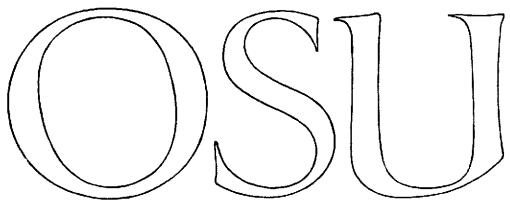
cc-69-5

MIMIC
A Digital-Analog Simulator

April, 1969



COMPUTER CENTER
Oregon State University
Corvallis, Oregon 97331

MIMIC

A Digital-Analog Simulator

cc-69-5

Computer Center
Oregon State University
Corvallis, Oregon 97331

April, 1969

TABLE OF CONTENTS

I. INTRODUCTION

MIMIC is a programming system written for a digital computer which, from the standpoint of the user, seems to make the sequential machine function like the parallel analog computer and at the same time virtually eliminates the problems of time and amplitude scaling. It does this by providing a language which is based on the idea of interconnecting basic computing elements such as adders, integrators, switches, etc., as in the wiring of a patch board for a real analog computer. In this respect MIMIC is like MIDAS and PACTOLUS but has a larger set of computing elements, is faster, more efficient and easier to use for it allows FORTRAN-like expressions, MIMIC language sub-programs and logical control of program execution.

This manual presupposes a limited acquaintance with analog techniques on the part of the reader.

II. DEFINITIONS

The following definitions will be used throughout this manual.

1. Variable Name - A group of 1 - 6 letters or digits of which one must be a letter.

   e.g. AB, IB2, 24C2

   The following symbols have special meanings to the MIMIC processor:

   T, DT, DTMAX, DTMIN, TRUE, FALSE.

2. Literals - There are two types of literals:
   (a) Numeric constants which must conform to the rules:
      (i) a decimal point
      (ii) at least one digit
      (iii) at most six characters

      e.g. 47., 63.0, 4.5278.

   (b) Logical constants TRUE, FALSE.

3. Function - A three letter mnemonic code for a MIMIC computing element. All MIMIC computing elements are specified in Table 1, Section III.

   e.g. LOG, ADD.

4. Expression - A MIMIC expression may be:
   (i) a literal
   or (ii) a variable name
   or (iii) a MIMIC function with its arguments enclosed in parentheses and separated by commas.
   or (iv) any algebraically meaningful combination of these formed using the arithmetic operators + - */** and parentheses.

NOTE: Multiplication cannot be implied,

e.g. A*(B + C)

is legal but A(B + C) is not.

e.g. 28.0

-IBX

LOG(X,-2.)

1BX+28.0*SIN(T*(THETA1-THETA2)/360.)

NOTE: The arguments of MIMIC functions may be expressions.

5. Statement - A MIMIC statement consists of two things: an expression and a variable name. The effect of the statement is to assign the result of evaluating the expression to the variable named.

   e.g. X ADD(Y,Z,-4.,T*2.0+NEG(1BX))

   The result of carrying out the operations specified by the expression will become the value of the result variable named X.

   Exceptions: CON, PAR, RSP, OUT, HDR, FIN, END statements do not require a result variable.

6. Program - A MIMIC program consists of a set of MIMIC statements, of which at least one must be a FIN statement, followed by an END statement.

7. Job - A MIMIC job consists of a MIMIC program punched on cards and possibly followed by data cards.

8. Processor - The MIMIC processor is a set of routines written in FORTRAN-IV and COMPASS for the 3300 which assembles and executes MIMIC programs.

9. Special Variables - The MIMIC processor reserves the following symbols for the special purposes listed:

   T - the independent variable (considered possibly as time)

   DT - the amount T changes between printouts of values

   DTMAX - maximum integration stepsize allowed

   DTMIN - minimum integration stepsize allowed

10. Program Run - The generation of the solution (to a problem programmed in MIMIC) from T = 0 until termination by a FIN statement will be called a run.

