The motivation, history, and basic concepts of user-oriented languages for digital simulation of continuous systems are presented. Reference is made to two illustrative programs, the IBM 1130 *and* SYSTEM/360 *Continuous System Modeling Programs* (CSMP).

Both programs accept user-oriented input statements for constructing simulation models and controlling simulation runs. The 1130 CSMP *also allows on-line interaction by the user. An engineer or scientist at the console can alter the model or change run conditions based on direct observation of simulation outputs. The* SYSTEM/360 CSMP *is intended for batch-mode operation. It has extended facilities for describing the model and for obtaining automatic program control of successive simulation runs.*

Two continuous system modeling programs

by R. D. Brennan and M. Y. Silberberg

Simulation is a well-established tool, with applications ranging from the study of information flow in business organizations to the investigation of the dynamic behavior of complex mechanical systems. The former has often been treated as a discrete process on digital computers through the use of such discrete system simulation programs as the General Purpose System Simulator (GPSS).[1] By contrast, those continuous dynamic systems that are the usual concern of engineers and scientists traditionally have been modeled on analog computers.

The analog computer has been used for innumerable studies of continuous systems and has proved to be a convenient and flexible tool. However, the necessity of scaling variables into reasonable voltage levels and the operational difficulties inherent in analog circuitry combine to present the analog user with a number of irksome problems. These difficulties mount as the size, complexity, and accuracy requirements of the problem increase. As a consequence, considerable attention has been given in the last several years to the use of digital computers for simulation of continuous systems.

This does not imply that the analog computer is passé. In general, for those situations in which raw computing speed is the decisive factor, the analog approach is superior. Moreover there is the possibility of combining an analog system with a digital system in a so-called hybrid configuration. For certain problems, a successful combination of the relative advantages of each has been realized.

However the tasks of implementing and utilizing such a combination are by no means trivial. It would be pointless to attempt to delineate precisely the problems that are best suited to one or another approach. The situation has been in constant flux as the capabilities of both digital and analog techniques have developed and the requirements imposed by the users have become more and more stringent.

An obstacle to the use of digital computers has been the reluctance of many engineers and scientists to enter into digital computer programming. Recognition of that reluctance has stimulated development of problem-oriented languages designed to facilitate communication between simulation users and digital machines.

This article describes the basic design and user-oriented concepts of these languages with specific reference to two programs developed and supported by IBM: the 1130 and SYSTEM/360 Continuous System Modeling Programs (CSMP).

Historical background

digital-analog simulators

Most past activity has been in the development of "digital-analog simulators." During the past ten years, upwards of thirty separate programs of this type have been reported. Each provided a complement of functional elements or blocks similar to those of the analog computer and a block-oriented language for specifying their interconnection. These "digital-analog simulators" model the elements and organization of analog computers and provide numerical routines that are equivalent to such standard analog elements as integrators, summers, inverters, multipliers, and function generators. In addition, they provide those special-purpose functions commonly assembled from several analog elements; for example, division, exponentiation, limiting, time delay and dead space functions. Just as the computer patchboard electrically links analog computing elements, the simulation language describes interconnections among the numerical routines.

continuous system simulators

Recently, developers have become interested in a somewhat different approach, the so-called "continuous system simulators." These programs combine the element or block modeling feature of the "digital-analog simulators" with algebraic and logical modeling capabilities. The input language permits configuration or structure statements to be prepared directly and simply from either a block diagram or differential equation representation of the system to be simulated.

The designers of each successive program have sought to put increased digital computing capability within the reach of the engineer and scientist. While the programs differ in details, there is a common thread—the block-oriented input language. A distinguishing characteristic of the engineering design-analysis approach is the conceptual breakdown of a system into its component parts. By training, many engineers and scientists visualize a system as a complex of interconnected subunits. For example, the control

engineer often represents a servo system by transfer functions grouped as input/output blocks in a feedback system diagram. A physiologist often visualizes a body mechanism as a complex of functional units, without necessarily preparing a formal statement of equations. Because of this user orientation, the block-oriented input language has persisted throughout the history of this type of system.

This history can be traced back to the digital-analog simulator developed by R. G. Selfridge[2] in 1955, a time when the digital computer was still in its infancy. Programmed for the IBM 701, Selfridge's program demonstrated the validity of his idea—that digital computers could be used effectively to simulate continuous phenomena. From that beginning, there has been a steady progression of programs as this new field has unfolded.[3,4] It would be less than honest, however, to suggest that Selfridge's notion met with immediate universal acceptance. The analog computer had been used successfully in simulation studies for nearly a decade. Although requiring considerable expertise, it had proved to be a flexible and valued tool. Its devotees were convinced that the claims made regarding digital simulation stemmed from empty parochialism. The MIDAS program, developed in 1963, finally overcame much of this resistance.[5]

Although intended primarily to check solutions obtained with an analog computer, MIDAS soon found acceptance as an alternative to analog simulation. Its block-oriented language was convenient and simple to use; the computer for which it was developed was capable of handling even the extensive simulations used in the aerospace and chemical process industries. Within months, MIDAS was in use almost everywhere an IBM 7090 was available; its strongest adherents were, more often than not, people who had previously considered themselves happily wedded to the analog computer.

But users of MIDAS soon began to chafe under the frustrations of batch-mode computer "turn-around time." The elimination of scaling and patching—the most irksome aspects of analog computer programming—did permit the user to formulate a simulation quickly, but this advantage was partially offset by the inability to interact directly with the simulation in a "conversational" mode. A program called PACTOLUS, which appeared in 1964, was designed to remedy that situation.[6]

user interaction

PACTOLUS demonstrated that, with an appropriate terminal, a modus operandi could be enjoyed that was very similar, and in some ways superior, to that of conventional analog computers. The program was developed for the IBM 1620, a comparatively small scientific computer. By means of the typewriter, sense switches, and the IBM 1627 Plotter, the user was able to modify the simulation configuration, parameters, and initial conditions at will, while observing the response of the system being simulated.

As with MIDAS, the PACTOLUS program had two big factors in its favor—it appeared at just the right time and was developed for a computer in wide use. Within a year, PACTOLUS was being used

extensively. Its simplicity made it particularly appealing to those previously uninitiated to the mysteries of either analog or digital programming.

Since the appearance of PACTOLUS in 1964, a whole new class of programs, the "continuous system simulators" previously mentioned, has emerged. DSL/90 and MIMIC were in the vanguard of this recent movement;[7,8] the new SYSTEM/360 CSMP is based on DSL/90 and is the latest and most comprehensive of this class.[9] It is intended for simulation users having access to large-scale digital systems.

