the 1-second time delay were similar to those obtained during the analog simulations. The best results were obtained with the integral gain between one and one-half to two times the proportional gain. The additional time delay did not seriously affect the stability of the control loop.

The effects of gain changing on the control loop varied. In some cases gain changing caused the control loop to oscillate when it had not done so before. In general, the ISE of the loop was not as good.

One other change was made that does not affect the control loop directly. Two sets of time constants and steady state gains were used in the mold response differential equation. The first set was used for heating and the second for cooling. The decision as to which set to use was made in the set point adjustment (SPADJ) subroutine. The two sets were used because experimental evidence indicated that significant differences existed between the observed heatup and cooldown responses. Experimental mold response data collected to that date³ were used in the simulations.

Several improvements were made in the computer program from the analog to the analog equivalent simulation package. Those improvements were primarily in the visibility of input information and run criteria data in the output. The two computer program source listings are presented in Appendices B and C.

The objectives of the first simulations were to determine the stable ranges of the PID constants and to evaluate the effects of additional delays on loop stability. The simulation packages performed that function prior to the availability of hardware. When hardware became available in January 1971, the results of the PID constant studies were used as starting points for loop tuning. When using the test molds (General K-2809 and K-2810), the PID constants, after tuning, were within one position (on the 10-position selector switches) of the settings predicted by the simulation program. The results were not as good when molds with other than nominal channeling configurations were actually used. In fact, the only consistency with other molds was the one and one-half to two times rule of thumb concerning the ratio of the integral to proportional gain.

The inconsistent results led to the analysis of the loop flow characteristics versus channel design and the impact of these characteristics on loop stability, as reported earlier.² The loop gain, which is nonlinear, is a function of the valve flow coefficients, valve stem position, the central system supply temperature, the return fluid temperature, and the loop flow rate.² The closed loop gain is the product of the loop gain and the proportional gain. The pole and zero location of the characteristic equation will shift as the gain product is changed. To illustrate the effect the loop gain has on the root location, the closed loop block diagram of Figure 6 will be assumed to represent the fluid loop. The closed loop transfer function is

$$\frac{T(S)}{R(S)} = \frac{C(S) G(S)}{1 + C(S) G(S)} , \quad (20)$$

Figure 6. Closed Loop Block Diagram, Root Locus Example

where

$T(S)$ = Fluid temperature,

$R(S)$ = Fluid temperature set point,

$C(S)$ = Controller dynamics, and

$G(S)$ = Fluid loop dynamics.

The characteristic equation for the closed loop is

$$1 + C(S) G(S) = 0 . \quad (21)$$

Substituting the transfer functions into Equation 21,

$$1 + \frac{K K_p (K_D S^2 + S + K_I)}{S(\tau S + 1)} = 0 . \quad (22)$$

Expansion of Equation 22 yields

$$S^2 + \frac{K K_p + 1}{K K_p K_D + \tau} S + \frac{K K_p K_I}{K K_p K_D + \tau} = 0 , \quad (23)$$

where

K = Loop gain,

K_p = Proportional gain,

K_I = Integral gain = $1/\tau_I$,

K_D = Derivative gain = τ_D , and

τ = Loop time constant.

The roots of Equation 23 will depend on the proportional gain-loop gain product ($K K_p$). To obtain roots identical to the roots of stable responses, assuming all other variables are constant, the proportional gain must decrease as the loop gain increases. The assumption that all other variables will be constant is not realistic, nor is the loop linear as illustrated, nor is the loop gain constant as assumed. However, the principle of the constant gain product holds.

The loop flow rate has the largest effect on loop gain.² It is not surprising from the preceding analysis that the discrepancies between the simulation results and the observed responses existed when molds with other than nominal channel configurations were used. At the time of the initial simulations, the loop flow relationship to loop gain was not known. It has since been demonstrated that when high process gains are encountered, caused by restricted flow, the proportional gain must be drastically reduced to achieve a stable response. Conversely, the addition of a parallel flow path to the restricted flow path, which increases the loop flow rate and decreases the loop gain, has also returned stability to previously unstable loops without seriously affecting heat transfer.

After the computer hardware was installed and checked out in June 1971, a program to simulate the direct digital control (DDC) loop was initiated.

Simulation of the Direct Digital Control Loop

The computer installed to control the fluid loop process has the characteristics listed below.

Core = 32,000 words,

Cycle time = 1.2 μ s,

Mass memory = 500,000 words (drum),

Word length = 16 usable bits + 1 parity bit + 1 memory protect bit,

Analog to digital converter = 12 bits,

Sample rate = 250 points per second,

Sample period = 1-second base or any multiple of 2 seconds,

Quasi Special Equipment (QSE) output = ± 10 -volt pulse maximum,

QSE output pulse duration = 5 milliseconds, and
 QSE gate pulse duration = 2 milliseconds.

Figure 7 illustrates the signal flow path for the automatic and computer modes of control. The automatic mode (analog) was described in an earlier section. Figure 7 illustrates the signal flow path when the Computer-Auto-Manual (CAM) station at the process site is in the Computer (C) mode. The 4- to 20-milliamp output of the millivolt-to-current converter is sent to the computer interface cabinet. At that location, it drops across a precision 8-ohm resistor generating a 32- to 160-millivolt signal. The analog-to-digital converter converts the 32- to 160-millivolt signal to a -50 to +50 percent of range number. The -50 to +50 percent of range represents a temperature range of 10 to 315°C (50 to 600°F).

The information is then stored in a special core resident table until it is required by the PID velocity control algorithm. The PID velocity algorithm computes a change in valve position according to Equation 24.⁶

$$\Delta m_n = K_p \left[\Delta e_n + \frac{K_I T}{60} e_n + \frac{K_D 60}{T} \Delta(\Delta e_n) \right], \quad (24)$$

where

Δm_n = Change in valve position (percent),

K_p = Proportional gain (percent/percent),

Δe_n = Change in error at n^{th} sample period (percent),

K_I = Integral gain (min^{-1}),

T = Sample period (s),

e_n = Error at n^{th} sample period (percent),

K_D = Derivative gain (min),

$\Delta(\Delta e_n)$ = Change in the change in error at n^{th} sample period, and

n = Current sample period.

Figure 7. Signal Flow of One Loop Under Computer Control

$$e_n = SP_n - PV_n , \quad (25)$$

$$\Delta e_n = e_n - e_{n-1} , \text{ and} \quad (26)$$

$$\Delta(\Delta e_n) = e_n - 2e_{n-1} + e_{n-2} , \quad (27)$$

where

SP_n = Set point at n^{th} sample period (percent),

PV_n = Controlled variable at n^{th} sample period (percent),

e_{n-1} = Error at $n^{\text{th}}-1$ sample period (percent), and

e_{n-2} = Error at $n^{\text{th}}-2$ sample period (percent).

The computed change in valve position is stored in a table accessed by the Quasi Special Equipment (QSE) device. That change in valve position will be between -50 and +50 percent. The QSE device uses the magnitude and sign of the computed change to generate a voltage pulse. If the computed valve change is between zero and +50 percent, the QSE will generate a pulse proportional to the computed valve change between zero and +10 volts. If the valve change is between zero and -50 percent, the QSE will generate a pulse proportional to the computed valve change between zero and -10 volts.

As an example, if the computed change in valve position is -30 percent, the voltage pulse generated would be -6 volts. The voltage pulse is sent out on a common line to all CAM stations. The correct CAM station will get a 2-millisecond gate pulse, which allows the voltage pulse to enter the Incremental Memory Module (IMM) circuit. The IMM circuit integrates the voltage pulse, causing a change in the 4- to 20-mA output of the CAM station. The gate pulse is only 2 milliseconds in duration, while the voltage pulse is 5 milliseconds. When the gate pulse drops out, the IMM circuit stops integrating the voltage pulse because the signal pulse is no longer there. Three milliseconds of the voltage pulse are currently not used.

The time constant of integration of the IMM is 125 milliseconds. The maximum change in output per sample period, as a result of the short gate pulse and time constant of integration, is 16 percent of range. By increasing the gate pulse to 5 milliseconds, the maximum output change per sample period is increased to

40 percent. The remaining 60 percent change must be obtained by reducing the integration time constant to 50 milliseconds. All the IMM devices currently utilize the 125-millisecond time constant, and the QSE is set to deliver 2-millisecond gate pulses.

The incremental memory module can be modeled as a zero order hold,⁶ plus a gain, which has the following transfer function:

$$\frac{M(S)}{V^*(S)} = \frac{K_{IMM} (1 - e^{-ST})}{S}, \quad (28)$$

where

$M(S)$ = IMM output (mA),

$V^*(S)$ = Voltage pulse (V),

K_{IMM} = IMM gain = 0.016 (mA/V),

T = Sample period = 2 (s), and

S = Laplace operator.

The output of the IMM, which is the same 4- to 20-milliamp output as that of the PID controller, is the input signal to the current-to-pressure converter (I/P). The I/P is the same device described in the model section.

For the purposes of computer simulation, the IMM was assumed to be instantaneous, since its time constant was so small compared to the time constant of the next smallest system. The increment size for numerical integration would have to be that small; otherwise, computer simulation would have become impractical from an execution time standpoint. A general rule of thumb for numerical integration step size selection is to divide the smallest time constant by 10. The time constants of the two smallest components are 3 seconds for the valve and 0.125 second for the IMM. Using the rule of thumb, if the IMM were not assumed to be instantaneous the simulation would have to be executed 80 times for every real-time second, compared to 3.3 times if it were assumed to be instantaneous. Consequently, the blocks labeled IMM in the simulation block diagrams are not shown as state variables.

Figure 8 illustrates in block diagram form the final evolution of the Direct Digital Control (DDC) model. Table 4 lists the parameters for the simulation. Refer to Appendix D for the computer program. The simulation was styled to emulate the purchased computer system and incorporates all the control options available as standard features for the DDC control algorithm. Those modes are the following:

Text continued on page 47.

Table 4. Parameter List for DDC Simulation, Revision 3, July 28, 1972

Parameter	Value	Description	Units
1	K_{AE}	Absolute Error Gain Multiplier	%/%
2	K_P	Controller Proportional Gain	%/%
3	K_I	Integral Gain	min ⁻¹
4	K_D	Derivative Gain	min
5	5.175 (0.75)	Current-to-Pressure Conversion	mA/kPa (mA/psi)
6	21 (3.0)	{ Pressure-to- Fraction Lift	kPa (psi)
7	41 (6.0)		kPa (psi)
8	62 (9.0)		kPa (psi)
9	-9.53×10^{-4} (-5.3×10^{-4})	{ Temperature to Specific Gravity	°C ⁻¹ (°F ⁻¹)
10	1.397 (1.414)		---
11	ΔP_H	Hot System Differential Pressure	kPa (psi)
12	ΔP_C	Cold System Differential Pressure	kPa (psi)
13	C_V	Valve Flow Coefficient	m ³ /s-kPa ^{-1/2} (gal/min-psi ^{-1/2})
14	-0.895 (-0.00461)	{ Temperature-to- Density Conversion	kg/m ³ -°C (lbm/gal-°F)
15	1377.53 (11.917)		kg/m ³ (lbm/gal)
16	7.5×10^{-4} (1.03×10^{-4})	{ Temperature-to- Enthalpy Conversion	kJ/kg-°C ² (Btu/lbm-°F ²)
17	1.134 (0.26287)		kJ/kg-°C (Btu/lbm-°F)
18	-0.284 (-8.686)		kJ/kg (Btu/lbm)

Table 4 Continued. Parameter List for DDC Simulation, Revision 3, July 28, 1972

Parameter	Value	Description	Units
19	-752.7 (-1276)	{ Enthalpy- to- Temperature Conversion	°C (°F)
20	1.28498 (0.0691)		(kJ/kg-°C) ² (Btu/lbm-°F) ²
21	-0.00302 (-0.000412)		kJ/kg-°C (Btu/lbm-°F)
22	0.2804 (8.686)		kJ/kg-°C ² (Btu/lbm-°F ²)
23	0.001506 (0.000206)		kJ/kg-°C ² (Btu/lbm-°F ²)
24	0.05229 (0.02909)	{ Temperature-to- Current Conversion Current to Voltage Millivolt to Percent of Range	mA/°C (mA/°F)
25	3.4771 (2.5454)		mA
26	8.0		mV/mA
27	0.7812		%/mV
28	-74.998		Percent
29	0.3268 (0.1818)	{ Temperature to Percent of Range	%/°C (%/°F)
30	-53.268 (-59.09)		Percent
31	0.46 (0.11)	Specific Heat of Steel	kJ/kg-°C (Btu/lbm-°F)
32	M	Mold Metal Mass	kg (lbm)
33	τ_v	Value Time Constant	s
34	τ_3	Thermocouple Time Constant	s
35	$K_{1,1}$	Heatup Metal Gain	°C/°C (°F/°F)
36	$K_{2,1}$	Cooldown Metal Gain	°C/°C (°F/°F)
37	$\tau_{1,2}$	First Heatup Time Constant	s
38	$\tau_{1,3}$	Second Heatup Time Constant	s

Table 4 Continued. Parameter List for DDC Simulation, Revision 3, July 28, 1972

Parameter	Value	Description	Units
39	$\tau_{2,2}$	First Cooldown Time Constant	s
40	$\tau_{2,3}$	Second Cooldown Time Constant	s
41	T	Computer Sample Period	s
42	Q_T	Total Loop Flow Rate	m ³ /s (gal/min)
43	K_A	Value of e_n Above Which $K_m > 1$	Percent
44	SP_i	Initial Loop Set Point	°C (°F)

Figure 8. DDC Simulation, Revision 3, July 27, 1972

- Gain multiplier on large error,
- Proportional cutout for the first sample period on set point changes,
- Use of process variable in the derivative mode instead of error, and
- Integral mode cutout when the error rate exceeds a specified amount.

Those modes were investigated to determine their impact on the control loop performance.

The major differences between the Analog Equivalent and the DDC simulations are the set point recipe, the controller, the delays added, the revised loop flow, and the valve flow characteristics.

A set point recipe routine and a set point adjustment routine were added to facilitate set point changes from run to run. The set point recipe routine reads in the set point recipe as it might be constructed by the engineer. The three variables read for each segment are ramp rate, set point, and dwell time. The set point adjustment routine determines what the set point will be each sample period in the same manner that the process control computer software determines the set point.

The controller was changed from a three-mode positional controller to a three-mode velocity digital algorithm. The switch from positional to velocity was necessary because the CAM hardware required the velocity form. The positional algorithm equivalent of the PID equation is⁶

$$m_n = K_p \left[e_n + \frac{K_I T}{60} \sum_{i=1}^n e_i + \frac{K_D 60}{T} \Delta e_n \right] + m_i, \quad (29)$$

where n is the current sample period. Equation 29 requires that initial controller output (m_i) be known when control is switched from Automatic to Computer mode. The velocity algorithm is obtained by subtracting the n th-1 form of Equation 29 from the n th form.⁶ The velocity form is restated for comparison purposes. The integration term (summation) and the initial controller output term (m_i) disappear in Equation 24.

$$\Delta m_n = K_p \left[\Delta e_n + \frac{K_I T}{60} e_n + \frac{K_D 60}{T} \Delta (\Delta e_n) \right], \quad (24)$$

Since there is no initial controller output (m_i) in Equation 24, no special provision must be made to supply the controller output to the algorithm when control is switched from Automatic to Computer.

Since there is no integration term in Equation 24, no special measures need be taken to protect against reset windup. A third feature incorporated in the control algorithm is the *bumpless transfer* feature, which means that no change in output occurs the first sample period after control is switched to Computer. No change in output forces the error in Equation 24 to be zero during the first sample period which means that the set point and process variable must be equal. When control is switched to Computer, the computer sets the set point equal to the process variable to achieve the zero error. The simulation program also does the same thing. Bumpless transfer prevents large perturbations caused by large errors from entering the control loop at the switchover time.

The switch symbols in Figure 8 were added to the model to indicate signal multiplexing. All within the bounds of the switches are functions carried out by the process control computer.

The time delays that were added were a result of the modeling activity associated with fluid flow analyses² and are pure transport delays. The location of the transport time delays plays an important part in the stability of the simulated loop.

The loop and valve flow characteristics were investigated extensively,² and the results were incorporated in the simulation program. Computer programs were written² to allow predicting the loop flow rates from channel configurations for both series and parallel channel designs. The predicted values of flow were used instead of the nominal design flow rate. The predicted loop flow rates were invariably lower in magnitude than nominal, resulting in higher process gains. Valve flow characteristics were assumed to be linear with valve stem lift in the first two simulations. Once hardware was on site, it was discovered that the valve flow characteristics were equal percentage and not linear. The fraction flow versus valve stem lift (L/Q_s) equation was subsequently changed to reflect the equal percentage characteristics. Equal percentage valves yield lower process gains during the first 50 percent of the valve lift than would linear valves in the fluid control loop.

The early DDC simulations, prior to revision 3, indicated unstable control when actual testing exhibited stable responses. Further investigation revealed that the instability was caused by misplacement of the delays and combining the delay blocks. When the transport delays were properly placed, the simulated control loop exhibited response characteristics very close to

those observed in field tests. It was not until the flow predictor computer programs² became available that acceptable correlation between test and simulation results was achieved.

Table 5 lists the results of a run made to evaluate the controller options. The integral of the squared error (ISE) and the integral of the absolute error (IAE) were used to judge performance (Equation 30).

$$IAE = \int_0^t |e(t)| dt . \quad (30)$$

The point at which both the IAE and ISE are minimum is run number 17 (Table 5). Control appears to improve by exercising the first three controller options versus standard control. Exercising the fourth option, integral cutout, appears to degrade control back to or beyond that achieved when using no options.

The ISE of the control loop decreases as the proportional (K_p) integral (K_I) gain product increases for the standard algorithms, with no options. Table 6 lists the controller gain product, the ISE, and the IAE for the standard algorithm from Table 5. The IAE of the loop also decreases with an increasing gain product up to a product of 40. Beyond this point, it again begins to increase.

A minimum ISE tends to minimize overshoot, but it does not penalize long settling times as does the IAE. The optimum controller settings would be where both the IAE and ISE were minimum. That point is at a gain product of 40. The integral to proportional gain ratio at that gain product was 1.6. All the integral to proportional gain ratios in Table 5 are between 1.5 and 2.5.

A random sample of the velocity algorithm PID constants currently in use on the cyclic zones of four-control packages was made. The PID constants, the PI gain product, and the gain ratio are presented in Table 7. The PI gain product and gain ratios are generally in the range indicated as best during simulation. The loop gain must also be considered along with the controller gain product, which is evident from Equation 23.

The pole location is affected by the loop gain as indicated in prior discussion. Drawing the same analogies, as the loop gain increases (loop flow reduced), the gain product must decrease to maintain the same pole location. The converse is also true, as the loop gain decreases (loop flow above nominal design), the gain product must increase to maintain identical pole location. The conclusion that can be made is that as long as the loop flow rate is close to the nominal design the controller gain product should be approximately 40 for the best results.

Table 5. Performance Results of DDC Simulation, August 17, 1972, General K-2810 Cavity Dynamics

Run	$K_m > 1$ When $ e_n > X$	Proportional Mode Equals Zero on Set Point. Changes First Period.	Process Variable Used in Derivative Mode	Integral Cutout When $ e_n > Y$	ISE $\times 10^5$	IAE $\times 10^3$	K_p (%/%)	K_I min^{-1}
1	No	No	No	No	1.35	4.92	3.0	5.0
2	Yes	Yes	Yes	No	0.98	4.46	3.0	5.0
3	Yes	Yes	Yes	Yes	1.11	4.85	3.0	5.0
4	No	No	No	No	1.00	4.12	6.0	10.0
5	Yes	Yes	Yes	No	0.93	4.07	6.0	10.0
6	Yes	Yes	Yes	Yes	1.06	4.46	6.0	10.0
7	No	No	No	No	1.29	4.71	2.5	6.0
8	Yes	Yes	Yes	No	0.97	4.39	2.5	6.0
9	Yes	Yes	Yes	Yes	1.07	4.68	2.5	6.0
10	No	No	No	No	1.84	7.19	2.0	4.0
11	Yes	Yes	Yes	No	1.27	6.23	2.0	4.0
12	Yes	Yes	Yes	Yes	1.35	6.50	2.0	4.0
13	No	No	No	No	1.03	3.77	4.0	8.0
14	Yes	Yes	Yes	No	0.93	3.89	4.0	8.0
15	Yes	Yes	Yes	Yes	1.05	4.24	4.0	8.0

Table 5 Continued. Performance Results of DDC Simulation, August, 17,
1972, General K-2810 Cavity Dynamics

Run	$K_m > 1$ When $ e_n > X$	Proportional Mode Equals Zero on Set Point. Changes First Period.	Process Variable Used in Derivative Mode	Integral Cutout When $ e_n > Y$	ISE $\times 10^5$	IAE $\times 10^3$	K_p (%/%)	K_I min^{-1}
16	No	No	No	No	1.03	3.68	5.0	8.0
17	Yes	Yes	Yes	No	0.92	3.70	5.0	8.0
18	Yes	Yes	Yes	Yes	1.06	4.06	5.0	8.0

K_m (%%) was 2 for all runs.

K_D (min) was 0.01 for all runs.

Table 6. Controller Gain Product Versus
IAE and ISE With No Options

Gain Product $K_p K_I$	ISE $\times 10^5$	IAE $\times 10^3$
8	1.84	7.19
15	1.29	4.71
32	1.03	3.77
40	1.03	3.68
60	1.00	4.12

Table 7. Random Sample of PID Velocity Algorithm
Constants in Use October 1, 1976

Package	Zone	K_p	K_I	K_D	$K_p K_I$	K_I/K_p
4	2	4.0	10.0	0.01	40	2.5
4	4	5.0	10.0	0.01	50	2.0
5	1	7.0	7.0	0.02	49	1.0
5	2	4.0	7.0	0.05	28	1.75
13	1	6.0	6.0	0.02	36	1.0
13	2	6.0	12.0	0.01	72	2.0
16	1	5.0	8.0	0.01	40	1.6
16	2	5.0	8.0	0.01	40	1.6

The process engineers do not attempt to tune the control loop to achieve the true optimum response. The IAE and ISE indices are not generated in the field for the process engineer to evaluate the responses that they see. The engineers tune the loop to achieve by visual inspection a response close to the quarter decay response. Any effort beyond visual checks would be wasteful, because the mold tends to filter out the undesirable response characteristics of near optimum PID settings. Compensation for mold temperature variations caused by less than optimum PID

constants can also be accomplished by adjusting the set point recipe. Large deviations from optimum PID constants, however, do not produce an acceptable mold response.

The primary objective for evolving the DDC simulation was to develop a mathematical model that would adequately depict the real physical mold temperature responses. With such a model, alternate direct metal temperature control schemes can be investigated off-line, requiring no real-time software development. The third iteration of the simulation provided that model.

An additional benefit derived was the capability to investigate the standard PID controller options and to determine their impact on the control loop performance. Two of the four options investigated, as listed in Table 5, are currently being used on all control loops. They are not using the proportional mode the first sample period after set point changes, and are using the process variable in the derivative mode. Both options were determined to be generally beneficial from simulations and were consequently tested and found desirable in the field.

Simulation of the Cascade Control Loop

Cascading the metal temperature and the fluid temperature was deemed the most likely conventional control scheme to succeed at closing the loop around the metal temperature. The general cascade configuration is shown in Figure 9, and the computer program is given in Appendix E. The cascade control configuration requires two transducers, two controllers, and two controlled variables. For cascade control to improve overall performance, the inner loop must be faster than the outer loop.⁶

The fluid and mold loops meet the basic requirements for response speed of the inner and outer loops. The mean ratio³ of the first order metal temperature time constant to fluid temperature time constant from available data is 15 with a standard deviation of 14. The inner loop is 15 times faster than the outer loop on the average.

The cascade system works in the following manner. The outer loop process variable is compared with the set point. The difference between desired and actual is used by the major loop controller to calculate the set point for the inner loop. The difference between the calculated set point and the actual inner loop value is used by the inner loop controller to force the inner loop value to the calculated set point. Any change in the inner loop value will cause a corresponding change in the outer loop value. Typically, the inner loop controller is a proportional or proportional plus integral controller.⁶ The outer loop is usually a proportional plus integral controller.⁶

Figure 9. Conventional Cascade Control Scheme Block Diagram

Figure 10 and Table 8 illustrate the cascade control model used for evaluation. The inner and outer loops are both shown as proportional-integral-derivative (PID) controllers. The inner loop was simulated, using the PID configuration since it had been tuned in the field for set point changes as a PID controller. The outer loop was simulated as a PI controller by setting K_D (P25) equal to zero. The computer program used to simulate the cascade system is given in Appendix E.

Late in 1972 field tests were conducted, using the cascade control configuration on a production mold, with poor dynamic response, and on the General K-2809 mold which had inverted thermocouples, resulting in rapid responses.³ The inverted thermocouples indicate the type of response possible if the air gaps can be eliminated from the mold design. The inverted thermocouple responses are two to three times faster than the production mold response times.