    The processor performs successive runs until it exhausts the supply of data cards provided for all PAR or PFN statements in the program.

III. TABLE OF MIMIC COMPUTING ELEMENTS

TABLE 1. LIST OF FUNCTIONS

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
1. ARITHMETIC FUNCTIONS			
Addition	ADD SUM	A,B(,C,D,E,F)*	R = A+B(+C+D+E+F)
Subtraction	SUB	A,B	R = A-B
Multiplication	MPY	A,B(,C,D,E,F)	R = A*B(*C*D*E*F)
Divide	DIV	A,B	R = A/B
Multiply and Add	MAD	A,B,C(,D,E,F)	R = A*B+C(*D+E*F)
Negation	NEG	A	R = -A
Absolute Value	ABS	A	R = |A|
Equality	EQL	A	R = A
2. ELEMENTARY TRANSCENDENTALS			
Square Root	SQR	A(>0)	R = $\sqrt{A}$
Sine	SIN	A(rads.)	R = sin A
Cosine	COS	A(rads.)	R = cos A
Arctangent	ATN	A(,B)	R = $\tan^{-1}$(A/B). If B is not specified it is assumed to be +1.
Exponential	EXP	A(,B)	R = B^A. If B is not specified B = e is assumed.
Logarithm	LOG	A(,B)	R = $\log_B A$. If B is not specified B = e is assumed.

* Operands enclosed in parentheses need not be specified.
1. B,C,D may be any logical-valued expression

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
3. LOGICAL FUNCTIONS			
Function Switch	FSW	A,B,C,D[1]	R = B A < 0 = C A = 0 = D A > 0
Logical Switch	LSW	A,B,C[1]	R = B if A TRUE = C if A FALSE
And	AND	A,B(,C,D,E,F)[1]	R = TRUE if A and B (and C and D and E and F) have value TRUE. FALSE, otherwise
Exclusive Or	EOR	A,B[1]	R = TRUE if A and B are different. FALSE if A and B are the same.
Inclusive Or	IOR	A,B(,C,D,E,F)[1]	R = TRUE if A or B (or C or D or E or F) have value TRUE. Otherwise R = FALSE.
Complement	COM NOT	A[1]	R = FALSE if A = TRUE. TRUE if A = FALSE.
4. INPUT/OUTPUT FUNCTIONS			
Name Constants	CON	A(,B,C,D,E,F)[2]	Enters constant names
Name Parameters	PAR	A(,B,C,D,E,F)[2]	Enters parameter names

1. A,B,C,D,E,F must be logical-valued expressions
2. Arguments must be variable names
3. Argument must be a numeric literal
4. SW is a control constant. If SW = 0., the data points will be plotted. If SW = 1., straight lines will be drawn between points. If SW=2., a quadratic will be fitted to the points. SCL is the multiplier to expand the scale in the X direction.

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
Name Function (Constant)	CFN	A[3]	The name of the array is entered in the result column. A = number of pairs or triples of points.
Name Function (Parameter)	PFN	A[3]	The name of the array is entered in the result column. A = number of pairs or triples of points.
Print Output	OUT PRI	(A,B,C,D,E,F)	Print A,B,C,D,E, F every DT units of T.
Print Headers	HDR HEA	(A,B,C,D,E,F)	Print heading names given in A,B,C,D,E,F
Plot[4]	PLO	SW,SCL,A(,B,C,D)	Supply A,B,C,D to subroutine PLO every DT units of T.
5. SUBPROGRAMS			
Begin Subprogram End Subprogram	BSP ESP	A(,B,C,D,E,F) A(,B,C,D,E,F)	The subprogram name appears in the result column of the BSP and ESP cards. The inputs are named on the BSP card and the outputs on the ESP card.
Call Subprogram Return Subprogram	CSP RSP	A(,B,C,D,E,F) A(,B,C,D,E,F)	The name of the called subprogram is given in the result field of the CSP card. The RSP card must immediately follow the CSP card.