However, the popularity of PACTOLUS demonstrated that, for many engineers and scientists who want to personally conduct their own simulation studies, the simplicity of an exclusively block-oriented programming language continues to have great appeal. This, then, led to the development of the 1130 Continuous System Modeling Program.[10]

1130 CSMP

The 1130 CSMP was developed specifically for the environment of the design engineer. Because the program operates on a small, fast computer, it is technically and economically feasible for the user to both simulate relatively complex processes and yet interact in an on-line mode. For a simulation study of typical complexity, a single simulation run requires only several minutes of computing time; thus, the user can feel free to experiment with the simulation until satisfied with the system design and performance. For many types of investigations, the 1130 CSMP obviates any requirement for an analog computer facility. Indeed, both the design and implementation of a simulation study are considerably simpler than would be the case with an analog computer.

functional elements

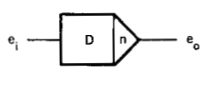
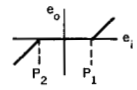
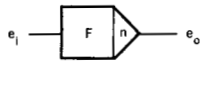
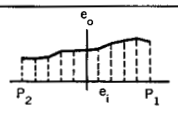
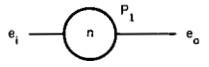
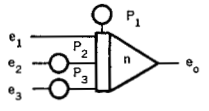
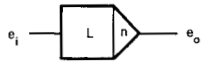
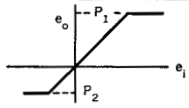
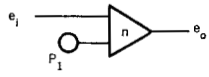
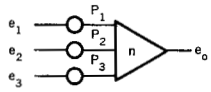
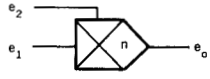
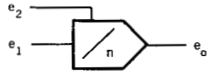
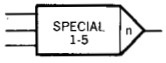
The 1130 CSMP provides a complement of 25 standard simulation elements or blocks, plus a group of five "Special" elements that the user can tailor to his particular needs. Figure 1 illustrates a representative group of these elements, their diagrammatic and language symbols, and descriptions of their functions. The user starts by developing a block diagram showing the interconnections among the elements required to implement his model. He then translates the diagram into a corresponding set of 1130 CSMP language statements.

input language

Translation of the block diagram into a computer program involves the use of three types of simple language statements. These permit easy development of a simulation program and provide a basis for convenient interaction with it by means of console switches and keyboard. The three types are:

- *Configuration statements*, which define the interconnections among the functional blocks and specify the desired functions.
- *Parameter statements*, which associate numerical constants with the elements to particularize their functions.
- *Function generator statements*, which define the input/output relationships for the Function Generator elements.

Figure 1 Representative 1130 CSMP functional elements

ELEMENT TYPE	LANGUAGE SYMBOL	DIAGRAMMATIC SYMBOL	DESCRIPTION
DEAD SPACE	D		
FUNCTION GENERATOR	F		
GAIN	G		$e_o = P_1 e_i$
INTEGRATOR	I		$e_o = P_1 + \int (e_1 + e_2 P_2 + e_3 P_3)\, dt$
LIMITER	L		
OFFSET	O		$e_o = e_i + P_1$
WEIGHTED SUMMER	W		$e_o = P_1 e_1 + P_2 e_2 + P_3 e_3$
MULTIPLIER	X		$e_o = e_1 e_2$
DIVIDER	/		$e_o = e_1 / e_2$
SPECIAL	1-5		SUBROUTINES SUPPLIED BY USER

n represents the block number

Referring again to Figure 1, it is seen that the types of statements required depend on the specific function. For example, some of the elements have one or more inputs but no associated parameters; each use of one of these elements in a model therefore requires only a configuration statement. Those elements that do have associated parameters in their definition usually require both configuration and parameter statements for each use. The Function Generator element requires all three types of statements.

A most important feature of the program is the option of entering these statements either by means of punched cards or directly from the console keyboard. In general, it is preferable to introduce the problem by means of cards and use the on-line console features only for modifying the statements. A standardized format simplifies the routine task of preparing the necessary punched cards. Note that the program is nonprocedural, in that block numbers can be assigned arbitrarily and the configuration statements can be entered in any order. The correct sequencing of the calculations is performed automatically by a sorting routine, as explained in the section on sorting.

on-line interaction

During the introduction of a problem, the user is provided with automatically typed instructions and diagnostics that guide him through the procedures. The instructions tell him how to initiate data entry, how to select the variables for printer and plotter output, and how to specify the integration interval, total run time, and output intervals by means of the console keyboard. There is even the option of suppressing some instructions, by means of a console switch, when the user has gained sufficient proficiency to no longer require that assistance.

Chiefly, of course, interaction means that, during a run, the user can experiment with the model as directly and spontaneously as he would on an analog computer. The user may interrupt a run at will to modify the simulation as desired and need not follow a prescribed pattern for development and testing of the simulation. The digital approach has the added advantage of providing positive documentation of his modifications and progress.

The operational flexibility essential for this on-line experimentation is obtained from use of the console entry switches for option selection and of the console keyboard for entering the modifications. By means of the console features, the user can modify the configuration, parameters, or control variables and can then either proceed from the point of interruption or initiate a new run. Table 1 lists the various options. Effective use of these interactive features requires a convenient means both for observing the model's response during the runs and for permanent documentation of the final data. These requirements are satisfied by use of the console printer and, optionally, the 1627 Plotter. The net effect is a very real capability for user interaction—easy modification of the model, easy run control, easy control of output devices, and adaptability in use.

sorting

The 1130 CSMP can be classified as a special type of "general differential equation solver." Its distinction is that the set of differential equations is specified by a special block-oriented language. Solution is accomplished by means of a sorting algorithm, which determines the order of computations, and a numerical integration formula, which approximates the continuous integration process.

In contrast to most digital programming, in which the order of coding is important, the 1130 CSMP is organized to provide a "parallel" language. That is, the order in which the configuration state-

Table 1 Console entry switch options

Interrupt run
Modify configuration
Change initial conditions or parameters
Change Function Generator intercepts
Change integration specifications
Set print interval
Define print variables
Specify new plot frame
Scale plotter x-axis
Scale plotter y-axis
Suppress instructions
Suppress typing of card input data
Punch updated data deck
Interrogate block outputs
Save status at interrupt point
Restart at previous interrupt point

ments are entered does not affect the subsequent solution. The sorting algorithm is used after each change in the configuration to determine the proper order for processing the functional elements; no element is allowed to be processed until updated values of its input variables are available. At each integration interval, it is assumed that constants and the outputs of memory-type elements are available. A memory-type element is one in which the current output depends only on past values of the input and output. Using these, it is possible to process one or more elements; these outputs then become available as inputs to additional elements. If the configuration is correct, it is possible to process all the blocks in accordance with the sorting algorithm. This operation is performed automatically and requires only a few seconds.

If the sorting algorithm indicates an improperly specified configuration, the program produces a diagnostic message. The most common cause for a sort error is the existence of an algebraic loop. This is a closed path in the simulation diagram that does not include a memory element. Special means provided in the program must then be used to "break" the loop and implement the necessary iteration mechanism for solution of the implicit function.

integration method

Integration is approximated by means of the second-order Runge-Kutta numerical method. The simplicity of this method is particularly important to the user who wishes to develop Special elements involving memory. The program uses centralized integration, and the set of differential equations is treated as a vector equation. Each time a block is processed, the independent variable, time, is incremented by one half the integration interval. At each such half step, the program calls into operation a subroutine that computes the outputs of all the blocks specified for the configuration, in effect, evaluating the derivative vector. This vector quantity is then used to compute the new value of the state variable vector; that is, the output of all the Integrators.

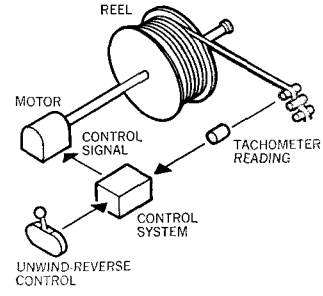
The user must specify the integration interval, consistent with the requirements of his problem. The "best" value for a particular problem can be determined only by experimentation. A good rule of thumb is to first try approximately one-tenth the period of the highest frequency expected. The best value is the largest time interval that can be used without degrading the accuracy below that desired; this choice results in the fastest solution times. It is best to experiment with several different integration intervals before initiating any series of parameter runs.

cable reel problem

The basic modeling capabilities of the 1130 CSMP can be illustrated by a simple but representative problem—a design study of a cable reel system. The objective is to devise a controller that will maintain a constant linear cable velocity as a cable unwinds from a large motor-driven reel. A sketch of the physical system is shown in Figure 2.

Control is to be established by measuring the cable velocity, comparing it to a desired or reference signal, and using the error to generate a motor control signal. This is the classical feedback method of control. A tachometer is used to sense the cable velocity and convert that measurement into a corresponding voltage, which can be compared to a reference. The operational characteristic of the tachometer is represented by a simple first-order transfer function.