The results of one set of test data for each mold are listed in Table 9. The inner loop PID constants were the same for both molds. The fast responding General K-2809 mold has a higher damping ratio and natural frequency than the slower responding production mold. The damping ratio differences are the most significant, since the lower the ratio the more oscillatory the response. The production mold closed loop response exhibited excessive overshoot and settling characteristics for a general purpose control strategy.

For a control strategy to be acceptable, overshoot must be held within a $\pm 3^\circ$ Celsius band ($\pm 5^\circ$ F) of set point. For a set point of 176° C (350° F), the percent overshoot is restricted to 1.5 percent. The damping ratio corresponding to a 1.5-percent overshoot is 0.8, which means that the fastest response possible is a slightly underdamped response. Any overdamped response would be acceptable in terms of overshoot. However, the more overdamped the closed loop response, the longer the response takes to reach

Figure 10. Cascade Simulation; Revision 1, March 4, 1974

Table 8. Parameter List for Cascade Simulation, Revision 1, March 4, 1974

Parameter	Value	Description	Units
1	0.3268 (0.1818)	{ Temperature to Percent of Range	%/°C (%/°F)
2	-53.268 (-59.09)		Percent
3	$K_{M,1}$	Major Loop Absolute Error Gain	%/%
4	$K_{I,1}$	Major Loop Integral Gain	min ⁻¹
5	$K_{p,1}$	Major Loop Proportional Gain	%/%
6	$K_{M,2}$	Minor Loop Absolute Error Gain	%/%
7	$K_{p,2}$	Minor Loop Proportional Gain	%/%
8	$K_{I,2}$	Minor Loop Integral Gain	min ⁻¹
9	$K_{D,2}$	Minor Loop Derivative Gain	min
10	5.175 (0.75)	Current-to-Pressure Conversion	mA/kPa (mA/psi)
11	62 (9.0)	{ Pressure-to- Fraction Lift	kPa (psi)
12	41 (6.0)		kPa (psi)
13	-0.895 (-0.00461)	{ Temperature-to- Density Conversion	kg/m ³ -°C (lbm/gal-°F)
14	1377.53 (11.917)		kg/m ³ (lbm/gal)
15	7.5×10^{-4} (1.03×10^{-4})	{ Temperature-to- Enthalpy Conversion	kJ/kg-°C ² (Btu/lbm-°F ²)
16	1.134 (0.26287)		kJ/kg-°C (Btu/lbm-°F)
17	-0.284 (-3.686)		kJ/kg (Btu/lbm)

Table 8 Continued. Parameter List for Cascade Simulation, Revision 1, March 4, 1974

Parameter	Value	Description	Units
18	-752.7 (-1276.0)	{ Enthalpy- to- Temperature Conversion	$^{\circ}\text{C}$ ($^{\circ}\text{F}$)
19	1.28498 (0.0691)		$(\text{kJ/kg-}^{\circ}\text{C})^2$ $(\text{Btu/lbm-}^{\circ}\text{F})^2$
20	-0.00302 (-0.000412)		$\text{kJ/kg-}^{\circ}\text{C}$ $(\text{Btu/lbm-}^{\circ}\text{F})$
21	0.2804 (8.686)		$\text{kJ/kg-}^{\circ}\text{C}^2$ $(\text{Btu/lbm-}^{\circ}\text{F}^2)$
22	0.001506 (0.000206)		$\text{kJ/kg-}^{\circ}\text{C}^2$ $(\text{Btu/lbm-}^{\circ}\text{F}^2)$
23	301 (575)	Hot Supply Temperature	$^{\circ}\text{C}$ ($^{\circ}\text{F}$)
24	32 (90)	Cold Supply Temperature	$^{\circ}\text{C}$ ($^{\circ}\text{F}$)
25	$K_{D,1}$	Major Loop Derivative Gain	min
26	M	Mold Metal Mass	kg (lbm)
27	0.46 (0.11)	Specific Heat of Steel	$\text{kJ/kg-}^{\circ}\text{C}$ $(\text{Btu/lbm-}^{\circ}\text{F})$
28	τ_{TC}	Thermocouple Time Constant	s
29	0.05229 (0.02909)	{ Temperature-to- Current Conversion	$\text{mA}/^{\circ}\text{C}$ $(\text{mA}/^{\circ}\text{F})$
30	3.4771 (2.5454)		mA
31	8.0	Current-to-Voltage Conversion	mV/mA
32	0.7812	{ Millivolt to Percent of Range	%/mV
33	-74.998		Percent
34	τ_V	Value Time Constant	s
35	T	Computer Sample Period	s
36	Q_T	Total Loop Flow Rate	m^3/s (gal/min)
37	C_V	Equivalent Valve Coefficient	$\text{m}^3/\text{s-kPa}^{-1/2}$ $(\text{gal/min-psi}^{-1/2})$

Table 8 Continued. Parameter List for Cascade Simulation, Revision 1, March 4, 1974

Parameter	Value	Description	Units
38	ΔP_H	Hot System Differential Pressure	kPa (psi)
39	ΔP_C	Cold System Differential Pressure	kPa (psi)
40	$K_{1,e}$	Value of $e_{1,n}$ Above Which $K_{m,1} > 1$	---
41	$K_{2,e}$	Value of $e_{2,n}$ Above Which $K_{m,2} > 1$	---
42	SP_i	Initial System Set Point	°C (°F)
43	$m_{1,n-1}$	Initial Major Loop Controller Output	Percent
44	$K_{m,H}$	Metal Heatup Gain	°C/°C (°F/°F)
45	$\tau_{1,H}$	First Heatup Time Constant	s
46	$\tau_{2,H}$	Second Heatup Time Constant	s
47	$K_{m,C}$	Metal Cooldown Gain	°C/°C (°F/°F)
48	$\tau_{1,C}$	First Cooldown Time Constant	s
49	$\tau_{2,C}$	Second Cooldown Time Constant	s
50	-9.53×10^{-4} (-5.3×10^{-4})	Temperature to	°C ⁻¹ (°F ⁻¹)
51	1.397 (1.414)	Specific Gravity	---
52	$Q_{h,v}$	Maximum Hot Valve Flow Rate	m ³ /s (gal/min)
53	$Q_{c,v}$	Maximum Cold Valve Flow Rate	m ³ /s (gal/min)

Table 9. Cascade Test Results of Modeling the Closed Loop Responses as Second Order Underdamped

Mold	Mold Half	Heat or Cool	K _p (%)	K _I (min ⁻¹)	Percent Overshoot	Time To Peak (s)	Damping Ratio	Natural Frequency (rad/sec x 10 ⁻³)
2809	Punch	Heat	0.25	4.0	7	266	0.645	15.40
2809	Punch	Cool	0.25	4.0	2	243	0.780	20.60
Prod*	Punch	Heat	0.25	4.0	17	672	0.491	5.37
Prod*	Punch	Cool	0.25	4.0	8	1220	0.627	3.30
2809	Cavity	Heat	0.35	4.0	1	349	0.826	16.0
2809	Cavity	Cool	0.35	4.0	3	348	0.745	13.50
Prod	Cavity	Heat	0.35	4.0	20	510	0.456	6.92
Prod	Cavity	Cool	0.35	4.0	11	1646	0.575	2.33
2809 Simu**	Cavity	Heat	0.35	5.0	12	225†	0.559	16.8
2809 Simu**	Cavity	Cool	0.35	5.0	25	345†	0.404	10.0
2809	Cavity	Heat	0.35	5.0	8	204	0.627	19.8
2809	Cavity	Cool	0.35	5.0	15	192	0.517	19.1

*Production Mold

**Simulation Data

†Minutes

set point. Production time is sacrificed as the response time lengthens. The production mold which had a slower open loop response exhibited a more underdamped closed loop response than did the test mold for identical controller settings. That result indicates that as the time constant of the mold increases the controller constants must be reduced to achieve a more damped response.

In an attempt to satisfy the 1.5-percent overshoot criterion, the set point recipe was altered. The initial step size was reduced from 176 to 149°C (350 to 300°F). As the initial response reached its peak overshoot to the initial set point change, the set point was then stepped again to 176°C. Figure 11 illustrates the technique. The results of that effort were satisfactory in that success in reducing overshoot of the desired cure temperature to 1.5 percent was achieved. The employee hours required to achieve success by using the cascade configuration were comparable to the time required to generate an equivalent fluid temperature recipe. Since the development times were equivalent, the only advantage of cascade control is direct control of metal drift during cure.

The simulation results of one run are also listed in Table 9. The differences between the simulated responses and the actual responses are a result of differences between the actual fluid response and the simulated fluid response. For reasons unknown, the simulated fluid response became unstable, indicating that the process gain of the simulated fluid loop was excessively high. Three variables could cause an excessively high gain:

- Loop flow rate too low,
- Incoming flow rate too high, and
- Erratic return fluid temperature.

For the inner loop response to be unstable, the roots of the characteristic equation, as described by Equation 23, must be on the right-hand side of the imaginary axis. For that to be the case, the process gain had to be too large.

Even though the inner loop instabilities existed during simulation, the trends indicated during simulation followed the trends observed in field tests. Specifically, as the mold temperature time constant was increased, the percentage overshoot increased and the natural frequency decreased. The conclusion that can be drawn from simulation is that unless the mold design is such that the response speed can be decreased to the minimum (no air gaps) cascade control will not satisfy the objective of the project. Simulation and field tests confirm that as the open loop response of the mold is increased the closed loop response becomes increasingly underdamped.

Figure 11. Set Point Adjusting Technique to Achieve a 1.5-Percent Maximum Overshoot.

In the past, it was believed that dead time was the major contributor to overshoot and oscillatory behavior of metal temperature feedback. It would appear from cascade simulation that the lag time of the response contributes heavily to the degree of closed loop damping. As the dead time increases, more phase shift results which forces the closed loop gain to be smaller to maintain a stable nonoscillatory response. Phase shifts also result from increasing time constants, creating the same phenomenon as increasing dead time.

The cascade simulation effort indicated that the dynamic response of the mold was crucial to the success of the cascade control strategy as a viable means to closed loop metal temperature control. There are so many variables that affect the dynamic response of the mold that achieving consistent mold response is not practical. A mold design program was started in the late 1960's in an attempt to achieve consistent designs for similar part configurations. An insert concept was formulated, using general purpose mold bases. Past experience has shown that even when molding parts of similar designs, using mold inserts of similar designs and mold bases of the same design, or in some cases the same base, the dynamic responses are not the same. If an insert is removed from the base and then replaced later, the dynamic response will often change. The cause of the inconsistency is the contact resistance between the insert and mold base³ (air gap).

It was decided from the results of the cascade tests and simulations that conventional PID algorithms would not provide a consistent general purpose means for direct metal temperature control. Even with dead time compensation techniques, such as the *Smith Predictor*,⁶ in conjunction with PID algorithms, the inconsistent lag times of the molds would give varying closed loop responses, making the control strategy marginally effective.

In late 1974 and early 1975, a program was initiated to develop a control strategy that uses the actual mold dynamics as part of the algorithm. The control strategy would be developed, using Z transforms and a math model of the dynamic responses of the control loop.

ACCOMPLISHMENTS

Math models were developed to allow off-line simulation of proposed closed loop metal temperature control strategies. Proportional cutout on set point change, and process variable in the derivative mode, were instituted as standard features of the current fluid loop algorithms as a result of simulation. Simulation activity, using the cascade control strategy, indicated that the closed loop response was highly dependent on the magnitude of the lag time. The simulations indicate that the closed loop response becomes more underdamped as the mold lag time increases. The specification for a satisfactory control system requires that the closed loop response be slightly underdamped to overdamped. The cascade system does not guarantee an overdamped system. Only one of the two molds tested could be tuned to produce an overdamped response.

FUTURE WORK

A program has been started to develop a control algorithm which uses the actual mold dynamics to replace the standard PID algorithm. That algorithm will be developed, using Z transforms and classical methods for developing control algorithms for sampled data systems. The specifications for the closed loop response are a part of the algorithm development procedure which tailors the algorithm to the desired response.

REFERENCES

¹R. H. Floersch, *Theory and Operation of the Bendix Mold Heating and Cooling Equipment* (Interim Report). UNCLASSIFIED. Bendix Kansas City: BDX-613-369, June, 1971 (Available from NTIS).

²R. H. Floersch, *Fluid Flow Analysis in the Mold Heating and Cooling Control Loop* (Interim Report). UNCLASSIFIED. Bendix Kansas City: BDX-613-742, September, 1972 (Available from NTIS).

³R. H. Floersch, *Experimental Testing for Mold Temperature Responses* (Interim Report). UNCLASSIFIED. Bendix Kansas City: BDX-613-111 (Rev.), May, 1975 (Available from NTIS).

⁴Moore, Smith, and Murrill, "Simplifying Digital Control Dynamics for Controller Tuning and Hardware Lag Effects," *Instrument Practice*, January, 1969.

⁵R. C. Dorf, *Modern Control Systems*. Reading, Mass: Addison-Wesley Publishing Co., 1967.

⁶C. L. Smith, *Digital Computer Process Control*. New York: International Textbook Company, Intext Educational Publishers, 1972.

Appendix A

EQUIVALENT VALVE FLOW ANALYSIS

Derived January, 1973

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix A

EQUIVALENT VALVE FLOW ANALYSIS

In the Bendix mold heating and cooling control loop the flow into the loop must pass through the supply valve, while the fluid leaving the loop must pass through the three-way return valve. The driving force to push the fluid through the valves is supplied by the central system differential pressure control. The control loop is a constant volume loop requiring that the amount of fluid entering must equal the amount leaving.

The fluid flowing through the two valves is flowing at the same rate and is being forced by the same pressure drop. Figure A-1 illustrates the effective flow network.

The equation for flow through a fully open valve is

$$Q = C_v \sqrt{\frac{\Delta P}{\rho}}, \quad (A-1)$$

where

Q = Flow rate (m^3/s) (gal/min),

C_v = Flow coefficient ($m^3/s-kPa^{-1/2}$) (gal/min-psi $^{-1/2}$),

ΔP = Pressure drop across valve kPa (psi), and

ρ = Fluid specific gravity.

Since the supply and return valves receive the same signal and are stroked the same way, their stem travels will be assumed identical and fully open. The flow through each valve is

$$Q_o = C_{v1} \sqrt{\frac{P_s - P_I}{\rho}}, \text{ and} \quad (A-2)$$

$$Q_o = C_{v2} \sqrt{\frac{P_I - P_r}{\rho}}. \quad (A-3)$$

The flow rates (Q_o) in Equations A-2 and A-3 are identical and may be equated as follows:

$$Q_o = C_{v1} \sqrt{\frac{P_s - P_I}{\rho}} = C_{v2} \sqrt{\frac{P_I - P_r}{\rho}}. \quad (A-4)$$

Figure A-1. Effective Flow Network Supply and Return Valves for Mold Heating and Cooling Control Loop

Solving Equation A-4 for the intermediate pressure P_I ,

$$P_I = \frac{P_s C_{v1}^2 + P_r C_{v2}^2}{C_{v1}^2 + C_{v2}^2} \quad (A-5)$$

Substituting Equation A-5 into A-2, squaring and reducing yields,

$$Q_o^2 = \frac{C_{v1}^2 C_{v2}^2}{C_{v1}^2 + C_{v2}^2} \frac{P_s - P_r}{\rho} \quad (A-6)$$

If the flow network of Figure A-1 were equated to a flow network with a single equivalent valve as in Figure A-2, the flow equation for the single valve network would be

$$Q_o^2 = C_{veq}^2 \frac{P_s - P_r}{\rho} \quad (A-7)$$

Since flow rates of Equations A-6 and A-7 are equal, the two equations may be equated as follows:

$$C_{veq}^2 \frac{P_s - P_r}{\rho} = \frac{C_{v1}^2 C_{v2}^2}{C_{v1}^2 + C_{v2}^2} \frac{P_s - P_r}{\rho}, \text{ and} \quad (A-8)$$

$$C_{veq} = \sqrt{\frac{C_{v1}^2 C_{v2}^2}{C_{v1}^2 + C_{v2}^2}} = \frac{C_{v1} C_{v2}}{\sqrt{C_{v1}^2 + C_{v2}^2}} \quad (A-9)$$

Figure A-2. Equivalent Flow Network

The supply and return valve flow characteristics can be calculated, using Equation A-7 with the equivalent coefficient of Equation A-9.

If the two valve coefficients of Figure A-1 are equal, Equation A-9 reduces to

$$C_{veq} = \frac{C_{v1}}{\sqrt{2}} . \quad (A-10)$$

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix B

COMPUTER PROGRAM FOR AN ANALOG CONTROL LOOP SIMULATION

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

PROGRAM DYSIMP(INPUT,OUTPUT,TAPE1=INPUT,TAPE6=OUTPUT)

```

C THIS PROGRAM IS A SIMULATION OF THE BENDIX MOLD HEATING AND
C COOLING SYSTEM IT REPRESENTS ONE LOOP OF A TWO ZONE SYSTEM
C UNDER ANALOG CONTROL
C THIS IS THE MAIN PROGRAM AND CONTROLS ALL SUBROUTINES THE FIRST
C SUBROUTINE CALLED IS INPUT NO2 IS INTFUN NO3 IS INTCAL NO4 IS
C DERFUN NO5 IS RKINT NO6 IS INTCAL NO5 AND NO6 ARE IN A DO LOOP
C NOS 7AND8 PRTTBL NOS 9 AND10 ARE PRTPLT AND NO 11 IS PARADJ
C INPUT -READS THE MODEL DATA-INITIAL X S Y S P S H S AND PRINTS ETC
C INTFUN-INITIALIZES DELAY BLOCKS AND IMPLICIT VARIABLES
C INTCAL-CONTAINS INTERMEDIATE EQUATIONS
C DERFUN-CONTAINS DIFFERENTIAL EQUATIONS
C RKINT -RUNGE KUTTA FOURTH ORDER INTEGRATION ROUTINE
C PRTTBL-PRINTS THE X AND Y ARRAYS IN TABULAR FORM
C PRTPLT-PLOTS THE X AND Y ARRAYS VERSUS TIME
C PARADJ-CHANGES MODEL PARAMETERS AND RERUNS THE PROGRAM
C OTHER PROGRAM SUBROUTINES ARE
C DELAY -SETS UP AND KEEPS TRACK OF TIME DELAYS
C SPADJ -CHANGES MODEL SET POINTS BASED ON TIME
C PLOT -DOES THE SCALING AND PLOTTING
C POT -IS USED BY PLOT
C
  DIMENSION NDATA(8)
  DIMENSION YSTOR(100,42),XSTOR(100,50)
  COMMON/CUNINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
  COMMON Y(42)
  COMMON/PARAM/P(50),H(8),NSYS
  COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)
  COMMON/OUTCON/NGY(42),NTY(42),NGX(50),NTX(50),IADJ
  DATA NR,NW/1,6/
  DATA IBUG/0/
13 READ(NR,1)(NDATA(I),I=1,8)
  1 FORMAT(8( 9X ,I1))
  NSUM = 0
  DO 4 I=1,8
  4 NSUM = NSUM + NDATA(I)
  IF(NSUM.EQ.0) GO TO 999
  CALL INPUT(NDATA)
  IF(NUMSIG.EQ.0) GO TO 70
  CALL INTFUN
70 IF(NX.EQ.0) GO TO 80
  CALL INTCAL
80 CALL DERFUN
  NSTEP = (ENDTIM-BEGTIM)/DELT
  IF((NSTEP/NWRITE).LT.100)GO TO 2
22 IF((100*NWRITE) .GT. NSTEP) GO TO 23
  NWRITE=NWRITE+1
  GO TO 22
23 WRITE(NW,3)NWRITE
  3 FORMAT(10X,47HNUMBER OF OUTPUTS EXCEEDS 300, NWRITE RESET TO
  1 15,/)
  2 IPAGE = 1
  IF((NSTEP/NWRITE).GT.200) IPAGE = 2
  JI = 1
  ITY=ITX=IGY=IGX=0
  NTMAX = 2*NY + 2

```

```

DO 64 I=1,NTMAX
  IF(NGY(I).NE.0) IGY = IGY+1
  IF(NTY(I).NE.0) ITY = ITY+1
64 YSTOR(JI,I) = Y(I)
  DO 65 I=1,NX
    IF(NTX(I).NE.0) ITX = ITX+1
    IF(NGX(I).NE.0) IGX = IGX+1
65 XSTOR(JI,I) = X(I)
902 FORMAT(8(3X,F12.5))
  WRITE(6,902)Y(NSYS+1)
  WRITE(6,902)(Y(I),I=10,NY)
  WRITE(6,902)(X(I),I=18,NX)
  NI = 0
  DO 60 KK = 1,NSTEP
    NI = NI+1
    CALL RKINT(KK,NY)
    IF(NX.EQ.0) GO TO 51
    CALL INTCAL
51 IF(NI.NE.NWRITE) GO TO 60
    NI = 0
    IF(IRUG.EQ.0) GO TO 444
901 FORMAT(8(2X,E12.3))
    WRITE(6,901)Y(NSYS+1)
    WRITE(6,901)(Y(I),I=10,15)
    WRITE(6,901)(X(I),I=18,43)
444 JI = JI + 1
    DO 66 I = 1,NTMAX
      66 YSTOR(JI,I) = Y(I)
      DO 67 I=1,NX
        67 XSTOR(JI,I) = X(I)
60 CONTINUE
    IF(ITY.EQ.0) GO TO 10
    K = 1
61 CALL PRTTBL(YSTOR,100,42,0,JI,NTY(1),NTY(K+1),NTY(K+2),NTY(K+3),
1 ,NTY(K+4),NTY(K+5),NTY(K+6),NTY(K+7))
    K = K + 8
    IF(K.GT.ITY) GO TO 10
    K=K-1
    GO TO 61
10 IF(ITX.EQ.0) GO TO 11
    K = 1
62 CALL PRTTBL(XSTOR,100,50,1,JI,NTX(1),NTX(K+1),NTX(K+2),NTX(K+3),
1 ,NTX(K+4),NTX(K+5),NTX(K+6),NTX(K+7))
    K = K + 8
    IF(K.GT.ITX) GO TO 11
    K=K-1
    GO TO 62
11 IF(IGY.EQ.0) GO TO 12
    K = 1
63 CALL PRTPLT(YSTOR,100,42,0,JI,IPAGE,NGY(1),NGY(K+1),NGY(K+2),
1 ,NGY(K+3),NGY(K+4))
    K = K + 4
    IF(K.GE.IGY) GO TO 12
    GO TO 63
12 IF(IGX.EQ.0) GO TO 14

```

```

      K = 1
59 CALL PRTPLT(XSTOR,100,50,1,JI,IPAGE,NGX(1),NGX(K+1),NGX(K+2),
      1  NGX(K+3),NGX(K+4))
      K = K + 4
      IF(K.GE.IGX) GO TO 14
      GO TO 59
14 IF(IADJ.NE.0) CALL PARADJ
      GO TO 13
999 STOP
      END
      SUBROUTINE SPADJ(T,SPT,SPDT)
C      THIS SUBROUTINE ADJUSTS THE MODEL SET POINT BASED ON TIME
C      THIS ROUTINE IS CALLED FROM INTCAL
      COMMON/PARAM/P(50),H(8),NSYS
      DATA T1,T2,T3,T4,T5,T6/180.,500.,1000.,1500.,1750.,1900./
      DATA TE1,TE2,TE3,TE4,TE5,TE6/150.,250.,350.,200.,275.,150./
      IF(T.LT.T1) SPT=TE1
      IF(T.GE.T1) SPT=TE2
      IF(T.GE.T2) SPT=TE3
      IF(T.GE.T3) SPT=TE4
      IF(T.GE.T4) SPT=TE5
      IF(T.GE.T5) SPT=TE6
      SPDT=0.0
      SPT=SPT*P(47)+P(48)
      RETURN
      END
      SUBROUTINE DERFUN
C      THIS SUBROUTINE CONTAINS ALL DIFFERENTIAL EQUATIONS IN THE MODEL
C      IN THEIR DIFFERENTIATED FORM EACH F IS THE TIME RATE OF CHANGE OF
C      A STATE VARIABLE Y THE COMPUTED VALUE OF EACH F IS PLACED IN A
C      Y ARRAY STARTING WITH THE NUMBER OF STATE EQUATIONS PLUS 2 TO BE
C      USED BY THE RUNGE KUTTA INTEGRATION ROUTINE THIS SUBROUTINE IS
C      CALLED FROM RKINT AND DYSIMP
      COMMON Y(42)
      COMMON/PARAM/P(50),H(8),NSYS
      COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)
      DIMENSION F(20)
C      PLACE DIFFERENTIAL EQUATIONS AFTER THIS CARD
      F(1) = (X(8)-Y(1))/P(5)
      F(2) = (X(3)-Y(2))/P(5)
      F(3) = Y(4)
      F(4) = (P(12)*X(6)-Y(3)-(P(10)+P(11))*Y(4))/(P(10)*P(11))
      F(8) = 0.0
      F(7) = P(14)*(P(15)-Y(7))
      F(5) = P(2)*(F(8)-F(1))+P(1)*(Y(8)-Y(1))
      F(6) = P(8)*X(13)
      F(9) = X(44)*X(44)
      F(10) = 0.0
      F(11) = X(18)*P(19)*P(18)/60.
      F(12) = (X(37)*P(41)-Y(12))/P(42)
      F(13) = (X(37)*P(45)-Y(13))/P(46)
      F(14) = (X(23)-Y(14))/P(49)
      F(15) = (X(45)-Y(15))/P(49)
      DO 1 II=1,NSYS
      IEQ = NSYS+2+II