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
6. SPECIAL FUNCTIONS			
Integration	INT	A,B(,C,D)	$R = B + \int A dT$ C,D must be logical valued expressions.
Limit Integrators	LIN	A,B,C,D	$R = 0 \quad B < C$ $= A \quad C \leq B \leq D$ $= 0 \quad B > D$
First Order Transfer Function	FTR	A,B	$R = L^{-1}[A(s)/(Bs+1)]$ where A is the variable operated on by $1/(Bs+1)$
Limiter	LIM	A,B,C	$R = B \quad A < B$ $= A \quad B \leq A \leq C$ $= C \quad A > C$
Dead Space	DSP	A,B,C	$R = A - B \quad A < B$ $= 0 \quad B \leq A \leq C$ $= A - C \quad A > C$
Time Delay	TDL	A,B(,C)	R = A(T-B). C is the number of points of A to be stored and must be a literal or constant. If C is not specified, C = 100 is assumed. B may be variable. If $T \leq B$, R = A(0).

D \ C	TRUE	FALSE
TRUE	OPERATE	HOLD
FALSE	RESET	OPERATE

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
Function	FUN	A,B A,B,C	R = A(B) R = A(B,C)
Implicit Function	IMP	A,B	R = A where $\|A-B(A)\| \leq 5\times10^{-6}\|A\|$
Maximum	MAX	A,B(,C,D,E,F)	R = max(A,B(,C,D,E,F))
Minimum	MIN	A,B(,C,D,E,F)	R = min(A,B(,C,D,E,F))
Random No. Generator[5]	RNG	A,B,C[6]	R = random sample from a gaussian distribution with mean = A and standard deviation B. C is a starting number.
Random No. Generator[5]	RNU	A,B,C[6]	R = random sample from a uniform distribution with lower limit A and upper limit B. C is a starting number.
Derivative	DER	A,B,C	R = dA/dB. R = C at T = 0.
7. HYBRID FUNCTIONS			
Monostable Multivibrator	MMV	A,B	R set TRUE when A TRUE and stays TRUE for B units of T after A goes FALSE. R = A at T = 0.
Track and Store	TAS	A,B,C	R = A when B TRUE = R_p when B FALSE. (R_p = previous R) R = C at T = 0.

FUNCTION	CODE	INPUT(S)	VALUE/REMARKS
Flip-Flop	FLF	A,B,C[7]	R^{8} = { TRUE if A = TRUE. TRUE if B = FALSE and R_p = TRUE. FALSE otherwise. R = C at T = 0.
Zero Order Hold	ZOH	A,B	A is sampled every B units of T. R = held sampled value. R = A at T = 0.
8. CONTROL FUNCTIONS			
Go to Next Case	FIN	A,B	Go to next run when A ≥ B.
End of Program	END		Signifies end of MIMIC program and beginning of MIMIC data.

5. Different starting numbers will produce independent sequences, hence many random number generators may be used in the same program.
6. C should be a literal, constant or parameter.
7. A,B,C must be logical valued expressions.
8. $R = A \vee (B' \wedge R_p)$.

IV. EXAMPLE

The method of forming expressions and statements defined in section II means that, unlike MIDAS and PACTOLUS, only those variables of particular interest need be named in the result portion of a MIMIC statement. Thus a MIMIC program could be written down from the analytic form of the system being solved without the necessity of a block diagram.