Figure 2 Cable reel control system



To maintain a constant linear cable velocity, angular reel velocity must increase as the cable unwinds, that is, as the effective radius of the reel decreases. The situation is complicated by the fact that the moment of inertia of the reel decreases as the cable unwinds, reducing the torque required to maintain constant cable velocity. Since the moment of inertia of the reel is proportional to the fourth power of its effective radius, this phenomenon is quite nonlinear. Common analytic control system techniques would, therefore, be inadequate for solving the design problem, and simulation seems the most suitable approach. The equations and specific physical data for the system are presented in Table 2. For a reel of width W and cable diameter D, there are W/D windings per layer. Thus, the rate of change of effective radius R is $D^2/2\pi W$ times the angular velocity of the reel. (In the simulation programs illustrated later, this constant, $K1$, is assigned a value of -0.0008.) The motor output/input relationship is represented as a simple first-order transfer function. The cable speed and reel acceleration equations describe the basic dynamics of the problem. The desired linear cable velocity is 50 feet/second.

procedure

After first gathering all this pertinent data, the designer using the 1130 CSMP would next develop an appropriate simulation block diagram. This phase of the simulation process is critical for successful use of the technique. In general, the user would be guided by the complement of standard 1130 CSMP functional elements and would minimize the programming of Special elements. During preparation of the block diagram the primary concern should be that the particular simulation implementation truly represents his visualization

Table 2 Equations for cable reel problem

Effective radius of reel

$R_o = 4.0$ ft (full reel)
$R_c = 2.0$ ft (empty reel)
$\dot{R} = -(D^2/2\pi W)\dot{\theta} = -K1\ \dot{\theta}$

where

$\dot{R}$ is time rate of change of reel radius and
$\dot{\theta}$ is angular velocity of the reel

Moment of inertia

$I = 18.5\ R^4 - 221.0$

Tachometer transfer function

$$\frac{V \text{ measured (volts)}}{V \text{ actual (fps)}} = \frac{2.0}{S + 2.0} = \frac{1}{0.5S + 1}$$

S is the Laplace Transform operator. This is equivalent to
$0.5\dot{V}$ measured $+ V$ measured $= V$ actual

Torque motor transfer function

$$\frac{\text{Torque (ft lbs)}}{\text{Control Signal (volts)}} = \frac{500.0}{S + 1.0} = 500.0\left(\frac{1}{S + 1}\right)$$

or $\dot{T}$orque + Torque = 500.0 (Control Signal)

Cable speed

V actual $= R\dot{\theta}$

Desired cable speed

50 ft/sec (represented by 50 volts at set point)

Reel Acceleration

$\ddot{\theta}$ = Torque/I

where $\ddot{\theta}$ is angular acceleration of the reel.

---

of the phenomenon. For this cable reel design problem, the designer might want to experiment with several control algorithms, observing the performance of each at cable start up, braking, and reversal. In these respects, the simulation design should be as flexible as possible.

simulation diagram

One possible simulation diagram for this problem is shown in Figure 3. Block numbers have been assigned in an arbitrary manner to emphasize that the 1130 CSMP configuration statements may appear in any order within the punched-card deck provided they all precede the set of parameter statements.

Note that block 43 is identified as a "Special #3" element; this element has been specifically developed to provide a convenient means for switching from cable unwind to braking or reversal. The simple FORTRAN program developed for this special purpose is shown in Figure 4. This listing shows that two parameters are associated with this new element—the first can be set to +50.0 (ft/sec) to signify that the reel should unwind; the second can be set to −50.0 (ft/sec) to signify reversal. The output of this element

Figure 3 Block diagram for cable reel control system

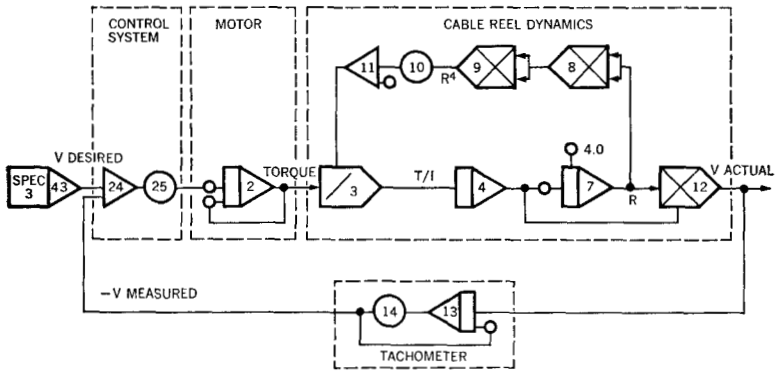


Figure 4 FORTRAN listing for unwind-reverse control function

```
// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE SUB3
C        OUTPUT EQUALS P1 WHEN SWITCH 3 IS ON, P2 WHEN SWITCH IS OFF
      REAL    REALS(395)
      INTEGER INTS(587)
      DIMENSION C(76),PAR1(75),PAR2(75)
      COMMON  REALS, INTS
      EQUIVALENCE  ( INTS(376),     I      ) , ( REALS(  2),     C(1)  )
      EQUIVALENCE  (REALS(156),  PAR2(1)   ) , ( REALS( 81),  PAR1(1)  )
      CALL DATSW( 3, KEY )
      GO TO ( 1,2) , KEY
    1       C(I) = PAR1(I)
      GO TO 3
    2       C(I) = PAR2(I)
    3 RETURN
      END
```

can be switched from one value to the other by use of console switch #3, allowing the designer to, in effect, "operate the controls" for the simulated system as he observes the plotted response during the runs. The designer has assumed that a simple proportional control scheme should be sufficient for this system. Accordingly, the control signal has been generated by just a Gain element, block 25 in Figure 3, having as input the error signal from the summing junction, block 24. Should this simple scheme prove unsatisfactory, the user can experiment with a more complex controller. For example, should this first approach yield wild oscillations, he might try to use a Limiter element between block 25 and block 2 to restrict the magnitude of the control signal applied to the motor.

After developing the block diagram, the designer's next step is to translate it into the corresponding set of 1130 CSMP statements. A configuration statement must be prepared for each block on the diagram; associated parameter and function generator statements must be prepared as required. Simple coding forms, such as those

Figure 5 1130 CSMP coding form with configuration statements for cable reel problem

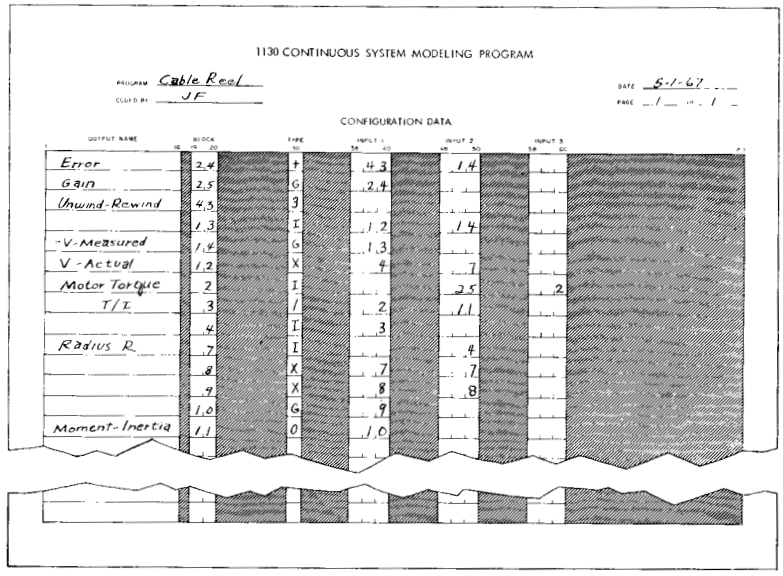


Figure 6 Portion of console printer output from 1130 CSMP run

CONFIGURATION SPECIFICATION

OUTPUT NAME	BLOCK	TYPE	INPUT 1	INPUT 2	INPUT 3
ERROR	24	+	43	14	0
GAIN	25	G	24	0	0
UNWIND - REWIND	43	3	0	0	0
	13	I	12	14	0
- V-MEASURED	14	G	13	0	0
V-ACTUAL	12	X	4	7	0
MOTOR TORQUE	2	I	0	25	2
T / I	3	/	2	11	0
	4	I	3	0	0
RADIUS R	7	I	0	4	0
	8	X	7	7	0
	9	X	8	8	0
	10	G	9	0	0
MOMENT - INERTIA	11	0	10	0	0

