INVESTIGATIONS OF SELECTED INITIAL VALUE PROBLEMS USING A DIGITAL SIMULATION LANGUAGE

by 620

RODNEY T. NASH

B. S., Kansas State University, 1967

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1969

Approved by:

Hugh S. Walker

Major Professor

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$\ddot{X},\ddot{Y}$	acceleration, feet/seconds squared
g	acceleration due to gravity, feet/seconds squared
l,L	length, feet
X,Y,Z	dependent variables, (units depend on problem)
T	period, seconds
θ	angular displacement, radians
θ_o	initial angular displacement, radians
T	temperature, °F
T_e	Environment temperature, °F
π	3.1416, dimensionless
t	Time, seconds
Bi	Biot modulus, dimensionless
ω	natural angular frequency, cycles/second
$\dot{\phi}$	driving frequency, cycles/second
m	mass, lb_m
E	Modulus of Elasticity, lb_f/in^2
I	Moment of Inertia, in^4
δ_x,δ_y	Displacement, inches
k	Spring Constant, lb_f/ft
F	Force, lb_f
P	Force, lb_f
W	Weight, lb_f
$\ddot{\theta}$	angular acceleration, radians/sec^2
$\ddot{\delta}_x,\ddot{\delta}_y$	acceleration, in/sec^2
a,b,α_1,α_2	combinations of variables defined when used

CHAPTER 1

INTRODUCTION

Occurrences of initial value problems in practical Engineering work are quite frequent. Sometimes referred to as propagation problems, these problems result from mathematical description of such phenomena as the diffusion of heat or transmission of disturbances by waves. Examples of this type of problem are the heat transfer analysis of systems with known initial temperature distribution, the investigation of motion of vibratory, mechanical systems, the solution of transient responses in electrical and fluidic systems, and the determination of paths for objects in flight. In essence, the initial value problem is taking a known value of a dependent variable and predicting what value that variable will assume at the next step of the independent variable.

Purpose of this Study

Formulation of an initial value problem usually consists of a set of differential equations relating the independent variable to the dependent one and the initial conditions to be employed in starting the solution. One method of solution that has found large acceptance in solving initial value problems is to simulate them on analog computers. The purpose of this report is to discuss a digital computer program which has been developed by International Business Machines, Inc., to carry out an analog type simulation using a digital computer. The name of this particular program is the Continuous System Modeling Program (CSMP) and it is for use with IBM's operating system 360.

Background

Engineers have been seeking solutions to initial value problems as long as the Calculus has been widely applied. The solution procedure took a major step forward when during World War II the operational amplifier was developed. This break through led to the development of the modern electronic analog computer. Analog solution of ordinary differential equations became the accepted method through the 1950's because digital computing was just getting started and to program a digital computer for such work was quite hard. Since the analog computer uses parallel operation, carrying out all the steps of the program simultaneously, it could yield solutions to initial value problems quicker in the execution phase than its digital counterpart in the early days of machine computation. This speed of execution helped to push analog computers to the fore-front in solving initial value problems.

The widespread availability of high speed digital computers along with the development of specialized software (special programs) has caused the advantages of analog over digital to decrease in intensity. For an initial value problem to be simulated on an analog computer, scaling of magnitude of both program variables and time must be done prior to execution in order to keep from over or under driving the amplifiers of the analog computer. Often a programmer attempting a new problem will find scaling a hard task as he will not know what values of his output variables to expect. As the complexity of the problem increases, the problem of scaling becomes even more confusing. This is especially true when there are nonlinearities in the system. Analog computers are further limited by the precision with which output can be interpreted. Analog outputs are usually on oscilloscopes, plotters, or strip chart recorders where the accuracy of interpretation of results depends on the operators ability to read various measuring scales. Further analog computers are limited in the size of system that can be simulated by the number of integrating amplifiers. Creating a wide range of different forcing functions in an analog computer requires a fairly sophisticated set of hardware.

With a digital computer, scaling of variables is not necessary. As many integrations as necessary can be accomplished because the same subroutine carries out all the integration. Almost any type of mathematical function can be generated from random numbers to sine functions. But programming a digital computer to march out the solution of an initial value problem is still a formidable task. Use of aids such as the Scientific Subroutine Package, a set of machine stored subroutines with numerical integration capabilities, still requires considerable programming time for solution of initial value problems.

Digital Simulation Languages

To overcome the lengthy time for coding a digital computer to solve initial value problems, the past few years have witnessed development of several special programs to facilitate solution with only a few steps. Some of these various programs are PACTOLUS (1)*, DIANA (2), MIDAS (3), MIMIC (3), and SIM 1 (4). The first two of these programs work on the so called smaller digital machine (IBM 1620) and the programmer may enter data during the simulation from the computer console during simulation just as he could change potentiometer settings during analog runs. The later programs have the simulation run under complete computer control. With this type of program the machine is charged with the responsibility of changing parameters during the run according to the programmers previous instructions.

One of the most sophisticated of this last type program is the IBM CSMP. The Continuous System Modeling Program is designed to allow problems to be programmed from analog circuit diagrams, block flow diagrams, or directly from the governing differential equations. The language of this program is made up of a set of relatively self explanatory statements which

*Numbers in parentheses refer to references in the List of References.

according to IBM doesn't require extensive understanding of either computer programming (Fortran) or digital computer operation. The CSMP program accepts the users source program and then translates it into Fortran. For this reason Fortran statements are acceptable inputs and some knowledge of this computer language will definitely be helpful in working with CSMP. After the source statements have been translated into Fortran, they are compiled (translated into machine language) in Fortran IV, Level E, linkedited, and then the simulation runs are carried out by the CSMP program. Input and Output statements are simplified by means of a free format. The order in which data statements are placed in a typical program isn't fixed since the CSMP program will sequence the statements so that the simulation may be completed. Output options are tabular printing of variables, print-plotting of variables, maximum-minimum range of variables, and preparation of variables to be used in user supplied plotting programs. CSMP offers the user a choice of seven different kinds of numerical integration. Two of these techniques are of sufficient sophistication to allow the user to specify acceptable error limits.

Typical Applications

Use of these digital simulation programs is receiving considerable use in design and research applications around the country. The Jet Propulsion Laboratory of the California Institute of Technology (2) reports that they have used their digital simulation program to analyze such complex problems as 3 degree of freedom autopilots and soft-landing of a space capsule. Other problems studied were the Mariner 1967 autopilot study, gimballed engine autopilot for Voyager, and a wide-angle gyro attitude control system for the Voyager capsule. Wright-Patterson Air Force Base (3) lists as a sample of problems they have attempted with their digital simulation program such things as pilot ejection studies, longitudinal motion of aircraft studies, and aircraft arresting gear system studies. NASA personnel (4) are reported to have attempted a six-degree-of-freedom problem which was programmed in 3 hours using a digital simulation language, and checked out in 1-1/2 hours of digital computer time. They estimated that programming and check out would have taken about six weeks on the analog computer.

CHAPTER II

DESCRIPTION OF CSMP PROGRAMMING

Procedure Prior to Programming

The first step in preparing an initial value problem for solution by the Continuous System Modeling Program is to set up a mathematical description of the problem in differential equation form. A complete description of the problem will include initial conditions and range of values of varying parameters in addition to the basic differential equations. As an example of this step, the following equations define the mathematical problem associated with the simulation of the free vibration of a simple pendulum.

$$\ddot{\theta} + g/L \sin\theta = 0.0$$
$$\text{initial angle; } \theta_o = 60^o$$
$$\text{initial angular velocity} = 0.0$$
$$g/L = 10.0$$

The appropriate differential equation was derived by using Newton's second law of motion. The initial displacement of 60 degrees from the vertical was selected in order to rule out small angle approximations. Initial velocity, acceleration of gravity, and length of the pendulum were the other system parameters required for a complete description of the problem.

The next step in setting up a CSMP program is to solve the differential equations for the highest order differential in each equation. This is the usual step one takes in setting up equations to be programmed on an analog computer. IBM suggests that people familiar with analog programming may wish to prepare circuit diagrams or block diagrams associated with analog programming to help them visualize the steps of the solution. However, this step doesn't seem to be necessary to the author as the CSMP program can be set up directly from the mathematical description of the problem. The reason for taking this second step is so that the highest order derivative can be integrated to yield the next lowest order derivative. A succession of these step integrations will yield the variables desired as output by the programmer. A further word of explanation of this second step is that the computer reduces higher order differential equations into a set of first order equations in order to use numerical integration on this set of equations. Arranging the derived differential equations in the form outlined in step two is the necessary format for this reduction of order to be carried out.

As an example of the second step, consider a two degree of freedom vibration problem. The example problem is the forced vibration of a shaft with a concentrated mass on the free end of the shaft. The shaft is circular with two flat sides milled on it. As a result of the milling, the principal moments of inertia are different in value. This problems description involves two second order differential equations. These two equations are set up with the second order differentials on the left side of the equations in the required form below.

$$\ddot{X} = \frac{a + b\ \cos(2\dot{\phi}t)}{b^2 - a^2} X + \frac{b\ \sin(2\dot{\phi}t)}{b^2 - a^2} Y$$

$$\ddot{Y} = \frac{b\ \sin(2\dot{\phi}t)}{b^2 - a^2} X + \frac{a - b\ \cos(2\dot{\phi}t)}{b^2 - a^2} Y$$

Language Elements

At this point, one is in a position to begin the programming of his problem. CSMP allows for a free format type of coding. This means that the programmer is free to enter source statements for his program in any column of a standard eighty column card between column one and column seventy-two. A system of equations to be solved by a CSMP program is described to the computer by a series of structure, data, and control statements.

These three kinds of CSMP statements are made up of constants, variable names, operators, and functions. Constants are unchanging quantities used in their numeric form in the various statements. The term constant implies the same meaning in CSMP as in other forms of digital or analog computing.

Variable names are symbolic representations of quantities that may be either changed during a simulation run or be changed between successive runs. Operators used in CSMP are of the same form and meaning as those used in basic Fortran. Table 1 lists these operators and their functions as well as the order in which the computer carries out the operations indicated. Those items of highest hierarchy in any statement are done first then the statements of next rank and so on through the complete statement. Functions are mathematical operations the CSMP program will execute automatically for the user. Chapter III on sample programs will give the reader a better understanding of the function of each of these elements in structuring the CSMP language.

TABLE 1

SYMBOL	OPERATION	ORDER
**	exponentiation	1th
*	multiplication	2nd
/	division	2nd
+	addition	3rd
-	subtraction	3rd

Structure statements describe the functional relationships between the variables of the problem. A person who is familiar with basic Fortran will find that a majority of these structure statements are Fortran statements. However, certain unique functions are provided by the CSMP program which serve to define relationships between variables. The most notable of these added functions is the integration operation. A listing of these functions which range from differentiation to a limiter function and from noise generation to picking the smallest value in a set of numbers can be found in Appendix A. As an example of a set of structure statements, those used in the simple pendulum problem discussed earlier are shown here.

```
X2DOT = - GOVERL * SIN(X)
XDOT = INTGRL(XDOTIC,X2DOT)
X = INTGRL(XIC,XDOT)
```

In these statements the second derivative of X with respect to time is set equal to the negative of the acceleration of gravity divided by the length of the pendulum (- GOVERL) times the sine of X. The parameter GOVERL has a previously defined value. The SIN(X) indicates that the sine series is to be evaluated using the argument X. In the second statement, the first derivative of X is set equal to the integral of X2DOT with XDOTIC as the appropriately defined initial condition to be applied in the integration. This statement tells the CSMP program that it is to apply numerical integration to the second derivative of X with respect to time using the initial condition XDOTIC inorder to obtain the first derivative of X. The last statement repeats the action of the second statement except the first derivative is integrated to yield the variable X. These structure statements need not appear in the source program in any set order. The CSMP program will arrange them in the proper sequence in the translation phase. They are, however, restricted to one statement per card.