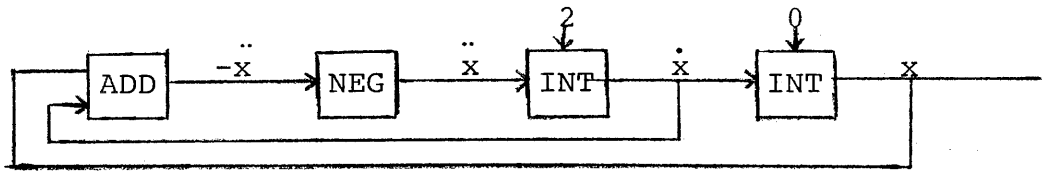
Suppose we wish to solve the problem

$$\ddot{x} + \dot{x} + x = 0 \quad \text{with } \dot{x}(0) = 2, \quad x(0) = 0$$

A block (circuit) diagram representation of this problem is obtained by writing the highest derivative in terms of lower order derivatives,

$$\text{eg.} \quad \ddot{x} = -\dot{x} - x$$

Thus we have



A MIMIC program which explicitly reflects the interconnection of each basic computing element is:

```
NEGDX2      ADD(X,DX1)
DX2         NEG(NEGDX2)
DX1         INT(DX2,2.)
X           INT(DX1,0.)
            OUT(X,DX1)
            FIN(T,2.5)
            END
```

The OUT and FIN statements cause the processor to print out a table of values of x, $\dot{x}$ for values of T = 0, .1, .2,..., 2.5.

A simpler way of writing this program is

```
DX1     INT(-(X+DX1),2.)
X       INT(DX1,0.)
        OUT(X,DX1)
        FIN(T,2.5)
        END
```

V. ARBITRARY FUNCTIONS

MIMIC allows arbitrary functions of 1 or 2 independent variables to be specified by tabulated data, i.e. by an array of values of independent variable(s) and corresponding values of the dependent variable.

These functions are declared as such by including a CFN or PFN statement in the program for each function generator desired and are used in calculations by means of a FUN statement.

For example the statement

```
F     CFN(5.)
```

has the effect of defining a function F to be specified by 5 sets of points which will be punched on data cards. Then the function F could be used, for example in the following statement

```
X     FUN(F,Z)
```

i.e. the value of X will become F(Z).

Similarly

```
X     FUN(F,U,V)
```

if F were a function of 2 independent variables.

The processor uses linear interpolation for values of the independent variable(s) between tabulated points. The function is assumed continuous and to have derivatives equal to zero for all points outside the tabulated range.

The number of data points as well as the number of arbitrary functions is limited only by the computer memory available for programs.

VI. NUMERICAL DATA FOR MIMIC PROGRAMS

Numerical values of variable names a programmer may have used to designate constant coefficients, initial conditions, arbitrary functions, etc., are supplied to the processor on data cards following the END statement of the program.

There are 4 types of data used in MIMIC programs and information concerning each type must appear in 2 places in a job. First, the name of the data must appear in a CON, PAR, CFN, or PFN statement and the corresponding value must appear on a data card.

Values for data declared by a CON (Constant) statement stay fixed for all runs. One data card must appear in the data section for each CON statement in the program section.

Values for data declared by a PAR (parameter) statement may change from run to run. One data card must appear for each PAR statement in the program section for each run desired.

Up to 6 variable names may be used as arguments of each PAR or CON statement.

Data for arbitrary functions of 1 (or 2) independent variable(s) declared by CFN (constant function) or PFN (parameter function) statements are punched as 1 pair (or triple) of points per data card in the following format:

- functions of 1 variable; independent variable punched in columns 1-12, dependent variable punched in columns 13-24 and the data cards ordered in increasing value of the independent variable.

- functions of 2 variables; 1st independent variable punched in columns 1-12, 2nd indpendent variable punched in columns 13-24, dependent variable punched in columns 25-36. The cards are arranged so that the values of the 1st independent variable form a non-decreasing sequence and at each value of the 1st independent variable, the values of the 2nd independent variable must form an algebraically increasing sequence.

  In addition the array must consist of at least 2 values of the second variable for each value of the first and the array must contain points less than and greater than the expected values of both independent variables.

The number of data cards for both is specified as the argument of the CFN or PFN statement.

Finally, any CON and CFN statements must come before any PAR and PFN statements and these cards must be the first cards in a program. The order of the data cards corresponding to each statement of each type is the same as the appearance of the statements in the program.