INITIAL CONDITIONS AND PARAMETERS

IC/PAR NAME	BLOCK	IC/PAR1	PAR2	PAR3
UNWIND - REWIND	43	50.0000	-50.0000	0.0000
MOTOR DYNAMICS	2	0.0000	500.0000	-1.0000
REEL CONSTANTS	7	4.0000	-0.0008	0.0000
	25	1.5000	0.0000	0.0000
MOMENT - INERTIA	11	-221.0000	0.0000	0.0000
TACHOMETER	13	0.0000	1.0000	0.0000
	10	18.5000	0.0000	0.0000
	14	-2.0000	0.0000	0.0000

(0.1) INTEGRATION INTERVAL

(30.0) TOTAL TIME

illustrated in Figure 5, greatly facilitate this task, especially for those who are unfamiliar with digital programming. The problem is most conveniently entered into the computer by means of corresponding punched cards.

Figure 6 is a reproduction of the first portion of the console printer record. The configuration statements correspond to those shown in Figure 5. Note that integration interval and total time are among the data entered on-line.

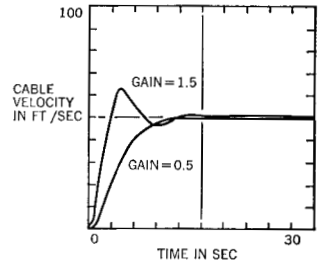
Figure 7 shows the 1627 Plotter output from two trial runs used to obtain an initial estimate of the gain setting of the controller. A gain of 1.5 resulted in some overshoot; the underdamped response obtained with a gain of 0.5 was more satisfactory to the designer. After making this initial analysis, the designer undoubtedly proceeded with a longer series of runs in which he manipulated console switch #3 to test unwind and reversal of the cable throughout the operating range, both with the reel nearly full and nearly empty.

output

SYSTEM/360 CSMP

The SYSTEM/360 CSMP combines the functional block modeling feature of a "digital-analog simulator," such as the 1130 CSMP, with extensive algebraic and logical modeling capabilities. It is intended for batch-mode operation on large-scale digital systems, rather than interactive operation on a small machine. On the other hand, it is a far more comprehensive simulation tool.

Figure 7 1627 Plotter output for cable reel system simulation (annotation added)

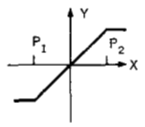
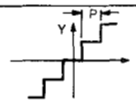
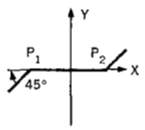


The SYSTEM/360 CSMP input language enables a user to prepare configuration or structure statements describing a physical system, starting from either a differential equation representation or a block diagram of that system. The program provides several means rather than one for defining functions specially suited to a particular simulation requirement. The SYSTEM/360 CSMP also accepts FORTRAN statements, thereby allowing the user to readily handle complex nonlinear and time-variant problems. Controlling input and output is facilitated by means of a free format for data entry and simple control statements. Like the 1130 CSMP, the SYSTEM/360 CSMP provides a parallel language; so that, with few exceptions, configuration or structure statements can be entered in any order and may be intermixed with data and control statements. Output options include printing of variables in standard tabular format, print-plotting in graphic form, and preparation of a data set for user-prepared plotting programs. An important feature is the facility for terminating a simulation run with a sequence of computations and logical tests. These can be designed to accomplish iterative simulations of the type required for parameter optimization studies.

A system to be simulated is described to the program by a series of structure, data, and control statements. Structure statements describe the functional relationships among the variables of the model, and, taken together, define the network to be simulated. A translator converts these structure statements into a FORTRAN subroutine, which is compiled and then executed alternately with a selected integration routine to accomplish the simulation. Data statements assign numerical values to the parameters, constants, initial conditions, and table entries associated with the problem. Control statements specify options relating to the translation, execution, and output phases of the SYSTEM/360 CSMP, such as run time, integration interval, and type of output.

input language

Figure 8 Sample SYSTEM/360 CSMP functions

GENERAL FORM	FUNCTION	
Y=AFGEN (FUNCT, X) ARBITRARY FUNCTION GENERATOR (LINEAR INTERPOLATION)	Y=FUNCT (X)	
Y=LIMIT (P_1, P_2, X) LIMITER	$Y=P_1$ IF $X<P_1$ $Y=P_2$ IF $X>P_2$ $Y=X$ IF $P_1 \leq X \leq P_2$	
Y=QNTZR (P, X) QUANTIZER	$Y=kP$ IF $(k-1/2)P<X\leq(k+1/2)P$ $k=0,\pm1,\pm2,\pm3\ldots$	
Y=DEADSP (P_1, P_2, X) DEAD SPACE	$Y=0$ IF $P_1 \leq X \leq P_2$ $Y=X-P_2$ IF $X>P_2$ $Y=X-P_1$ IF $X<P_1$	

Although the functional blocks of the SYSTEM/360 CSMP are used in the same way as those of the 1130 CSMP, the semantics differ. Figure 8 illustrates the basic form of the SYSTEM/360 CSMP structure statements. A specific example is

$$Y = \text{LIMIT}\ (-5., 10., X)$$

which states that a limiting operation was performed on the input variable X to obtain the output variable Y.

The basic program library contains thirty-four such functions. It includes all the standard functions of analog computers, plus a complement of special-purpose functions often encountered in simulation problems. The basic set is augmented by the FORTRAN library routines, including, for example, cosine and absolute value routines. Note that the simple arithmetic operations are performed by the conventional FORTRAN operators, rather than by functional blocks.

Special functions can be defined either through FORTRAN programming, or, more simply, through a MACRO capability, which permits the combining of individual existing functions into a larger functional block. This subject of user-defined functions is covered in somewhat more detail in a later section. The essential point is that the user has a high degree of flexibility. By properly preparing a set of special functions, he can restructure the SYSTEM/360 CSMP to provide a problem-oriented language for chemical kinetics, control system analysis, or biochemistry. In effect, the SYSTEM/360 CSMP does not have to operate within the framework of a digital-analog simulator language; it can take on the characteristics of a language oriented to any special field in continuous system simulation.

Table 3 SYSTEM/360 CSMP data labels

Label	*Type of Data*
PARAM	Parameters
CONST	Constants
INCON	Initial conditions
AFGEN	Coordinates of an arbitrary function (linear interpolation)
NLFGEN	Coordinates of an arbitrary function (quadratic interpolation)
TABLE	Entries in a stored array

For most simulation studies, the user first prepares configuration statements and follows them by the data and control statements, in that order. The use of data statements to assign numerical values to variables that are to be fixed during a given run provides a means for automatically changing these values between successive runs of the same model. An example of a data statement is:

```
PARAM   RATE = 550.0, DIST = 1000.0
```

where PARAM is the card label identifying it as a parameter card, RATE and DIST are the variables to be assigned numerical values, and 550.0 and 1000.0 are, respectively, the values assigned. The different types of data that can be specified are shown in Table 3.