Some general rules which must be followed in connection with use of structure statements can be found on page 15 of the CSMP users manual (5). In paraphrased form, the most significant of these requirements are:

a. The arithmetic operators (+,-,*,/,**) may not appear consecutively.
b. Any arguements used in CSMP functions must appear in parenthesis after the functions name and be separated by commas.
c. Parenthesis may be used to clarify the order of arithmetic operations.
d. There is a set of words which are reserved in the CSMP language and cannot be used as variables by the programmer. These are given in the CSMP users manual.
e. Statements may in general be continued for eight cards by use of three periods in a row to indicate that the statement will be continued on the next card.
f. An asterisk in card column one indicates a comment card. On this type of card the information is ignored by the processor and the programmer may enter explanatory information which will appear in the source listing of the program.
g. Columns 73-80 of a standard computer card are ignored by the compiler in processing the structure statements.

Data Statements

The second kind of statement used in programming an initial value problem to be solved with IBM's CSMP are data statements. These statements, as their name suggests, are used to supply information to the program about parameters, constants, initial conditions, and tabular data. The distinction between parameters and constants is that a parameter is free to be changed as the simulation of a problem progresses while a constant is fixed at one value. Only one parameter may vary through a single simulation. Initial conditions are those values to be supplied to the integration routines to start the simulation. Tables of data correspond to subscripted variables in Fortran. That is, it contains variables which take on a different value as the index of its subscript varies.

As an example of some data statements here are those that were used in the two degree of freedom vibration problem mentioned earlier.

```
CONST C1=0.1875,C2=0.2500,...
      E=.300E8,M=0.0060103,L=10.20
PARAM PHIDOT =(104.,10*.1)
INCON XIC=1.0, YIC =0.0,XDOTIC=0.0,YDOTIC=0.0
```

The first data statement is signified by the lable CONST which designates variables appearing on this card as constants. The constants C1 and C2 are assigned numeric values, each constant and its value being separated from the others by a comma. Notice that a blank must follow the card lable CONST but as to other spacing the programmer is free. The three dots after the specification for C2 indicate that more constants will follow on the next card. The value assigned to the constant E is the standard Fortran method of expressing scientific notation. E is set equal to 0.3 time 10 to the eight power or thirty million. PARAM is the label used with the second example data statement, which indicates that variables identified on that card are parameters. In this case the forcing frequency, PHIDOT, is set equal to 104.0 for the first simulation then PHIDOT is increased by 0.1 and another simulation takes place. The number 10 indicates that PHIDOT is to be increased in the above manner 10 times. That is the parameter is to range from 104.0 to 105.0 in increments of one tenth. INCON is the final example card label shown. This card assigned values to the initial conditions which were used in integration functions. Note that a blank follows PARAM and INCON, this blank is required after all card labels.

Another kind of data statement that finds some application is the TABLE data statement. Using this type of data statement allows the programmer to handle large sets of data by referencing it with a single name followed by a varying subscript. Variables defined in this manner are handled just like subscripted variables in Fortran. That is each data point in the table of data is referred to by the name of the table followed by a number in parenthesis that is the position of the desired data element in the table. The number 5 would be associated with the fifth item in the table. As an example this type of statement was used in a CSMP program to simulate the motion of a pole vaulter.

```
TABLE ICG(1-21)=3*20.,19.,17.,8.,6.,8.,14.,18.,11*20.
```

This sample statement defines the moment of inertia with respect to the pole vaulters center of gravity. The moment of inertia has 21 different values as the problem progresses. The elements of ICG are indexed sequentially as they appear in the defining statement. That is the element number 1 is 20.0, the number 2 element is 20.0, the same for the third, and the 4th element has a value of 19.0. To use the variable ICG in a structure statement, it would be referenced as ICG(K) where K would be the integer associated with the Kth element of the table. In order to use a table statement, prior to its appearance in the source program, a certain area of the computer memory must be set aside to store the components of the table. A STORAG label on a card will do this task. The corresponding statement for the above example would be.

```
          STORAG ICG(21)
```

There are two more kinds of data statements available for use with CSMP. The first of these allows for linear interpolation between a select number of data entries by the programmer. The second type does a parabolic interpolation between points. The same sample statement that was discussed for the TABLE data statement would look like this when entered in the program using the arbitrary function generation capabilities just mentioned.

```
AFGEN CURVE1=0.,20.,.05,20.,.1,20.,.15,19.,.2,17.,.25,8..3,6,...
             .35,8.,.4,.4.,.45,18.,.5,20.,.55,20.,.6,20.,.7,20.,
             .75,20.,.8,20.,.85,20.,.9,20.,.95,20.,1.,20.
```

The form of this statement is an independent variable followed by the associated value of a dependent variable. In this case, time is the independent value and for each specification of time there is an enumeriation of the moment of inertia. To use the moment of inertia in the program, the following type of statement would be necessary.

```
ICG=AFGEN(CURVE1,TIME)
```

The action of the program in responce to such a statement would be to assign a value to ICG for any given value of TIME by linearly interpolating the set of values given in the data statement.

Control Statements

The last kind of statements necessary to program CSMP are program control statements. This category is subdivided into three separate kinds of control statements. These are: Translation Control statements; Execution Control statements; and Output Control statements. These titles are self-explanatory as to the functions carried out by the three different kinds of control statements.

Translation Control Statements specify how the structure and data statements are to be translated from CSMP statements into machine language. Each of the different translation control statements are indicated by a different card label. The most often used are discussed here. As before a blank space must follow each card label. Some examples from previously discussed programs of these statements are:

```
RENAME  TIME=TEMP
FIXED  I,J
STORAG  ICG(21)
DECK
NOSORT
   .
   .
   .
SORT
INIT
   .
   .
   .
DYNAM
TERMIN
END
STOP
ENDJOB
```

A card identified by the label RENAME allows the programmer to change the name of one or more of the variables that are assigned a name by CSMP. The independent variable of the standard CSMP simulation is TIME. The example listed above shows this variable being renamed TEMP which is an abbreviation for temperature. Any six character name may be substituted for one of the CSMP variables so long as the new name begins with an alphabetic character and does not contain any embedded blanks or special characters. Renaming variables will facilitate a better description of the physical problem when the CSMP variables do not apply. FIXED on a card allows the variables listed on that card to be consider integers. Whereas, in Fortran I,J,K,L,M, and N as well as any variable whose name begins with these letters are consider integer variables, only those specified in CSMP by a FIXED statement are considered to be integers. STORAG is the label used to reserve space for variables to be entered into the program using a TABLE data statement and was discussed under that topic. DECK allows the user to request that a translated deck of cards be punched so that he may enter his program without the translation phase for successive runs. This feature would be helpful for use in programs that would require extensive simulation runs. Data, execution control, and output control statements are not included in the translated deck and these statements must then be supplied with the punched deck when it is used. The combination of the statements, NOSORT and SORT, allows the programmer to place statements in the CSMP source program which are not sorted by the CSMP program at translation time. These statements that the programmer wishes not to be sequenced by the compiler must come between the two cards with the labels NOSORT and SORT. The included statements are then translated in the order they appear in the NOSORT section. The combination INIT and DYNAM allows the user to put a series of operation in the program which are done only once at the outset of the execution phase. TERMIN indicates that all the statements following the card with the TERMIN label are to be carried out at the completion of the simulation run. END, STOP, and ENDJOB labels signify the end of the CSMP source statements. The reader is referred to the sample programs presented in

Chapter III for examples of application of the Translation Control statements.

The second kind of control statements are Execution control statements. This type of statement is used to specify the actual mechanics of a simulation run. Such things as length of the run, the kind of numerical integration to be used, and the step size to be used with the integration are specified with this category of statement. The three most often used Execution Control statements are examplified by the following examples.

```
TIMER PRDEL=.05, OUTDEL=.05, FINTIM=10.0, DELT=.05
FINISH X=1.1, Y=1.1
METHOD RKSFX
```

The program information supplied by the statement TIMER is data for the integration interval used in each step while marching out the solution. The variable PRDEL is the name associated with the printing out of output variables. That is if the user desires time sequenced values of any of the variables in his program, he can assign a value to PRDEL and receive printed output at his specified increments. OUTDEL has the same function for print plotting of variables against the independent variable. FINTIM places a final value on the independent variable which in this example is 10.0 seconds. The unit of time was determined from the units of physical data in the program. DELT is the value of the integration interval. Each of these variables has program assigned values and unless specified in an Execution Control statement by the programmer, these default options will be used by the CSMP program. Thus the programmer doesn't have to put a value on each of these items unless he chooses to do so. Some experimentation with each problem will usually be necessary before the optimum combination will be determined. FINISH is the statement label of the second statement shown in the example. This statement causes a simulation run to terminate if the values indicated on the card are reached or exceeded. For the example if X or Y were to become larger in absolute value than 1.1 the run would be terminated. The final example is the METHOD statement. Through the use of this card the CSMP user tells the computer which integration technique to choose in carrying out the simulation. RKSFX in the example indicates that a fixed interval Forth-order Runge-Kutta routine is to be employed. The various methods available for use and the appropriate symbolic names are listed below.

SYMBOL	METHOD
ADAMS	Second-order Adams integration with fixed interval
MILNE	Variable-step, fifth-order, predictor-corrector Milne integration
RECT	Rectangular integration
RKS	Fourth-order Runge-Kutta with variable integration interval; Simpson's Rule used for error estimation

RKSFX	Fourth-order Runge-Kutta with fixed interval
SIMP	Simpson's Rule integration with fixed interval
TRAPZ	Trapezoidal integration

The last kind of a control statement is the Output Control statement. These statements are used to indicate which variables are to be printed, which are to be plotted, and what labels are to be put on the tables and graphs. As with other kinds of CSMP statements there is unique set of statement names used for Output Control statements of which these may serve as examples.

```
PRINT X,Y,O,XDOT,YDOT,ODOT,RAD,ICG
TITLE OUTPUT VARIABLES
PRTPLT X,Y,O,C,ETA
LABEL DISPLACEMENT
RANGE X,Y
```

The variables listed after PRINT will be printed out over the range of time the simulation lasts in increments specified by PRDEL. The statement after the card label TITLE will appear at the top of each page of printing. PRTPLT indicates which variables are to be print-plotted against time according to the increment size defined by OUTDEL. The label specified by LABEL will appear on each print-plot. RANGE causes the maximum and minimum values of a variable to be printed out along with the time they occurred after each simulation.

Procedural Blocks

The discussion of the basic approach to and elements of CSMP programming is now complete. This material is not an exhaustive treatment of the subject and the reader may wish to refer to the various IBM reference manuals listed in the List of References as well as to the sample programs of Chapter III for more information. Just the basic elements of the CSMP program have been outlined so far. To complete Chapter II, grouping these basic elements into groups to act as procedural blocks will be discussed. CSMP provides for three different ways of combining the basic elements of CSMP and Fortran together in procedural blocks.