```

```
1 Y(IEQ) = F(II)
```

```
RETURN
```

```
END
```

```
SUBROUTINE INTCAL
```

```
C THIS SUBROUTINE IS THE INTERMEDIATE EQUATION SUBROUTINE AND MUST  
C CONTAIN ALL INTERMEDIATE CALCULATION EQUATIONS
```

```
C ALL XT(I) VARIABLES ARE THE INPUTS TO A TIME DELAY BLOCK
```

```
C THE VARIABLE XTH(I) ARE THE OUTPUTS FROM THE TIME DELAY BLOCKS
```

```
C THIS SBUROUTINE IS CALLED FROM DYSIMP AND CALLS SPADJ AND DELAY  
COMMON Y(42)
```

```
COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)
```

```
COMMON/PARAM/P(50),H(8),NSYS
```

```
DATA L01,L02,L03,L04,L05/10,11,12,13,14/
```

```
X(1)=Y(NSYS+1)
```

```
X(3)=Y(7)
```

```
X(11)= Y(8)
```

```
XT(1)= P(9)*Y(6)
```

```
XT(2)= X(3)
```

```
XT(3)= XTH(2)
```

```
XT(4)= Y(3)+XTH(3)
```

```
XT(5)= Y(L03)
```

```
CALL DELAY
```

```
Y(15) = OUTLIM(Y(15),0.0,100.0)
```

```
Y(14) = OUTLIM(Y(14),0.0,100.0)
```

```
Y(11) = OUTLIM(Y(11),4.0,20.0)
```

```
X(42)=Y(L04)
```

```
X(43)=P(47)*X(42)+P(48)
```

```
IF(X(43) .LT. 4.0) X(43)=4.0
```

```
IF(X(43) .GT. 20.0) X(43)=20.0
```

```
CALL SPADJ(X(1),Y(10),F10DT)
```

```
X(18)=Y(L01)-X(43)
```

```
X(19)=P(18)*X(18)
```

```
X(20)=P(20)*P(18)*60.*( F10DT -P(47)*(X(37)*P(45)-Y(L04))/P(46))
```

```
X(21)=X(20)+X(19)+Y(L02)
```

```
X(21)=OUTLIM(X(21),4.0,20.0)
```

```
X(22)=P(21)*X(21)+P(22)
```

```
IF(X(22) .GT. 15.0) X(22)=15.0
```

```
IF(X(22) .LT. 0.0) X(22)=0.0
```

```
X(23)=(X(22)-P(23))*P(25)/P(24)
```

```
X(45)=(P(23)-X(22))*P(25)/P(23)
```

```
X(45)=OUTLIM(X(45),0.0,100.0)
```

```
X(23)=OUTLIM(X(23),0.0,100.0)
```

```
X(24)=.8*Y(14)/100.
```

```
X(46)=.8*Y(15)/100.
```

```
X(46)=OUTLIM(X(46),0.0,0.8)
```

```
X(24)=OUTLIM(X(24),0.0,0.8)
```

```
X(28)=P(28)*X(46)
```

```
X(26)=P(28)-X(25)-X(28)
```

```
X(25)=P(28)*X(24)
```

```
X(27)=X(26)*(P(29)*X(41)+P(30))
```

```
X(31)=100.
```

```
X(32) = 475.0
```

```
X(29)=X(25)*(P(31)*X(32)+P(32))
```

```
X(47)= X(28)*(P(31)*X(31)+P(32))
```

```
X(30) = X(27)*X(40)
```

```

X(48)=P(33)*X(31)*X(31)+P(34)*X(31)+P(35)
X(33)=P(33)*X(32)*X(32)+P(34)*X(32)+P(35)
X(49)=X(48)*X(47)
X(34)=X(29)*X(33)
X(35)=X(34) + X(30)+X(49)
X(36)= X(35)/(X(27)+X(29)+X(47))
X(37)=P(36)+(SQRT(P(37)+(P(38)*(P(39)+X(36))))/P(40))
X(38)=Y(L03)
X(39) = XTH(5)
X(40)=X(36)-(P(43)*P(44)*((X(37)*P(41)-Y(L03))/P(42))/(X(27)+X(29)
1 +X(47)))
X(41)=P(36)+(SQRT(P(37)+(P(38)*(P(39)+X(40))))/P(40))
X(4)=XTH(2)
X(6)=XTH(1)
X(5)=XTH(3)
X(8)=XTH(4)
X(9)= Y(5)-Y(2)
Y10=(X(8)-Y(1))/P(5)
X100=X(15)-Y10
X(10) = X(11)-Y(1)
Y50=P(2)*X100+ P(1)*X(10)
X90=Y50-Y10
X(15) = P(3)*X(9)+ P(4)*X90
X(2) = P(7) * X(15)
X(12) = X(2)-Y(6)
IF(X(12).GT.0.0) X(13)=1.0
IF(X(12).LT.0.0) X(13)=-1.0
IF(ABS(X(12)).LE.P(6)) X(13)=0.0
X(7)= X(5)+ Y(3)
X(14)= P(9)*Y(6)
X(44) = (Y(L01) - P(48))/P(47) -X(37)
IF(X(14).GT.(1.-P(13))) GO TO 60
IF(X(14).LT.(-P(13))) GO TO 61
RETURN
60 X(14)=1.0-P(13)
Y(6)=X(14)*90.0
RETURN
61 X(14)=-P(13)
Y(6)= X(14)*90.0
RETURN
END
SUBROUTINE INTFUN
C THIS SUBROUTINE IS THE INITIAL FUNCTION SUBROUTINE AND IS CALLED
C FROM DYSIMP ONLY ONCE THE SUBROUTINE IS USED TO INITIALIZE
C IMPLICIT VARIABLES AND TO DETERMINE THE SIZE OF AND INITIALIZE THE
C DELAY BLOCKS
COMMON Y(42)
COMMON/PARAM/P(50),H(8),NSYS
COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)
DATA L01,L02,L03,L04,L05/10,11,12,13,14/
IMAX = 0
DO 2 I=1,NUMSIG
IH(I) = H(I)/Y(NSYS+2) + 1
IF(IH(I).GE.IMAX) IMAX = IH(I)
2 CONTINUE

```

```

DO 3 KK=1,IMAX
XSTR(1,KK)=0.0
XSTR(2,KK)=Y(7)
XSTR(3,KK)=Y(8)
XSTR(5,KK)=Y(L03)/P(41)
3 XSTR(4,KK)=Y(8)
DO 4 I=1,NUMSIG
4 XTH(I)=XSTR(I,IMAX)
X(37)=Y(L03)/P(41)
X(41)=Y(L03)*P(41)
X(42)=P(33)*X(41)*X(41)+P(34)*X(41)+P(35)
Y(L01)=Y(L01)*P(47)+P(48)
TEMP = Y(L03)/P(41)
X(40)= P(33)*TEMP*TEMP +P(34)*TEMP + P(35)
RETURN
END

```

SUBROUTINE DELAY

C THIS SUBROUTINE IS THE DELAY SUBROUTINE THE METHOD IS LIKE A PIPE
C THE INPUTS GO IN THE FRONT AND OUTPUTS AFTER A DELAY COME OUT THE
C BACK THE INPUTS ARE XSTR AND THE OUTPUTS XTH
C THIS SUBROUTINE IS CALLED FROM INTCAL

COMMON Y(42)

COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)

COMMON/PARAM/P(50),H(8),NSYS

ISTOP = IMAX -1

DO 3 KK =1,ISTOP

IP = IMAX -KK

DO 3 II = 1,NUMSIG

3 XSTR(II,IP+1)=XSTR(II,IP)

DO 4 II=1,NUMSIG

JJ = IH(II)

XTH(II)=XSTR(II,JJ)

4 XSTR(II,1)=XT(II)

RETURN

END

SUBROUTINE PRTPLOT(STRMAT,NROW,NCLM,IVAR,MAXLIN,NPAGE,NAXES,NOR1,
1 NOR2,NOR3,NOR4)

C THIS SUBROUTINE SETS UP THE X AND Y ARRAYS FOR THE PLOTTING

C SUBROUTINE IT IS CALLED FROM DYSIMP AND CALLS PLOT

DIMENSION STRMAT(NROW,NCLM),NPLT(4),NP(5),SYM(2),X(3000),Y(3000)

DIMENSION Z(3000)

DATA NW/6/

DATA SYM,PARL,PARR/1HY,1HX,1H(.1H)/

IF(NAXES.EQ.0) RETURN

NPLT(1)=NOR1

NPLT(2)=NOR2

NPLT(3)=NOR3

NPLT(4)=NOR4

NPLT = 0

IFM =1

IF(IVAR.GT.0) IFM= 2

DO 1 I=1,4

IF(NPLT(I).EQ.0) GO TO 1

NPLT= NPLT +1

NP(NPLT)=NPLT(I)

1 CONTINUE

NPT = MAXLIN*NPLOT

DO 2 II =1,MAXLIN

DO 2 IC =1,NPLOT

KK = NP(IC)

NCRV = (IC-1)*MAXLIN + II

X(NCRV) = STRMAT(II,NAXES)

Y(NCRV)=STRMAT(II,KK)

2 CONTINUE

CALL PLOT(X,0,Y,0,Z,0,NPT,NPLOT,1,0,2,0,NPAGE)

WRITE(NW,3)((SYM(IFM),PARL,NP(IC),PARR,SYM(IFM),PARL,NAXES,

1 PARR) ,IC=1,NPLOT)

3 FORMAT(/,38H THE ABOVE PLOTS, RESPECTIVELY, ARE ,4(/,38X,A1,A1,I
12, A1,8H VS ,A1,A1,I2,A1))

RETURN

END

FUNCTION OUTLIM(XIN,BOT,TOP)

OUTLIM = XIN

IF(XIN.GT.TOP) OUTLIM = TOP

IF(XIN.LT.BOT) OUTLIM = BOT

RETURN

END

SUBROUTINE PRITBL(STRMAT,NROW,NCLM,IVAR,MAXLIN,NC1,NC2,NC3,NC4,

1 NC5,NC6,NC7,NC8)

C THIS SUBROUTINE TABULATES THE X AND Y ARRAYS FOR OUTPUT

C THIS ROUTINE IS CALLED FROM DYSIMP

DIMENSION STRMAT(NROW,NCLM),PRTARY(100,8),NPRT(8),SYM(2)

DATA SYM,PARL,PARR/1HY,1HX,1H(,1H)/

DATA NW/6/

333 FORMAT(1H1,8H PAGE ,I2,/,8(7XA1,A1,I2,A1,2X))

334 FORMAT(1H1,8H PAGE ,I2,/,8(7X,2HX(,I2,1H),2X))

4 FORMAT(8(2X,F12.4))

NPRT(1)=NC1

NPRT(2)=NC2

NPRT(3)=NC3

NPRT(4)=NC4

NPRT(5)=NC5

NPRT(6)=NC6

NPRT(7)=NC7

NPRT(8)=NC8

IFM=1

IF(IVAR.GT.0) IFM=2

DO 1 I=1,MAXLIN

DO 1 JJ=1,8

KK = NPRT(JJ)

1 PRTARY(I,JJ)=STRMAT(I,KK)

NCOL = 0

DO 2 I=1,8

IF(NPRT(I).NE.0) NCOL =I

2 CONTINUE

NPAGE = 1

LINE = 1

WRITE(NW,333)NPAGE,((SYM(IFM),PARL,NPRT(I),PARR),I=1,NCOL)

DO 3 JLIN=1,MAXLIN

WRITE(NW,4)(PRTARY(JLIN,I),I=1,NCOL)

```

      LINE = LINE+1
      IF (LINE.LT.60 ) GO TO 3
      LINE =1
      NPAGE = NPAGE +1
      WRITE (NW,333) NPAGE, ((SYM(IFM),PARL,NPRT(I),PARR),I=1,NCOL)
3     CONTINUE
      RETURN
      END
      SUBROUTINE INPUT(NDATA)
C     THIS SUBROUTINE READS IN THE PARAMETER LIST THE DELAYS THE
C     INITIAL CONDITIONS OF THE STATE VARIABLES THE PROGRAM CONTROL
C     STATEMENTS THE OUTPUT CONTROL STATEMENTS AND WRITES OUT THEIR
C     VALUES THIS ROUTINE IS CALLED FROM DYSIMP
      COMMON Y(42)
      COMMON/SIGSTR/XSTR(8,600),XTH(8),XT(8),IH(8),NUMSIG,IMAX,X(50)
      COMMON/PARAM/P(50),H(8),NSYS
      COMMON/CUNINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
      COMMON/OUTCON/NGY(42),NTY(42),NGX(50),NTX(50),IADJ
      DIMENSION NDATA(8),SYM(4)
      DATA SYM,PARL,PARR/1HY,1HX,1HH,1HP, 1H(,1H)/
      DATA NR,NW/1,6/
      2 FORMAT(8(F10.0))
      13 FORMAT(1H1,5X,21HORDER OF SYSTEM      = ,I12)
      24 FORMAT(      5X,21HINITIAL TIME        = ,E12.3)
      25 FORMAT(      5X,21HFINAL TIME          = ,E12.3)
      26 FORMAT(      5X,21HINTEGRATION STEP    = ,E12.3)
      27 FORMAT(      5X,21HOUTPUT INTERVAL     = ,I12)
      28 FORMAT(      5X,21HNUMBER OF X ES      = ,I12)
      14 FORMAT(4(5X,A1,A1,I2,A1,3H = ,E12.3,2X),/)
      15 FORMAT(22I3)
      6 FORMAT(3I5,3F10.0)
      7 FORMAT(8F10.0)
      IF (NDATA(1).EQ.0) GO TO 1
      READ(NR,6) NY,NWRITE,NX,BEGTIM,ENDTIM,DELT
      NSYS=NY
      Y(NY+2)=DELT
      Y(NY+1)=BEGTIM
      1 IF (NDATA(2).EQ.0) GO TO 4
      READ(NR,7) (Y(I),I=1,NY)
      NP2 = NY+2
      NTMAX = 2*NY+2
      GO TO 20
      4 IF (NDATA(7).NE.0) GO TO 20
      DO 5 I=1,NP2
      5 Y(I)=YINT(I)
      20 DO 9 I=1,NP2
      9 YINT(I)=Y(I)
      IF (NDATA(3).EQ.0) GO TO 8
      READ(NR, 6) NUMSIG
      READ(NR,2) ( H(I),I=1,NUMSIG)
      8 IF (NDATA(4).EQ.0) GO TO 11
      READ(NR, 6) NP
      READ(NR,2) ( P(I),I=1,NP)
      11 IADJ = 0
      IF (NDATA(8).NE.0) IADJ = 1

```



```

IF (NDATA(5).EQ.0) GO TO 12
DO 50 I= 1,42
50 NGY(I)=NTY(I)=0
DO 51 I= 1,50
51 NGX(I)=NTX(I)=0
READ(NR,15) (NTY(I),I=1,NTMAX)
READ(NR,15) (NGY(I),I=1,NTMAX)
READ(NR,15) (NTX(I),I=1,NX)
READ(NR,15) (NGX(I),I=1,NX)
12 IF (NDATA(6).EQ.0) RETURN
WRITE(NW,13) NY
WRITE(NW,24) BEGTIM
WRITE(NW,25) ENDTIM
WRITE(NW,26) DELT
WRITE(NW,27) NWRITE
WRITE(NW,28) NX
WRITE(NW,14) ((SYM(3),PARL,I,PARR,H(I)),I=1,NUMSIG)
WRITE(NW,14) ((SYM(4),PARL,I,PARR,P(I)),I=1,NP)
RETURN
END
SUBROUTINE PARADJ
C THIS SUBROUTINE ADJUSTS ANY PARAMETER AND RERUNS THE PROGRAM
C THIS ROUTINE IS CALLED FROM DYSIMP
COMMON Y(42)
COMMON/PARAM/P(50),H(8),NSYS
WRITE(6,1)Y(NSYS+1),Y(9)
1 FORMAT(/,10X,*THE ISE OF THE OIL LOOP IS FROM TIME 0.0 TO *,F7.1,*
  1= *,F12.2,/)
READ(1,2)P(18),P(19),P(20)
2 FORMAT(3F10.0)
RETURN
END
SUBROUTINE RKINT(LL,NSYS)
C THIS SUBROUTINE IS A FOURTH ORDER RUNGE KUTTA NUMERICAL
C INTEGRATION ROUTINE
C THIS SUBROUTINE IS CALLED BY DYSIMP AND CALLS DERFUN
DIMENSION YU(20),Y(42),BET(3), DELY(4,20)
COMMON Y
DOUBLE PRECISION YU
IF (LL.NE.1) GO TO 1001
BET(1)=0.5
BET(2)=0.5
BET(3)=1.0
N2=NSYS+2
NP1=NSYS+1
XV=Y(NP1)
CALL DERFUN
DO 320 I=1,NSYS
320 YU(I)=Y(I)
1001 DO 1034 K=1,4
  IF (K.EQ.1) GO TO 1002
  CALL DERFUN
1002 DO 1340 I=1,NSYS
  IPN2=I+N2
1340 DELY(K,I)=Y(N2)*Y(IPN2)

```

```

      IF (K.EQ.4) GO TO 1034
      DO 1350 I=1,NSYS
1350  Y(I)=YU(I)+BET(K)*DELY(K,I)
      Y(NP1)=XV+BET(K)*Y(N2)
1034  CONTINUE
      DO 1039 I=1,NSYS
      DEL=(DELY(1,I)+2.0*DELY(2,I)+2.0*DELY(3,I)+DELY(4,I))/6.0
      YU(I)=YU(I)+DEL
      Y(I)=YU(I)
1039  CONTINUE
      Y(NP1)=XV+Y(N2)
      CALL DERFUN
      XV=Y(NP1)
      RETURN
      END
      SUBROUTINE PLOT(X,LX,Y,LY,Z,LZ,NPT,NPLOT,NCOPY,NCD,NDIM,NXTREM,
*NPAGE)
C  PURPOSE
C  THIS SUBROUTINE PLOTS UP TO TEN RELATIONSHIPS OF THE FORM  $Y = F(X)$ 
C  IT WILL ALSO PREPARE TOPOGRAPHIC MAPPINGS OF THREE DIMENSIONAL
C  SURFACES.
C  USAGE
C  CALL PLOT(X,LX,Y,LY,Z,LZ,NPT,NPLOT,NCOPY,NCD,NDIM,NXTREM,NPAGE)
C  X, Y, Z - COORDINATES OF DATA POINTS
C  LX, LY, LZ - SCALING FACTORS. THE SCALES ON EACH AXIS MAY BE LINE
C  ONE CYCLE LOG, TWO CYCLE LOG, ...SIX CYCLE LOG
C  NPT - NUMBER OF POINTS TO BE PLOTTED
C  NPLOT - NUMBER OF RELATIONSHIPS TO BE PLOTTED IN DIFFERENT
C  SYMBOLS IF PLOTS ARE TWO DIMENSIONAL. IF A TOPOGRAPHIC
C  MAPPING IS CALLED FOR NPLOT IS NUMBER OF CONTOURS TO BE PLOTT
C  NCOPY - NUMBER OF DUPLICATE PLOTS TO BE MADE
C  NCD - NUMBER OF INFORMATION CARDS TO BE READ. IF NCD IS ZERO NO
C  CARDS ARE READ. OTHERWISE SEE WRITE UP.
C  NDIM - IF THE RELATIONSHIPS WHICH ARE TO BE PLOTTED ARE TWO
C  DIMENSIONAL NDIM IS 2, AND IF A TOPOGRAPHIC MAPPING IS DESIR
C  NDIM IS 3.
C  NXTREM - IF NXTREM IS 1, A CARD WITH THE LIMITING VALUES OF
C  THE X, Y, AND Z SCALES IS READ. OTHERWISE THEY ARE
C  CALCULATED WITHIN THE SUBROUTINE SO THAT ALL DATA POINTS WILL
C  APPEAR ON THE PLOT.
C  NPAGE - NUMBER OF PAGES THE PLOT IS TO COVER. IF NPAGE IS 1 THE P
C  IS 56 LINES HIGH. EACH INCREASE OF 1 IN NPAGE CAUSES THE
C  PLOT TO EXTEND ACROSS AN ADDITIONAL 66 LINES.
C  SUBROUTINES REQUIRED
C  THE FORTRAN IV SUBROUTINE POT IS INCLUDED AS PART OF THE
C  PLOTTING PACKAGE.
C
      REAL N
      DIMENSION X(1),Y(1),Z(1),SX(7),TITLE(20),L(134),NCH(41),NH(41)
* ,MOP(42),FM1(6),FM2(6),FM3(6),FM4(8),FM5(8),FM6(8),FMT(6),FRT(8)
1,SCL(3)
      DIMENSION IZP(3000)
      DATA FM1/4H(1H , 4HA1,F,4H9.2,,4H121A,4H1) ,4H /
      DATA FM2/4H(1H , 4HA1,F,4H9.4,,4H121A,4H1) ,4H /
      DATA FM3/4H(1H , 4HA1,1,4HPE9,,4H2,12,4H1A1),4H /

```

```

DATA FM4/4H(1X,,4H1PE1,4H6.2,,4H5E20,4H.2,E,4H15.2,4H) ,4H /
DATA FM6/4H(5X,,4HF9.2,4H,5(1,4H1X, ,4HF9.2,4H),9X,4H,F9.,4H2) /
DATA FM5/4H(5X,,4HF9.4,4H,5(1,4H1X, ,4HF9.4,4H),9X,4H,F9.,4H4) /
DATA NCH/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HA,1HB,1HC,1HD,
*1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1
*HU,1HV,1HW,1HX,1HY,1H0,1H*,1HS,1H=,1H+,1H-/
DATA ND,NP,NM,NB/1H.,1H.,1H-,1H /
DATA SCL/1HX,1HY,1HZ/
1  FORMAT(20A4)
2  FORMAT(52A1,7A4)
3  FORMAT(1H ,26X,20A4)
5  FORMAT(132A1)
6  FORMAT(5H ALL ,A1,57H VALUES ARE NEGATIVE. LOG SCALE MAY NOT BE S
*PECIFIED. )
7  FORMAT(1PE17.2,E115.2)
8  FORMAT(1PE17.2,E61.2,E54.2)
9  FORMAT(1PE17.2,2E40.2,E35.2)
10 FORMAT(1PE17.2,3E30.2,E25.2)
11 FORMAT(1PE17.2,4E24.2,E19.2)
12 FORMAT(1HK,59X,7A4)
74 FORMAT(6E12.2)
76 FORMAT(1H1)
80 FORMAT(70H NPT(NO. OF PTS.) MUST BE EVENLY DIVISIBLE BY NPLOT(NO.
*OF CURVES.) )
84 FORMAT(1X,36H ONE DIMENSIONAL PLOT NOT ALLOWED. )
94 FORMAT(42A1,10X,7A4)
D095I=1,41
95 NH(I)=NCH(I)
N=50.66*(NPAGE-1)
LLX=LX+1
NDD=NCD+1
GOTO(15,13,14,13),NDD
13 READ(1,1)(TITLE(I),I=1,20)
14 IF(NDD.LT.3)GOTO15
IF(NDIM.NE.3)GOTO93
READ(1,94)(MOP(I),I=1,42),TAB1,TAB2,TAB3,TAB4,TAB5,TAB6,TAB7
GO TO 81
93 READ(1,2)(MOP(I),I=1,42),( NH(I),I=1,10),TAB1,TAB2,TAB3,TAB4,TAB5,
1TAB6,TAB7
GOTO81
15 D082I=1,42
82 MOP(I)=NB
81 NCH(41)=NB
IF (FLOAT(NPT/NPLOT)-FLOAT(NPT)/FLOAT(NPLOT))101,85,101
85 IF (NXTREM.EQ.1)GOTO72
YMIN=1.00E 75
XMIN=YMIN
ZMIN=YMIN
YMAX=1.00E-78
XMAX=YMAX
ZMAX=YMAX
D051I=1,NPT
GOTO(103,52,53),NDIM
53 IF (Z(I).GE.ZMAX)ZMAX=Z(I)
IF (Z(I).LE.ZMIN)ZMIN=Z(I)