Logically, the user's next step would be to specify operations associated with the translation, execution, and output phases of the program by means of control statements. Like data statements, these can be changed between runs under control of the simulation program. An example of a control statement is:

```
PRINT   X,XDOT,X2DOT
```

where PRINT is the label of a card specifying that lists of the variables X, XDOT, and X2DOT are to be printed. Other control labels and their effects are shown in Table 4.

user-defined functions

As has been noted, the user has several means for building his own special-purpose functional blocks. These functions may involve just a few statements or represent an extremely complex model of a complete plant; they may be defined by SYSTEM/360 CSMP statements, FORTRAN statements, or a combination of both. Three different types of functions, identified by the names MACRO, PROCED, and subprogram, may be defined. These functions differ somewhat in their use and the way in which they are handled by the SYSTEM/360 CSMP.

The MACRO capability is a particularly significant feature of the language. It allows the user to build larger functional blocks from the basic functions available in the library. Thus, the user can identify a subsection of a simulation block diagram, or the corresponding subset of structure statements, as a parallel functional

Table 4 SYSTEM/360 CSMP control labels

Control Label	*Operation*
NOSORT . . . SORT	Specify groups of structure statements not to be sorted
INIT . . . DYNAM	Define a block of computations to be executed only once at the beginning of the run
TIMER	Specify print interval, print-plot interval, finish time, and integration interval
FINISH	Specify a condition for termination of run
RELERR	Specify relative error for integration routine
ABSERR	Specify absolute error for integration routine
METHOD	Specify integration method
PRINT	Identify variables to be printed
PRTPLT	Identify variables to be print-plotted
TITLE	Print page headings for printed output
LABEL	Print page headings for print-plot output
RANGE	Obtain minimum and maximum values of specified variables

entity. Once defined, the MACRO subset can be used any number of times within the simulation, just like any of the other basic library functions.

The PROCED type of user-defined function allows simple application of the logic capabilities of FORTRAN. During sorting, the statements that define the PROCED function are treated as a single functional group, and the entire set is moved around as an entity in order to satisfy the input/output sequencing requirements of the sorting algorithm. There is no internal sorting of statements within a PROCED group. Generally, a particular PROCED group can be used only once within a simulation program. However, the PROCED block can be imbedded within a MACRO block and thereby used repeatedly.

The FORTRAN subprogram approach permits the user to prepare a separate subprogram that represents the functional characteristics of the phenomenon to be simulated. This approach actually

adds little to the algebraic and logical capabilities available through use of the PROCED technique. However, use of the subprogram permits the new block to be permanently added to the system library.

As an illustration of a user-defined function, consider a simulation study that involves several transfer functions with differing parameter values but the same general form:

$$\frac{Z(s)}{X(s)} = \frac{s^2 + as + b}{s^2 + cs + d}$$

The user may define a MACRO to represent this general functional relationship, assigning to it some unique name, such as FILTER. The SYSTEM/360 CSMP statements to define this MACRO might be as follows:

```
MACRO     OUT = FILTER (A, B, C, D, IN)
           S2Y = IN - C*SY - D*Y
            SY = INTGRL (0.0, S2Y)
             Y = INTGRL (0.0, SY)
           OUT = S2Y + A*SY + B*Y
ENDMAC
```

where the MACRO and ENDMAC cards are translation control statements that bound the set of statements defining the new function.

NOSORT option

Although SYSTEM/360 CSMP was designed to provide a parallel language, specifically to free the user from the task of correctly sequencing structure statements, the automatic sorting can be bypassed by using the NOSORT option, PROCED functional blocks, or subroutines. The user can thereby include any type of procedural statement capability, such as branching on conditions and logical testing, within a sequence of either SYSTEM/360 CSMP or FORTRAN statements. In particular, the NOSORT option can be used to identify a section of coding that divides the other structure statements into groups, each of which is separately sorted. Such a NOSORT section might be used to test simulation response as a basis for switching portions of the configuration into or out of the simulation. This might be done to decrease run time or alter the information flow. Problem structure variations that can be anticipated can thereby be included in a single run.

integration methods

As in the 1130 CSMP, centralized integration is used to ensure that all integrator outputs are computed simultaneously at the end of the iteration cycle. The user may choose from among several different types of integration routines provided with the program. These include both fixed-step and variable-step integration routines. Five fixed-step routines are available: fixed-step Runge-Kutta, Simpson, trapezoidal, rectangular, and second-order Adams. Two variable-step routines are available: fifth-order Milne predictor-corrector and fourth-order Runge-Kutta. In the latter two routines, the integration interval is varied, under program control, during problem execution. Both routines provide an estimate of the integration error, which is compared with a user-specified error bound. The step size is adjusted accordingly by the program to

achieve the specified accuracy with the maximum step size. If none of the above methods satisfies the user's requirement, he can specify his own integration method.

To repeat an idea, selection of an integration method and integration interval for a particular simulation study should be made only after consideration of a number of interrelated factors. The objective is to choose that combination of routine and interval that provides the fastest execution while maintaining sufficient accuracy and providing sufficient output data for easy interpretation of results. In general, as the complexity of the integration method increases, the computer time required for a single step also increases. On the other hand, stability may also increase, permitting larger time steps. Thus, a number of trade-offs between accuracy and running time are possible.

If no integration method is specified, the program automatically uses the fixed-step Runge-Kutta method. This method is generally a good choice for the initial runs of a simulation study if the user is unsure of the dynamic response of the simulated phenomenon. It is advisable to make the initial choice of integration interval sufficiently small to ensure accurate, stable solution, even at the expense of a longer-than-necessary initial computer run. After becoming familiar with a particular problem, the user may choose a more optimal combination of timing specifications and integration method. Again, the user must be prepared to do some experimentation to achieve that end.

programmed simulation sequences

Because the SYSTEM/360 CSMP is a batch-mode program, several means are provided for obtaining an automatic sequence of runs of the same structure statements with different parameters, output variables, and output options. For example, CONTIN, END, and STOP execution statements can be used as follows:

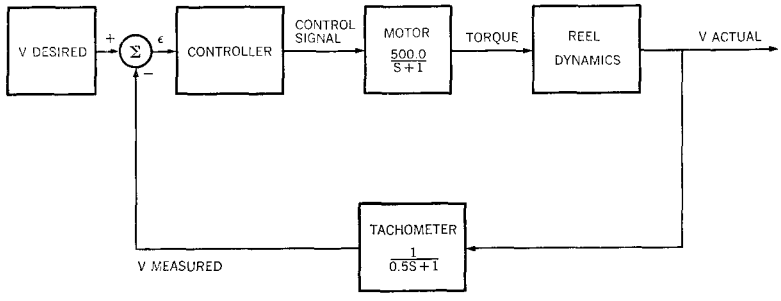
- A CONTIN statement permits data and control statements to be changed without resetting the independent variable, and causes the run to continue. This feature permits such items as integration method, integration interval, and data output interval to be modified during the progress of a simulation study.
- An END statement permits data and control statements to be changed, resets the independent variable to zero, and initiates a new run.
- A combination of END and STOP statements indicates termination of the sequence.

Changes in a parameter, an initial condition, or a constant can also be performed automatically by specifying multiple values on an appropriate data card. For example, the statement:

```
PARAM  RATE = (5.0,5.5,6.0)
```

will initiate a sequence of three runs in which RATE = 5.0 for the first run, RATE = 5.5 for the second run, and RATE = 6.0 for the third run. Only one variable at a time can be specified using this multiple-parameter option.