The first of these ways is referred to as a MACRO. It is a function-defining capability which allows the CSMP user to designate a group of mathematical statements as a function and then reference this block by a single name. That is when the programmer has a set of mathematical steps in his program which is used several times but with different input parameters each time, he may define a functional block and then supply the appropriate variables to the function instead of writing the equations each time he wants the calculations performed. The form for this block is the card label MACRO followed by a dummy variable which is set equal to the function with appropriate inputs in parenthesis after the name. Following the MACRO card are the cards with statements defining the function. Any CSMP structure statement may be used. Somewhere in these statements the dummy variable used on the first card has to be set equal to the desired output. Multiple outputs may be used but multiple dummy variables must be placed in the MACRO statement and these separated by commas. After the cards with the mathematical statements, a card with the label ENDMAC is required to terminate the function block. This block of cards must appear in the source program prior to any structure statements. A MACRO would look something like the following sample.

```
MACRO IDOT,I,C,ETA,DEM1=CALC(ICG,RAD,TIME,DELT,M,O,Y,X)
  .
  .   (structure statements)
  .
ENDMAC
```

The second of these methods for handling several statements as a unit, is referenced by the card label PROCED. Its format is the same as a MACRO's except the end statement for this block is ENDPRO. The only differences between these first two blocks is that the PROCED block can be used only once per simulation and the PROCED block is not sorted by the CSMP program. If a PROCED block is put inside a MACRO block it can be used as often as the MACRO. PROCED blocks may be placed anywhere in the CSMP program except inside another PROCED block.

The third method of handling groups of statements is the use of Fortran subroutines. Both the function subroutine and the regular subroutine are allowed to be used in CSMP. As with Fortran, the function subprogram is used for multiple outputs. All Fortran rules apply to subroutine usage, however, the reader who is familiar with Fortran will note that the CSMP statement for calling a subroutine is different from Fortran. The statement

```
IDOT,I,C,ETA,DEM1=CALC(ICG,RAD,TIME,DELT,M,O,Y,X)
```

calls a subroutine whose symbolic name is CALC. This is the same form used to reference MACROs and PROCEDs. The inputs to this subroutine from the main CSMP program are listed in parenthesis after the subroutine name and the outputs are listed on the left side of the equal sign. The standard Fortran card to go at the beginning of CALC would be.

```
SUBROUTINE CALC(ICG,R,TIME,DELT,M,O,Y,X,IDOT,I,C,ETA,DEM1)
```

All subroutines must be placed after the STOP card and before the ENDJOB card prior to processing a CSMP program.

CHAPTER III

SAMPLE CSMP PROGRAMS

This chapter consists of description of four sample CSMP programs. Since the main emphasis here is on the digital simulation language, little discussion of the problems themselves will be included. In each case, however, some background information will be presented along with the sample program and a discussion of what each statement in it does.

Simple Pendulum

When small angle approximation are applicable ($\sin \theta = \theta$, $\cos \theta = 1$), the analytical solution of the differential equation is a linear combination of sine and cosine terms. However, when the displacement of the pendulum exceeds, approximately twenty degrees from the vertical, the analytical solution becomes much more complex since the small angle approximations are no longer valid. For displacements greater than twenty degrees, the mathematical procedure for solution yields an elliptical integral that defines the period of oscillation.

As a test of CSMP the governing differential equation for a simple pendulum

$$\ddot{x} = -\frac{g}{l} \sin x$$

was programmed using an initial displacement of sufficient magnitude as to rule out the use of small angle approximations. In this problem the ratio of acceleration due to gravity to the length of the pendulum ($\frac{g}{l}$) was arbitrarily picked as 10.0. The other physical parameters chosen to define the problem were initial displacement (60°), initial pendulum acceleration (0.0), and problem duration (5 seconds). The CSMP statements used to simulate this problem are listed below. Those statements which appear in the program with an asterisk as the first character are comment cards. The numbers which appear at the right hand side of the program are merely identification numbers.

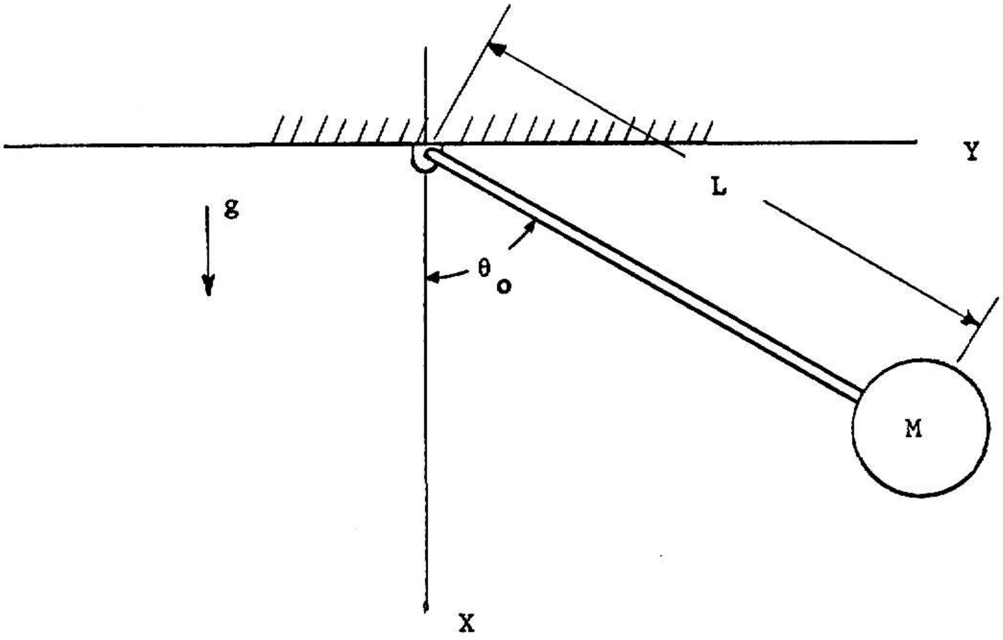


Figure 1. Schematic View of a Simple Pendulum.

```
*                 ***PENDULUM SIMULATION***
*
*                    **DATA STATEMENTS**
*
PARAM GOVERL=10.0                                                  1
INCON XIC=1.0471975,XDOTIC=0.0                                     2
*
*              **EXECUTION CONTROL STATEMENTS**
*
TIMER PRDEL=.05,OUTDEL=.05,FINTIM=5.0                              3
*
*                **OUTPUT CONTROL STATEMENTS**
*
PRINT X2DOT,XDOT,X                                                 4
TITLE SIMPLE PENDULUM ACCELERATION, VELOCITY, AND DISPLACEMENT     5
PRTPLT X                                                           6
LABEL PENDULUM MOTION                                              7
*
*                 **STRUCTURE STATEMENTS**
*
       X2DOT = -GOVERL*SIN(X)                                      8
       XDOT=INTGRL(XDOTIC,X2DOT)                                   9
       X=INTGRL(XIC,XDOT)                                         10
*
*                     **TRANSLATION CONTROL STATEMENTS**
*
END                                                               11
STOP                                                              12
ENDJOB                                                            13
```

The first card of the program is labeled PARAM. This card defines the ratio g/l that is to be used in this particular simulation of a pendulum. By using a parameter card to designate a value of g/l, the programmer allows himself the option of changing this ratio for other runs. A separate simulation of the problem will be carried out for each value that appears on the PARAM card up to fifty runs.

The second card of the program specifies the initial angular displacement and initial angular velocity. 1.0471975 is the initial angle in radians which is equivalent to sixty degrees.

The timer card sets the printing increment for output at one hundredth of a second, the increment for plotting at the same value, and specifies the length of the simulation is to be five seconds. The fourth and fifth cards indicate which variables are to be printed and what heading is to appear at the top of the page of printed output. Cards six and seven signify which variables to plot against time and what lable to put on the dependent variable axis.

The governing differential equation appears on the eighth card. All these cards allow free format and this statement was only set further to the right in order to distinguish it as a structure statement. However, such a step is neither necessary or strongly recommended. The ninth and tenth cards complete the mathematical structuring of the problem. The final three statements merely tell the compiler that the program is finished.

Inorder to evaluate the results obtained by this particular simulation, the period for this pendulum was evaluated using

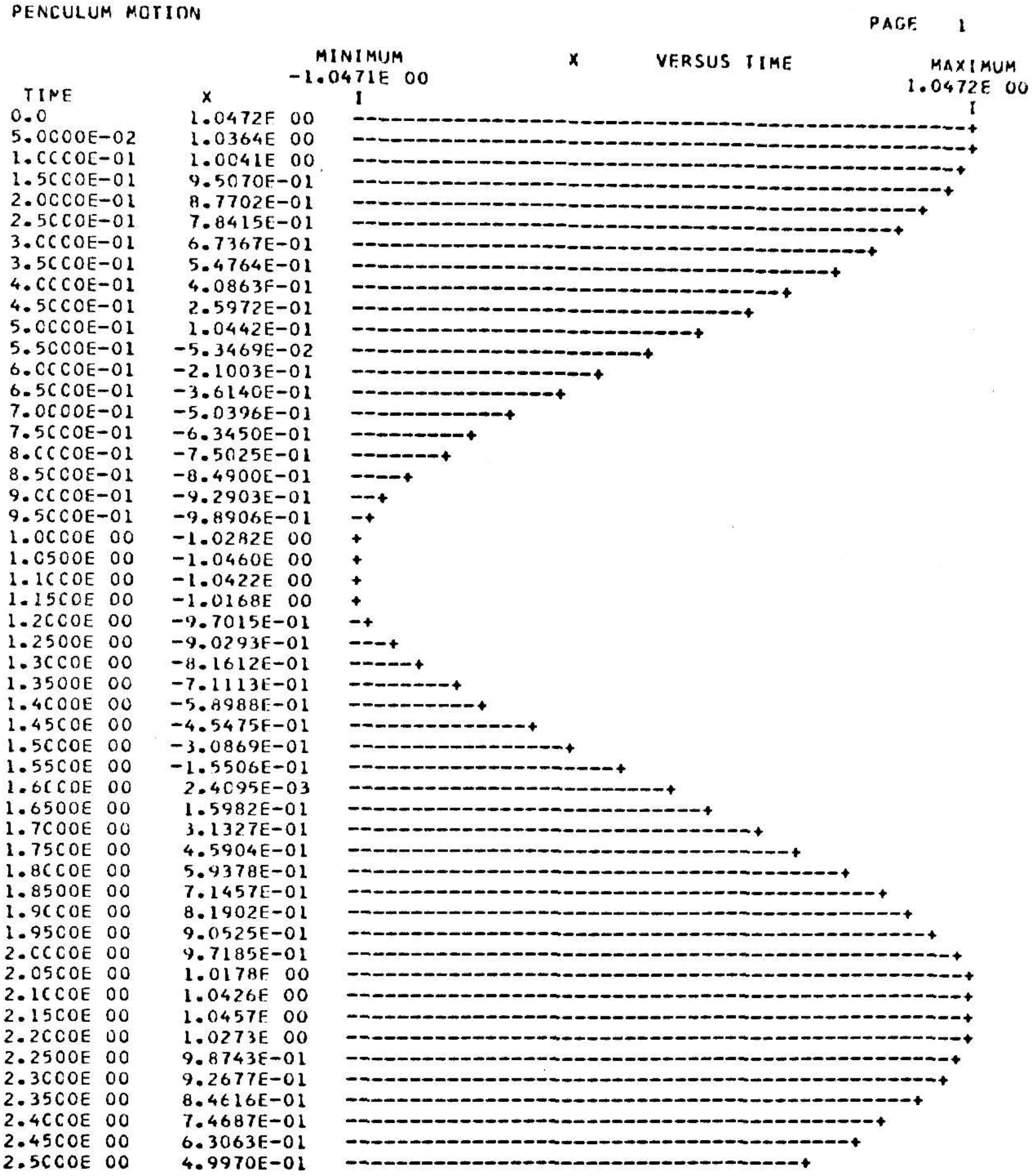
$$T = (4/\sqrt{g/l})\int_0^{\pi/2} \frac{d\theta}{\sqrt{1-\sin^2(\frac{\theta_o}{2})\sin^2\theta}}$$