For example, if

```
      CON(A,B)
F     CFN(3.)
G     PFN(4.)
      PAR(X)
```

are statements in a program, data cards might be

```
        12          24          36
|-------------|-----------|-----------|
|    -4.      |   10.     |           |   card 1:  constant values of A,B
|    -1.0     |   -2.     |           |        2)  specify constant
|     0.      |    0.     |           |        3)  function F of 1
|     1.      |    2.     |           |        4)  variable
|$MIMC,DCH3   |           |           |
|  2          |           |           |
|     2.      |   -2.     |   17.     |        5)
|     2.      |    5.     |   10.8    |        6)  specify function G of
|             |           |           |
```

3.	-2.	25.0	7	2 variables for 1st run
4.	6.5	100.63	8	
-25.36			9	value of X for 1st run
1.5	-3.0	25.0	10	
2.5	-2.0	14.0	11	function G for
2.5	4.0	73.0	12	2nd run
3.	0.0	28.5	13	
-50.2			14	value of X for 2nd run

NOTE: If *MIMIC load option is used, it must follow constant function followed by number of parameter sets, in this case 2, in I4 format columns 1 - 4.

VII. OUTPUT

1. Description of Printed Output

Values of up to 6 variables may be printed by means of an OUT statement every DT units of T from T = 0 until a FIN statement terminates the run. More than one OUT statement is allowed per program.

e.g. OUT(A,X) will print columns of values of A and X.

A value for DT may be

(i) read in by a CON or PAR card

(ii) calculated by operations in a MIMIC program

(iii) assumed to be .1 if not specified by (i) or (ii).

Blank columns can be introduced by using a comma with no preceding argument. For example OUT(A,,X).

Blank lines may be inserted in the output by an OUT statement with no arguments. That is OUT ONLY.

Up to 6 column headings to appear at the top of the output may be specified by HDR statements. The actual characters to be printed are the arguments of HDR and can consist of 1-6 alphabetic or numeric characters. Blank column headings can

be obtained by using a comma with no preceding argument and blank lines in headings can be obtained by using HDR alone.

In addition the processor prints out the names and values of any data it reads in for each run.

1. Description of Plotted Output

The plotter will supply a plotted output with the independent variable as the abscissa and the function as the ordinate.

Up to four functions can be plotted simultaneously from the data points which are stored in the plotting subroutine. 300 data points can be stored for each function; however, if this number is exceeded a diagnostic will result.

The Y axis has a length of 20.0 inches, therefore, the user must scale his output to fit this. A mean value of 10.0 is desirable.

The X axis is 30.0 inches long and it is possible to scale the X-array by a multiplier supplied by the user. The time scale is normally marked at 1 second per inch.

The user has the choice of plotting:-

1. Data points only---------- PLO(0.,SCL,--)
2. Straight lines between points--------- PLO(1.,SCL,--)
3. A quadratic curve fit between points-- PLO(2.,SCL,--)

Care should be taken using the third option if discontinuities are expected.

The second parameter in the call to plot, SCL, is the multiplier used to expand the graph in the X direction and is normally 1. The annotations of the X axis are modified accordingly. It is possible to reduce the scale of the whole graph. This is sometimes useful for photographic purposes. The decimal fraction of the first parameter is used to divide all dimensions in the plot.

Thus PLO(2.2,-----) will give a quadratic curve fit with the graph reduced by 2. If the decimal fraction is blank or zero, the graph is of normal size (20" x 30").

VIII. SOME SPECIAL MIMIC FEATURES

1. Logical Control Variables

In addition to variables which can take on numeric values in the limits $|1.7| \times 10^{\pm 38}$, MIMIC allows logical variables which can take on only the values TRUE or FALSE. These variables can be used to provide control over the evaluation of expressions.

Logical variables may be assigned values by the use of element FWS (function switch) and the value can be changed by any of the logic functions (see parts 3, 7 of Table 1).