Figure 9 Rough block diagram for cable reel control system



The foregoing methods are, of course, applicable only when the user can specify his desired sequence a priori. Frequently, however, it is desirable to control the sequence by using results obtained in a prior run, possibly to decide whether or not the sequence should be continued. Examples are parameter optimization and two-point boundary value problems, which require modification of parameters between iterative simulations. A criterion is required in these cases for terminating this iterative sequence when a "sufficiently accurate" solution has been obtained. The SYSTEM/360 CSMP has a control feature called TERMIN that allows the user to identify a sequence of statements that are to be performed only on the termination of a run and thereby achieve this type of program control. Use of this feature is shown later in a sample problem.

cable reel problem

The cable reel design problem described before will also serve to illustrate the basic features of the SYSTEM/360 CSMP. When approaching this problem, the designer could conceivably work from either the system equations or a very detailed block diagram of the type shown in Figure 3. Most likely, however, he would sketch the basic operational units in the sort of rough block diagram shown in Figure 9. The SYSTEM/360 CSMP statements would then be developed from a composite block diagram-differential equation representation of the dynamics of the problem. In any case, the designer must, as always, keep account of the approximations introduced by his visualization of the system.

problem statement

A complete listing of the statements that might be prepared for this problem is shown in Figure 10. It must be emphasized that this is simply one possibility and that there is no single or "best" way to describe the system. Some programs might be more direct or efficient, but any complete SYSTEM/360 CSMP statement of the equations should produce equivalent results.

In this case, the designer decided to provide flexibility in the program by entering D and W as parameters and directing the computer to determine the composite coefficient $K1$. For efficiency, this coefficient is computed only once during the initialization phase of the run. The necessary structure and data statements are identified as initializing operations by means of the translation control cards INIT and DYNAM, which, respectively, precede and follow

Figure 10 SYSTEM/360 CSMP statements for cable reel problem

```
TITLE              CABLE REEL CONTROL DESIGN
*                        SEPTEMBER 23, 1966
INIT
                K1 = (D**2)/(2.0*PI*W)
PARAM            D = 0.1       ,        W = 2.0
CONST           PI = 3.14159
PARAM        RFULL = 4.0       ,  REMPTY = 2.0
*
DYNAM
                 I = 18.5 * (R**4)   - 221.0
            TH2DOT = TORQUE / I
            TH1DOT = INTGRL( 0.0   TH2DOT )
                 R = INTGRL( RFULL , ( -K1 * TH1DOT ))
             ERROR = VDESIR - VM
PARAM        VDESIR = 50.0
              CONTL = GAIN * ERROR
*     (REMEMBER TO ADD CONTROL LIMITING ON NEXT TRY)
PARAM          GAIN = 0.5
            TORQUE = 500.0 * DUMMY
             DUMMY = REALPL(0.0, 1.0, CONTL)
              VACT = R * TH1DOT
                VM = REALPL(0.0, 0.5, VACT)
FINISH           R = 2.0
TIMER  DELT=.05, FINTIM = 20.0, PRDEL=0.5,OUTDEL=0.5
PRINT    VACT,VM,ERROR,CONTL  ,TORQUE,R,I
PRTPLT   VACT
LABEL        PRELIMINARY TEST OF SYSTEM STABILITY (GAIN = 0.5)
METHOD   RECT
END
PARAM          GAIN = 1.5
RESET LABEL
LABEL        PRELIMINARY TEST OF SYSTEM STABILITY (GAIN = 1.5)
END
STOP
ENDJOB
```

Figure 11 Output of cable reel control system for gain=1.5

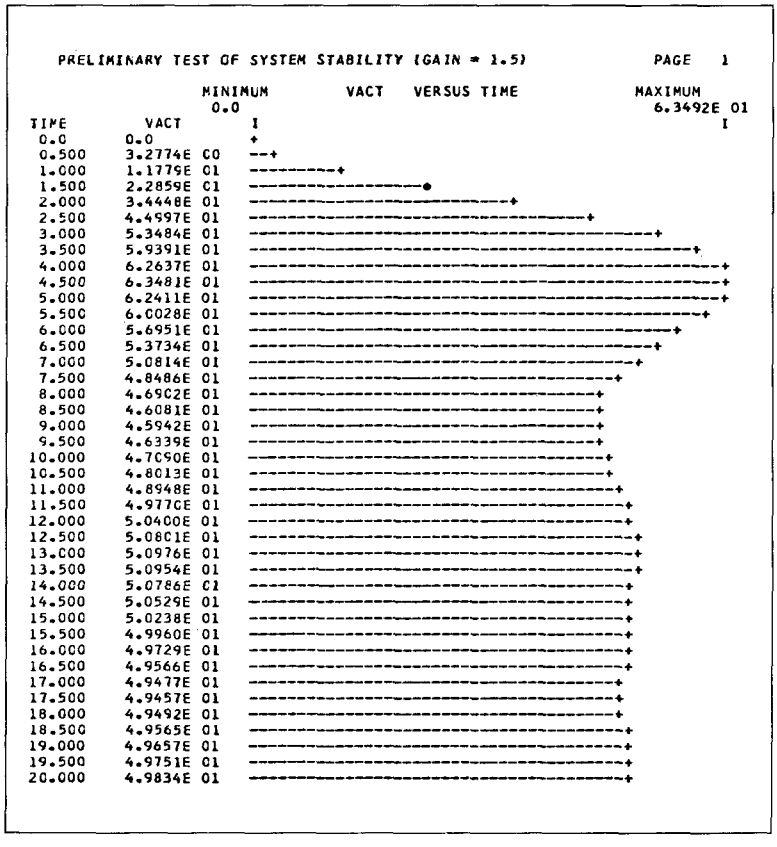
```
                CABLE REEL CONTROL DESIGN

 TIME       VACT          VM          ERROR        CONTL        TORQUE       R            I
   0.0  0.0          0.0          5.0000E 01 7.5000E 01 0.0          4.0000E 00 4.5150E 03
   0.50 3.2774E 00 7.6835E-01 4.9232E 01 7.3847E 01 1.4996E 04 3.9999E 00 4.5146E 03
   1.00 1.1779E 01 5.1595E 00 4.4841E 01 6.7261E 01 2.3250E 04 3.9992E 00 4.5113E 03
   1.50 2.2859E 01 1.3256E 01 3.6744E 01 5.5116E 01 2.6294E 04 3.9976E 00 4.5035E 03
   2.00 3.4448E 01 2.3573E 01 2.6427E 01 3.9641E 01 2.5307E 04 3.9948E 00 4.4902E 03
   2.50 4.4997E 01 3.4426E 01 1.5574E 01 2.3360E 01 2.1492E 04 3.9908E 00 4.4718E 03
   3.00 5.3484E 01 4.4405E 01 5.5955E 00 8.3932E 00 1.6040E 04 3.9859E 00 4.4488E 03
   3.50 5.9391E 01 5.2527E 01-2.5272E 00-3.7909E 00 1.0029E 04 3.9803E 00 4.4224E 03
   4.00 6.2637E 01 5.8273E 01-8.2726E 00-1.2409E 01 4.3296E 03 3.9742E 00 4.3940E 03
   4.50 6.3481E 01 6.1531E 01-1.1531E 01-1.7296E 01-4.4140E 02 3.9679E 00 4.3646E 03
   5.00 6.2411E 01 6.2517E 01-1.2517E 01-1.8776E 01-3.9332E 03 3.9615E 00 4.3354E 03
   5.50 6.0028E 01 6.1672E 01-1.1672E 01-1.7508E 01-6.0364E 03 3.9553E 00 4.3070E 03
   6.00 5.69[illegible] 5.9550E 01-9.5502E 00-1.4325E 01-4.837[illegible] 3.94[illegible] 4.2800E
                                                    -1.009[illegible]
  19.50 4.9751E 01 4.9663E 01 3.3765E-01 5.0558E-01 3.0433E 02 3.8122E 00 3.6865E 03
  20.00 4.9834E 01 4.9750E 01 2.4983E-01 3.7475E-01 2.7062E 02 3.8070E 00 3.6652E 03
```

the initialization statements. The DYNAM card indicates the end of the initialization statements and the beginning of the dynamic portion of the simulation program. No provision has been made in this particular illustration for wind-unwind control.