which is the analytical result for large oscillations. (9) Using a θ_0 of 60 degrees, the above expression yielded a value of 2.133 seconds as the length of the period. Using the CSMP

```
SIMPLE PENDULUM ACCELERATION, VELOCITY, AND DISPLACEMENT

 TIME           X2DOT          XDOT            X
0.0            -8.6603E 00  0.0            1.0472E 00
5.00C0E-02 -8.6056E 00 -4.3210E-01  1.0364E 00
1.00C0E-01 -8.4366E 00 -8.5866E-01  1.0041E 00
1.50C0E-01 -8.1382E 00 -1.2736E 00  9.5070E-01
2.00C0E-01 -7.6883E 00 -1.6699E 00  8.7702E-01
2.5000E-01 -7.0622E 00 -2.0395E 00  7.8415E-01
3.0CC0E-01 -6.2386E 00 -2.3729E 00  6.7367E-01
3.50C0E-01 -5.2067E 00 -2.6599E 00  5.4764E-01
4.00C0E-01 -3.9735E 00 -2.8901E 00  4.0863E-01
4.5CC0E-01 -2.5681E 00 -3.0543E 00  2.5972E-01
5.00C0E-01 -1.0423E 00 -3.1449E 00  1.0442E-01
5.50C0E-01  5.3443E-01 -3.1577E 00 -5.3469E-02
6.00C0E-01  2.0849E 00 -3.0919E 00 -2.1003E-01
6.50C0E-01  3.5359E 00 -2.9508E 00 -3.6140E-01
7.0000E-01  4.8290E 00 -2.7410E 00 -5.0396E-01
7.50C0E-01  5.9277E 00 -2.4712E 00 -6.3450E-01
8.00C0E-01  6.8182E 00 -2.1516E 00 -7.5025E-01
8.50C0E-01  7.5062E 00 -1.7927E 00 -8.4900E-01
9.00C0E-01  8.0104E 00 -1.4041E 00 -9.2903E-01
9.50C0E-01  8.3551E 00 -9.9433E-01 -9.8906E-01
1.00C0E 00  8.5638E 00 -5.7083E-01 -1.0282E 00
1.0500E 00  8.6543E 00 -1.3991E-01 -1.0460E 00
1.10C0E 00  8.6350E 00  2.9278E-01 -1.0422E 00
1.15C0E 00  8.5042E 00  7.2174E-01 -1.0168E 00
1.20C0E 00  8.2497E 00  1.1411E 00 -9.7015E-01
1.25C0E 00  7.8514E 00  1.5443E 00 -9.0293E-01
1.30C0E 00  7.2849E 00  1.9235E 00 -8.1612E-01
1.35C0E 00  6.5269E 00  2.2696E 00 -7.1113E-01
1.40C0E 00  5.5626E 00  2.5727E 00 -5.8988E-01
1.45C0E 00  4.3924E 00  2.8224E 00 -4.5475E-01
1.50C0E 00  3.0381E 00  3.0088E 00 -3.0869E-01
1.55C0E 00  1.5444E 00  3.1238E 00 -1.5506E-01
1.60C0E 00 -2.4095E-02  3.1620E 00  2.4095E-03
1.65C0E 00 -1.5914E 00  3.1214E 00  1.5982E-01
1.70C0E 00 -3.0817E 00  3.0041E 00  3.1327E-01
1.75C0E 00 -4.4309E 00  2.8156E 00  4.5904E-01
1.80C0E 00 -5.5950E 00  2.5641E 00  5.9378E-01
1.85C0E 00 -6.5529E 00  2.2595E 00  7.1457E-01
1.90C0E 00 -7.3048E 00  1.9123E 00  8.1902E-01
1.95C0E 00 -7.8658E 00  1.5322E 00  9.0525E-01
2.00C0E 00 -8.2593E 00  1.1285E 00  9.7185E-01
2.05C0E 00 -8.5097E 00  7.0869E-01  1.0178E 00
2.10C0E 00 -8.6369E 00  2.7954E-01  1.0426E 00
2.15C0E 00 -8.6528E 00 -1.5316E-01  1.0457E 00
2.20C0E 00 -8.5588E 00 -5.8391E-01  1.0273E 00
2.25C0E 00 -8.3461E 00 -1.0071E 00  9.8743E-01
2.30C0E 00 -7.9969E 00 -1.4162E 00  9.2677E-01
2.35C0E 00 -7.4874E 00 -1.8041E 00  8.4616E-01
2.40C0E 00 -6.7934E 00 -2.1619E 00  7.4687E-01
2.45C0E 00 -5.8966E 00 -2.4800E 00  6.3063E-01
2.50C0E 00 -4.7916E 00 -2.7481E 00  4.9970E-01
```

Figure 2. Printed Output from Pendulum Problem.



```
PENCULUM MOTION
                                                                    PAGE   1
                        MINIMUM             X      VERSUS TIME          MAXIMUM
                      -1.0471E 00                                      1.0472E 00
 TIME          X           I                                                I
0.0           1.0472E 00   -------------------------------------------------+
5.0000E-02    1.0364E 00   -------------------------------------------------+
1.0000E-01    1.0041E 00   ------------------------------------------------+
1.5000E-01    9.5070E-01   -----------------------------------------------+
2.0000E-01    8.7702E-01   ---------------------------------------------+
2.5000E-01    7.8415E-01   -------------------------------------------+
3.0000E-01    6.7367E-01   -----------------------------------------+
3.5000E-01    5.4764E-01   --------------------------------------+
4.0000E-01    4.0863E-01   ----------------------------------+
4.5000E-01    2.5972E-01   -------------------------------+
5.0000E-01    1.0442E-01   ---------------------------+
5.5000E-01   -5.3469E-02   -----------------------+
6.0000E-01   -2.1003E-01   -------------------+
6.5000E-01   -3.6140E-01   ----------------+
7.0000E-01   -5.0396E-01   ------------+
7.5000E-01   -6.3450E-01   ---------+
8.0000E-01   -7.5025E-01   -------+
8.5000E-01   -8.4900E-01   ----+
9.0000E-01   -9.2903E-01   --+
9.5000E-01   -9.8906E-01   -+
1.0000E 00   -1.0282E 00   +
1.0500E 00   -1.0460E 00   +
1.1000E 00   -1.0422E 00   +
1.1500E 00   -1.0168E 00   +
1.2000E 00   -9.7015E-01   -+
1.2500E 00   -9.0293E-01   ---+
1.3000E 00   -8.1612E-01   -----+
1.3500E 00   -7.1113E-01   --------+
1.4000E 00   -5.8988E-01   ----------+
1.4500E 00   -4.5475E-01   --------------+
1.5000E 00   -3.0869E-01   -----------------+
1.5500E 00   -1.5506E-01   ---------------------+
1.6000E 00    2.4095E-03   -------------------------+
1.6500E 00    1.5982E-01   ----------------------------+
1.7000E 00    3.1327E-01   --------------------------------+
1.7500E 00    4.5904E-01   -----------------------------------+
1.8000E 00    5.9378E-01   ---------------------------------------+
1.8500E 00    7.1457E-01   ------------------------------------------+
1.9000E 00    8.1902E-01   --------------------------------------------+
1.9500E 00    9.0525E-01   ----------------------------------------------+
2.0000E 00    9.7185E-01   ------------------------------------------------+
2.0500E 00    1.0178E 00   -------------------------------------------------+
2.1000E 00    1.0426E 00   -------------------------------------------------+
2.1500E 00    1.0457E 00   -------------------------------------------------+
2.2000E 00    1.0273E 00   -------------------------------------------------+
2.2500E 00    9.8743E-01   ------------------------------------------------+
2.3000E 00    9.2677E-01   -----------------------------------------------+
2.3500E 00    8.4616E-01   ---------------------------------------------+
2.4000E 00    7.4687E-01   ------------------------------------------+
2.4500E 00    6.3063E-01   ----------------------------------------+
2.5000E 00    4.9970E-01   ------------------------------------+
```

Figure 3. Print-plotted Output from Pendulum Problem.

program it was determined that the period of the pendulum was between 2.13 and 2.14 seconds. By reducing the printing increment, this result may have been improved. A sample of both the printed and plotted output of this program is shown on page

Cooling of an Object

The physical problem to be modeled in this example problem is that of a small object at a given temperature which is placed in an environment at some lower temperature. For the simulation problem, it was assumed that the internal conductive resistance of the object was so small that the object temperature was always uniform throughout the object at each instance of time. This simplification is justified when the external thermal resistance between the surface of the system and the surrounding medium is so large compared to the internal resistance of the object that it controls the heat transfer process. Although there are no materials in nature that would conform to this assumption exactly, there are several problems which can be approximated using this analysis. A typical example of this type of transient heat flow is the cooling of a small metal casting or a billet in a quenching bath after its removal from a hot furnace. (10)

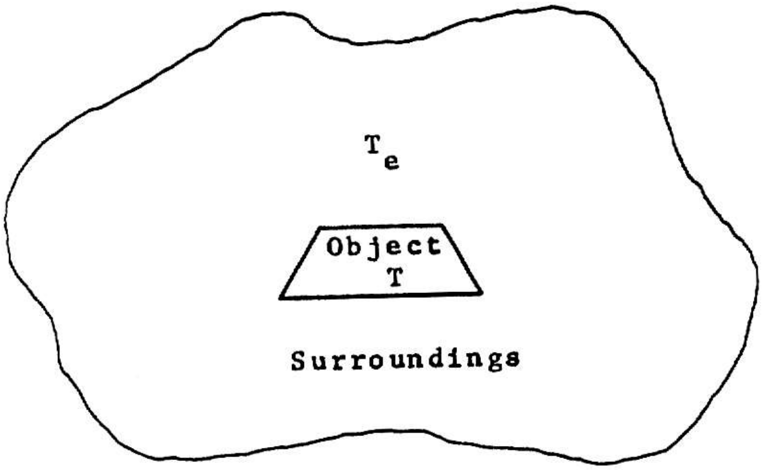


Figure 4. Cooling Problem.

Newton's law of cooling

$$-\frac{dT}{dt} = Bi\ (T-T_e)$$

describes the time rate change of temperature in the object under analysis. For the sample problem, the parameter Bi was assumed constant at a value of 0.1. This factor is referred to in the heat transfer literature as the Biot number and it is the dimensionless ratio of the internal resistance of the object to the external thermal resistance. The criterion used to establish the applicability of the analysis outlined here is that the Biot number be 0.1 or less (10).

The analytical solution of the governing time dependent differential equation would predict that the temperature the test object decreases according to some exponential function. To complicate this solution and to display one of the mathematical functions of CSMP, the sample problem cooling environment temperature was defined to remain constant at 100°F for the first second the object was in it. Then the environment temperature was defined to increase as a ramp function with a slope of five for the rest of the simulation run.