```

```

52 IF (Y(I).GE.YMAX)YMAX=Y(I)
   IF (Y(I).LE.YMIN)YMIN=Y(I)
   IF (X(I).GE.XMAX)XMAX=X(I)
   IF (X(I).LE.XMIN)XMIN=X(I)
51 CONTINUE
   IF (LX.GT.0)GOTO900
   XMIN=XMIN-(XMAX-XMIN)/75.0
   XMAX=XMAX+(XMAX-XMIN)/75.0
900 IF (LY.GT.0)GOTO73
   YMIN=YMIN-(YMAX-YMIN)/30.0
   YMAX=YMAX+(YMAX-YMIN)/30.0
   GOTO73
72 READ(1,74)XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX
73 IF (YMAX.GT.10000.0.OR.YMIN.LT.(-10000.0))GOTO60
   IF (YMAX-YMIN.LT.5.0)GOTO60
   IF (YMAX.GT.10.0.OR.YMIN.LT.(-10.0))GOTO62
   IF (YMAX.LT.0.01.OR.ABS(YMIN).LT.0.01)GOTO60
   D059I=1,6
59 FMT(I)=FM2(I)
   GOTO65
60 D061I=1,6
61 FMT(I)=FM3(I)
   GOTO65
62 D063I=1,6
63 FMT(I)=FM1(I)
65 IF (XMAX.GT.10000.0.OR.XMIN.LT.(-10000.0))GOTO67
   IF (XMAX-XMIN.LT.5.0)GOTO67
   IF (XMAX.GT.10.0.OR.XMIN.LT.(-10.0))GOTO69
   IF (XMAX.LT.0.01.OR.ABS(XMIN).LT.0.01)GOTO67
   D066I=1,8
66 FRT(I)=FM5(I)
   GOTO70
67 D068I=1,8
68 FRT(I)=FM4(I)
   GOTO70
69 D071I=1,8
71 FRT(I)=FM6(I)
70 IF (LX.GT.0)GOTO17
   CX=120./(XMAX-XMIN)
   SX(1)=XMIN
   SX(7)=XMAX
   U=XMIN
   D016K=2,6
   U=(XMAX-XMIN)/6.*U
16 SX(K)=U
   GOTO19
17 XLX=LX
   IF (XMAX.LE.0.0)GOTO100
   XMIN=10.0** (ALOG10(XMAX*9.999999)-FLOAT(LX))
   MS=1
   CX=120./XLX
   NX=ALOG10(XMIN)
   IF (XMIN.LT.1.0)NX=ALOG10(XMIN)-0.999999
   XMIN=10.0**NX
   D018K=1,LLX

```

```

18 SX(K)=10.**(NX+K-1)
19 CALLPOT(X,XMIN,LX,NPT,0,120.,CX)
   IF (LY.GT.0)GOTO20
   CY=N/(YMAX-YMIN)
   GOT021
20 YLY=LY
   IF (YMAX.LE.0.0)GOTO100
   YMIN=10.0**(ALOG10(YMAX*9.999999)-FLOAT(LY))
   MS=2
   CY=N/YLY
   KY=CY
   NY=ALOG10(YMIN)
   IF (YMIN.LT.1.0)NY=ALOG10(YMIN)-0.9999999
   YMIN=10.0**NY
21 CALLPOT(Y,YMIN,LY,NPT,1,N,CY)
   IF (NDIM.LT.3)GOTO24
   FPLOT=NPLOT
   IF (LZ.GT.0)GOTO22
   CZ=FPLOT/(ZMAX-ZMIN)
   GOT023
22 ZLZ=LZ
   IF (ZMAX.LE.0.0)GOTO100
   ZMIN=10.0**(ALOG10(ZMAX*9.999999)-ZLZ)
   MS=3
   CZ=FPLOT/ZLZ
   IF (ZMIN.GE.1.0)GO TO 23
   IA=ALOG10(ZMIN)-.9999999
   ZMIN=10.0**IA
23 CALL POT(Z,ZMIN,LZ,NPT,0,FPLOT,CZ)
   D077I=1,NPT
   IZ=Z(I)+1.0
   IZP(I) = NCH(IZ)
77 CONTINUE
   GOT099
24 NPN=NPT/NPLOT
   DO 78I=1,NPLOT
   JJ=(I-1)*NPN+1
   JK=I*NPN
   D078J=JJ,JK
   IZP(J)=NH(I)
78 CONTINUE
99 D0102I=1,NPT
   D0102J=1,NPT
   IF (Y(I).GE.Y(J))GOTO102
   A=Y(I)
   B=X(I)
   Y(I)=Y(J)
   X(I)=X(J)
   Y(J)=A
   X(J)=B
   IC = IZP(I)
   IZP(I) = IZP(J)
   IZP(J)=IC
102 CONTINUE
   D056NO=1,NCOPY

```

```

M1=1
M2=1
LL=0
LYY=LY
TT=N
WRITE(6,76)
IF(NDD.EQ.1.OR.NDD.EQ.3)GO TO 97
WRITE(6,3)(TITLE(I),I=1,20)
97 NN= N+1.
DO43KK=1,NN
M=1
NNN=NP
JED=1
T=INT(N)-KK+1
DO25J=1,133
25 L(J)=NB
L(133)=ND
L(13)=NP
IF(LY.GT.0)GOTO26
L(133)=NP
IF(T.GT.TT)GOTO30
SCALE=T/CY+YMIN
IF(T.LE.0.)SCALE=YMIN
M=0
TT=TT-5.
GOTO30
26 GOTO(27,27,28,28,27,28),LY
27 SS=KY*LYY
GOTO29
28 SS=KY*LYY+1
29 L(13)=ND
IF(T.GT.SS)GOTO30
SCALE=10.** (NY+LYY)
M=0
LYY=LYY-1
L(13)=NP
L(133)=NP
30 IF(N.EQ.T)GOTO31
IF(0..NE.T)GOTO37
31 DO32J=14,133
32 L(J)=NM
IF(LX.GT.0)GOTO34
DO33J=13,133,10
33 L(J)=NP
GOTO36
34 KX=120/LX
DO35J=13,133,KX
35 L(J)=NP
36 IF(N.EQ.T)L(133)=ND
37 LL=LL+1
IF(LL.GT.NPT)GOTO79
IF(Y(LL)-T)79,92,37
92 J=X(LL)
L(J+13)=IZP(LL)
GOTO37

```

```

79 LL=LL-1
   IF (M1.GT.4+( INT(N)-50)/2 .AND.M1.LT.47+( INT(N)-50)/2 )GOTO54
   L(2)=NB
   GOT055
54 L(2)=MOP(M2)
   M2=M2+1
55 M1=M1+1
41 IF (M.EQ.1)GOTO42
   WRITE(6,FMT)L(2),SCALE,(L(J),J=13,133)
   GOT043
42 WRITE(6,5) (L(J),J=1,11),(L(J),J=13,133)
43 CONTINUE
   GOT0(44,45,46,47,48,49,44),LLX
44 WRITE(6,FRT) (SX(K),K=1,7)
   GOT050
45 WRITE(6,7) (SX(K),K=1,LLX)
   GOT050
46 WRITE(6,8) (SX(K),K=1,LLX)
   GOT050
47 WRITE(6,9) (SX(K),K=1,LLX)
   GOT050
48 WRITE(6,10) (SX(K),K=1,LLX)
48 WRITE(6,10) (SX(K),K=1,LLX)
   GOT050
49 WRITE(6,11) (SX(K),K=1,LLX)
50 IF (NDD.LT.3)GOTO56
   WRITE(6,12)TAB1,TAB2,TAB3,TAB4,TAB5,TAB6,TAB7
56 CONTINUE
83 RETURN
100 WRITE(6,6)SCL(MS)
   RETURN
101 WRITE(6,80)
   RETURN
103 WRITE(6,84)
   RETURN
   END
SUBROUTINEPOT(V,VMIN,LV,NP,J,VC,C)
   DIMENSIONV(1)
   IF (LV.GT.0)GOTO2
   DO1I=1,NP
1  V(I)=FLOAT( INT(C*(V(I)-VMIN)+.5))
   GOT04
2 VMIN=10.0**(- INT(ALOG10(VMIN)))
   DO3I=1,NP
   IF (V(I).LE.0.0)GOTO3
   V(I)=FLOAT( INT(C*(ALOG(V(I)/VMIN)/2.302585)+.5))
   VT=V(I)
3 CONTINUE
4 DO6I=1,NP
   IF (V(I).LT.0.)GOTO5
   IF (V(I).LE.VC)GOTO6
5 V(I)=VC+1.
6 CONTINUE
   RETURN
   END

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Appendix C

COMPUTER PROGRAM FOR AN ANALOG
EQUIVALENT TO A DIRECT DIGITAL
CONTROL LOOP

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
PROGRAM DICK(INPUT,OUTPUT,TAPE1=INPUT,TAPE6=OUTPUT)
```

```

C
C THIS IS A SIMULATION OF THE BENDIX MOLD HEATING AND COOLING SYSTEM
C UNDER ANALOG EQUIVALENT CONTROL      ADDITIONAL DEAD TIME
C
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/XSYSTO/YSTOR(200,42),XSTOR(200,60)
COMMON/CUNINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
COMMON Y(42)
COMMON/PARAM/P(99),H(9),NSYS
COMMON/SIGSTR/XSTR(9,500),XTH(9),XT(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/OUTCON/NGY(42),NTY(42),NGX(60),NTX(60),IADJ
DATA NR,NW/1,6/
C
C CHANGED CONTROL FORMAT FROM 8(9X,I1) TO 10(1X,I1) AND NDATA(8) TO
C NDATA(10) AND PUT IT IN COMMON
C
C FUNCTIONS AVAILABLE ARE
C
C GSABSE - GAIN SWITCH ON ABSOLUTE ERROR
C
C FOLWUG - FIRST ORDER LAG WITH UNITY GAIN
C
C FOLNUG - FIRST ORDER LAG NON-UNITY GAIN
C
C FLOWCL - FLOW CALCULATION
C
C OUTLIM - CHECKS FOR OUT OF LIMIT CONDITIONS
103 FORMAT(/)
100 FORMAT(1X,*THE FOLLOWING ARE INITIAL CONDITIONS ON THE Y S*)
101 FORMAT(1X,*THE FOLLOWING ARE INITIAL VALUES OF XS BEFORE 1ST INTEG
IRATION*)
902 FORMAT(4(5X,*Y(*,I2,*) = *,F12.5,2X))
903 FORMAT(4(5X,*X(*,I2,*) = *,F12.5,2X))
904 FORMAT(1H,* KP = *,F7.3,4X,* KI = *,F7.3,4X,* KD = *,F7.3)
1 FORMAT(10(1X,I1))
13 READ(1,1) NDATA
NSUM = 0
DO 4 I=1,20
4 NSUM = NSUM + NDATA(I)
IF(NSUM.EQ.0) GO TO 999
IBUG=NDATA(9)
CALL INPUT
IF(NUMSIG.EQ.0) GO TO 70
CALL INTFUN
70 IF(NX.EQ.0) GO TO 80
CALL INTCAL
80 CALL DERFUN
WRITE(6,100)
WRITE(6,902)(I,Y(I),I=1,NSYS)
WRITE(6,103)
WRITE(6,101)
WRITE(6,903)(I,X(I),I=1,NX)
WRITE(6,904)P(18),P(19),P(20)
NSTEP = (ENDTIM-BEGTIM)/DELT
8 IF((200*NWRITE) .LT. NSTEP) GO TO 23
GO TO 7
23 NWRITE=NSTEP/200+1
WRITE(6,3) NWRITE
3 FORMAT(10X,*NUMBER OF OUTPUTS EXCEEDS 200 , NWRITE RESET TO *,I5, /
1/)
GO TO 8

```

```

7 IF((NSTEP/NWRITE).LT.100) IPAGE = 1
  IF((NSTEP/NWRITE).GT.200) IPAGE = 2
  NPRNO=NSTEP/NWRITE
  IF(NPRNO.GT.100.AND.NPRNO.LE.200) IPAGE = 2
  JI = 1
  ITY=ITX=IGY=IGX=0
  NTMAX = 2*NY + 2
  DO 64 I=1,NTMAX
    IF(NGY(I).NE.0) IGY = IGY+1
    IF(NTY(I).NE.0) ITY = ITY+1
64 YSTOR(JI,I) = Y(I)
    DO 65 I=1,NX
      IF(NTX(I).NE.0) ITX = ITX+1
      IF(NGX(I).NE.0) IGX = IGX+1
65 XSTOR(JI,I) = X(I)
    NI = 0
    DO 60 KK = 1,NSTEP
      NI = NI+1
      CALL RKINT(KK,NY)
      IF(NX.EQ.0) GO TO 51
      CALL INTCAL
51 IF(NI.NE.NWRITE) GO TO 60
    NI = 0
    IF(IBUG.EQ.0) GO TO 444
901 FORMAT(8(2X,E12.3))
    WRITE(6,901)Y(NSYS+1)
    WRITE(6,901)(Y(I),I=10,15)
    WRITE(6,901)(X(I),I=18,43)
444 JI = JI + 1
    DO 66 I = 1,NTMAX
66 YSTOR(JI,I) = Y(I)
    DO 67 I=1,NX
67 XSTOR(JI,I) = X(I)
60 CONTINUE
    IF(ITY.EQ.0) GO TO 10
    K = 1
61 CALL PRITBL(YSTOR,200,42,0,JI,NTY(1),NTY(K+1),NTY(K+2),NTY(K+3),
1,NTY(K+4),NTY(K+5),NTY(K+6),NTY(K+7))
    K = K + 8
    IF(K.GT.ITY) GO TO 10
    K=K-1
    GO TO 61
10 IF(ITX.EQ.0) GO TO 11
    K = 1
62 CALL PRITBL(XSTOR,200,60,1,JI,NTX(1),NTX(K+1),NTX(K+2),NTX(K+3),
1,NTX(K+4),NTX(K+5),NTX(K+6),NTX(K+7))
    K = K + 8
    IF(K.GT.ITX) GO TO 11
    K=K-1
    GO TO 62
11 IF(IGY.EQ.0) GO TO 12
    K = 1
63 CALL PRIPLT(YSTOR,200,42,0,JI,IPAGE,NGY(1),NGY(K+1),NGY(K+2),
1,NGY(K+3),NGY(K+4))
    K = K + 4

```

```

IF(K.GE.IGY) GO TO 12
GO TO 63
12 IF(IGX.EQ.0) GO TO 14
K = 1
59 CALL PRTPLT(XSTOR,200,60,1,JI,IPAGE,NGX(1),NGX(K+1),NGX(K+2),
1 NGX(K+3),NGX(K+4))
K = K + 4
IF(K.GE.IGX) GO TO 14
GO TO 59
14 IF(IADJ.NE.0) CALL PARADJ
GO TO 13
999 STOP
END
SUBROUTINE SPADJ(T,SPT,SPDT,GAIN,TAU1,TAU2,T1)
COMMON/PARAM/P(99),H(9),NSYS
COMMON/FM/TM(7),TP(7),PP(3)
COMMON/NIDAT/NDATA(20),IFL,F(20)
DATA TM/120.,150.,1180.,1360.,1420.,2800.,4000./
DATA TP/120.,450.,170.,550.,320.,365.,90./
13 FORMAT(7F10.0)
IF(NDATA(12).NE.1) GO TO 12
READ(1,13)(TM(I),I=1,7)
READ(1,13)(TP(I),I=1,7)
12 IF(T.LT.TM(1)) GO TO 1
IF(T.LT.TM(2)) GO TO 2
IF(T.LT.TM(3)) GO TO 3
IF(T.LT.TM(4)) GO TO 4
IF(T.LT.TM(5)) GO TO 5
IF(T.LT.TM(6)) GO TO 6
IF(T.LT.TM(7)) GO TO 7
1 SPT=TP(1)
GAIN=P(57)
TAU1=P(58)
TAU2=P(60)
SPDT=0.0
GO TO 8
2 SPT=TP(2)
GAIN=P(57)
TAU1=P(58)
TAU2=P(60)
SPDT=0.0
GO TO 8
3 SPT=TP(3)
GAIN=PP(1)
TAU1=PP(2)
TAU2=PP(3)
SPDT=0.0
GO TO 8
4 SPT=TP(4)
GAIN=P(57)
TAU1=P(58)
TAU2=P(60)
SPDT=0.0
GO TO 8
5 SPT=TP(5)

```

GAIN=PP(1)

TAU1=PP(2)

TAU2=PP(3)

SPDT=0.0

GO TO 8

6 SPT=TP(6)

GAIN=P(57)

TAU1=P(58)

TAU2=P(60)

SPDT=0.0

GO TO 8

7 SPT=TP(7)

GAIN=PP(1)

TAU1=PP(2)

TAU2=PP(3)

SPDT=0.0

8 SPT=SPT*P(47)+P(48)

SPDT=SPDT*P(47)

RETURN

END

SUBROUTINE DERFUN

COMMON Y(42)

COMMON/PARAM/P(99),H(9),NSYS

COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)

COMMON/NIDAT/NDATA(20),IFL,F(20)

C PLACE DIFFERENTIAL EQUATIONS AFTER THIS CARD

F(1)=FOLWUG(X(8),Y(1),P(5))

F(2)=FOLWUG(X(3),Y(3),P(5))

F(3)=Y(4)

F(4)=(P(12)*X(6)-Y(3)-(P(10)+P(11))*Y(4))/(P(10)*P(11))

F(8)=0.0

F(6)=P(8)*X(13)

F(7)=P(14)*(P(15)-Y(7))

F(5)=P(2)*(F(8)-F(1))+P(1)*(Y(8)-Y(1))

C THE FOLLOWING CARD CALCULATES THE ISE OF THE OIL LOOP

F(9)=AISEC(X(44))

F(10)=X(56)

IF(NDATA(11).NE.1) GO TO 3

CALL INTCUT

IF(IFL.EQ.1) GO TO 2

3 F(11)=X(18)*P(19)*P(18)/60.

2 F(12)=FOLNUG(P(41),X(38),Y(12),P(42))

F(13)=FOLNUG(P(45),X(37),Y(13),P(46))

F(14)=FOLWUG(X(23),Y(14),P(49))

F(15)=FOLWUG(X(45),Y(15),P(49))

F(16)=(P(26)*X(37)-(P(27)+P(31))*Y(16)-Y(17))/(P(27)*P(31))

F(17)=Y(16)

C THE FOLLOWING PLACES THE NUMERICAL VALUE OF THE DERIVATIVE IN

C Y(N+2+J)

DO 1 II=1,NSYS

IEQ = NSYS+2+II

1 Y(IEQ) = F(II)

RETURN

END

SUBROUTINE INTCUT

```
COMMON/PARAM/P(99),H(9),NSYS
```

```
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
```

```
COMMON Y(42)
```

```
COMMON/NIDAT/NDATA(20),IFL,F(20)
```

```
IFL=0
```

```
OSP=BNSP
```

```
BNSP=Y(10)
```

```
SPD=BNSP-OSP
```

```
IF (SPD.NE.0) GO TO 1
```

```
TO=TN
```

```
TN=X(44)
```

```
DIT=(TN-TO)/Y(NSYS+2)
```

```
IF (DIT.GT.-11.0) GO TO 1
```

```
F(11)=0.0
```

```
Y(11)=0.0
```

```
IFL=1
```

```
1 RETURN
```

```
END
```

```
SUBROUTINE INTFUN
```

```
COMMON Y(42)
```

```
COMMON/PARAM/P(99),H(9),NSYS
```

```
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
```

```
COMMON/FM/TM(7),TP(7),PP(3)
```

```
COMMON/NIDAT/NDATA(20),IFL,F(20)
```

```
C THIS SUBROUTINE IS CALLED ONLY ONCE AND CONTAINS INITIAL FUNCTIONS
```

```
C IT ALSO INITIALIZES THE DELAY BLOCKS
```

```
PP(1)=P(26)
```

```
PP(2)=P(27)
```

```
PP(3)=P(31)
```

```
X(56)=0.0
```

```
X(17)=0.0
```

```
X(41)=Y(17)
```

```
X(55)=X(41)*X(41)*P(33)+X(41)*P(34)+P(35)
```

```
X(40)=X(55)
```

```
IMAX=0
```

```
DO 2 I=1,NUMSIG
```

```
IH(I)=H(I)/Y(NSYS+2)+1
```

```
IF (IH(I) .GE. IMAX) IMAX=IH(I)
```

```
2 CONTINUE
```

```
DO 3 KK=1,IMAX
```

```
XSTR(1,KK)=0.0
```

```
XSTR(2,KK)=Y(7)
```

```
XSTR(3,KK)=Y(8)
```

```
XSTR(4,KK)=Y(8)
```

```
XSTR(5,KK)=Y(17)
```

```
XSTR(6,KK)=12.0
```

```
XSTR(7,KK)=X(55)
```

```
XSTR(8,KK)=X(40)
```

```
3 XSTR(9,KK)=X(41)
```

```
DO 4 I=1,NUMSIG
```

```
4 XTH(I)=XSTR(I,IMAX)
```

```
Y(13)=Y(17)/P(26)
```

```
X(21)=12.0
```

```
X(37)=Y(17)/P(26)
```

```
Y(10)=Y(10)*P(47)+P(48)
```

```

X(42)=Y(13)
X(44)=((Y(10)-P(48))/P(47))-X(37)
BNSP=Y(10)
TN=X(44)
RETURN
END
SUBROUTINE INTICAL
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON Y(42)
COMMON/PARAM/P(99),H(9),NSYS
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
C PLACE INTERMEDIATE EQUATIONS AFTER THIS CARD
X(1)=Y(NSYS+1)
X(3)=Y(7)
X(11)=Y(8)
XT(1)=P(9)*Y(6)
XT(2)=X(3)
XT(3)=XTH(2)
XT(4)=Y(3)+XTH(3)
XT(5)=X(38)
XT(6)=X(21)
XT(7)=X(55)
XT(8)=X(40)
XT(9)=X(41)
CALL DELAY
Y(15)=OUTLIM(Y(15),0.0,100.)
Y(14)=OUTLIM(Y(14),0.0,100.)
Y(11)=OUTLIM(Y(11),4.0,20.)
X(42)=Y(13)
X(43)=P(47)*X(42)+P(48)
X(43)=OUTLIM(X(43),4.0,20.)
CALL SPADJ(X(1),Y(10),F10DT,P(26),P(27),P(31),P(59))
X(56)=F10DT
X(57)=((Y(10)-P(48))/P(47))
X(18)=Y(10)-X(43)
X(19)=P(18)*X(18)
X(20)=P(20)*P(18)*60.*(X(56)-P(47)*(X(37)+P(45)-Y(13))/P(46))
X(50)=GSABSE(X(44),P(52),P(51))
IF(NDATA(10).EQ.1) X(50)=1.0
X(51)=X(19)+X(20)+Y(11)
X(21)=X(50)*X(51)
X(21)=OUTLIM(X(21),4.0,20.)
X(52)=XTH(6)
X(22)=P(21)*X(52)+P(22)
X(22)=OUTLIM(X(22),0.0,15.0)
X(23)=(X(22)-P(23))*P(25)/P(24)
X(45)=(P(23)-X(22))*P(25)/P(23)
X(23)=OUTLIM(X(23),0.,100.)
X(45)=OUTLIM(X(45),0.,100.)
X(24)=FLOWCL(X(23),P(53),P(54),P(55),P(28),P(32))
X(46)=FLOWCL(X(45),P(53),P(54),P(55),P(28),P(32))
X(28)=P(28)*X(46)
X(25)=P(28)*X(24)
X(26)=P(28)-X(28)-X(25)
X(53)=XTH(9)

```



```

X(27)=X(26)*(P(29)*X(53)+P(30))
X(31)=90.
X(32)=600.
X(29)=X(25)*(P(29)*X(32)+P(30))
X(47)=X(28)*(P(29)*X(31)+P(30))
X(54)=XTH(8)
X(30)=X(27)*X(54)
X(48)=P(33)*X(31)*X(31)+P(34)*X(31)+P(35)
X(33)=P(33)*X(32)*X(32)+X(32)*P(34)+P(35)
X(49)=X(47)*X(48)
X(34)=X(33)*X(29)
X(35)=X(34)+X(30)+X(49)
X(36)=X(35)/(X(27)+X(29)+X(47))
X(36)=OUTLIM(X(36),4.715,186.116)
X(37)=P(36)+(SQRT(P(37)+(P(38)*(P(39)+X(36)))))/P(40)
X(37)=OUTLIM(X(37),50.,600.)
X(38)=Y(17)
X(39)=XTH(5)
X(55)=X(36)-((P(43)*P(44)*Y(16))/(X(27)+X(29)+X(47)))
X(55)=OUTLIM(X(55),4.715,186.116)
X(40)=XTH(7)
X(41)=P(36)+(SQRT(P(37)+(P(38)*(P(39)+X(40)))))/P(40)
X(41)=OUTLIM(X(41),50.,600.)
X(44)=((Y(10)-P(48))/P(47))-X(37)
X(4)=XTH(2)
X(6)=XTH(1)
X(5)=XTH(3)
X(8)=XTH(4)
X(9)=Y(5)-Y(2)
Y1D=(X(8)-Y(1))/P(5)
X10D=X(15)-Y1D
X(10)=X(11)-Y(1)
Y5D=P(2)*X10D+P(1)*X(10)
X9D=Y5D-Y1D
X(15)=P(3)*X(9)+P(4)*X9D
X(2)=P(7)*X(15)
X(12)=X(2)-Y(6)
IF(X(12).GT.0.0)X(13)=1.0
IF(X(12).LT.0.0)X(13)=-1.0
IF(ABS(X(12)).LE.P(6))X(13)=0.0
X(7)=X(5)+Y(3)
X(14)=P(9)*Y(6)
IF(X(14).GT.(1.-P(13)))GO TO 60
IF(X(14).LT.(-P(13)))GO TO 61
RETURN
60 X(14)=1.0-P(13)
Y(6)=X(14)*90.
RETURN
61 X(14)=-P(13)
Y(6)=X(14)*90.
RETURN
END
SUBROUTINE INPUT
COMMON Y(42)
COMMON/CONINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)