If the name of a logical variable is entered in the LCV field of a MIMIC statement, the expression on that card will be evaluated only when that logical variable has the value TRUE. That is, logical control variables may be used to select, under programmed control, which of several paths should be evaluated.

For example, define a logical variable which is TRUE only at T = 0 and use it to control precomputation of constants, initial conditions, etc. (see example 2, Appendix I)

NOTE: If a logical control variable is used on statements containing expressions involving INT or FTR, these functions will not be initialized unless the variable is TRUE at T = 0.

2. Integrator Mode Control

The mode of each integrator in MIMIC can be individually set to one of reset, operate or hold. In reset mode, the output of the integrator is set equal to the initial condition argument and no integration takes place. In operate, the integrator is updated by the integration routine. In hold, the input to the integrator is removed.

Two extra arguments which must have logical values are used with INT to control the mode. If these are not present, the mode is assumed to be "operate".

For example

```
X    INT(Y,XO,C,D)
```

is a completely specified integrator. Logical variables C,D control the mode of the integrator by the following scheme:-

D \ C	TRUE	FALSE
TRUE	OPERATE	HOLD
FALSE	RESET	OPERATE

3. Integrator Limiting

MIMIC contains a function LIM to limit between some lower and upper bounds the values that a variable may assume. However, if the variable to be limited is the result of an integration, it is necessary to set the derivative of the variable to zero whenever the variable is equal to a limit value by use of function LIN before using it as input to an integrator.

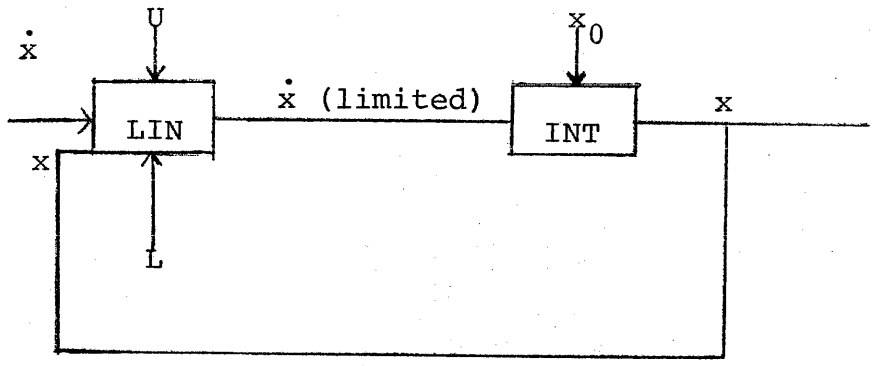
For example, if x is the output of an integrator but is to be limited such that $L \leq x \leq U$, MIMIC statements might be

```
XDOTLM    LIN(XDOT,X,L,U)
X         INT(XDOTLM,XO)
```

or simply

```
X         INT(LIN(XDOT,X,L,U),XO)
```

A block diagram for this is



4. Implicit Functions

MIMIC contains an element IMP for solving equations of the form x = f(x). An iterative method is used and thus a starting value must be given by a PAR statement (or assigned in a calculation performed before any integration starts).

For example

```
      PAR(X)
        :
X     IMP(X,SIN(X)+Y)
        :
```

would solve x = sin x + y where y is specified elsewhere in the program.

Some computational efficiency can be obtained with implicit functions by using dummy variables since the processor function-sorting routine will insure that all computations involving the result of an IMP will be placed after the IMP.

eg.

```
      PAR(X1)
        :
X     IMP(X1,SIN(X1)+Y)
        :
```

NOTE: $x = f(x)$ is solved by the scheme

$$x_0 = \text{supplied value}$$

$$x_{n+1} = \frac{f(x_n) - c_n x_n}{1 - c_n}$$

$$c_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

$$\text{until } \left| \frac{x_n - f(x_n)}{x_n} \right| \leq .5 \times 10^{-5}$$

5. Time Delay

The expression TDL(A,B,C) has the effect of delaying the variable A by B units of T. Here, B may be any expression; C, if specified must be a literal or constant, and represents the number of values of the input variable A stored at any one time for providing the delay. If C is not specified, it is assumed to be 100.