The sample program for modeling the cooling of an object is shown below. As before and in all CSMP programs, those statements which begin with an asterisk as the first symbol are merely comment statements. These cards in the program have information for the programmer about the structure of the program and are ignored by the CSMP program translator.

```
*            ***COOLING SIMULATION***
*
*              **DATA STATEMENTS**
*
CONST  BI=0.1                                              1
INCON  TIC=500.0                                           2
*
*            **EXECUTION CONTROL STATEMENTS**
*
TIMER PRDEL=0.05,OUTDEL=0.05,FINTIM=10.0                   3
METHOD RECT                                                4
*
*              **OUTPUT CONTROL STATEMENTS**
*
PRINT T,TDOT,TSUR,Y                                        5
PRTPLT T,TDOT,TSUR,Y                                       6
*
*              **STRUCTURE STATEMENTS**
*
       TDOT=-BI*(T-TSUR)                                   7
       T=INTGRL(TIC,TDOT)                                  8
       Y=RAMP(1.0)                                         9
       TSUR=100.0-5.0*Y                                   10
*
*              **TRANSLATION CONTROL STATEMENTS**
```

```
*
END
STOP
ENDJOB
```

Card number one begins the problem definition. This card defines the value to be associated with the symbol BI throughout the simulation run. The next card sets the initial conditions to be used for temperature. Card number three sets the increment for printing and plotting output as well as the time at which the run is to be terminated. The METHOD card defined the particular numerical integration technique to be employed during the simulation. PRINT and PRTPLT designate those variables which are to be listed for each time increment and those which are to be print-plotted. The first statement after the comment, STRUCTURE STATEMENTS, is the governing differential equation. The following statement integrates the governing differential equation. Statement number 9 is an application of one of the CSMP function blocks. In this case it is a ramp function. This statement indicates that Y is to be assigned the ordinate value of a ramp function with time as the independent value beginning at TIME=1.0. Thus Y is a 45° line passing through the point 1.0 on the time axis. The 10th statement defines the environment temperature as 100°F plus five times the current value of Y. The last three statements indicate that the source program is complete.

Table 2: Comparison of execution times for CSMP integration methods used in the object cooling problem.

Integration Method	Execution Time (min.)
RKS	3.67
TRAPZ	3.16
MILNE	2.88
SIMP	3.23
RECT	3.07
ADAMS	3.00
RKSFX	3.11

Table 3: Comparison of Temperatures at various times for the cooling problem using the seven numerical integration techniques.

Time	RKS	TRAPZ	MILNE	SIMP	RECT	ADAMS	RKSFX
0.0	500.0	500.0	500.0	500.0	500.0	500.0	500.0
1.0	461.9	461.9	461.9	461.9	461.9	461.9	461.9
2.0	427.7	427.7	427.7	427.7	427.7	427.7	427.7
3.0	397.3	397.2	397.3	397.2	397.2	397.2	397.2
4.0	370.2	370.1	370.2	370.1	370.0	370.1	370.1
5.0	346.1	346.0	346.1	346.0	346.0	346.0	346.0
6.0	324.8	324.7	324.8	324.7	324.7	324.7	324.7
7.0	306.1	305.9	306.9	305.9	305.9	305.9	305.9
8.0	289.5	289.4	289.5	289.4	289.4	289.4	289.4
9.0	275.1	274.9	275.1	274.9	274.9	274.9	274.9
10.0	262.5	262.3	262.4	262.3	262.2	262.3	262.3

As a comparison of the effect of the use of the various kinds of numerical integration available with CSMP, this sample program was run using each kind. The results of these runs are compared in tables 1 and 2 which are on page . As one may observe from the tabulated results, the fifth order Milne predictor corrector method had the best execution time. A general conclusion about the speed or accuracy of these various techniques cannot be drawn from this test. Each individual simulation would require some investigation of this type to determine the best method for the particular problem. The CSMP program will use RKSFX if a method is not specified and for most short problems and initial attempts at simulation this method will suffice.

Stability Study of a Rotating Shaft with Unsymmetrical Cross Section

The problem simulated by this sample program was that of a shaft with a concentrated mass on the free end whose cross section had different principal moments of inertia. Simulation runs were carried out to determine the range of speeds at which the vibratory motion became unstable. A cross-section of the shaft is shown in figure 3. In magnitude, the moment of inertia with respect to the x axis exceeded the moment of inertia with respect to the y axis.

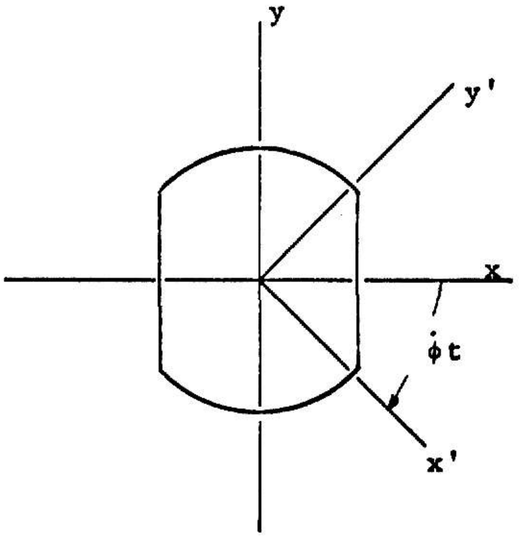


Figure 5. Shaft Cross-section with Fixed and Rotating Coordinate Systems.

Procedurally the solution consisted of programming the governing differential equations, mathematically displacing the shaft from equilibrium, stepping the solution out through time for a wide range of shaft speeds, and observing where the amplitude of the vibrations grew. The unstable region was then defined as being those shaft speeds which caused the displacements to keep growing in magnitude as time progressed in each simulation.

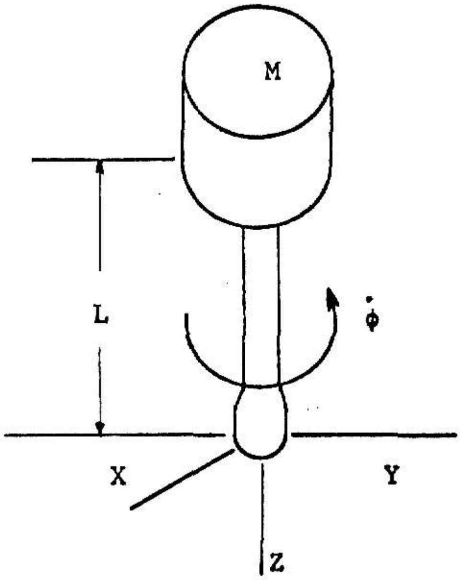


Figure 6. Frontal View of the Shaft.

Bolotin (11) predicted that the unstable region would be between the exciting frequencies which were numerically equal to the natural frequencies of the shaft acting as a cantilever beam in single degree of freedom vibration along the y axis, for one bound and the x axis for the other. That is to say if one looks at the shaft as a cantilever beam with the mass as a load supported at the free end, then the natural frequency associated with that system will be equivalent to the frequency that defines the boundary of the unstable region for the two degree of freedom problem. Since the shaft can move either along y or x axis, then one obtains the two boundary points necessary to define the region. Algebraically these natural frequencies are:

$$\omega_{\text{upper}} = \sqrt{\frac{3\ E\ I_x}{m\ L^3}}$$

$$\omega_{\text{lower}} = \sqrt{\frac{3\ E\ I_y}{m\ L^3}}$$

These equations were developed by considering the shaft as having a spring constant of $3EI/L^3$ and plugging this spring constant into the expression for the natural frequency of a spring-mass system, $\sqrt{k/m}$.

To do the CSMP analysis of the two degree of freedom problem, the coupled differential equations were derived using a rotating coordinate system which was fixed on the equilibrium center line of the shaft. The coordinate system rotated with the angular speed of the forcing device. The derivation is a straight forward application of dynamics and since it is not the main topic, it is presented in the appendix. The resulting differential equations from the derivation were:

$$\ddot{x} + \frac{a + b\ \cos 2\dot{\phi} t}{a^2 - b^2}\ x + \frac{b\ \sin 2\dot{\phi} t}{a^2 - b^2}\ y = 0$$

$$\ddot{y} + \frac{b\ \sin 2\dot{\phi} t}{a^2 - b^2}\ x + \frac{a - b\ \cos 2\dot{\phi} t}{a^2 - b^2}\ y = 0$$

where

$$a = \frac{m(\alpha_2+\alpha_1)}{2} \quad ; \quad b = \frac{m(\alpha_2-\alpha_1)}{2}$$

$$\alpha_1 = \frac{L^3}{3EI_y} \quad ; \quad \alpha_2 = \frac{L^3}{3EI_x}$$

As mentioned earlier in this report, to program these equations using CSMP it is necessary to set them up so that the highest order differential appears on the left of both equations. This procedure transforms the differential equations into the following form.

$$\ddot{x} = \frac{a + b\ \cos 2\dot{\phi}t}{b^2 - a^2}\ x + \frac{b \sin(2\dot{\phi}t)}{b^2 - a^2}\ y$$

$$\ddot{y} = \frac{b\ \sin 2\dot{\phi}t}{b^2 - a^2}\ x + \frac{a - b\ \cos 2\dot{\phi}t}{b^2 - a^2}\ y$$

The sample program for this problem is listed below.

```
*       ***SHAFT SIMULATION***
*
*       **INITIALIZATION OF PHYSICAL PARAMETERS**
*
INTT
        C3=(C1*C1)-((C2*C2)/4.0)
        C4=((C1*C1)*C2)/4.
        C5=(C1**4)/2.
        C6=C2/(2.*C1)
        IXP=(C2/6.)*(C3--3)+C4*(C3**.5)+C5*ARSIN(C6)
        IYP=-(C2/2.)*(C3**1.5)+C4*(C3**.5)+C5*ARSIN(C6)
        ALPHA1=(L**3)/(3.*E*IYP)
        ALPHA2=(L**3)/(3.*E*IXP)
        A=M*((ALPHA1+ALPHA2)/2.)
        B=M*((ALPHA2-ALPHA1)/2.)
        DEM=(B*B)-(A*A)
DYNAM
```

```
*
*                    **DATA STATEMENTS**
*
CONST C1=0.1875,C2=0.2500,...
      E=.300E8,M=0.0060103,L=10.20
PARAM PHIDOT=(103.4,1*.1)
INCON XIC=1.0,YIC=0.0,XDOTIC=0.0,YDOTIC=0.0
*
*                      **EXECUTION CONTROL STATEMENTS**
*
TIMER PRDEL=.05,FINTIM=30.0
RANGE X,Y
METHOD RKSFX
*
*                      **OUTPUT CONTROL STATEMENTS**
*
PRINT X,Y
TITLE X AND Y DISPLACEMENTS
*
*                      **STRUCTURE STATEMENTS**
*
      C=COS(2.0*PHIDOT*TIME)
      S=SIN(2.0*PHIDOT*TIME)
      MULTX1=(A+B*C)/DEM
      MULTX2=(B-S)/DEM
      MULTY1=(A-B*C)/DEM
      MULTY2=MULTX2
      X2DOT=MULTX1*X+MULTY2*Y
      Y2DOT=MULTX2*X+MULTY1*Y
      XDOT=INTGRL(XDOTIC,X2DOT)
      YDOT=INTGRL(YDOTIC,Y2DOT)
      X=INTGRL(XIC,XDOT)
      Y=INTGRL(YIC,YDOT)
TERMIN
      PERIOD=((2.0*3.1415927)/PHIDOT)*1000.0
      WRITE(6,1)PERIOD
      FORMAT(20X,13H***PERIOD=E14.6,12HMILLISECOND)
END
PARAM PHIDOT=(76.0,10.*.1)
END
STOP
ENDJOB
```

The section of the program that is started by INIT and ended by DYNAM is an initialization section. Its operations were performed before the simulation run began. It was used in this program to calculate the moments of inertia and the constants α_1, α_2, a and b.

Cards numbered 14 and 15 were used to define certain physical constants of the system. C1 and C2 were dimensions of the shaft, E is the Modulus of Elasticity, M is the mass concentrated at the shaft's end, and L is the shaft's length. The definition of constants is continued from card 14 to card 15 by the three periods after C2. The next card in the data statements section is a PARAM card which defines the shaft speeds to be used in the simulation runs. This example starts with PHIDOT equal to 103.4 and preforms one more simulation with PHIDOT equal to 103.5. The statement specifies that after the first simulation using PHIDOT = 103.4, the value is to be incremented once by 0.1. The rest of the statements in this program down to TERMIN are similar to those that appeared in the first two examples.