```

```

COMMON/PARAM/P(99),H(9),NSYS
COMMON/SIGSTR/XSTR(9,500),XTH(9),XT(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/OUTCON/NGY(42),NTY(42),NGX(60),NTX(60),IADJ
COMMON/NIDAT/NDATA(20),IFL,F(20)
DIMENSION SYM(4)
DATA SYM,PARL,PARR/1HY,1HX,1HH,1HP, 1H(,1H)/
DATA NR,NW/1,6/
DATA MSP,MSU,MSN,MSC,MSH/1HP,1HU,1HN,1HC,1HH/
DATA MSC,MSA,MSV,MSI,MST,MSY,MSB/1HC,1HA,1HV,1HI,1HT,1HY,1H /
2 FORMAT(8F10.0)
13 FORMAT(1H1,4X,*ORDER OF SYSTEM = *,I12,14X,*MOLD NUMBER = *
1,I4)
24 FORMAT(5X,*INITIAL TIME*,6X,*= *,F12.3,14X,*MOLD SECTION = *,A1,
1A1,A1,A1,A1,A1)
25 FORMAT(5X,*FINAL TIME*,8X,*= *,F12.3,14X,*MOLD MASS = *,F7.0)
26 FORMAT(5X,*INTEGRATION STEP = *,F12.5,14X,*MOLD TEST NUMBER = *,
1I2)
27 FORMAT(5X,*OUTPUT INTERVAL = *,I12)
28 FORMAT(5X,*NUMBER OF X ES = *,I12)
14 FORMAT(4(5X,A1,A1,I2,A1,* = *,F12.5,2X))
99 FORMAT(/)
15 FORMAT(22I3)
6 FORMAT(3I5,3F10.0)
7 FORMAT(8F10.0)
100 FORMAT(I2)
101 FORMAT(I1)
30 FORMAT(I4,1X,I1,2X,I2)
IF(NDATA(1).EQ.0) GO TO 1
READ(1,30)MMFR,MSFR,MTEST
READ(NR,6) NY,NWRITE,NX,BEGTIM,ENDTIM,DELT
NSYS=NY
Y(NY+2)=DELT
Y(NY+1)=BEGTIM
1 IF(NDATA(2).EQ.0) GO TO 4
READ(NR,7)(Y(I),I=1,NY)
NP2 = NY+2
NTMAX = 2*NY+2
GO TO 20
4 IF(NDATA(7).NE.0) GO TO 20
DO 5 I=1,NP2
5 Y(I)=YINI(I)
20 DO 9 I=1,NP2
9 YINI(I)=Y(I)
IF(NDATA(3).EQ.0) GO TO 8
READ(NR,101) NUMSIG
READ(NR,2) ( H(I),I=1,NUMSIG)
8 IF(NDATA(4).EQ.0) GO TO 11
READ(NR,100) NP
READ(NR,2) ( P(I),I=1,NP)
11 IADJ = 0
IF(NDATA(8).NE.0) IADJ = 1
IF(NDATA(5).EQ.0) GO TO 12
DO 50 I= 1,42
50 NGY(I)=NTY(I)=0
DO 51 I=1,60

```

```

51 NGX(I)=NIX(I)=0
   READ(NR,15) (NTY(I),I=1,NTMAX)
   READ(NR,15) (NGY(I),I=1,NTMAX)
   READ(NR,15) (NTX(I),I=1,NX)
   READ(NR,15) (NGX(I),I=1,NX)
12 IF (NDATA(6).EQ.0) RETURN
   WRITE(NW,13) NY,MMFR
   IF (MSFR .NE. 0) GO TO 31
   WRITE(6,24) BEGTIM,MSB,MSP,MSU,MSN,MSC,MSH
   GO TO 32
31 WRITE(6,24) BEGTIM,MSC,MSA,MSV,MSI,MST,MSY
32 WRITE(NW,25) ENDTIM,P(43)
   WRITE(6,26) DELT,MTEST
   WRITE(NW,27) NWRITE
   WRITE(NW,28) NX
   WRITE(6,99)
   WRITE(NW,14) ((SYM(3),PARL,I,PARR,H(I)),I=1,NUMSIG)
   WRITE(6,99)
   WRITE(NW,14) ((SYM(4),PARL,I,PARR,P(I)),I=1,NP)
   WRITE(6,99)
   RETURN
END
FUNCTION AISEC(VOISE)
  AISEC=VOISE*VOISE
  RETURN
END
FUNCTION GSABSE(AES,SLIM,GAIN)
  IF (ABS(AES) .GE. SLIM) GSABSE = GAIN
  IF (ABS(AES) .LT. SLIM) GSABSE = 1.0
  RETURN
END
FUNCTION FOLWUG(XIN,YOUT,TAU)
  FOLWUG=(XIN-YOUT)/TAU
  IF (TAU .EQ. 0.0) FOLWUG = XIN
  RETURN
END
FUNCTION FOLNUG(GAIN,XIN,YOUT,TAU)
  FOLNUG=(GAIN*XIN-YOUT)/TAU
  IF (TAU .EQ. 0.0) FOLNUG = GAIN*XIN
  RETURN
END
FUNCTION FLOWCL(PL,ACON,BCON,CCON,TFLO,ATCON)
  IF (PL .LE. 10.) GO TO 2
  FLOWCL=(ACON*PL*PL+BCON*PL+CCON)/(TFLO*60.)
  RETURN
2 FLOWCL=PL*ATCON/(TFLO*60.)
  IF (FLOWCL .LE. 0.0) FLOWCL=0.0
  RETURN
END
FUNCTION OUTLIM(XIN,BOT,TOP)
  OUTLIM = XIN
  IF (XIN.GT.TOP) OUTLIM = TOP
  IF (XIN.LT.BOT) OUTLIM = BOT
  RETURN
END

```

```

SUBROUTINE DELAY
COMMON Y(42)
COMMON/PARAM/P(99),H(9),NSYS
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
ISTOP = IMAX -1
DO 3 KK =1,ISTOP
IP = IMAX -KK
DO 3 II = 1,NUMSIG
3 XSTR(II,IP+1)=XSTR(II,IP)
DO 4 II=1,NUMSIG
JJ = IH(II)
XTH(II)=XSTR(II,JJ)
4 XSTR(II,1)=XT(II)
RETURN
END
SUBROUTINE PRTPLT(STRMAT,NROW,NCLM,IVAR,MAXLIN,NPAGE,NAXES,NOR1,
1 NOR2,NOR3,NOR4)
COMMON/FORMXY/X(1500),Y(1500)
DIMENSION STRMAT(NROW,NCLM),NPLT(4),NP(5),SYM(2)
DIMENSION Z(2)
DATA NW/6/
DATA SYM,PARL,PARR/1HY,1HX,1H( ,1H)/
IF(NAXES.EQ.0) RETURN
NPLT(1)=NOR1
NPLT(2)=NOR2
NPLT(3)=NOR3
NPLT(4)=NOR4
NPLOT = 0
IFM =1
IF(IVAR.GT.0) IFM= 2
DO 1 I=1,4
IF(NPLT(I).EQ.0) GO TO 1
NPLOT= NPLOT +1
NP(NPLOT)=NPLT(I)
1 CONTINUE
NPT = MAXLIN*NPLOT
DO 2 II =1,MAXLIN
DO 2 IC =1,NPLOT
KK = NP(IC)
NCRV = (IC-1)*MAXLIN + II
X(NCRV) = STRMAT(II,NAXES)
Y(NCRV)=STRMAT(II,KK)
2 CONTINUE
CALL PLOT(X,0,Y,0,Z,0,NPT,NPLOT,1,0,2,0,NPAGE)
WRITE(NW,3)((SYM(IFM),PARL,NP(IC),PARR,SYM(IFM),PARL,NAXES,
1 PARR) ,IC=1,NPLOT)
3 FORMAT(/,38H THE ABOVE PLOTS, RESPECTIVELY, ARE ,4(/,38X,A1,A1,I
12, A1,8H VS ,A1,A1,I2,A1))
RETURN
END
SUBROUTINE PRTTBL(STRMAT,NROW, NCLM,IVAR,MAXLIN,NC1,NC2,NC3,NC4,
1 NC5,NC6,NC7,NC8)
COMMON/ARYPRT/PRTARY(200,8)
DIMENSION STRMAT(NROW,NCLM),NPRT(8),SYM(2)
DATA SYM,PARL,PARR/1HY,1HX,1H( ,1H)/

```

PAGE NO. 11

DATA NW/6/

333 FORMAT(1H1,8H PAGE ,I2,/,8(7XA1,A1,I2,A1,2X))

334 FORMAT(1H1,8H PAGE ,I2,/,8(7X,2HX(,I2,1H),2X))

4 FORMAT(8(2X,F12.4))

NPRT(1)=NC1

NPRT(2)=NC2

NPRT(3)=NC3

NPRT(4)=NC4

NPRT(5)=NC5

NPRT(6)=NC6

NPRT(7)=NC7

NPRT(8)=NC8

IFM=1

IF (IVAR.GT.0) IFM=2

DO 1 I=1,MAXLIN

DO 1 JJ=1,8

KK = NPRT(JJ)

1 PRTRY(I,JJ)=STRMAT(I,KK)

NCOL = 0

DO 2 I=1,8

IF(NPRT(I).NE.0) NCOL =I

2 CONTINUE

NPAGE = 1

LINE = 1

WRITE(NW,333)NPAGE,((SYM(IFM),PARL,NPRT(I),PARR),I=1,NCOL)

DO 3 JLIN=1,MAXLIN

WRITE(NW,4) (PRTRY(JLIN,I),I=1,NCOL)

LINE = LINE+1

IF(LINE.LT.60) GO TO 3

LINE =1

NPAGE = NPAGE +1

WRITE(NW,333)NPAGE,((SYM(IFM),PARL,NPRT(I),PARR),I=1,NCOL)

3 CONTINUE

RETURN

END

SUBROUTINE PARADJ

COMMON Y(42)

COMMON/PARAM/P(99),H(9),NSYS

1 FORMAT(/,10X,*THE ISE OF THE OIL IS = *,E12.4,3X,*AND IS DEGREES F-
1-SECONDS*/)

WRITE(6,1)Y(9)

READ(1,2)P(18),P(19),P(20)

2 FORMAT(3F10.0)

RETURN

END

SUBROUTINE RKINT(LL,NSYS)

C THIS SUBROUTINE SOLVES DIFFERENTIAL EQUATIONS BY USING A RUNGE KUT
C METHOD

DIMENSION YU(20),Y(42),BET(3), DELY(4,20)

COMMON Y

DOUBLE PRECISION YU

IF (LL.NE.1) GO TO 1001

BET(1)=0.5

BET(2)=0.5

BET(3)=1.0

```

N2=NSYS+2
NP1=NSYS+1
XV=Y(NP1)
CALL DERFUN
DO 320 I=1,NSYS
320 YU(I)=Y(I)
1001 DO 1034 K=1,4
IF (K.EQ.1) GO TO 1002
CALL DERFUN
1002 DO 1340 I=1,NSYS
IPN2=I+N2
1340 DELY(K,I)=Y(N2)*Y(IPN2)
IF (K.EQ.4) GO TO 1034
DO 1350 I=1,NSYS
1350 Y(I)=YU(I)+BET(K)*DELY(K,I)
Y(NP1)=XV+BET(K)*Y(N2)
1034 CONTINUE
DO 1039 I=1,NSYS
DEL=(DELY(1,I)+2.0*DELY(2,I)+2.0*DELY(3,I)+DELY(4,I))/6.0
YU(I)=YU(I)+DEL
Y(I)=YU(I)
1039 CONTINUE
Y(NP1)=XV+Y(N2)
CALL DERFUN
XV=Y(NP1)
RETURN
END

```

SUBROUTINE PLOT(X,LX,Y,LY,Z,LZ,NPT,NPLOT,NCOPY,NCD,NDIM,NXTREM,
*NPAGE)

C PURPOSE

C THIS SUBROUTINE PLOTS UP TO TEN RELATIONSHIPS OF THE FORM $Y = F(X)$
C IT WILL ALSO PREPARE TOPOGRAPHIC MAPPINGS OF THREE DIMENSIONAL
C SURFACES.

C USAGE

C CALL PLOT(X,LX,Y,LY,Z,LZ,NPT,NPLOT,NCOPY,NCD,NDIM,NXTREM,NPAGE)

C X, Y, Z - COORDINATES OF DATA POINTS

C LX, LY, LZ - SCALING FACTORS. THE SCALES ON EACH AXIS MAY BE LINE
C ONE CYCLE LOG, TWO CYCLE LOG, ...SIX CYCLE LOG

C NPT - NUMBER OF POINTS TO BE PLOTTED

C NPLOT - NUMBER OF RELATIONSHIPS TO BE PLOTTED IN DIFFERENT

C SYMBOLS IF PLOTS ARE TWO DIMENSIONAL. IF A TOPOGRAPHIC
C MAPPING IS CALLED FOR NPLOT IS NUMBER OF CONTOURS TO BE PLOTT

C NCOPY - NUMBER OF DUPLICATE PLOTS TO BE MADE

C NCD - NUMBER OF INFORMATION CARDS TO BE READ. IF NCD IS ZERO NO
C CARDS ARE READ. OTHERWISE SEE WRITE UP.

C NDIM - IF THE RELATIONSHIPS WHICH ARE TO BE PLOTTED ARE TWO
C DIMENSIONAL NDIM IS 2, AND IF A TOPOGRAPHIC MAPPING IS DESIR
C NDIM IS 3.

C NXTREM - IF NXTREM IS 1, A CARD WITH THE LIMITING VALUES OF

C THE X, Y, AND Z SCALES IS READ. OTHERWISE THEY ARE

C CALCULATED WITHIN THE SUBROUTINE SO THAT ALL DATA POINTS WILL
C APPEAR ON THE PLOT.

C NPAGE - NUMBER OF PAGES THE PLOT IS TO COVER. IF NPAGE IS 1 THE P
C IS 50 LINES HIGH. EACH INCREASE OF 1 IN NPAGE CAUSES THE
C PLOT TO EXTEND ACROSS AN ADDITIONAL 66 LINES.

C SUBROUTINES REQUIRED

C THE FORTRAN IV SUBROUTINE POT IS INCLUDED AS PART OF THE
C PLOTTING PACKAGE.

C

REAL N

DIMENSION X(1),Y(1),Z(1),SX(7),TITLE(20),L(134),NCH(41),NH(41)
*,MOP(42),FM1(6),FM2(6),FM3(6),FM4(8),FM5(8),FM6(8),FMT(6),FRT(8)

1,SCL(3)

DIMENSION IZP(3000)

DATA FM1/4H(1H , 4HA1,F,4H9.2,,4H121A,4H1) ,4H /

DATA FM2/4H(1H , 4HA1,F,4H9.4,,4H121A,4H1) ,4H /

DATA FM3/4H(1H , 4HA1,1,4HPE9,,4H2,12,4H1A1),4H /

DATA FM4/4H(1X,,4H1PE1,4H6.2,,4H5E20,4H.2,E,4H15.2,4H) ,4H /

DATA FM6/4H(5X,,4HF9.2,4H,5(1,4H1X, ,4HF9.2,4H),9X,4H,F9,,4H2) /

DATA FM5/4H(5X,,4HF9.4,4H,5(1,4H1X, ,4HF9.4,4H),9X,4H,F9,,4H4) /

DATA NCH/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HA,1HB,1HC,1HD,
*1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1

HU,1HV,1HW,1HX,1HY,1H0,1H,1H\$,1H=,1H+,1H-/

DATA ND,NP,NM,NB/1H,,1H*,1H=,1H /

DATA SCL/1HX,1HY,1HZ/

1 FORMAT(20A4)

2 FORMAT(52A1,7A4)

3 FORMAT(1H ,26X,20A4)

5 FORMAT(132A1)

6 FORMAT(5H ALL ,A1,57H VALUES ARE NEGATIVE. LOG SCALE MAY NOT BE S
*PECIFIED.)

7 FORMAT(1PE17.2,E115.2)

8 FORMAT(1PE17.2,E61.2,E54.2)

9 FORMAT(1PE17.2,2E40.2,E35.2)

10 FORMAT(1PE17.2,3E30.2,E25.2)

11 FORMAT(1PE17.2,4E24.2,E19.2)

12 FORMAT(1HK,59X,7A4)

74 FORMAT(6E12.2)

76 FORMAT(1H1)

80 FORMAT(70H NPT(NO. OF PTS.) MUST BE EVENLY DIVISIBLE BY NPLOT(NO.
*OF CURVES.)

84 FORMAT(1X,36H ONE DIMENSIONAL PLOT NOT ALLOWED.)

94 FORMAT(42A1,10X,7A4)

D095I=1,41

95 NH(I)=NCH(1)

N=50+66*(NPAGE-1)

LLX=LX+1

NDD=NCD+1

GOTO(15,13,14,13),NDD

13 READ(1,1)(TITLE(I),I=1,20)

14 IF(NDD.LT.3)GOTO15

IF(NDIM.NE.3)GOTO93

READ(1,94)(MOP(I),I=1,42),TAB1,TAB2,TAB3,TAB4,TAB5,TAB6,TAB7

GO TO 81

93 READ(1,2)(MOP(I),I=1,42),(NH(I),I=1,10),TAB1,TAB2,TAB3,TAB4,TAB5,
1TAB6,TAB7

GOTO81

15 D082I=1,42

82 MOP(I)=NB

81 NCH(41)=NB

```

IF (FLOAT(NPT/NPLOT)-FLOAT(NPT)/FLOAT(NPLOT))101,85,101
85 IF (NXTREM.EQ.1)GOTO72
   YMIN=1.00E 75
   XMIN=YMIN
   ZMIN=YMIN
   YMAX=1.00E-78
XMAX=YMAX
ZMAX=YMAX
   DO51 I=1,NPT
   GOTO(103,52,53),NDIM
53 IF (Z(I).GE.ZMAX)ZMAX=Z(I)
   IF (Z(I).LE.ZMIN)ZMIN=Z(I)
52 IF (Y(I).GE.YMAX)YMAX=Y(I)
   IF (Y(I).LE.YMIN)YMIN=Y(I)
   IF (X(I).GE.XMAX)XMAX=X(I)
   IF (X(I).LE.XMIN)XMIN=X(I)
51 CONTINUE
   IF (LX.GT.0)GOTO900
XMIN=XMIN-(XMAX-XMIN)/75.0
XMAX=XMAX+(XMAX-XMIN)/75.0
900 IF (LY.GT.0)GOTO73
   YMIN=YMIN-(YMAX-YMIN)/30.0
   YMAX=YMAX+(YMAX-YMIN)/30.0
   GOTO73
72 READ(1,74)XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX
73 IF (YMAX.GT.10000.0.OR.YMIN.LT.(-10000.0))GOTO60
   IF (YMAX-YMIN.LT.5.0)GOTO60
   IF (YMAX.GT.10.0.OR.YMIN.LT.(-10.0))GOTO62
   IF (YMAX.LT.0.01.OR.ABS(YMIN).LT.0.01)GOTO60
   DO59 I=1,6
59 FMT(I)=FM2(I)
   GOTO65
60 DO61 I=1,6
61 FMT(I)=FM3(I)
   GOTO65
62 DO63 I=1,6
63 FMT(I)=FM1(I)
65 IF (XMAX.GT.10000.0.OR.XMIN.LT.(-10000.0))GOTO67
   IF (XMAX-XMIN.LT.5.0)GOTO67
   IF (XMAX.GT.10.0.OR.XMIN.LT.(-10.0))GOTO69
   IF (XMAX.LT.0.01.OR.ABS(XMIN).LT.0.01)GOTO67
   DO66 I=1,8
66 FRT(I)=FM5(I)
   GOTO70
67 DO68 I=1,8
68 FRT(I)=FM4(I)
   GOTO70
69 DO71 I=1,8
71 FRT(I)=FM6(I)
70 IF (LX.GT.0)GOTO17
   CX=120./(XMAX-XMIN)
   SX(1)=XMIN
   SX(7)=XMAX
   U=XMIN
   DO16K=2,6

```



```

      U=(XMAX-XMIN)/6.+U
16  SX(K)=U
      GOTO19
17  XLX=LX
      IF (XMAX.LE.0.0) GOTO100
      XMIN=10.0** (ALOG10(XMAX*9.999999)-FLOAT(LX))
      MS=1
      CX=120./XLX
      NX=ALOG10(XMIN)
      IF (XMIN.LT.1.0) NX=ALOG10(XMIN)-0.9999999
      XMIN=10.0**NX
      DO18K=1,LLX
18  SX(K)=10.**(NX+K-1)
19  CALLPOT(X,XMIN,LX,NPT,0,120.,CX)
      IF (LY.GT.0) GOTO20
      CY=N/(YMAX-YMIN)
      GOTO21
20  YLY=LY
      IF (YMAX.LE.0.0) GOTO100
      YMIN=10.0** (ALOG10(YMAX*9.999999)-FLOAT(LY))
      MS=2
      CY=N/YLY
      KY=CY
      NY=ALOG10(YMIN)
      IF (YMIN.LT.1.0) NY=ALOG10(YMIN)-0.9999999
      YMIN=10.0**NY
21  CALLPOT(Y,YMIN,LY,NPT,1,N,CY)
      IF (NDIM.LT.3) GOTO24
      FPLOT=NPLOT
      IF (LZ.GT.0) GOTO22
      CZ=FPLOT/(ZMAX-ZMIN)
      GOTO23
22  ZLZ=LZ
      IF (ZMAX.LE.0.0) GOTO100
      ZMIN=10.0** (ALOG10(ZMAX*9.999999)-ZLZ)
      MS=3
      CZ=FPLOT/ZLZ
      IF (ZMIN.GE.1.0) GO TO 23
      IA=ALOG10(ZMIN)-.9999999
      ZMIN=10.0**IA
23  CALL POT(Z,ZMIN,LZ,NPT,0,FPLOT,CZ)
      DO77I=1,NPT
      IZ=Z(I)+1.0
      IZP(I) = NCH(IZ)
77  CONTINUE
      GOTO99
24  NPN=NPT/NPLOT
      DO 78I=1,NPLOT
      JJ=(I-1)*NPN+1
      JK=I*NPN
      DO78J=JJ,JK
      IZP(J)=NH(I)
78  CONTINUE
99  DO102I=1,NPT
      DO102J=I,NPT

```

```

IF (Y(I).GE.Y(J))GOTO102
A=Y(I)
B=X(I)
Y(I)=Y(J)
X(I)=X(J)
Y(J)=A
X(J)=B
IC = IZP(I)
IZP(I) =IZP(J)
IZP(J)=IC
102 CONTINUE
D056N0=1,NCOPY
M1=1
M2=1
LL=0
LYY=LY
TT=N
WRITE(6,76)
IF (NDD.EQ.1.OR.NDD.EQ.3)GO TO 97
WRITE(6,3) (TITLE(I),I=1,20)
97 NN= N+1.
D043KK=1,NN
M=1
NNN=NPN
JED=1
T=INT(N)-KK+1
D025J=1,133
25 L(J)=NB
L(133)=ND
L(13)=NP
IF (LY.GT.0)GOTO26
L(133)=NP
IF (T.GT.TT)GOTO30
SCALE=T/CY+YMIN
IF (T.LE.0.)SCALE=YMIN
M=0
TT=TT-5.
GOTO30
26 GOTO(27,27,28,28,27,28),LY
27 SS=KY*LYY
GOTO29
28 SS=KY*LYY+1
29 L(13)=ND
IF (T.GT.SS)GOTO30
SCALE=10.** (NY+LYY)
M=0
LYY=LYY-1
L(13)=NP
L(133)=NP
30 IF (N.EQ.T)GOTO31
IF (0..NE.T)GOTO37
31 D032J=14,133
32 L(J)=NM
IF (LX.GT.0)GOTO34
D033J=13,133,10