To prepare the dynamic portion of the simulation program, the designer began with the reel dynamics and developed the appropriate structure statements directly from the differential equations. He then proceeded around the block diagram starting at the comparison point. Note that both the motor and tachometer blocks were modeled using the library function REALPL, which represents a first-order transfer function. Data statements specifying parameter values were inserted wherever the designer found it convenient.

Figure 12 Print-plot output of cable reel control system



```
   PRELIMINARY TEST OF SYSTEM STABILITY (GAIN = 1.5)                PAGE    1

                    MINIMUM          VACT   VERSUS TIME              MAXIMUM
                      0.0                                           6.3492E 01
 TIME          VACT          I                                              I
  0.0        0.0             +
  0.500      3.2774E 00    --+
  1.000      1.1779E 01    ----------+
  1.500      2.2859E 01    ----------------------●
  2.000      3.4448E 01    ------------------------------+
  2.500      4.4997E 01    ----------------------------------+
  3.000      5.3484E 01    ----------------------------------------+
  3.500      5.9391E 01    ------------------------------------------+
  4.000      6.2637E 01    --------------------------------------------+
  4.500      6.3481E 01    --------------------------------------------+
  5.000      6.2411E 01    --------------------------------------------+
  5.500      6.0028E 01    ------------------------------------------+
  6.000      5.6951E 01    ----------------------------------------+
  6.500      5.3734E 01    --------------------------------------+
  7.000      5.0814E 01    ------------------------------------+
  7.500      4.8486E 01    ----------------------------------+
  8.000      4.6902E 01    --------------------------------+
  8.500      4.6081E 01    --------------------------------+
  9.000      4.5942E 01    --------------------------------+
  9.500      4.6339E 01    --------------------------------+
 10.000      4.7090E 01    ---------------------------------+
 10.500      4.8013E 01    ---------------------------------+
 11.000      4.8948E 01    ----------------------------------+
 11.500      4.977CE 01    -----------------------------------+
 12.000      5.0400E 01    -----------------------------------+
 12.500      5.08C1E 01    ------------------------------------+
 13.000      5.0976E 01    ------------------------------------+
 13.500      5.0954E 01    ------------------------------------+
 14.000      5.0786E 01    -----------------------------------+
 14.500      5.0529E 01    -----------------------------------+
 15.000      5.0238E 01    -----------------------------------+
 15.500      4.9960E 01    -----------------------------------+
 16.000      4.9729E 01    -----------------------------------+
 16.500      4.9566E 01    -----------------------------------+
 17.000      4.9477E 01    ----------------------------------+
 17.500      4.9457E 01    ----------------------------------+
 18.000      4.9492E 01    ----------------------------------+
 18.500      4.9565E 01    -----------------------------------+
 19.000      4.9657E 01    -----------------------------------+
 19.500      4.9751E 01    -----------------------------------+
 20.000      4.9834E 01    -----------------------------------+
```

The programming was completed with a group of control statements specifying the run termination condition ($R = 2$), integration and data output intervals, variables to be plotted and print-plotted, a label for the print-plot, and the integration method.

As in the 1130 CSMP illustration, the designer assumed that simple proportional control would prove sufficient, and decided to make his initial runs with gains of 0.5 and 1.5. However, he added a comment card, which was included in the documentation, to remind himself that a modification should be considered in the next series of trial runs. The first END statement indicates the completion of data entry for the first run, which was made with a controller gain of 0.5. The designer requested that a second run be made immediately after the first, with a controller gain of 1.5. This shows how easily the user can prepare, in advance, the desired parameters, execution options, and output options for a series of runs. Figures 11 and 12 illustrate the type of tabular and print-plot outputs, respectively, that are provided by the program. A title has been printed as a heading for the tabular output and a label has been printed for the print-plot. The minimum and maximum values of the dependent variable, VACT, have also been given in the print-plot output.

heat dissipation problem

Another simple example that illustrates several SYSTEM/360 CSMP features, including TERMIN, is the design of a radiating heat fin of the type shown in Figure 13. The fins are attached to the coolant tube in order to increase heat dissipation by thermal radiation. This method might, for example, be used to control the thermal environment of a space station power plant.[11]

A typical design problem would be to dimension the fins such that each dissipated a specified amount of heat per hour, with a specified temperature along the root end of the fin. If the metal and the surface roughness are fixed by other considerations, the available design parameters are fin length, L, and fin thickness, H. Since physical constraints on fin length make that dimension less readily manipulated, the engineer would generally first seek a solution using only fin thickness as the design parameter.

differential equation representation

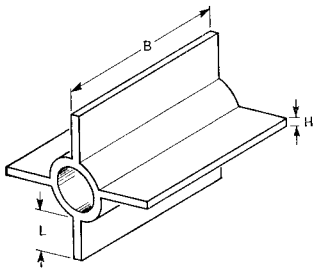
In most simulation studies, it is the time or transient response that is of interest. Here, however, the engineer is concerned with the steady-state heat flow within the fin, at thermal equilibrium. Assuming negligible thermal interaction among fins, the situation is described by the following differential equation:

$$d^2T/dX^2 = 2\,E(T^4 - T^4_s)/KH$$

where

X = distance from fin root
T = temperature in degrees R
T_s = temperature of surrounding space
σ = Stefan-Boltzmann constant
E = thermal emissivity
K = thermal conductivity
H = fin thickness

Figure 13 Cooling fin



The independent variable is X, the distance from the fin root, rather than time.

The simplicity of this differential equation is disarming. Although only a few SYSTEM/360 CSMP structure statements are needed to represent the relationship, design requires solution of a two-point boundary value problem. Two conditions are needed to solve the second-order differential equation. One is given at $X = 0.0$ and the other at $X = L$. At the point $X = 0.0$, T must equal the specified, constant fin root temperature, T_o. At the edge of the fin ($X = L$), radiation must approach zero; that is, dT/dX must equal zero.

solution method

Solution of this type of problem involves a trial and error process. The basic procedure is to search for a value of dT/dX at $X = 0.0$ that, together with the specified temperature condition at $X = 0.0$, yields a solution that also satisfies the other end-point condition, $dT/dX = 0.0$ at $X = L$.

In this case, the actual implementation involves a search over values of the design parameter H by use of the relationship:

$$\frac{dT}{dX} = \frac{-Q_o}{KBH} \qquad \text{at } X = .00$$

Figure 14 SYSTEM/360 CSMP statements for cooling fin design

```
LABEL     COOLING FIN DESIGN
 RENAME        TIME = X ,     FINTIM = FINX
*
 INIT
                   H = 0.5* ( HIGH + LOW )
                COEF = ( 2.*SIGMA*E ) / ( K*H )
               DTDX0 =-Q0 / ( K*B*H )
  CONST           Q0 = 1000 ,   T0 = 2000.,    LOW = 0.0 ,   HIGH = .01,...
                SIGMA = 0.173E-8,  E = 0.8,        K = 25.0,      B = 0.5
*
*
 DYNAM
              D2TDX2 = COEF * ( TEMP**4 )
                DTDX = INTGRL( DTDX0, D2TDX2 )
                   T = INTGRL( T0, DTDX )
                TEMP = LIMIT( 0.0, T0, T )
 TIMER          DELT = 0.0001,       FINX = 0.25
*
*
 TERMIN
       IF ( HIGH - LOW - 0.00001 ) 5, 5, 1
     1 IF ( DTDX ) 3, 5, 2
     2   GO TO 4
     3 LOW = H
     4 CALL RERUN
     5 CONTINUE
 END
*
  TIMER      OUTDEL = 0.01
  PRTPLT  TEMP(DTDX,D2TDX2,H)
 END
 STOP
```

where Q_o is the specified heat dissipation rate and B is the width of the fin. The fin thickness, H, is varied until the initial condition on dT/dX yields a value at the fin edge sufficiently close to zero to be considered a solution. A common approach is to make an arbitrary first guess for the design parameter and then, by means of some algorithm, successively modify that value after each run until all constraints are satisfied. This procedure may be easily automated within the SYSTEM/360 CSMP.