Statements that appear between TERMIN and END are executed only at the end of each simulation run. The statements in the TERMIN section calculated the period associated with the shafts circular frequency. Then this value was written out along with regular output using a standard Fortran WRITE statement. An END statement terminates the TERMIN section as well as the specifications for the simulation run.

After the first END statement, a PARAM card is used to set new values for PHIDOT. This card calls for eleven more runs, each with a different value of PHIDOT. All the statements and variables accept PHIDOT stay the same for this set of runs. If other statements would have appeared between the two END statements they would have replaced the original statements in the second set of simulation runs. The last three cards tell the processor that the source program is completed.

Pole Vaulter

The problem simulated in this example is that of a person engaged in pole vaulting. Simulation begins the instant the vaulter plants his pole in the pit and continues until he has cleared the bar. Inorder to mathematically describe the forces in the pole, the Theory of the Elastica (14) was used. The vaulter was described as a body with time varying radius between its center of rotation (the point where the vaulter holds the pole) and its center of gravity. Also the vaulter was considered to have a moment of inertia about his center of gravity which was time dependent. Purely arbitrary curves where used to portray these time dependent properties. Complete investigation of this non-linear problem would necessitate several runs with different sets of the time dependent curves to improve the mathematical model. However, mathematical analysis is not the objective of this report and further discussion of the model will not be presented here. A brief outline of the model development is presented in the Appendix for the readers consideration.

With CSMP, the programmer has considerable leeway in attacking initial value problems, that is there are several alternate ways of handling various problems. For that reason, this problem was worked out using two different methods. The sample programs used in these different approaches are listed below. They are referred to as programs one or two and will be referenced according to that notation. The reader may observe that these programs call for access to Fortran subroutines CAL, CALC, ROOT. As a point of clarification, the subroutine ROOT further calls another subroutine FUN. These subroutines are written in Fortran IV, level E, and appear in the Appendix.

Program 1

```
AFGEN  CURVE1=0.,20.,.05,20.,.1,20.,.15,19.,.2,17.,.25,8.,.3,6.,...   1
       .35,8.,.4,14.,.45,18.,.5,20.,.55,20.,.6,20.,.65,20.,.7,20.,...2
       .75,20.,.8,20.,.85,20.,.9,20.,.95,20.,1.,20.                    3
AFGEN  CURVE2=0.,3.5,.2,3.5,.25,3.4,.3,3.1,.35,2.7,.4,2.,.45,1.,...    4
       .5,.5,.55,.5,.6,2.7,.65,3.5,.7,3.5,1.,3.5                       5
CONST  LL=14.0,BETA=7000.,W=200.,G=32.174,PI=3.1415926                 6
INCON  XIC=9.57,XDOTIC=-27.3,YIC=4.83,YDOTIC=10.0,OIC=.872,ODOTIC=3.57
TIMER  PRDEL=.05,OUTDEL=.05,FINTIM=1.0,DELT=.05                        8
PRINT  X,Y,I,C1,ETA,RAD,ICG,O,IDOT,KK,FX,FY                            9
       M=W/G                                                          10
       ICG=AFGEN(CURVE1,TIME)                                         11
       RAD=AFGEN(CURVE2,TIME)                                         12
       I,IDOT,C1,ETA,DEM1,=CAL(ICG,RAD,O,Y,X,M,DELT,TIME)             13
       L=SQRT(DEM1)                                                   14
       KK1=ROOT(L,LL)                                                 15
       KK=KK1*PI/2.                                                   16
       FORCE=(4.*BETA*KK*KK)/(LL*LL*L)                                17
       FX=FORCE*C1                                                    18
       FY=FORCE*ETA                                                   19
       Y2DOT=FY/M-G                                                   20
       X2DOT=FX/M                                                     21
       O2DOT=(1./I)*(FY*(C1-X)-FX*(ETA-Y)-IDOT*ODOT)                  22
       YDOT=INTGRL(YDOTIC,Y2DOT)                                      23
       XDOT=INTGRL(XDOTIC,X2DOT)                                      24
```

```
      ODOT=INTGRL(ODOTIC,O2DOT)
      Y=INTGRL(YIC,YDOT)
      X=INTGRL(XIC,XDOT)
      O=INTGRL(OCI,ODOT)
END
STOP
ENDJOB
```

Program 2

```
CONST LL=14.0,BETA=7000.,W=200.,G=32.172,PI=3.1415926
INCON XIC9.57,XDOTIC=-27.3,YIC=4.83,YDOTIC=10.0,OIC=.872,ODOTIC=3.5
TIMER PRDEL=.05,OUTDEL=.05,FINTIM=1.0,DELT=.05
PRINT X,Y,O,XDOT,YDOT,ODOT,IDOT,I,C1,ETA,ICG,RAD
PRTPLT X,Y,O,C1,ETA
     M=W/G
     I,IDOT,C1,ETA,DEM1,ICG,RAD=CALC(O,Y,X,M)
     L=SQRT(DEM1)
     KK1=ROOT(L,LL)
     KK=KK1*PI/2.
     FORCE=(4.*BETA*KK*KK)/(LL*LL*L)
     FX=FORCE*C1
     FY=FORCE*ETA
     Y2DOT=FY/M-G
     X2DOT=FX/M
     O2DOT=(1./I)*(FY*(C1-X)-FX*(ETA-Y)-IDOT*ODOT)
     YDOT=INTGRL(YDOTIC,Y2DOT)
     XDOT=INTGRL(XDOTIC,X2DOT)
     ODOT=INTGRL(ODOTIC,O2DOT)
     Y=INTGRL(YIC,YDOT)
     X=INTGRL(XIC,XDOT)
     O=INTGRL(OIC,ODOT)
END
STOP
ENDJOB
```

The two programs just presented differ only in the way they handle the data for the time dependent moment of inertia, ICG, and the time dependent distance between the center of gravity and the hands of the vaulter, RAD. Program one used the arbitrary function generation capability of CSMP for this purpose while program two read the data from punched cards.

The first statement of program one is started with the card label AFGEN. This symbol indicates that the numbers following the label define a curve to be used later in the program. This curve is given the name, CURVE1 and is set up with a value of the independent variable, TIME, followed by the associated value of the dependent variable, ICG. A comma separates each of the data points. The curve is read by noting that the initial value of TIME is 0.0 and the associated value of ICG is 20.0. One may note that the data points in this curve are continued on two more cards by the usual three periods. A similar set of statements start with card 4 and define the function for ICG.

Statements in the example program down to statement 12 add nothing new to the information presented in the previous problems. Statements 12 and 13 indicate that the variables ICG and RAD are to be assigned a value for the current value of TIME by linearly interpolating between the points of the curves presented in statements 1 through 6.

The rest of this sample program is made up of previously discussed structure statements. Two exceptions to this generalization, are statements 14 and 16. These two statements indicate that the variable appearing to the left of the equal signs in those statements are to be evaluated in the Fortran subroutines CAL and ROOT.

Program number two sets up another way to carry out the simulation of the pole vaulter. Basically, the statements in this program are the same as in program one. However, this program calls for a Fortran subroutine to read the data for ICG and RAD off of data cards. This subroutine shown in the appendix, is written so that at each time step another card is read and thus determines the current value of the variables. The reader will note the use of a variable in the subroutine by the name of KEEP. This is a CSMP control variable which is set equal to 1 each time the program is to print output. Subroutine CALC tests the value of KEEP at each iteration of the solution and when KEEP equals 1 a new data card is read. These data cards are placed directly ahead of the ENDJOB card at the time the program is processed. Using this technique allows the programmer to change data points by adding a new card with the point in question instead of having to replace the entire data set.

The two programs certainly don't exhaust the ways in which this problem could have been programmed but further discussion would lead to repetition of previously discussed CSMP statements.

For example, ICG and RAD could have been defined in a table data statement. However, the intent of this section on the pole vaulter was merely to point out that there was more than one way to simulating a given problem with CSMP.

CHAPTER 4

CONCLUSION AND RESULTS

Several conclusions about CSMP can be drawn from sample programs used to simulate the problems discussed in Part III of this report. The first conclusion that one may draw is that programming differential equations for solution with CSMP can be done quickly if the programmer has some basic knowledge of the CSMP language. Secondly, there is a wide range of mathematical functions available in the CSMP language to use in modeling problems. Accurate solutions of linear and non-linear initial value problems can be obtained with CSMP with reasonable computer run times. CSMP has shown itself to be a flexible tool for solution of initial value problems with several overlapping avenues of approach to these types of problems.

Also conclusions about using CSMP can be drawn from the experience gained in investigating the sample problems in Part III of the report. A basic knowledge of Fortran programming would seem to be beneficial to a person wanting to use CSMP. Terminology used to describe CSMP features parallels those terms used in Analog computer literature.

Two extensions of this report that may provide useful information can be stated. First, IBM suggests in their literature that CSMP can be used for programming two point boundary problems. This process involves an iterative technique in which final values of a simulation run are compared against boundary values to determine the starting conditions for the next simulation. An investigation in to this procedure should prove informative. Also a comparison of the features of CSMP with other digital simulation languages should be of benefit. This comparison might be along the lines of the complexity of the digital simulation language, features available in the language, input-output procedures, and computer run times for similar problems.

LIST OF REFERENCES

1. Brennan, R. D., and Sano, H. "PACTOLUS - A Digital Analog Simulator Program for the IBM 1620." 1620 General Program Library No. 13. 0.019, International Business Machine Corporation, San Jose, California, September 1964.

2. Mankovitz, R. J. "DIANA: A Digital-Analog Simulation Program for the IBM 1620 II Computer." *JPL Technical Report No. 32-1011*, March 1, 1967.

3. Sansom, F. J., and Peterson, H. E. "Mimic Programming Manual." *Air Force Systems Command Technical Report SE6-TR-67-31*, Wright-Patterson Air Force Base, Ohio, July, 1967.

4. Goering, C. E., Shukla, L. N., and Weathers, B. D. "Is the Analog Computer Obsolete?" *ASAE Paper No. MC-68-105*, April 5, 1968.

5. "System/360 Continuous System Modeling Program (360A-CX-16X) User's Manual." *IBM Publication H20-0367-2*, 1968.

6. "System/360 Continous System Modeling Program (360A-CX-16X) Application Description." *IBM Publication H20-0240-1*, 1967.

7. "System/360 Continuous System Modeling Program (360A-CX-16X) Operator's Manual." *IBM Publication H20-0368-1*, 1967.

8. "System/360 Continuous System Modeling Program (360A-CX-16X) System Manual." IBM Publication Y20-0111-0, 1967.

9. Thomson, W. T. *Vibration Theory and Applications*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1965, p. 139.

10. Kreith, F. *Principles of Heat Transfer*, Scranton, Pennsylvania: International Textbook Company, 1958, pp. 128-129.

11. Bolotin, V. V. *The Dynamic Stability of Elastic Systems*, San Francisco: Holden-Day, Inc., 1964, p. 131.

12. "IBM System/360 FORTRAN IV Language." *IBM Publication C28-6515-4*, 1965.

13. Crandall, S. H. *Engineering Analysis*, New York: McGraw-Hill Book Company, Inc., 1956, pp. 125-135.

14. Timoshenko, S. *Theory of Elastic Stability*, New York: McGraw-Hill Book Company, Inc., 1936, pp. 69-75.

15. *Handbook of Mathematical Functions*, Washington, D.C.: U.S. Government Printing Office, 1966, p. 591.