```

```

33 L(J)=NP
   GOT036
34 KX=120/LX
   D035J=13,133,KX
35 L(J)=NP
36 IF(N.EQ.T)L(133)=ND
37 LL=LL+1
   IF(LL.GT.NPT)GOT079
   IF(Y(LL)-T)79,92,37
92 J=X(LL)
   L(J+13)=IZP(LL)
   GOT037
79 LL=LL-1
   IF(M1.GT.4+(INT(N)-50)/2 .AND.M1.LT.47+(INT(N)-50)/2 )GOT054
   L(2)=NB
   GOT055
54 L(2)=MOP(M2)
   M2=M2+1
55 M1=M1+1
41 IF(M.EQ.1)GOT042
   WRITE(6,FMT)L(2),SCALE,(L(J),J=13,133)
   GOT043
42 WRITE(6,5)(L(J),J=1,11),(L(J),J=13,133)
43 CONTINUE
   GOT0(44,45,46,47,48,49,44),LLX
44 WRITE(6,FRT)(SX(K),K=1,7)
   GOT050
45 WRITE(6,7)(SX(K),K=1,LLX)
   GOT050
46 WRITE(6,8)(SX(K),K=1,LLX)
   GOT050
47 WRITE(6,9)(SX(K),K=1,LLX)
   GOT050
48 WRITE(6,10)(SX(K),K=1,LLX)
   GOT050
49 WRITE(6,11)(SX(K),K=1,LLX)
50 IF(ND.LT.3)GOT056
   WRITE(6,12)TAB1,TAB2,TAB3,TAB4,TAB5,TAB6,TAB7
56 CONTINUE
83 RETURN
100 WRITE(6,6)SCL(MS)
   RETURN
101 WRITE(6,80)
   RETURN
103 WRITE(6,84)
   RETURN
   END
   SUBROUTINEPOT(V,VMIN,LV,NP,J,VC,C)
   DIMENSIONV(1)
   IF(LV.GT.0)GOT02
   D01I=1,NP
   1 V(I)=FLOAT( INT(C*(V(I)-VMIN)+.5))
   GOT04
   2 VMIN=10.0** ( INT(ALOG10(VMIN)))
   D03I=1,NP

```

PAGE NO. 18

~~IF (V(I).LE.0.0)GOTO3~~

~~V(I)=FLOAT(INT(C*(ALOG(V(I)/VMIN)/2.302585)+.5))~~

~~VT=V(I)~~

~~3 CONTINUE~~

~~4 DO6I=1,NP~~

~~IF (V(I).LT.0.)GOTO5~~

~~IF (V(I).LE.VC)GOTO6~~

~~5 V(I)=VC+1.~~

~~6 CONTINUE~~

~~RETURN~~

~~END~~

Appendix D

COMPUTER PROGRAM FOR A DIRECT DIGITAL CONTROL LOOP

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

PROGRAM DDCSYM(INPUT,OUTPUT,TAPE1=INPUT,TAPE6=OUTPUT)

C THIS IS A SIMULATION OF THE BENDIX MOLD HEATING AND COOLING SYSTEM
C UNDER DDC CONTROL VELOCITY CONTROLLER HEREIN.
C

COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/XSYSTO/YSTOR(800,8),XSTOR(800,8),NXCUM(8),NYCUM(8)
COMMON/CONINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
COMMON/COMPPR/NSAMPL,NTIME,E1,E2,T01,T02,DE,DDE,DTO
COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
COMMON/PSOU/GAV(2),XGRAPH(8,2),YGRAPH(8,2)
COMMON/OUTCON/NGY(8),NTY(8),NGX(8),NTX(8),IADJ,NNTX,NNTY,IPAGE
COMMON Y(42),AMASS

C NDATA(1) =1 WILL READ IN MOLD IDENTITY AND INTEGRATION CONTROL.
C NDATA(2) =1 WILL READ IN INITIAL CONDITIONS OF THE Y'S.
C NDATA(3) =1 WILL READ IN THE DELAYS.
C NDATA(4) =1 WILL READ IN THE PARAMETER LIST.
C NDATA(5) =1 WILL READ IN THE SUBSCRIPTS OF THE X AND Y'S TO BE
C PRINTED AND PLOTTED.
C NDATA(6) =1 WILL PRINT OUT INTEGRATION CONTROLS, MOLD IDENTITY,
C THE DELAYS, AND THE PARAMETER LIST.
C NDATA(7) =0 WILL SET THE INITIAL VALUE OF THE Y'S EQUAL TO
C THEIR FINAL VALUE FROM THE PREVIOUS RUN.
C NDATA(7) =1 WILL SET THE INITIAL VALUE OF THE Y'S EQUAL TO
C THE VALUE ON THE INITIAL CONDITION DATA CARDS.
C NDATA(8) =1 SETS IADJ = 1 WHICH WILL ALLOW DYSIMP TO CALL PARADJ
C NDATA(9) =1 SETS IBUG = 1 WHICH CAUSES THE X'S AND Y'S TO BE
C PRINTED AFTER EACH RETURN FROM RUNGE KUTTA.
C NDATA(10)=1 ALLOWS THE GAIN MULTIPLIER KM TO BE OTHER THAN 1 IF
C THE ABSOLUTE ERROR EXCEEDS THE PARAMETER P(43).
C NDATA(11)=1 CAUSES THE PROPORTIONAL MODE TO BE SET = 0.0 IF A
C SET POINT CHANGE OCCURS FOR THE 1 ST SAMPLE PERIOD.
C NDATA(12)=1 CAUSES THE ACTUAL FLUID TEMP RATHER THAN THE ERROR
C TO BE USED IN THE DERIVATIVE MODE.
C NDATA(13)=1 CAUSES A NEW SET POINT RECID TO BE READ IN.
C NDATA(14)=0 CAUSES THE SCALE SUBROUTINE TO SET SCALE LIMITS FOR
C EACH PLOT.
C NDATA(14)=1 CAUSES THE SCALE ROUTINE TO READ IN THE UPPER
C AND LOWER SCALE LIMITS FOR ALL PLOTS.
C NDATA(14)=2 CAUSES THE SCALE ROUTINE TO SEARCH EACH PLOT FOR
C THE MAX AND MIN LIMIT. IT THEN PLOTS ALL PLOTS
C WITHIN THOSE TWO LIMITS.
C NDATA(14)=3 CAUSES THE SCALE ROUTINE TO READ IN THE UPPER AND
C LOWER LIMITS FOR EACH PLOT.
C NDATA(15)=1 CAUSES INTEGRAL CUT OUT TO OCCUR WHEN THE PROCESS
C VARIABLE RATE EXCEEDS 11 DEGREES PER SECOND.

C THE LOOP FLOW RATE X(44) MUST BE DETERMINED USING PROGRAM
C PFLOW OR MFLOW.
C

```

100 FORMAT(1H1,5X,48HFLUID FEEDBACK SIMULATION UNDER DDC          MODI,
      *17HFICATION NUMBER 3,11X,8HRUN NO. ,I2,4X,9HPAGE NO. ,I3)
101 FORMAT(1X,51HTHE FOLLOWING ARE THE INITIAL CONDITIONS OF THE Y'S/)
102 FORMAT(1X,52HTHE FOLLOWING ARE THE VALUES OF X AFTER THE 1 ST CAL,
      *42HCULATIONS BUT BEFORE THE 1 ST INTEGRATION./)
103 FORMAT(4(5X,2HY(,I2,4H) = ,F12.5,2X))
104 FORMAT(4(5X,2HX(,I2,4H) = ,F12.5,2X))
105 FORMAT(///,14X,45HTHE FLUID LOOP CONTROLLER GAIN CONSTANTS ARE,/)
106 FORMAT(14X,5HKM = ,F3.0,2X,5HKP = ,F7.3,3X,5HKI = ,F7.3,3X,
      *5HKD = ,F7.3/)
107 FORMAT(25X,42H* THE FLUID LOOP SET POINT RECIDE IS      *)
108 FORMAT(25X,42(1H*))
109 FORMAT(25X,42H* RAMP * SETPOINT * DWELL * SEGMENT *)
110 FORMAT(25X,42H* F/MIN * DEGREE F * MIN * NUMBER *)
111 FORMAT(25X,2H* ,F5.1,2H * ,3X,F5.1,2X,3H* ,F5.2,3H * ,4X,I2,5X,
      *1H*)
112 FORMAT(20(1X,I1))
113 FORMAT(1X,120(1H*))
114 FORMAT(5(/))
115 FORMAT(///,10X,46HTHE OUTPUT INTERVAL IS TOO SMALL OUTPUT ARRAY ,
      *30HEXCEEDS 800 . NOUTPT RESET TO ,2X,I5///)
116 FORMAT(/,10X,29HTHE ISE OF THE FLUID LOOP IS ,1PE14.7/10X,
      *29HTHE IAE OF THE FLUID LOOP IS ,1PE14.7/)
117 FORMAT(1X,24(1H*),11H KM WILL = ,F5.0,23H IF THE ABSOLUTE ERROR ,
      *8HEXCEEDS ,F5.1,10H PER CENT ,43(1H*))
118 FORMAT(1X,24(1H*),44H THE PROPORTIONAL MODE WILL BE SET = 0.0 FOR,
      *46H THE FIRST SAMPLE PERIOD ON SET POINT CHANGES ,15(1H*))
119 FORMAT(1X,24(1H*),44H THE FLUID TEMPERATURE RATHER THAN THE ERROR,
      *37H WILL BE USED IN THE DERIVATIVE MODE ,24(1H*))
120 FORMAT(1X,24(1H*),42H INTEGRAL CUT OUT WILL OCCUR IF THE FLUID ,
      *41HTEMPERATURE RATE EXCEEDS 11 DEGREES/SEC. ,21(1H*))
121 FORMAT(//)

```

C

```

      IRUN = 0
1  IPAGE = 0
      IRUN = IRUN + 1
      READ(1,112)(NDATA(I),I=1,20)
      IF(EOF(1))999,2
2  IBUG = NDATA(9)
      NSUM = 0
      DO 3 I=1,20
3  NSUM = NSUM + NDATA(I)
      IF(NSUM.EQ.0) GO TO 999
      CALL INPUT
      IF(NUMSIG.EQ.0) GO TO 4
      CALL INTFUN
4  IF(NX.EQ.0) GO TO 5
      CALL INTCAL
5  CALL DEREUN
      WRITE(6,101)
      WRITE(6,113)
      WRITE(6,103)(I,Y(I),I=1,NSYS)
      WRITE(6,113)
      WRITE(6,102)
      WRITE(6,113)

```



```

WRITE(6,104) (I,X(I),I=1,NX)
WRITE(6,113)
IPAGE = IPAGE + 1
WRITE(6,100) IRUN,IPAGE
WRITE(6,113)
WRITE(6,105)
WRITE(6,113)
WRITE(6,106) P(1),P(2),P(3),P(4)
WRITE(6,113)
WRITE(6,114)
WRITE(6,108)
(#I%E(6,107)
WRITE(6,108)
WRITE(6,109)
WRITE(6,110)
WRITE(6,108)
WRITE(6,111) (RAM(I),STPT(I),TIM(I),I,I=1,NSEG)
WRITE(6,108)
WRITE(6,121)
IF(NDATA(10).NE.0) WRITE(6,117) P(1),P(43)
IF(NDATA(11).NE.0) WRITE(6,118)
IF(NDATA(12).NE.0) WRITE(6,119)
IF(NDATA(15).NE.0) WRITE(6,120)
IPAGE = IPAGE + 1

```

C

```

NSTEP = (ENDTIM-BEGTIM)/DELT
6 NWRITE = IFIX(NOUTPT/DELT+0.5)
NMOG = NSTEP/NWRITE
IF(NMOG.GT.800) GO TO 7
GO TO 8
7 NOUTPT = NSTEP/(800*DELT)+1
WRITE(6,115) NOUTPT
GO TO 6

```

C

```

8 JI = 1
DO 9 I=1,NNTX
9 XSTOR(JI,I) = X(NXCUM(I))
DO 10 I=1,NNTY
10 YSTOR(JI,I) = Y(NYCUM(I))
NI=0
DO 20 KK=1,NSTEP
NI=NI+1
CALL RKINT(KK,NY)
IF(IRUG.EQ.0) GO TO 11
NSI = NSYS + 1
WRITE(6,103) NSI,Y(NSI)
WRITE(6,103) (I,Y(I),I=1,NSYS)
WRITE(6,104) (I,X(I),I=1,NX)
11 IF(NX.EQ.0) GO TO 12
CALL INTCAL
12 IF(NI.NE.NWRITE) GO TO 20
NI = 0
IF(IRUG.EQ.0) GO TO 13
WRITE(6,103) NSI,Y(NSI)
WRITE(6,103) (I,Y(I),I=1,NSYS)

```

```

WRITE(6,104)(I,X(I),I=1,NX)
13 JI = JI + 1
   IF(NNTX.EQ.0) GO TO 15
   DO 14 I=1,NNTX
14 XSTOR(JI,I) = X(NXCUM(I))
15 IF(IGX.EQ.0) GO TO 17
   DO 16 I=1,NNTX
   IF(XSTOR(JI,I).GE.XGRAPH(I,2)) XGRAPH(I,2) = XSTOR(JI,I)
   IF(XSTOR(JI,I).LE.XGRAPH(I,1)) XGRAPH(I,1) = XSTOR(JI,I)
16 CONTINUE
17 IF(NNTY.EQ.0) GO TO 20
   DO 18 I=1,NNTY
18 YSTOR(JI,I) = Y(NYCUM(I))
   IF(IGY.EQ.0) GO TO 20
   DO 19 I=1,NNTY
   IF(YSTOR(JI,I).GE.YGRAPH(I,2)) YGRAPH(I,2) = YSTOR(JI,I)
   IF(YSTOR(JI,I).LE.YGRAPH(I,1)) YGRAPH(I,1) = YSTOR(JI,I)
19 CONTINUE
20 CONTINUE
   NOUT = JI
   IF(ITY.EQ.0) GO TO 21
   CALL PRTTBL(JI,0)
21 IF(ITX.EQ.0) GO TO 22
   CALL PRTTBL(JI,1)
22 IF(IGY.EQ.0) GO TO 23
   NGRAPH = IGY
   CALL PLOT(YSTOR,1,YGRAPH)
   IPAGE = IPAGE + 1
23 IF(IGX.EQ.0) GO TO 24
   NGRAPH = IGX
   CALL PLOT(XSTOR,0,XGRAPH)
   IPAGE = IPAGE + 1
24 IF(IADJ.NE.0) CALL PARADJ
   WRITE(6,116) Y(6),Y(7)
   GO TO 1
999 STOP
END
SUBROUTINE INPUT

```

```

C
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/XSYSTO/YSTOR(800,8),XSTOR(800,8),NXCUM(8),NYCUM(8)
COMMON/CONINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
COMMON/OUTCON/NGY(8),NTY(8),NGX(8),NTX(8),IADJ,NNTX,NNTY,IPAGE
COMMON/ARYPRI/PRTARY(800,8)
COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
COMMON/COMPPR/NSAMPL,NTIME,E1,E2,T01,T02,DE,ODE,DTO
COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
COMMON/PSOU/GAV(2),XGRAPH(8,2),YGRAPH(8,2)
COMMON Y(42),AMASS

```

```

C
DIMENSION MSC(2)
C

```

DATA DASH/1H-/

DATA MSC/6HPUNCH ,6HCAVITY/

C

```

100 FORMAT(1H1,5X,48HFLUID FEEDBACK SIMULATION UNDER DDC          MODI,
      *17HFICATION NUMBER 3,11X,8HRUN NO. ,I2,4X,9HPAGE NO. ,I3)
102 FORMAT(1X,120(1H*)/)
101 FORMAT(8F10.0)
103 FORMAT(5X,27HTHE ORDER OF THE SYSTEM IS ,I12,17X,9HMOLD NO. ,A4,1X
      *.I6,1X,A1,1X,I3)
104 FORMAT(5X,21HTHE BEGINNING TIME IS,6X,F12.3,2X,10HIN SECONDS,5X,
      *13HMOLD SECTION ,A6)
105 FORMAT(5X,23HTHE TERMINATION TIME IS,4X,F12.3,2X,10HIN SECONDS,5X,
      *9HMOLD.MASS,5X,F7.0,1X,3HLB.)
106 FORMAT(5X,27HTHE INTEGRATION STEP SIZE ,F12.5,2X,10HIN SECONDS,
      *5X,14HMOLD TEST NO. ,I2)
107 FORMAT(5X,22HTHE OUTPUT INTERVAL IS,5X,I12,2X,10HIN SECONDS)
108 FORMAT(5X,28HTHE NUMBER OF X EQUATIONS IS,8X,I3)
109 FORMAT(4(5X,A1,A1,I2,A1,3H = ,F12.5,2X))
110 FORMAT(///)
111 FORMAT(8I3)
112 FORMAT(3I5,3F10.0)
113 FORMAT(I2)
114 FORMAT(I1)
115 FORMAT(I6,1X,I1,2X,I2,4X,A4,1X,I3)

```

C

```

DO 1 I=1,8
DO 1 J=1,2
XGRAPH(I,J) = 0.0
1 YGRAPH(I,J) = 0.0
IF(NDATA(5).EQ.0) GO TO 3
DO 2 I=1,8
NGX(I) = 0
NGY(I) = 0
NTX(I) = 0
NTY(I) = 0
NXCUM(I) = 0
2 NYCUM(I) = 0
ITY=ITX=IGY=IGX=0
NNTX=NNTY=0
3 IF(NDATA(1).EQ.0) GO TO 4
READ(1,115)MMFR,MSFR,MTEST,MDES,IDMO
READ(1,112)NY,NOUTPT,NX,BEGTIM,ENDTIM,DELT
NSYS = NY
Y(NSYS+2) = DELT
Y(NSYS+1) = BEGTIM
4 IF(NDATA(2).EQ.0) GO TO 5
READ(1,101)(Y(I),I=1,NY)
NP2 = NY + 2
NIMAX = 2*NY + 2
GO TO 7
5 IF(NDATA(7).NE.0) GO TO 7
DO 6 I=1,NP2
6 Y(I) = YINT(I)
7 DO 8 I=1,NP2
8 YINT(I) = Y(I)

```

```

IF (NDATA(3).EQ.0) GO TO 9
READ(1,114) NUMSIG
READ(1,101) (H(I), I=1, NUMSIG)
9 IF (NDATA(4).EQ.0) GO TO 10
READ(1,113) NP
READ(1,101) (P(I), I=1, NP)
10 IADJ = 0
IF (NDATA(8).NE.0) IADJ = 1
IF (NDATA(5).EQ.0) GO TO 19
READ(1,111) (NTY(I), I=1, 8)
READ(1,111) (NGY(I), I=1, 8)
READ(1,111) (NTX(I), I=1, 8)
READ(1,111) (NGX(I), I=1, 8)
DO 11 I=1, 8
IF (NTX(I).NE.0) ITX = ITX + 1
IF (NTY(I).NE.0) ITY = ITY + 1
IF (NGX(I).NE.0) IGX = IGX + 1
IF (NGY(I).NE.0) IGY = IGY + 1
11 CONTINUE
DO 12 I=1, 8
IF (NTX(I).EQ.0) GO TO 12
NNTX = NNTX + 1
NXCUM(NNTX) = NTX(I)
12 CONTINUE
DO 14 I=1, 8
IF (NGX(I).EQ.0) GO TO 14
DO 13 J=1, NNTX
IF (NGX(I).EQ.NXCUM(J)) GO TO 14
13 CONTINUE
NNTX = NNTX + 1
IF (NNTX.GT.8) GO TO 15
NXCUM(NNTX) = NGX(I)
14 CONTINUE
15 IF (NNTX.GT.8) NNTX = 8
DO 16 I=1, 8
IF (NTY(I).EQ.0) GO TO 16
NNTY = NNTY + 1
NYCUM(NNTY) = NTY(I)
16 CONTINUE
DO 18 I=1, 8
IF (NGY(I).EQ.0) GO TO 18
DO 17 J=1, NNTY
IF (NGY(I).EQ.NYCUM(J)) GO TO 18
17 CONTINUE
NNTY = NNTY + 1
IF (NNTY.GT.8) GO TO 19
NYCUM(NNTY) = NGY(I)
18 CONTINUE
19 IF (NNTY.GT.8) NNTY = 8
IPAGE = IPAGE + 1
IP = 1
IF (MSFR.NE.0) IP = 2
WRITE(6,100) IRUN, IPAGE
WRITE(6,102)
WRITE(6,110)

```

```

WRITE(6,103)NY,MDES,MMFR,DASH,IDMO
WRITE(6,104)BEGTIM,MSC(IP)
WRITE(6,105)ENDTIM,P(32)
WRITE(6,106)DELT,MTEST
WRITE(6,107)NOUTPT
WRITE(6,108)NX
WRITE(6,102)
WRITE(6,109)(SYMB(12),PARL,I,PARR,H(I),I=1,NUMSIG)
WRITE(6,102)
WRITE(6,109)(SYMB(13),PARL,I,PARR,P(I),I=1,NP)
WRITE(6,102)
IF(NDATA(13).NE.0) CALL RECIPE
RETURN
END
SUBROUTINE RKINT(LL,NSYS)

```

```

C
C THIS ROUTINE SOLVES DIFFERENTIAL EQUATIONS USING THE RUNGE KUTTA
C METHOD.
C

```

```

C
C DIMENSION YU(20),Y(42),BET(3),DELY(4,20)
COMMON Y
DOUBLE PRECISION YU

```

```

C
C IF(LL.NE.1) GO TO 2
BET(1) = 0.5
BET(2) = 0.5
BET(3) = 1.
N2 = NSYS + 2
N1 = NSYS + 1
XV = Y(N1)
CALL DERFUN
DO 1 I=1,NSYS
1 YU(I) = Y(I)
2 DO 6 K=1,4
IF(K.EQ.1) GO TO 3
CALL DERFUN
3 DO 4 I=1,NSYS
I2 = I+N2
4 DELY(K,I) = Y(N2)*Y(I2)
IF(K.EQ.4) GO TO 6
DO 5 I=1,NSYS
5 Y(I) = YU(I)+BET(K)*DELY(K,I)
Y(N1) = XV + BET(K)*Y(N2)
6 CONTINUE
DO 7 I = 1,NSYS
DEL = (DELY(1,I)+2.*DELY(2,I)+2.*DELY(3,I)+DELY(4,I))/6.
YU(I) = YU(I) + DEL
Y(I) = YU(I)
7 CONTINUE
Y(N1) = XV + Y(N2)
CALL DERFUN
XV = Y(N1)
RETURN
END
SUBROUTINE SPADJ

```

```

COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4

```

```

THIS IS THE SET POINT ADJUSTMENT ROUTINE. THE RECIDE IS READ IN
FROM SUBROUTINE RECIPE. SOME OF THE FEATURES OF THIS ROUTINE ARE:

```

- 1) NO RAMP MAY BE SPECIFIED IN SEGMENT 1.
- 2) DWELL PERIODS ARE IN MINUTES AND APPLY ONLY TO SPECIFIED SEG.
- 3) IF DESIRED THE SET POINT MAY BE SPECIFIED AS ZERO (EXCEPT IN SEGMENT 1) AND THAT SEGMENT WILL USE THE SET POINT FROM THE PRECEDING SEGMENT.
- 4) TWO CONSECUTIVE SEGMENTS MUST HAVE AT LEAST ONE SET POINT SPECIFIED.
- 5) THE DWELL IN ANY ONE SEGMENT MAY BE SPECIFIED AS ZERO AND THAT SEGMENT WILL BE PASSED OVER.
- 6) A DWELL MUST BE SPECIFIED IN A RAMP SEGMENT. THIS OPTION IS USUALLY USED JUST BEFORE A RAMP TO DEFINE THE START POINT FOR THE RAMP.
- 7) NO TWO CONSECUTIVE SEGMENTS MAY BE SPECIFIED WITH A ZERO DWELL.
- 8) POSITIVE RAMPS ARE DEFINED BY A PLUS BEFORE THE RAMP MAGNITUDE OR BY NO SIGN.
- 9) A NEGATIVE RAMP MUST HAVE A MINUS SIGN PRIOR TO THE MAGNITUDE DEFINITION.
- 10) THE RAMP SEGMENT MUST HAVE A SEGMENT TERMINATION SET POINT AND A TERMINATION TIME LIMIT. THE TIME LIMIT MAY BE SPECIFIED GREATER THAN THE ACTUAL RAMP TIME.

```

IF(JFLAG.EQ.1) GO TO 2

```

```

STIME = 0.0

```

```

DWELL(1) = TIM(1)*60.

```

```

JFLAG = 1

```

```

DO 1 I=2,NSEG

```

```

STIME = TIM(I)*60.

```

```

1 DWELL(I) = DWELL(I-1) + STIME

```

```

2 IF(X(1).GE.DWELL(I)) 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 = RAM(I)

```

```

IF(IRAMP.NE.0) GO TO 6

```

```

GO TO 7

```

```

6 BETM = DWELL(I-1)

```

```

SLP = RAM(I)/60.