Suppose now that a solution is to be obtained for a specific fin for which:

T_s = 0 deg R
E = 0.8
K = 25 Btu/hr/ft/ deg R
B = 0.5 ft
L = 0.25 ft
T_o = 2000 deg R, and
Q_o = 1000 Btu/hr

Suppose, too, that the fin thickness, H, is constrained to be less than 0.01 foot.

problem statement

Figure 14 shows the complete set of SYSTEM/360 CSMP statements for one possible approach to this design problem. A LABEL card provides a heading for the output, which will be generated at the end of the computation. The RENAME feature is used so that, for convenience, the independent variable will be represented by the symbol X, rather than by TIME.

Note that the structure statements are separated into three segments by the INIT, DYNAM, and TERMIN control statements. The computations in the initial segment are performed only at the beginning of each run. For the first run, the value of H is computed

using values of HIGH and LOW, as specified in a CONST statement. On subsequent runs during the iterative search, HIGH and LOW are adjusted by the algorithm imbedded in the terminal segment. Using this value for H, the program next calculates the variables COEF and DTDX0 for use in the dynamic segment. It should be remembered that structure statements within the initial segment are presumed to represent parallel structure. Accordingly, the actual order of the statements within this segment is of no concern. Unless the user deliberately chooses otherwise, by specifying the NOSORT option or by placing statements within a PROCED group, the sorting algorithm automatically determines an appropriate computational sequence.

The statements between the DYNAM and TERMIN control cards represent the dynamics of the differential equation. Note again that this is merely one possible representation of this relationship. Some users prefer to "nest" expressions to achieve a very compact problem formulation; others find that fewer errors occur if a single functional relationship is expressed in each statement. Note, in particular, that the integrator that develops DTDX uses, as an initial condition, the value DTDX0 computed in the initial segment. An interesting refinement of the search procedure is the use of a limiting function, applied to T, to obtain the variable TEMP. Clearly, in a correct solution, the temperature along the fin must be less than that at the fin root and greater than the temperature of the surrounding space. The limiting function imposes this restriction explicitly, as a precaution, in the event that an estimate of H causes a "wild" solution. A TIMER control card specifies the integration interval, and also specifies that the dynamic computation should terminate in each run when X reaches 0.25, the fin length. Since no integration method has been specified, the program will automatically use the fixed-step Runge-Kutta method.

By definition, the structure statements within the terminal segment represent procedural programming. In this example, the terminal segment is an implementation of a binary search algorithm that adjusts the values of HIGH and LOW according to the final value of DTDX. Consistent with the physics of the problem, this algorithm reduces the estimate of H used in the next run if the final value of DTDX is positive, and increases the estimate if DTDX is negative. The algorithm is designed to ensure convergence subject to the dimensional constraints on fin thickness imposed by the HIGH and LOW values specified on the CONST data card. The designer has made his own definition of "sufficient accuracy" by his choice of the constant term in the first IF statement. When convergence is obtained, the program bypasses the adjustment algorithm. Until convergence is obtained, the SYSTEM/360 CSMP statement RERUN is executed, thereby signifying that yet another iteration is required.

Note that no output statements are used prior to the first END card. This allows the entire search to occur without either tabular or plot output until convergence. When convergence occurs, the

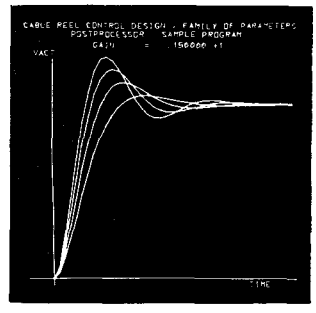
program next encounters statements requesting a print-plot of the temperature profile. The SYSTEM/360 CSMP then performs one additional run, using the final value of H obtained from the iterative search, and writes out the results of this run as requested on the TIMER and PRTPLT control cards.

Summary comments

One objective of this paper was to demonstrate that continuous system modeling programs have reached a useful level of maturity. Certainly, digital-analog simulators now have a reasonably stable set of design criteria. In addition, it appears that the basic development phase is past for the continuous system simulators. Several programs of this type are currently available with a number of advanced features. There does remain, though, a need to accumulate application experience, which could serve to chart the requirements for new developments.

Even now, however, certain directions are apparent. In the opinion of the authors, the next few years will bring extensive use of graphic devices for simulation data input and display. For example, Figure 15 shows output obtained on an IBM 2250 for the cable reel problem through use of the SYSTEM/360 CSMP PREPAR data set and a homemade display routine. Remote consoles operating within a time-sharing environment will give the simulation user a computing tool that can be used to handle complex problems and yet provide the desired on-line interaction. Such developments will necessarily have a profound effect upon the day-to-day practice of engineering and many of the allied sciences.

Figure 15 2250 display of output from cable reel problem



ACKNOWLEDGMENT

The authors would like to acknowledge the contribution of their colleagues on the CSMP development team: C. A. Luke, R. E. Moore, W. S. Mossie, W. T. Prewett, and M. M. Rogson. Many of their ideas and comments are reflected in this paper.

CITED REFERENCES

1. H. Herscovitch and T. Schneider, "GPSS III—An expanded general purpose simulator," *IBM Systems Journal* **4**, No. 3, 174–183 (1965).
2. R. G. Selfridge, "Coding a general-purpose digital computer to operate as a differential analyzer," *Proceedings of the Western Joint Computer Conference*, The Institute of Radio Engineers, New York, 82–84 (1955).
3. R. N. Linebarger and R. D. Brennan, "A survey of digital simulation: Digital-analog simulator programs," *Simulation* **3**, No. 6, 22–36 (December 1964).
4. J. J. Clancy and M. S. Fineberg, "Digital simulation languages: A critique and a guide," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **27**, Part I, 23–36 (November 1965).
5. R. T. Harnett, J. F. Sansom, and L. M. Warshawsky, "MIDAS . . . An analog approach to digital computation," *Simulation* **3**, No. 3, 17–43 (September 1964).
6. R. D. Brennan and H. Sano, "PACTOLUS—A digital simulator program for the IBM 1620," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **26**, 299–312 (October 1964).

7. W. M. Syn and R. N. Linebarger, "DSL/90—a digital simulation program for continuous system modeling," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **28**, 165–187 (April 1966).
8. F. J. Sansom and H. E. Peterson, "MIMIC—Digital simulator program," *SESCA Internal Memo 65-12*, Wright-Patterson Air Force Base, Ohio (May 1965).
9. SYSTEM/360 *Continuous System Modeling Program*, H20-0240, IBM Data Processing Division, White Plains, New York.
10. *1130 Continuous System Modeling Program*, H20-0209-1, IBM Data Processing Division, White Plains, New York.
11. M. L. James, G. M. Smith, and J. C. Wolford, *Analog and Digital Computer Methods in Engineering Analysis*. International Text Book Co., Scranton, Pennsylvania, 171–179 (1964).