APPENDIX A

A LIST OF MATHEMATICAL FUNCTIONS AVAILABLE IN CSMP

CSMP NAME	FUNCTION
INTGRL	Integration
DERIV	Differentiation
DELAY	Define a dead time
ZHOLD	Zero-order hold
IMPL	Iterate an implicit function
MODINT	Mode-controlled integrator
REALPL	1st order lag(real pole)
LEDLAG	Lead-lag
CMPXPL	2nd order lag(comples pole)
FCNSW	Function switch
INSW	Input switch (relay)
OUTSW	Output switch
COMPAR	Comparator
RST	Resetable Flip-Flop
AFGEN	Linear interpolation
NLFGEN	Quadratic interpolation
LIMIT	Limiter
QNTZ	Quantizer
DEADSP	Dead Space
HSTRSS	Hysteresis loop
STEP	Step function
RAMP	Ramp function
IMPULS	Impulse generator
PULSE	Pulse generator
SINE	Trigonometric sine wave with delay, amplitude, and phase parameters
GAUSS	Noise generator with normal distribution
RNDGEN	Noise generator with uniform distribution

APPENDIX B

Development of equations for Rotating Shaft Problem

Assume that the shaft rotates with angular velocity ϕ and that $I_y < I_x$.

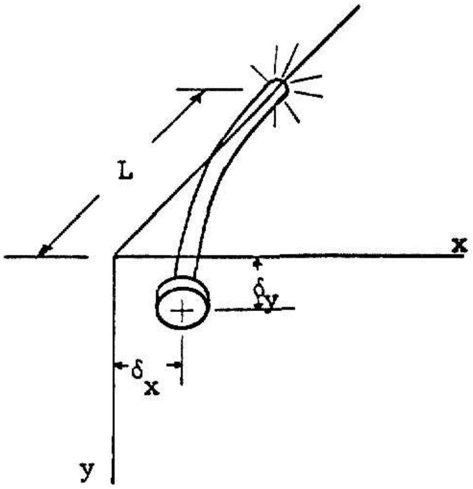


Figure 7a. Relation of Deflections in the x and y Directions to the Fixed Coordinate System.

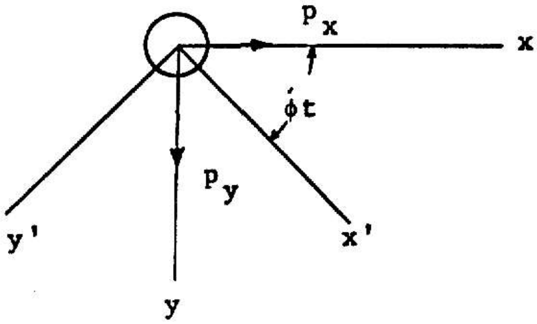


Figure 7b. Relation of Rotating and Fixed Coordinates.

For a cantilever beam

$$\delta = \frac{PL^3}{3EI} = \alpha P \qquad [1]$$

(P is the force in the direction of δ)

Now

$$P_x' = P_x \cos\dot{\phi}t + P_y \sin\dot{\phi}t \qquad [2]$$

$$P_y' = -P_x \sin\dot{\phi}t + P_y \cos\dot{\phi}t \qquad [3]$$

Then

$$\delta_x' = \alpha_1(P_x \cos\dot{\phi}t + P_y \sin\dot{\phi}t) \qquad [4]$$

$$\delta_y' = \alpha_2(-P_x \sin\dot{\phi}t + P_y \cos\dot{\phi}t) \qquad [5]$$

where

$$\alpha_1 = \frac{L^3}{3EI_y'} \quad ; \quad \alpha_2 = \frac{L^3}{3EI_x'}$$

Also

$$\delta_x = \delta_x' \cos\dot{\phi}t - \delta_y' \sin\dot{\phi}t \qquad [6]$$

$$\delta_y = \delta_x' \sin\dot{\phi}t + \delta_y' \cos\dot{\phi}t \qquad [7]$$

$$\delta_x = \alpha_1(P_x\cos\dot{\phi}t + P_y\sin\dot{\phi}t)\cos\dot{\phi}t - \alpha_2(-P_x\sin\dot{\phi}t + P_y\sin\dot{\phi}t)\sin\dot{\phi}t \qquad [8]$$

$$\delta_y = \alpha_1(P_x\cos\dot{\phi}t + P_y\sin\dot{\phi}t)\sin\dot{\phi}t + \alpha_2(-P_x\sin\dot{\phi}t + P_y\cos\dot{\phi}t)\cos\dot{\phi}t \qquad [9]$$

which reduces to,

$$\delta_x = \frac{(\alpha_1+\alpha_2)P_x}{2} + \frac{(\alpha_1-\alpha_2)P_x}{2} \cos 2\dot{\phi}t + \left(\frac{\alpha_1-\alpha_2}{2}\right)P_y \sin 2\dot{\phi}t \qquad [10]$$

$$\delta_y = \left(\frac{\alpha_1+\alpha_2}{2}\right)P_y - \frac{(\alpha_1-\alpha_2)}{2}P_y \cos 2\dot{\phi}t + \frac{(\alpha_1-\alpha_2)}{2}P_x \sin 2\dot{\phi}t \qquad [11]$$

From Newton's law

$$P_x = m\ddot{\delta}_x \quad ; \quad P_y = -m\ddot{\delta}_y \qquad [12]$$

substituting

$$m[\frac{\alpha_1+\alpha_2}{2} + \frac{\alpha_1-\alpha_2}{2}\cos2\dot{\phi}t]\ddot{\delta}_x + \delta_x + m\frac{\alpha_1-\alpha_2}{2}\sin2\dot{\phi}t\ \ddot{\delta}_y = 0 \qquad [13]$$

$$m[\frac{\alpha_1+\alpha_2}{2} - \frac{\alpha_1-\alpha_2}{2}\cos2\dot{\phi}t]\ddot{\delta}_y + \delta_y + m\frac{\alpha_1-\alpha_2}{2}\sin2\dot{\phi}t\ \ddot{\delta}_x = 0 \qquad [14]$$

For convenience let

$$m(\frac{\alpha_1+\alpha_2}{2}) = a \quad \text{and} \quad m(\frac{\alpha_2-\alpha_1}{2}) = b$$

Then

$$(a-b\ \cos2\dot{\phi}t)\ddot{\delta}_x - (b\sin2\dot{\phi}t)\ddot{\delta}_y + \delta_x = 0 \qquad [15]$$

$$(-b\sin2\dot{\phi}t)\ddot{\delta}_x + (a+b\ \cos2\dot{\phi}t)\ddot{\delta}_y + \delta_y = 0 \qquad [16]$$

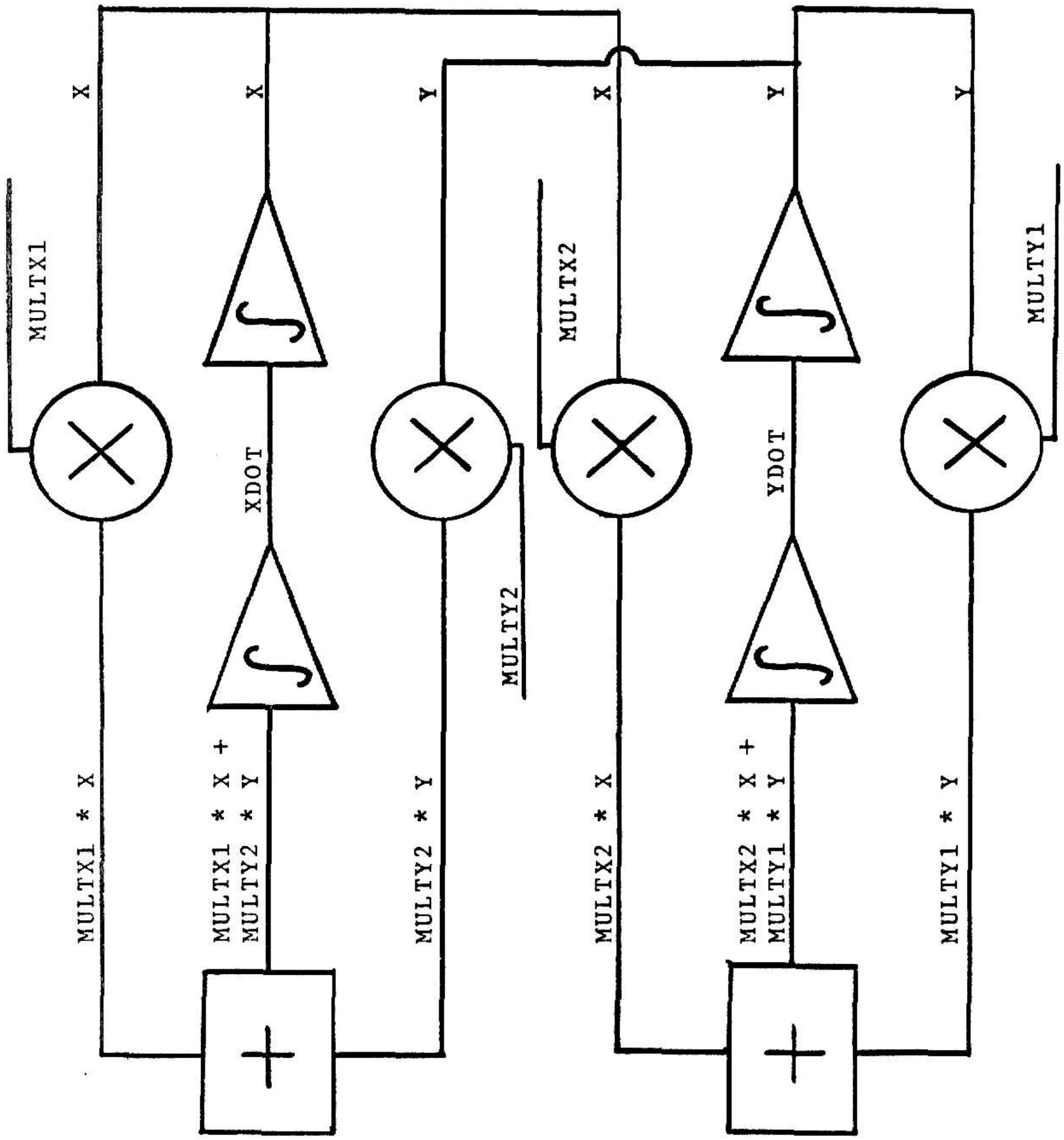
Now by appropriate algebraic manipulation, one may transform equations 15 and 16 into the form which is required for CSMP programing.

$$\ddot{\delta}_x = \frac{1}{b^2-a^2}\left((a+b\ \cos2\dot{\phi}t)\delta_x + (b\sin2\dot{\phi}t)\delta_y\right) \qquad [17]$$

$$\ddot{\delta}_y = \frac{1}{b^2-a^2}\left((b\sin 2\dot{\phi}t)\delta_x + (a-b\cos 2\dot{\phi}t)\delta_y\right) \qquad [18]$$

APPENDIX C

Block Diagram
For Shaft Simulation



APPENDIX D

Outline of Developement of the Equations Used in Analyzing the Pole Vaulter

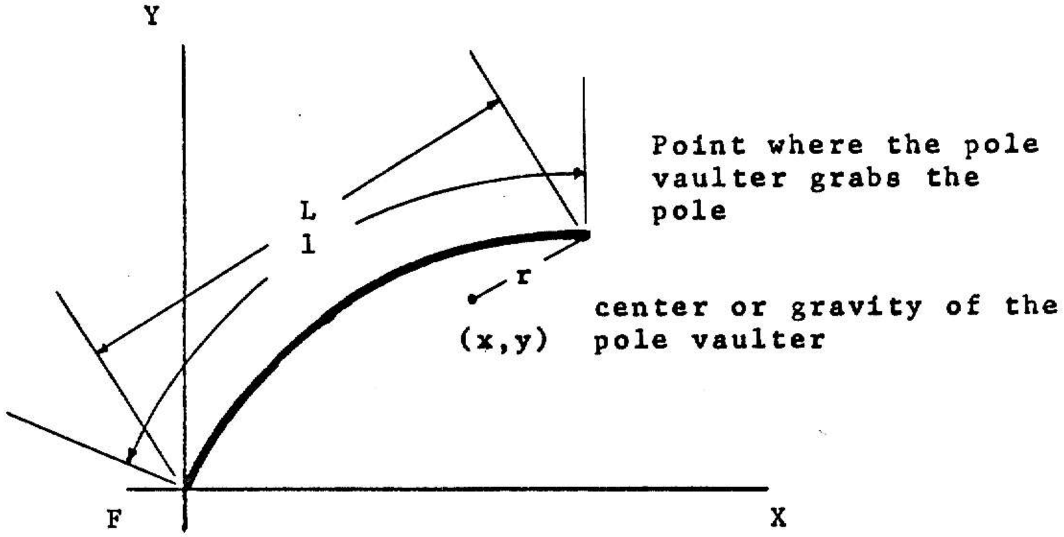

Y
L
1
Point where the pole vaulter grabs the pole
r
(x,y)
center or gravity of the pole vaulter
F
X

Figure 8. Schematic Representation of Pole Vaulter Problem.