```

```

SP = SLP*(X(1)-BETM) + STPT(I-1)

```

```

IF(STPT(I-1).LT.50.) SP = SLP*(X(1)-BETM) + STPT(I-2)

```

```

IF(ABS(SP-STPT(I)).LT.0.5.AND.X(1).LT.DWELL(I)) GO TO 12

```

```

IF(RAM(I).LT.0.0) GO TO 11

```

```

GO TO 10

```

```

7 SP = STPT(I)

```

```

      IF(SP.GE.50.) GO TO 9
      SP = STPT(I-1)
      IF(I.GT.2) GO TO 8
      GO TO 14
8     CHOC = SP-STPT(I-2)
      IF(CHOC.LT.0.0) GO TO 11
      GO TO 10
9     CHOC = SP-STPT(I-1)
      IF(CHOC.LT.0.0) GO TO 11
10    P1 = P(35)
      P2 = P(37)
      P3 = P(38)
      GO TO 14
11    P1 = P(36)
      P2 = P(39)
      P3 = P(40)
      GO TO 14
12    ADJTIM = DWELL(I) - X(1)
      SP = STPT(I+1)
      IF(SP.LE.50.) SP = STPT(I)
      DWELL(I) = DWELL(I)-ADJTIM
      I=I+1
      IF(I.GT.NSEG) GO TO 14
      DO 13 K=I,NSEG
13    DWELL(K) = DWELL(K-1) + TIM(K)*60.
      CHOC = SP-STPT(I-1)
      IF(SP.EQ.STPT(I-1)) CHOC = SP - STPT(I-2)
      IF(CHOC.LT.0.0) GO TO 11
      GO TO 10
14    X(42) = SP
      X(43) = SP*P(29) + P(30)
      RETURN
      END
      SUBROUTINE INTFUN

```

C

```

COMMON/CONINT/BEGTIM,ENDTIM,DELT,NY,NX,NWRITE,YINT(42)
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/COMPPR/NSAMPL,NTIME,E1,E2,T01,T02,DE,DDE,DT0
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4
COMMON Y(42),AMASS

```

C

```

      IMAX = 0
      DO 1 I=1,NUMSIG
      IH(I) = H(I)/Y(NSYS+2)+1
      IF(IH(I).GE.IMAX) IMAX = IH(I)
1     CONTINUE
      DO 2 I=1,NX
2     X(I) = 0.0
      P1 = P(35)
      P2 = P(37)
      P3 = P(38)
      ID1 = NDATA(10)
      ID2 = NDATA(11)

```

```

ID3 = NDATA(12)
ID4 = NDATA(15)
SAMPLE = P(41)/DELT + 0.5
NSAMPL = SAMPLE
NTIME = 0
X(1) = BEGTIM
X(44) = 575.
X(45) = 90.
X(20) = P(16)*X(44)*X(44) + P(17)*X(44) + P(18)
X(21) = P(16)*X(45)*X(45) + P(17)*X(45) + P(18)
X(46) = P(42)/60.
X(42) = P(44)
X(43) = P(44)*P(29) + P(30)
X(41) = X(43)
T01 = X(41)
T02 = X(41)
E1 = E2 = 0.0
SGC = X(45)*P(9) + P(10)
X(10) = 12.
X(11) = 9.
X(30) = P(44)
X(29) = P(44)
X(35) = P(16)*P(44)*P(44) + P(17)*P(44) + P(18)
X(28) = P(44)
X(27) = X(35)
X(31) = P(44)
AMASS = (P(15)*P(44) + P(16))*X(46)
X(36) = X(35)
X(37) = P(44)
SGH = X(44)*P(9) + P(10)
X(47) = P(13)*SQRT(P(12)/SGC)
X(48) = P(13)*SQRT(P(11)/SGH)
DO 3 I = 1,IMAX
  XSTR(1,I) = X(28)
  XSTR(2,I) = X(29)
  XSTR(3,I) = X(27)
  XSTR(4,I) = AMASS
  XSTR(5,I) = X(31)
3 XSTR(6,I) = X(35)
DO 4 I=1,NUMSIG
4 XTH(I) = XSTR(I,IMAX)
RETURN
END

```

SUBROUTINE VELCON

C

```

COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/COMPPR/NSAMPL,NTIME,E1,E2,T01,T02,DE,DDE,DTO
COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4

```

C

```

SP1 = SP
CALL SPADJ
E2 = E1

```



```

E1 = X(2)
T02 = T01
T01 = X(41)
X(41) = X(40)*P(27) + P(28)
X(2) = X(43) - X(41)
DE = X(2) - E1
DDE = X(2) - 2.*E1 + E2
DT0 = -(X(41)-2.*T01+T02)
IF(ID3.EQ.0) GO TO 1
X(6) = P(4)*60.*DT0/P(41)
GO TO 2
1 X(6) = P(4)*60.*DDE/P(41)
2 X(5) = P(3)*P(41)*X(2)/60.
X(4) = DE
NAVO = SP - SP1
IF(ID2.EQ.1.AND.NAVO.NE.0) X(4)=0.0
DPV = X(41) - T01
IF(ID4.EQ.1.AND.DPV.GT.4.) X(5) = 0.0
X(7) = X(4) + X(5) + X(6)
IF(ABS(X(2)).LT.P(43)) X(3) = 1.
IF(ID1.NE.1) X(3) = 1.
IF(ID1.EQ.1.AND.ABS(X(2)).GE.P(43)) X(3) = P(1)
X(8) = X(3)*P(2)
X(9) = X(8)*X(7)
X(9) = ALIM(X(9),-50.,50.)
X(10) = AIMM(X(9),X(10))
X(10) = ALIM(X(10),4.,20.)
NTIME = 0
RETURN
END
SUBROUTINE RECIPE

```

```

C
COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
C
C THIS SUBROUTINE READS IN THE SET POINT RECIPE.
C RAM(1) = RAMP RATE IN DEGREES/MIN
C STPT(1) = FLUID SET POINT
C TIM(1) = DWELL TIME IN MINUTES
C NSEG = NUMBER OF SEGMENTS
C THIS ROUTINE IS CALLED FROM INPUT IF NDATA(13) = 1.
C
1 FORMAT(3F10.0)
2 FORMAT(I2)
C
READ(1,2)NSEG
C
DO 3 I=1,NSEG
3 READ(1,1)RAM(I),STPT(I),TIM(I)
C
JFLAG = 0
RETURN
END
SUBROUTINE PLOT(SAVE,NSS,GRAPH)
C
REAL LINE(101),DEL(8)

```

```

DIMENSION SAVE(800,8),IN(8)
DIMENSION GRAPH(8,2),STEPS(10),NG(8)
COMMON/PSOU/GAV(2),XGRAPH(8,2),YGRAPH(8,2)
COMMON/XSYSTO/YSTOP(800,8),XSTOR(800,8),NXCUM(8),NYCUM(8)
COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
COMMON/OUTCON/NGY(8),NTY(8),NGX(8),NTX(8),IADJ,NNTX,NNTY,IPAGE

```

```

DATA PLUS,BLANK/1H+,1H /
DATA IX,IY/1HX,1HY/

```

```

100 FORMAT(1H1,5X,48HFLUID FEEDBACK SIMULATION UNDER DDC      MODI,
      *17HFICATION NUMBER 3,11X,8HRUN NO. ,12,4X,9HPAGE NO. ,13)
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 **,15,2X,17HPOINTS WERE NOT W
      *,22HITHIN THE GRAPH RANGE.)
106 FORMAT(1H ,20X,101A1)
107 FORMAT(1H+,7X,F11.2)

```

```

MSS = NSS
NN = 2
IF(MSS.EQ.1) GO TO 2
DO 1 I=1,IGX
DO 1 J=1,NNTX
IF(NGX(I).EQ.NXCUM(J)) IN(I) = J
1 CONTINUE
GO TO 4
2 DO 3 I = 1,IGY
DO 3 J = 1,NNTY
IF(NGY(I).EQ.NYCUM(J)) IN(I) = J
3 CONTINUE
4 CALL SCALE(NN,GRAPH,IN)
DO 6 I=2,NGRAPH
IF(GRAPH(IN(I),1).NE.0.OR.GRAPH(IN(I),2).NE.0) GO TO 5
GRAPH(IN(I),1) = -0.1
GRAPH(IN(I),2) = 0.1
GO TO 6
5 IF(GRAPH(IN(I),1).NE.GRAPH(IN(I),2)) GO TO 6
GRAPH(IN(I),1)=GRAPH(IN(I),1)-(0.1*FLOAT(I)*ABS(GRAPH(IN(I),1)))
GRAPH(IN(I),2)=GRAPH(IN(I),2)+0.1*ABS(GRAPH(IN(I),2))
6 DEL(I) = (GRAPH(IN(I),2)-GRAPH(IN(I),1))/10.
IPAGE = IPAGE + 1
WRITE(6,100)IRUN,IPAGE
WRITE(6,102)
IF(MSS.EQ.1) GO TO 7
WRITE(6,103)(IX,PARL,NXCUM(IN(I)),PARR,GRAPH(IN(I),1),SYMB(I),
      *DEL(I),IX,PARL,NGX(I),PARR,GRAPH(IN(I),2),I=2,NGRAPH)
GO TO 8
7 WRITE(6,103)(IY,PARL,NYCUM(IN(I)),PARR,GRAPH(IN(I),1),SYMB(I),
      *DEL(I),IY,PARL,NGY(I),PARR,GRAPH(IN(I),2),I=2,NGRAPH)
8 WRITE(6,104)
IF((NGRAPH-1).EQ.1) GO TO 9

```

```

WRITE(6,101)SYMB(9)
9 DO 10 I=2,NGRAPH
10 GRAPH(IN(I),2) = 100./(GRAPH(IN(I),2) - GRAPH(IN(I),1))
  NPRINT = 0
  NOTON = 0
  DO 20 K=1,NOUT
    NPRINT = NPRINT + 1
    IF(K.NE.1.AND.K.NE.NOUT) GO TO 11
    NPRINT = 5
11 IF(NPRINT.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,NGRAPH
  N=FIX((SAVE(K,IN(I))-GRAPH(IN(I),1))*GRAPH(IN(I),2)+1.5)
  IF(N.GE.1.AND.N.LE.101) GO TO 17
  NOTON = NOTON + 1
  GO TO 19
17 IF(LINE(N).EQ.BLANK.OR.LINE(N).EQ.PLUS) GO TO 18
  LINE(N)=SYMB(9)
  GO TO 19
18 LINE(N)=SYMB(I)
19 CONTINUE
  WRITE(6,106)LINE
  IF(NPRINT.NE.5) GO TO 20
  WRITE(6,107)SAVE(K,IN(1))
  NPRINT = 0
20 CONTINUE
  IF(NOTON.NE.0) WRITE(6,105)NOTON
21 RETURN
END
SUBROUTINE PRTTBL(MAXLIN,IVAR)

```

```

COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
COMMON/OUTCON/NGY(8),NTY(8),NGX(8),NTX(8),IADJ,NNTX,NNTY,IPAGE
COMMON/ARYPRT/PRTARY(800,8)
COMMON/XSYSTO/YSTOR(800,8),XSTOR(800,8),NXCUM(8),NYCUM(8)

```

```

C
DIMENSION NPRT(8)

```

```

C
100 FORMAT(1H1,5X,48HFLUID FEEDBACK SIMULATION UNDER DDC MOD1,
  *17HFICATION NUMBER 3,11X,8HRUN NO. ,12,4X,9HPAGE NO. ,13)
101 FORMAT(8(7X,A1,A1,I2,A1,2X))
102 FORMAT(8(2X,F12.4))
103 FORMAT(1X,120(1H*))

```

```

C
IF(IVAR.EQ.0) GO TO 3
IFM = 10
NCOL = IIX
DO 1 I = 1,MAXLIN
DO 1 J = 1,IIX

```

```

1  PRTARY(I,J) = XSTOR(I,J)
   DO 2 I=1,ITX
2  NPRT(I) = NTX(I)
   GO TO 6
3  IFM = 11
   NCOL = ITY
   DO 4 I = 1,MAXLIN
   DO 4 J = 1,ITY
4  PRTARY(I,J) = YSTOR(I,J)
   DO 5 I=1,ITY
5  NPRT(I) = NTY(I)
6  LINE = 1
   WRITE(6,100)IRUN,IPAGE
   WRITE(6,103)
   WRITE(6,101)(SYMB(IFM),PARL,NPRT(I),PARR,I=1,NCOL)
   DO 7 KK=1,MAXLIN
   WRITE(6,102)(PRTARY(KK,I),I=1,NCOL)
   LINE = LINE + 1
   IF(LINE.LT.55) GO TO 7
   LINE = 1
   IPAGE = IPAGE + 1
   WRITE(6,103)
   WRITE(6,100)IRUN,IPAGE
   WRITE(6,103)
   WRITE(6,101)(SYMB(IFM),PARL,NPRT(I),PARR,I=1,NCOL)
7  CONTINUE
   WRITE(6,103)
   RETURN
   END
   SUBROUTINE SCALE(JJ,GRAPH,NG)

```

```

C
COMMON/PSOU/GAV(2),XGRAPH(8,2),YGRAPH(8,2)
COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
DIMENSION GRAPH(8,2),STEPS(10),NG(8)

```

```

C
DATA STEPS/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
101 FORMAT(2F10.0)

```

```

C
IF(NDATA(14).EQ.0) GO TO 7
IF(NDATA(14).EQ.2) GO TO 3
IF(NDATA(14).EQ.3) GO TO 1
READ(1,101)YMIN,YMAX
GO TO 5
1 DO 2 I=2,NGRAPH
  READ(1,101) YMIN,YMAX
  GRAPH(NG(I),1) = YMIN
2 GRAPH(NG(I),2) = YMAX
  RETURN
3 YMIN = GRAPH(NG(1),1)
  YMAX = GRAPH(NG(1),2)
  IF((NGRAPH-1).EQ.1) GO TO 7
  DO 4 I=JJ,NGRAPH
  IF(YMIN.GT.GRAPH(NG(I),1)) YMIN = GRAPH(NG(I),1)
  IF(YMAX.LT.GRAPH(NG(I),2)) YMAX = GRAPH(NG(I),2)

```

```

4 CONTINUE
5 DO 6 I=JJ,NGRAPH
  GRAPH(NG(I),1) = YMIN
6 GRAPH(NG(I),2) = YMAX
  IF(NDATA(14).EQ.2) RETURN
7 DO 14 K=JJ,NGRAPH
  YMIN = GRAPH(NG(K),1)
  YMAX = GRAPH(NG(K),2)
  YRNG = (YMAX-YMIN)/10.
  IF(YRNG.LE.0.)GO TO 14
  TEMP = ALOG10(YRNG)
  IRNG = TEMP
  IF(TEMP.LT.0.0) IRNG = TEMP-1.
  YRNG = YRNG*10.**(-IRNG)-0.005
  DO 8 J = 1,10
  IF(STEPS(J).LT.YRNG) GO TO 8
  I=J
  GO TO 9
8 CONTINUE
  I=10
9 CONTINUE
  IF(I.LE.10) GO TO 10
  I=I-10
  IRNG = IRNG+1
10 CONTINUE
  COR = STEPS(I)*10.**IRNG
  IF(I.EQ.10) IRNG = IRNG + 1
  IF(YMIN.GE.0.) GO TO 11
  MIN = YMIN/COR - .995
  GO TO 12
11 CONTINUE
  MIN = YMIN/COR + .005
12 CONTINUE
  TEMP = MIN*COR
  TEMPA = TEMP*10.*COR
  IF(TEMPA.GE.YMAX) GO TO 13
  I=I+1
  GO TO 9
13 GRAPH(NG(K),1) = TEMP
  GRAPH(NG(K),2) = TEMPA
14 CONTINUE
  RETURN
END
SUBROUTINE INTCAL

```

C

```

COMMON/NIDAT/NDATA(20),IFL,F(20)
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
COMMON/COMPPR/NSAMPL,NTIME,E1,E2,T01,T02,DE,DDE,DT0
COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4
COMMON/REC/RAM(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG
COMMON Y(42),AMASS

```

C

```

X(1) = Y(NSYS+1)
IF(X(1).LE.0.0) GO TO 1

```

```

      IF (NTIME.EQ.NSAMPL) CALL VELCON
      NTIME = NTIME + 1
1  XT(1) = X(28)
      XT(2) = X(29)
      XT(3) = X(27)
      XT(4) = AMASS
      XT(5) = X(31)
      XT(6) = X(35)
      CALL DELAY
      Y(1) = ALIM(Y(1),0.,1.)
      Y(2) = ALIM(Y(2),0.,1.)
      X(11) = P(5)*X(10)
      X(11) = ALIM(X(11),3.,15.)
      X(12) = (X(11)-P(8))/P(7)
      X(13) = (P(8)-X(11))/P(7)
      X(12) = ALIM(X(12),0.,1.)
      X(13) = ALIM(X(13),0.,1.)
      X(14) = FLOW(Y(1),X(48),P(42))
      X(15) = FLOW(Y(2),X(47),P(42))
      X(22) = X(14)+X(15)
      X(23) = X(46)-X(22)
      X(16) = (P(14)*X(44)+P(15))*X(14)
      X(17) = (P(14)*X(45)+P(15))*X(15)
      X(18) = X(16)*X(20)
      X(19) = X(17)*X(21)
      X(29) = XTH(1)
      X(30) = XTH(2)
      X(33) = XTH(3)
      X(34) = XTH(4)
      X(32) = XTH(5)
      X(36) = XTH(6)
      X(37) = P(19)+(SQRT(P(20)+P(21)*(P(22)+X(36)))/P(23))
      X(37) = ALIM(X(37),X(45),X(44))
      X(24) = X(23)*(P(14)*X(37)+P(15))
      X(36) = ALIM(X(36),X(21),X(20))
      X(25) = X(24)*X(36)
      X(26) = X(18) + X(19) + X(25)
      AMASS = X(16) + X(17) + X(24)
      X(27) = X(26)/AMASS
      X(27) = ALIM(X(27),X(21),X(20))
      X(28) = P(19)+(SQRT(P(20)+P(21)*(P(22)+X(27)))/P(23))
      X(31) = Y(4)
      X(35) = X(33) - (P(31)*P(32)*Y(3)/X(34))
      X(35) = ALIM(X(35),X(21),X(20))
      X(38) = Y(5)
      X(39) = P(24)*X(38) + P(25)
      X(40) = X(39)*P(26)
      RETURN
      END
      SUBROUTINE DERFUN

```

C

```

      COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
      COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)
      COMMON/NIDAT/NDATA(20),IFL,F(20)
      COMMON/FM/SP,P1,P2,P3,ID1,ID2,ID3,ID4

```

COMMON Y(42),AMASS

C

```

F(1) = FOL(1.,X(12),Y(1),P(33))
F(2) = FOL(1.,X(13),Y(2),P(33))
F(3) = (P1*X(30)-(P2+P3)*Y(3)-Y(4.))/(P2*P3)
F(4) = Y(3)
F(5) = FOL(1.,X(29),Y(5),P(34))
F(6) = X(2)*X(2)
F(7) = ABS(X(2))

```

C

```

DO 1 L=1,NSYS
IEQ = NSYS + 2 + L
1 Y(IEQ) = F(L)
RETURN
END
SUBROUTINE DELAY

```

C

```

COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
COMMON/SIGSTR/XSTR(9,500),XT(9),XTH(9),IH(9),NUMSIG,IMAX,X(60)

```

C

```

ISTOP = IMAX-1
DO 1 KK=1,ISTOP
IS=IMAX-KK
DO 1 II=1,NUMSIG
1 XSTR(II,IS+1)=XSTR(II,IS)
DO 2 II=1,NUMSIG
JJ = IH(II)
XTH(II) = XSTR(II,JJ)
2 XSTR(II,1) = XT(II)

```

C

```

RETURN
END
FUNCTION ALIM(V,B,T)
ALIM = V
IF(V.GT.T) ALIM = T
IF(V.LT.B) ALIM = B
RETURN
END
FUNCTION FLOW(PL,VF,FM)
QS = EXP(4.*(PL-1.))*VF
IF(QS.GT.FM) QS = FM
IF(PL.LE.0.) QS = 0.0
FLOW = QS/60.
RETURN
END
FUNCTION AIMM(DMN,OMA)
DMA = 0.0512*DMN
AIMM = OMA + DMA
RETURN
END
FUNCTION FOL(GA,XIN,YOUT,TAU)
FOL = (GA*XIN-YOUT)/TAU
IF(TAU.EQ.0.) FOL = GA*XIN
RETURN
END

```

PAGE NO. 18

SUBROUTINE PARADJ

```
C
COMMON/PARAM/P(60),H(9),NSYS,NOUTPT
C
1 FORMAT(4F10.0)
  READ(1,1)P(1),P(2),P(3),P(4)
  RETURN
  END
  BLOCK DATA
C
COMMON/OUT/ITY,ITX,IGY,IGX,NGRAPH,SYMB(14),PARL,PARR,IRUN,NOUT,MSS
COMMON/PSOU/GAV(2),XGRAPH(8,2),YGRAPH(8,2)
C
DATA SYMB/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H*,1HX,1HY,1HH,1HP,1H8/
DATA PARL,PARR/1H(.1H)/
DATA GAV/1HY,1HX/
C
END
```


Appendix E

COMPUTER PROGRAM FOR CASCADE CONTROL PID ALGORITHMS

 OVERLAY(CASCADE,0,0)

 PROGRAM CASCAD(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)

 THIS IS A SIMULATION OF THE BENDIX MOLD HEATING AND COOLING SYSTEM
 UNDER CASCADE CONTROL. MOLD AND FLUID LOOP FEEDBACK.

 COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMS
 COMMON/BB/XSTR(9,500),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),I1,I2,I3,I4,ID1,ID2,ID3,NPT,NSP,NTM,PTIME
 COMMON/HH/BTIME,ETIME,DELT,I1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
 COMMON/II/ISAMP,NTIME,E1,E2,T01,T02,DE,DDE,DT0,TMASS
 COMMON/JJ/EM1,EM2,TM1,TM2,DEM,DDEM,DTM

 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(9) = 1 SETS I1=1 CAUSING PROPORT. CUTOUT ON DELTA SETPOINT
 ND(10) = 1 SETS I2=1 CAUSING INTEGRAL CUTOUT IN POSCON (2%/SEC)
 ND(11) = 1 SETS I3=1 CAUSING POSCON TO USE PROC. VAR. IN RATE
 ND(12) = 1 SETS I4=1 CAUSING USE OF GAIN MULTIPLIER IN POSCON
 ND(13) = 1 SETS ID1=1 CAUSING VELCON TO USE PROC. VAR. IN RATE
 ND(14) = 1 SETS ID2=1 CAUSING INTEGRAL CUTOUT IN VELCON.
 ND(15) = 1 SETS ID3=1 ALLOWS GAIN MULTIPLIER IN VELCON.
 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.

 THE LOOP FLOW RATE IN GAL/SEC MUST BE DETERMINED FROM PFLON OR
 MFLOW.

 P'S ARE PARAMETER VARIABLES READ IN OR DEFINED IN INTFUN.

 X'S ARE INTERMEDIATE VARIABLES.

Y'S ARE STATE VARIABLES.
 F'S ARE STATE DERIVATIVES.
 H'S ARE TIME DELAY MAGNITUDES.

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,11))
 104 FORMAT(4(5X,2HY(,12,4H) = ,F12.5,2X))
 105 FORMAT(4(5X,2HX(,12,4H) = ,F12.5,2X))

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)

ABOVE READS IN NDATA AND TERMINATES RUN IF NO CHANGES IN ND

CALL INPUT
 CALL INTFUN
 IF(NX.EQ.0) GO TO 4
 CALL INTCAL
 4 CALL DIFFEQ
 CALL ALIST

ABOVE INITIALIZES PROGRAM AND PRINTS OUT HEADER INFO.

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

PAGE NO. 3

```
NW = NPT + (1./DELT)
PTIME = NW*DELT
WRITE(2,101)PTIME
```

C
C
C

ABOVE ESTABLISHES PRINT INTERVAL

```
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
C
C

STORE IN PRINT MATRIX INITIAL X AND Y PRINT/PLOT VARIABLES.

PROCEED WITH PROGRAM EXECUTION BELOW.