The moment of inertia (I) of the pole vaulter about his center of gravity and the distance from where he grabs the pole to his center gravity (r) can be given either as functions of position or time. Applying Newton's second law to the vaulter:

$$\frac{w}{g}\ddot{y} = F_y(x,y,\theta) - w \qquad [1]$$

$$\frac{w}{g}\ddot{x} = F_x(x,y,\theta) \qquad [2]$$

$$I\ddot{\theta} + \dot{\theta}\dot{I} = F_y\ r\cos\theta - F_x\ r\sin\theta \qquad [3]$$

F_x, F_y are the x and y components of the force exerted on the vaulter by the pole.

The straight line distance from one end of the pole to the other is derived by using the theorem of Pythagoreous.

$$L = \sqrt{(x+r\cos\theta)^2 + (y+r\sin\theta)^2}$$

$$= \sqrt{x^2 + y^2 + r^2 + 2xr\cos\theta + 2yr\sin\theta} \qquad [4]$$

From the theory of the Elastica (14) for a free-pinned beam the following expressions for the distance between ends of the pole and the force in the pole were used.

$$L = \frac{2}{\sqrt{F/\beta}} \int_0^{\pi/2} \frac{1-2k^2\sin^2\phi}{\sqrt{1-k^2\sin^2\phi}}\, db = \frac{2}{\sqrt{F/\beta}} \left[2\int_0^{\pi/2} \sqrt{1-k^2\sin^2\phi}\; d\phi - \int_0^{\pi/2} \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}} \right] \qquad [5]$$

where $k = \sin\alpha/2$ and α is the initial angular displacement

$$F = \beta \left[\frac{2}{L_c} \int_0^{\pi/2} \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}} \right]^2 \qquad [6]$$

where L_c is the poles length.

These elliptical integrals were converted to polynomial approximations for numeric evaluation using the following relationships from the Handbook of Mathematical Functions (15).

$$K(k^2) = \int_0^{\pi/2} \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}} = \frac{\pi}{2}\left(1+(\tfrac{1}{2})^2k^2+(\tfrac{1.3}{2.4})^2k^4+(\tfrac{1.3.5}{2.4.6})^2k^6+\ldots\right) \quad [7]$$

$$E(k^2) = \int_0^{\pi/2} \sqrt{1-k^2\sin^2\phi}\; d\phi = \frac{\pi}{2}\left(1-(\tfrac{1}{2})^2k^2-(\tfrac{1.3}{2.4})^2k^4-(\tfrac{1.3.5}{2.4.6})^2k^6-\ldots\right) \quad [8]$$

When these relationships were combined with equations 5 and 6, the following resulted.

$$L = \frac{2}{\sqrt{F/\beta}}\,[2\ E(k^2) - K(k^2)] \quad [9]$$

$$F = \frac{4\beta}{L_c^{\,2}}\ K^2(k^2) \quad [10]$$

The x and y components of the force were:

$$F_x = F\left(\frac{x+r\cos\theta}{L}\right) \quad [11]$$

$$F_y = F\left(\frac{y+r\sin\theta}{L}\right) \quad [12]$$

These relationships then yielded sufficient information to carry out the simulation.

APPENDIX E

FORTRAN SUBROUTINE CAL USED WITH PROGRAM 1 IN THE POLE VAULTER ANALYSIS

```
  SUBROUTINE CAL(ICG,RAD,O,Y,X,M,DELT,TIME,I,IDOT,C1,ETA,DEM1)
  REAL ICG,ICUR,ILAST,I,IDOT,M
  ICUR=ICG+M*RAD*RAD
  IF(TIME) 2,2,3
2 ILAST=ICUR
  GO TO 4
3 ILAST=SAVE
4 CONTINUE
  I=ICUR
  IDOT=(ICUR-ILAST)/DELT
  C1=RAD*COS(O)+X
  ETA=RAD*SIN(O)+Y
  DEM1=X*X+Y*Y+RAD*RAD+2.*X*RAD*COS(O)+2.*Y*RAD*SIN(O)
  SAVE=ICUR
  RETURN
  END
```

APPENDIX F

FORTRAN SUBROUTINE CALC USED WITH PROGRAM 2 IN THE POLE VAULTER ANALYSIS

```
      SUBROUTINE CALC(O,Y,X,M,I,IDOT,C1,ETA,DEM1,ICG,RAD)
COMMON   MEM
      REAL ICG,ICUR,ILAST,I,IDOT,M
      EQUIVALENCE (C(1),TIME)
    1 FORMAT(2F12.8)
      IF(TIME) 23,23,21
   21 IF(KEEP-1) 22,23,22
   22 RETURN
   23 CONTINUE
      READ(1,1) ICG,RAD
C
C             DATA FOR 0.0 MUST BE ENTERED THREE TIMES
C             THAT IS THREE CARDS WITH THE SAME DATA MUST
C             COME AT THE BEGINNING OF THE DATA
C
      ICUR=ICG+M*RAD*RAD
      IF(TIME) 2,2,3
    2 ILAST=ICUR
      GO TO 4
    3 ILAST=SAVE
    4 CONTINUE
      I=ICUR
      IDOT=(ICUR-ILAST)/.05
      C1=RAD*COS(O)+X
      ETA=RAD*SIN(O)+Y
      DEM=X*X+Y*Y+RAD*RAD+2.*X*RAD*COS(O)
      DEM1=DEM+2.*Y*RAD*SIN(O)
      SAVE=ICUR
      RETURN
      END
```

APPENDIX G

FORTRAN SUBROUTINE USED IN POLE VAULTER ANALYSIS

```
      FUNCTION ROOT(L,LL)
      REAL KK,L,LL,KZERO,KLAST
      DEL=.1
      KZERO=.1
      KLAST=.00001
   98 CONTINUE
      TEST=KZERO-KLAST
      IF(TEST-.001) 100,100,1
    1 CONTINUE
      CALL FUN(KLAST,LL,L,H,KK)
      IF(H-.001) 99,99,2
    2 CONTINUE
      KLAST=KZERO
      KZERO=KLAST+DEL
      GO TO 98
   99 DEL=DEL/4.0
      KZERO=KLAST+DEL
      GO TO 98
  100 ROOT=KK
      RETURN
      END
```

APPENDIX H

FORTRAN SUBROUTINE USED IN POLE VAULTER ANALYSIS

```
      SUBROUTINE FUN(K,LL,L,H,KK)
      REAL KSQ,K,LL,L,KK
      SUMK=1.
      SUME=1.
      SAVEK=0.
      SAVEE=0.
      XK=1.
      KSQ=K*K
      C=0.
      A=1.
      B=1.
  100 C=C+1.
      E=C
      A=A*C
      C=C+1.
      B=B*C
      D=A/B
      D=D*D
      F=D/E
      XK=XK*KSQ
      SUMK=SUMK+D*XK
      SUME=SUME-F*XK
      T=ABS(SUMK-SAVEK)
      U=ABS(SUME-SAVEE)
      IF(T-.00001) 101,101,1
    1 CONTINUE
      IF(U-.00001) 101,101,2
    2 CONTINUE
      H=LL*(2.*(SUME/SUMK)-1.)-L
      KK=SUMK
      RETURN
  101 SAVEK=SUMK
      SAVEE=SUME
      GO TO 100
  102 RETURN
      END
```

VITA

Rodney T. Nash

Candidate for the Degree of

Master of Science

Report: INVESTIGATIONS OF SELECTED INITIAL VALUE PROBLEMS USING A DIGITAL SIMULATION LANGUAGE

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born at Pratt, Kansas, May 11, 1945.

Education: Graduated from Greensburg Rural High School, Greensburg, Kansas, in 1963; received the Bachelor of Science degree from Kansas State University, with a major in Mechanical Engineering, in August, 1967; completed requirements for the Master of Science degree in January, 1969.

Professional experience: Employed as a summer engineer by Texaco, Inc., in the summer of 1966; taught in the Mechanical Engineering Department of Kansas State University as a full-time instructor in the fall of 1968.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Dr. H. S. Walker of the Department of Mechanical Engineering, Kansas State University, for his guidance and help throughout the course of this work as his advisor.

The author is also grateful to Dr. P. E. McNall, Head, Department of Mechanical Engineering, Kansas State University, for his confidence shown by providing employment for the author while the work was being completed.

INVESTIGATIONS OF SELECTED INITIAL VALUE PROBLEMS USING A DIGITAL SIMULATION LANGUAGE

by

RODNEY T. NASH

B. S., Kansas State University, 1967

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1969

Name: Rodney T. Nash   Date of Degree: Jan., 1969

Institution: Kansas State University   Location: Manhattan, Kansas

Title of Study: INVESTIGATIONS OF SELECTED INITIAL VALUE PROBLEMS USING A DIGITAL SIMULATION LANGUAGE

Pages in Study: Candidate for Degree of Master of Science

Major Field: Mechanical Engineering

Scope and Method of Study: Digital simulation languages are specialized digital computer software which facilitates rapid programming of initial value problems. This type of digital computer program is defined in much the same terms as analog computer equipment thus causing the digital computer to solve many of the initial value problems previously relegated to analog computers. There are several different digital simulation languages in use today. This study deals with a simulation language developed by International Business Machines called Continuous System Modeling Program (CSMP).

The study was carried out by solving four sample problems with CSMP. These example problems were: non-linear simple pendulum; cooling of a uniform temperature body; stability analysis of a whirling shaft; and simulation of a pole vaulter. A Program for each of these sample problems was written and run on an IBM system 360/50 computer. Where available numerical results were compared to analytical results. Several of the various features of CSMP were employed in the sample programs. Two of the problems were

programmed in different ways to check the versitility of CSMP.

Findings and Conclusions: The results of these sample problems indicated that CSMP provides a rapid method of programming initial value problems for digital solution. In the estimation of the author, a basic knowledge of Fortran programming is necessary for efficient use of CSMP. CSMP provides a wide range of mathematical functions necessary to simulation. Accurate answers are readily attainable. CSMP is quite flexible, that is it allows several avenues to be persued in solving initial value problems CSMP terminology closely parallels that used in Analog computer literature thus allowing those familiar with Analogs to adapt rapidly to CSMP.