```
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      NOUT = JI
      IF(ITY.EQ.0) GO TO 16
C
C      PRINT Y TABLES.
C
C      CALL PRTTBL(JI,0)
16 IF(ITX.EQ.0) GO TO 17
C
C      PRINT X TABLES.
C
C      CALL PRTTBL(JI,1)
17 IF(IGY.EQ.0) GO TO 18
C
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
C      PLOT X DATA.
C
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(6),Y(7)
   IF(JSTOP.NE.0) GO TO 999
   GO TO 1
999 STOP
   END
   SUBROUTINE INPUT
C
C
COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP
COMMON/BB/XSTR(9,500),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),I1,I2,I3,I4,ID1,ID2,ID3,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   ZERO OUT ARRAYS
C
  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
C   IF (ND(3).EQ.0) GO TO 9
C   READ(1,100) (H(I),I=1,NH)
C   9 IF (ND(4).EQ.0) GO TO 10
C   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

C

~~DETERMINE HOW MANY OF EACH FUNCTION ARE TO BE PERFORMED.~~

~~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

C

~~STORE PRINT X INDICES IN ARRAY JX , Y INDICES IN JY.BELOW~~

~~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

C

C

~~CHECK TO SEE IF PLOT X AND Y INDICES ARE THE SAME AS PRINT INDICES~~

~~IF SO DO NOT STORE INDEX. IF NOT SAME STORE INDEX IN JX OR JY.~~

~~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

~~READS IN THE MAJOR LOOP SET POINT RECIPE.~~

```

C      COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
C
C      1 FORMAT(3F10.0)
C      2 FORMAT(I2)
C
C      READ(1,2)NSEG
C
C      DO 3 I=1,NSEG
C      3 READ(1,1)RAMP(I),STPT(I),TIM(I)
C
C      JFLAG = 0
C      RETURN
C      END
C      SUBROUTINE ALIST
C
C      C
C      C      HEADER OUTPUT ROUTINE.
C
C      C
C      C      COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP
C      C      0:99:1/NN/XSTR(9,5<),XT(9),XT4(9),54(9),:4,59MX,4(9)
C      C      COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
C      C      COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
C      C      COMMON/GG/ND(20),I1,I2,I3,I4,ID1,ID2,ID3,NPT,NSP,NTM,PTIME
C      C      COMMON/HH/BTIME,ETIME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
C
C
C
C
C      100 FORMAT(1H1,9X,35HDDC CASCADE SIMULATION ** RUN DATE ,3A8,5X,
C      *8HRUN NO. ,I2,5X,9HPAGE NO. ,I3/)
C      101 FORMAT(1X,120(1H*))
C      102 FORMAT(65X,10HDESCRIPTOR/)
C      103 FORMAT(20X,10A8/20X,10A8//)
C      104 FORMAT(5X,19H* PROGRAM CONTROLS,24X,1H*,16X,17H* MOLD IDENTIFIE,
C      *2HRS,33X,1H*)
C      105 FORMAT(5X,1H*,42X,1H*,16X,1H*,51X,1H*)
C      106 FORMAT(5X,24H* NUMBER OF Y EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
C      *10HMOLD TITLE,13X,3A8,2X,1H*)
C      107 FORMAT(//5X,44(1H*),16X,53(1H*))
C      108 FORMAT(5X,24H* NUMBER OF X EQUATIONS,4X,I3,12X,1H*,16X,1H*,2X,
C      *11HMOLD NUMBER,12X,3A8,2X,1H*)
C      109 FORMAT(5X,28H* NUMBER OF P PARAMETERS ,I3,12X,1H*,16X,1H*,2X,
C      *12HMOLD SECTION,11X,3A8,2X,1H*)
C      110 FORMAT(5X,21H* NUMBER OF H DELAYS,7X,I3,12X,1H*,16X,1H*,2X,
C      *15HSIMULATION DATE,8X,3A8,2X,1H*)
C      111 FORMAT(5X,13H* START TIME,12X,F6.2,3X,10HSECONDS *,16X,1H*,2X,
C      *23HMOLD METAL MASS (LBM) ,F7.0,19X,1H*)
C      112 FORMAT(5X,11H* END TIME,11X,F9.2,3X,10HSECONDS *,16X,1H*,2X,
C      *23HLOOP FLOW RATE (GPM) ,F7.2,19X,1H*)
C      113 FORMAT(5X,18H* OUTPUT INTERVAL,8X,F5.2,3X,10HSECONDS *,16X,
C      *53(1H*))
C      114 FORMAT(5X,25H* INTEGRATOR STEP SIZE ,F6.3,3X,10HSECONDS */5X,
C      *44(1H*))
C      115 FORMAT(//5X,44HTHE FOLLOWING ARE THE TIME DELAYS IN SECONDS/)
C      116 FORMAT(4(5X,2HH(,I2,4H) = ,F12.3,2X))
C      117 FORMAT(//5X,38HTHE FOLLOWING IS THE PARAMETER LISTING/)

```



```

118 FORMAT(4(5X,2HP(,12,4H) = ,F12,6,2X))
119 FORMAT(//5X,48HENSUING ARE THE INITIAL CONDITIONS OF THE STATE ,
*9H VARIABLES/)
120 FORMAT(4(5X,2HY(,12,4H) = ,F12,5,2X))
121 FORMAT(//5X,49HENSUING ARE THE X VALUES AFTER INTFUN AND INTCAL ,
*33HBUT BEFORE THE FIRST INTEGRATION./)
122 FORMAT(4(5X,2HX(,12,4H) = ,F12,4,2X))
123 FORMAT(8(/),50X,20HCONTROLLER CONSTANTS//32X,10HMETAL LOOP,32X,
*10HFLUID LOOP//)
124 FORMAT(32X,5HKM = ,F7.3,30X,5HKM = ,F7.3/32X,5HKP = ,F7.3,30X,
15HKP = ,F7.3/32X,5HKI = ,F7.3,30X,5HKI = ,F7.3/74X,5HKD = ,F7.3/)
C
125 FORMAT(///45X,29HMETAL LOOP CONTROLLER OPTIONS//)
126 FORMAT(///45X,29HFLUID LOOP CONTROLLER OPTIONS//)
127 FORMAT(1X,24(1H.),44H THE PROCESS VARIABLE RATHER THAN THE ERROR ,
*36HWILL BE USED IN THE DERIVATIVE MODE. ,15(1H.))
128 FORMAT(1X,24(1H.),44H INTEGRAL CUT OUT WILL OCCUR IF THE PROCESS ,
*34H VARIABLE RATE EXCEEDS 11 DEG/SEC ,17(1H.))
129 FORMAT(1X,24(1H.),44H THE PROPORTIONAL MODE WILL BE SET = 0.0 FOR ,
*40H ONE SAMPLE PERIOD ON SET POINT CHANGES ,11(1H.))
130 FORMAT(1X,24(1H.),11H KM WILL = ,F5.0,23H IF THE ABSOLUTE ERROR ,
*8HEXCEEDS ,F5.1,10H PERCENT ,33(1H.))
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(///)

```

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,P(26)
QT = P(36)
WRITE(2,112)ETIME,QT
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(2,123)
WRITE(2,124) P(3),P(6),P(5),P(7),P(4),P(8),P(9)
WRITE(2,101)
IPAGE = IPAGE + 1
WRITE(2,100) (JD(I),I=1,3),IRUN,IPAGE
WRITE(2,101)
WRITE(2,125)
IF(I3.EQ.1) WRITE(2,127)
IF(I2.EQ.1) WRITE(2,128)
IF(I4.EQ.1) WRITE(2,130) P(3),P(40)
IF(I1.EQ.1) WRITE(2,129)
WRITE(2,136)
WRITE(2,101)
WRITE(2,126)
WRITE(2,101)
IF(ID1.EQ.1) WRITE(2,127)
IF(ID2.EQ.1) WRITE(2,128)
IF(ID3.EQ.1) WRITE(2,130) P(6),P(41)
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 INTFUN

```

```

C
C
C
INITILIZE FUNCTION SUBROUTINE.

```

```

COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSF
COMMON/BB/XSTR(9,500),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),I1,I2,I3,I4,ID1,ID2,ID3,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,E1,E2,T01,T02,DE,DDE,DT0,TMASS
COMMON/JJ/EM1,EM2,TM1,TM2,DEM,DDM,DTM

```

```

C
C
100 FORMAT(1H1,5X,23H** TIME DELAY VARIABLE .11.18H 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
      P1 = P(44)
      P2 = P(45)
      P3 = P(46)
      X(1) = BTIME
      X(2) = P(42)
      SP = X(2)
      SP1 = X(2)
      SAM = P(35)/DELT
      ISAMP = SAM
      NTIME = 0

```

```

C
      I1 = ND(9)
      I2 = ND(10)
      I3 = ND(11)
      I4 = ND(12)
      ID1 = ND(13)
      ID2 = ND(14)
      ID3 = ND(15)

```

```

C
      X(53) = P(23)
      X(54) = P(24)

```

```

C
      X(28) = X(53)*X(53)*P(15) + X(53)*P(16) + P(17)
      X(29) = X(54)*X(54)*P(15) + X(54)*P(16) + P(17)

```

```

C
      SGH = X(53)*P(50) + P(51)
      SGC = X(54)*P(50) + P(51)

```

```

C
      X(31) = P(36)/60.
      X(7) = P(43)*P(1)+P(2)
      X(8) = X(7)
      X(3) = X(2)*P(1) + P(2)
      X(52) = X(3)
      X(41) = X(2)
      X(50) = X(41)*P(29) + P(30)
      X(51) = X(50)*P(31)
      P(4) = P(4)*P(35)/60.
      P(8) = P(8)*P(35)/60.

```

P(9) = P(9)*60./P(35)

P(25) = P(25)*60./P(35)

X(18) = 12.

X(19) = X(18)*P(10)

X(5) = P(3)

X(11) = P(6)

X(9) = X(8)

X(44) = X(9)

C

X(38) = X(2)

X(36) = X(38)*X(38)*P(15) + X(38)*P(16) + P(17)

X(45) = X(36)

X(47) = X(2)*P(15)*X(2) + P(16)*X(2) + P(17)

X(48) = X(47)

X(49) = X(2)

X(37) = X(38)

X(39) = X(38)

Y(5) = X(38)

X(42) = Y(5)*P(29) + P(30)

X(43) = X(42)*P(31)

X(33) = X(31)*(X(49)*P(13)+P(14))

X(32) = X(31)

X(34) = X(33)*X(48)

X(35) = X(34)

X(40) = X(2)

X(46) = X(33)

P(52) = P(37)*SQRT(P(38)/SGH)

P(53) = P(37)*SQRT(P(39)/SGC)

TMASS = X(33)

X(55) = (X(9) - P(2))/P(1)

E1=E2=0.0

T01 = X(44)

T02 = X(44)

EM1 = EM2 = 0.0

TM1 = X(52)

TM2 = X(52)

DO 5 I=1,IMAX

XSTR(1,I) = X(37)

XSTR(2,I) = X(38)

XSTR(3,I) = X(36)

XSTR(4,I) = X(33)

XSTR(5,I) = X(40)

---

5 XSTR(6,I) = X(47)

DO 6 I=1,NH

6 XTH(I) = XSTR(I,IMAX)

RETURN

END

SUBROUTINE INTCAL

C

C

THIS IS THE INTERMEDIATE CALCULATION SUBROUTINE.

C

COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP

COMMON/BB/XSTR(9,500),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),I1,I2,I3,I4,ID1,ID2,ID3,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,E1,E2,T01,T02,DE,DDE,DT0,TMASS

C
C
C

X(1) = Y(NY+1)
 IF(X(1).LE.BTIME) GO TO 1
 IF(NTIME.EQ.ISAMP) CALL POSCON
 NTIME = NTIME + 1
 1 XT(1) = X(37)
 XT(2) = X(38)
 XT(3) = X(36)
 XT(4) = TMASS
 XT(5) = X(40)
 XT(6) = X(47)

C
C

CALL DELAY

X(38) = XTH(1)
 X(39) = XTH(2)
 X(45) = XTH(3)
 X(46) = XTH(4)
 X(41) = XTH(5)
 X(48) = XTH(6)

C

X(49) = P(18) + (SQRT(P(19)+P(20)*(P(21)+X(48)))/P(22))
 X(49) = ALIM(X(49),X(54),X(53))
 X(48) = ALIM(X(48),X(29),X(28))

C

X(18) = ALIM(X(18),4.,20.)
 X(19) = P(10)*X(18)
 X(19) = ALIM(X(19),3.,15.)

C

X(20) = (X(19)-P(11))/P(12)
 X(21) = (P(11)-X(19))/P(12)
 X(20) = ALIM(X(20),0.,1.)
 X(21) = ALIM(X(21),0.,1.)

C

Y(1) = ALIM(Y(1),0.,1.)
 Y(2) = ALIM(Y(2),0.,1.)
 X(22) = FLOW(Y(1),P(52),P(36))
 X(23) = FLOW(Y(2),P(53),P(36))
 X(30) = X(22) + X(23)
 X(32) = X(31) - X(30)
 X(32) = ALIM(X(32),0.0,X(31))
 X(33) = X(32)*(P(13)*X(49)+P(14))
 X(34) = X(33)*X(48)

C

X(24) = X(22)*(P(13)*X(53)+P(14))
 X(25) = X(23)*(P(13)*X(54)+P(14))
 X(26) = X(24)*X(28)
 X(27) = X(25)*X(29)

C

X(35) = X(26) + X(34) + X(27)
 TMASS = X(24) + X(33) + X(25)

C

```

X(36) = X(35)/TMASS
X(36) = ALIM(X(36),X(29),X(28))
X(37) = P(18)+(SQRT(P(19)+P(20)*(P(21)+X(36)))/P(22))

```

C

```

X(40) = Y(4)
X(42) = Y(5)*P(29) + P(30)
X(43) = X(42)*P(31)

```

C

```

X(50) = X(41)*P(29) + P(30)
X(51) = X(50)*P(31)

```

C

```

X(47) = X(45) - (P(26)*P(27)*Y(3)/X(46))
IF(ABS(Y(3)).LT.1.E-4) GO TO 2
GO TO 3
2 TP = X(39)*P1
X(47) = TP*TP*P(15)+TP*P(16)+P(17)
3 X(47) = ALIM(X(47),X(29),X(28))

```

C

```

RETURN
END
FUNCTION ALIM(V,B,T)
ALIM = V
IF(V.GT.T) ALIM = T
IF(V.LT.B) ALIM = B
RETURN
END
FUNCTION FLOW(AL,VF,FM)
QS = EXP(4.*(AL-1.))*VF
IF(QS.GT.FM) QS = FM
IF(AL.LE.0.0) QS = 0.0
FLOW = QS/60.
RETURN
END
FUNCTION AIMM(DMN,OMA)
DMA = 0.0256*DMN
AIMM = OMA + DMA
RETURN
END
FUNCTION FOL(GA,XIN,YOUT,TAU)
FOL = (GA*XIN-YOUT)/TAU
IF(TAU.LE.0.0) FOL = GA*XIN
RETURN
END
SUBROUTINE POSCON

```

C

C

C

```

THIS IS A P+I POSITION CONTROLLER

```

```

COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP
COMMON/CC/RAMP(14),STPT(14),TIM(14),DWELL(14),JFLAG,NSEG,SP,SP1
COMMON/GG/ND(20),I1,I2,I3,I4,I01,I02,I03,NPT,NSP,NTM,PTIME
COMMON/II/ISAMP,NTIME,E1,E2,T01,T02,DE,DDE,DTO,TMASS
COMMON/JJ/EM1,EM2,TM1,TM2,DEM,DDEM,DTM

```

C

```

EM2 = EM1

```

```

EM1 = X(4)
TM2 = TM1
TM1 = X(52)
X(52) = X(51)*P(32) + P(33)
SP1 = SP

```

C

```
CALL SPADJ
```

C

```

DSP = SP - SP1
X(4) = X(3) - X(52)

```

C

```

DEM = X(4) - EM1
DDEM = X(4) - 2.*EM1 + EM2
DTM = -(X(52) - 2.*TM1 + TM2)
X(6) = DEM
IF(I1.NE.0.AND.DSP.NE.0) X(6) = 0.0
X(7) = P(4)*X(4)
IF(I2.EQ.1) GO TO 1
GO TO 2

```

```

1 DET = (X(4)-EM1)/P(35)
IF(X(4).LT.0.0.AND.DET.GT.2.0) X(7) = 0.0
IF(X(4).GT.0.0.AND.DET.LT.-2.0) X(7) = 0.0
2 X(56) = P(25)*DDEM
IF(I3.NE.0) X(56) = P(25)*DTM

```

C

```
X(8) = X(6) + X(7) + X(56)
```

C

```

X(5) = 1.
SWT = P(40)*P(1) + P(2)
IF(I4.EQ.1.AND.ABS(X(4)).GT.SWT) X(5) = P(3)
X(57) = X(5)*P(5)
X(58) = X(57)*X(8)
X(9) = X(9) + X(58)
X(9) = ALIM(X(9),-50.,50.)
X(55) = (X(9) - P(2))/P(1)

```

C

C

```
CALL VELCON
```

C

C

```

X(18) = AIMM(X(17),X(18))
NTIME = 0

```

C

```

RETURN
END
SUBROUTINE VELCON

```

C

C

```
THIS IS THE VELOCITY CONTROLLER ALGORITHM.
```

C

```

COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP
COMMON/GG/ND(20),I1,I2,I3,I4,ID1,ID2,ID3,NPT,NSP,NTM,PTIME
COMMON/II/ISAMP,NTIME,E1,E2,T01,T02,DE,DDE,OTO,TMASS

```

C

C

C

E2=E1
E1=X(10)
T02=T01
T01=X(44)

C
X(44) = X(43)*P(32) + P(33)
X(10) = X(9) - X(44)

C
DE = X(10) - E1
DDE = X(10) - 2.*E1 + E2
DT0 = -(X(44) - 2.*T01 + T02)
X(14) = P(9)*DDE
IF(I01.EQ.1) X(14) = P(9)*DT0
X(13) = P(8)*X(10)

C
DPV = (X(44) - T01)/P(35)
IF(I02.EQ.1) GO TO 1
GO TO 2
1 IF(X(10).LT.0.0.AND.DPV.GT.2.0) X(13) = 0.0
IF(X(10).GT.0.0.AND.DPV.LT.-2.) X(13) = 0.0
2 X(12) = DE

C
X(15) = X(12) + X(13) + X(14)

C
SMT = P(41)*P(1) + P(2)
X(11) = 1.
IF(I03.EQ.1.AND.ABS(X(10)).GT.SMT) X(11) = P(6)

C
X(16) = X(11)*P(7)
X(17) = X(16)*X(15)

C
RETURN
END
SUBROUTINE SPADJ

C
COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,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 ... 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 P1 = P(44)
P2 = P(45)
P3 = P(46)
GO TO 14
11 P1 = P(47)
P2 = P(48)
P3 = P(49)
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
I=I+1
IF(I.GT.NSEG) GO TO 14

```

PAGE NO. 17

```
DO 13 K=1,NSEG
13 DWELL(K) = DWELL(K-1) + TIM(K)*60.
CH = SP - STPT(I-1)
IF(SP.EQ.STPT(I-1)) CH = SP-STPT(I-2)
IF(CH.LT.0.0) GO TO 11
GO TO 10
```

```
14 X(2) = SP
X(3) = SP*P(1) + P(2)
```

C

MMSP = I

RETURN

END

SUBROUTINE DELAY

C

C THIS SUBROUTINE SETS UP THE DELAY PIPELINES.

C

COMMON/BB/XSTR(9,500),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(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP

C

F(1) = FOL(1.,X(20),Y(1),P(34))

F(2) = FOL(1.,X(21),Y(2),P(34))

F(3) = (P1*X(39)-(P2+P3)*Y(3)-Y(4))/(P2*P3)

F(4) = Y(3)

F(5) = FOL(1.,X(38),Y(5),P(28))

F(6) = X(4)*X(4)

F(7) = ABS(X(4))

C

C

C

DO 1 LK = 1,NY

IQ = NY + 2 + LK

1 Y(IQ) = F(LK)

C

RETURN

END

SUBROUTINE RKINT(LL)

```

C
C RUNGE-KUTTA NUMERICAL INTEGRATION ROUTINE FOR SOLVING DIFFERENTIAL
C EQUATIONS.
C
C COMMON/AA/X(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP
C
C DIMENSION YU(42),BET(3),DY(4,20)
C DOUBLE PRECISION YU
C
C IF(LL.NE.1) GO TO 2
C BET(1) = 0.5
C BET(2) = 0.5
C BET(3) = 1.0
C N2 = NY + 2
C N1 = NY + 1
C XV = Y(N1)
C CALL DIFFEQ
C DO 1 I=1,NY
C 1 YU(I) = Y(I)
C 2 DO 6 K=1,4
C IF(K.EQ.1) GO TO 3
C CALL DIFFEQ
C 3 DO 4 I=1,NY
C IJ = I + N2
C 4 DY(K,I) = Y(N2)*Y(IJ)
C IF(K.EQ.4) GO TO 6
C DO 5 I=1,NY
C 5 Y(I) = YU(I) + BET(K)*DY(K,I)
C Y(N1) = XV + BET(K)*Y(N2)
C 6 CONTINUE
C DO 7 I=1,NY
C DEL = (DY(1,I)+2.*DY(2,I)+2.*DY(3,I)+DY(4,I))/6.
C YU(I) = YU(I) + DEL
C Y(I) = YU(I)
C 7 CONTINUE
C Y(N1) = XV + Y(N2)
C CALL DIFFEQ
C XV = Y(N1)
C RETURN
C END
C SUBROUTINE PRTTBL(MAXLIN,IV)
C
C THIS ROUTINE PRINTS THE SPECIFIED TABLES.
C
C COMMON/DD/ITY,ITX,IGY,IGX,NG,SB(12),PL,PR,IRUN,IPAGE,IX,IY,NOUT
C COMMON/EE/XG(8,2),YG(8,2),MSS,NGY(8),NGX(8),NTY(8),NTX(8)
C COMMON/FF/IADJ,MX,MY,JX(8),JY(8),XS(800,8),YS(800,8)
C COMMON/HH/BTIME,ETDME,DELT,IT1(10),IT2(10),MT(3),MN(3),JD(3),MS(3)
C COMMON ARY(800,8),NPRT(8)
C
C 100 FORMAT(1H1,9X,35HDDC CASCADE SIMULATION ** RUN DATE ,3A8,9X,
C *8HRUN NO. ,I2,5X,9HPAGE NO. ,I3)
C 101 FORMAT(8(7X,A1,A1,I2,A1,2X))
C 102 FORMAT(8(2X,F12.4))

```

---

103 FORMAT(IX,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)
```

```
DATA PLUS,BLANK/1H+,1H /
```

```
100 FORMAT(1H1,9X,35HDDC CASCADE SIMULATION ** RUN DATE ,3A8,9X,
  *8HRUN NO. ,I2,5X,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 **,15,2X,17HPOINTS WERE NOT W,
  *22HWITHIN THE GRAPH RANGE.)
106 FORMAT(1H ,20X,101A1)
107 FORMAT(1H+,7X,F11.2)
```

```
IN(1) IS RESERVED FOR TIME.
```

```
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
```

```
4 CALL SCALE(NN,GR,IN)
```

```
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
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
      COMMON/GG/ND(20),I1,I2,I3,I4,I01,I02,I03,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./

```

~~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

PAGE NO. 23

```
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(70),P(70),Y(42),YT(42),F(20),NY,NX,NP,P1,P2,P3,MMSP/
```

```
C DIMENSION NPA(20)
```

```
C 1 FORMAT(20(I2,1X))
C 2 FORMAT(I2)
C 3 FORMAT(8F10.0)
```

```
C READ(1,2)JP
C 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./
C DATA PL,PR,IX,IY/1H(-,1H),1HX,1HY/
```

```
C END
```


BDX-613-1764 (Rev.)

COMPUTER SIMULATION OF THE MOLD HEATING AND COOLING CONTROL LOOP, USING CONVENTIONAL CONTROL STRATEGIES, R. H. Floersch, D/861, UNCLAS Topical, February 1978.

Mathematical models of the mold and fluid temperature responses were developed experimentally. Those models were then used with a digital computer simulation program in an effort to duplicate mathematically the physical plant dynamic response. The simulation program underwent a series of evolutionary steps until satisfactory duplication of the processes existed. Various control strategies were then simulated. The evolution of the

PLASTICS: Computer Simulation of the Mold Heating and Cooling System

COMPUTER SIMULATION OF THE MOLD HEATING AND COOLING CONTROL LOOP, USING CONVENTIONAL CONTROL STRATEGIES, R. H. Floersch, D/861, UNCLAS Topical, BDX-613-1764 (Rev.), February 1978.

Mathematical models of the mold and fluid temperature responses were developed experimentally. Those models were then used with a digital computer simulation program in an effort to duplicate mathematically the physical plant dynamic response. The simulation program underwent a series of evolutionary steps until satisfactory duplication of the processes existed. Various control strategies were then simulated. The evolution of the

COMPUTER SIMULATION OF THE MOLD HEATING AND COOLING CONTROL LOOP, USING CONVENTIONAL CONTROL STRATEGIES, R. H. Floersch, D/861, UNCLAS Topical, BDX-613-1764 (Rev.), February 1978.

Mathematical models of the mold and fluid temperature responses were developed experimentally. Those models were then used with a digital computer simulation program in an effort to duplicate mathematically the physical plant dynamic response. The simulation program underwent a series of evolutionary steps until satisfactory duplication of the processes existed. Various control strategies were then simulated. The evolution of the

simulation program and the simulation results
are presented...

simulation program and the simulation results
are presented...

simulation program and the simulation results
are presented...