If $T \leq B$, the result of TDL is the value of A at T = 0.

6. MIMIC Subprograms

MIMIC language subprograms present a way of extending the list of available functions for a given program. In particular, it allows functions with multiple outputs to be generated.

A subprogram is defined as those statements occurring between BSP (begin subprogram) card and an ESP (end subprogram) card. The subprogram name is entered in the result field of the BSP and ESP cards. The dummy inputs to the subprogram are specified as arguments of the BSP element and the dummy outputs as arguments of the ESP element. Up to 6

inputs and outputs may be used and must be single variable names.

A subprogram is used by entering its name in the result field of a CSP (call subprogram) card with the actual inputs listed as arguments of the CSP element and the actual outputs as arguments of the RSP element. Actual outputs must be single variable names but actual inputs may be expressions.

The following rules apply in writing subprograms:

(i) A dummy variable must be defined as an input or a result before it can be used as an argument.

(ii) The names used in a subprogram must not be used elsewhere in the program.

(iii) The RSP card must be the card immediately after the CSP card.

For example, we define a subprogram SAMPLE to evaluate

$$x_1 = x \cos z + y \sin z$$

and $$y_1 = -x \sin z + y \cos z.$$

The inputs are x,y,z and the outputs are x_1, y_1.

```
SAMPLE BSP(X,Y,Z)
X1     X*COS(Z) + Y*SIN(Z)
X1     -X*SIN(Z) + Y*COS(Z)
SAMPLE ESP(X1,Y1)
```

We could use SAMPLE in several places in a program possibly as follows:

```
SAMPLE   CSP(U1,U2,U3)
         RSP(V1,V2)
           :
SAMPLE   CSP(2.*T,45.,PI)
         RSP(P1,P2)
           :
```

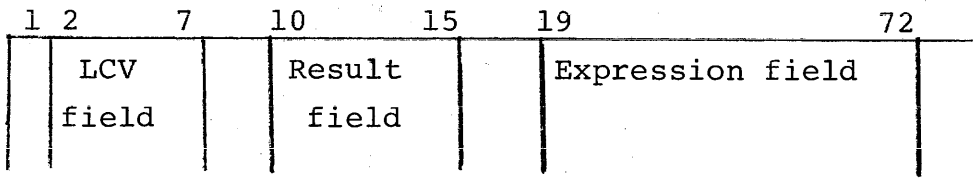
Restrictions:

(i) The functions FTR and INT may not be used in subprograms.

(ii) Arbitrary functions defined by CFN or PFN statements cannot be used as input to a subprogram; however, they may be used in FUN expressions within the subprogram.

IX. CARD FORMATS

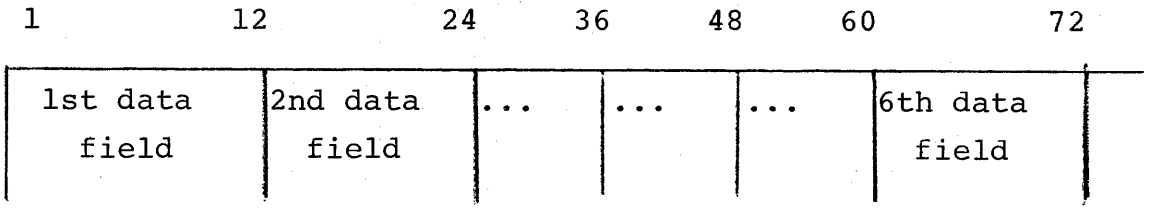
1. Program Cards



Any symbol in column 1 results in the entire card being treated as a comment.

A single variable name only can appear in the LCV or Result field.

2. Data Cards



Up to six 12-column fields allowed per card.

For CON and PAR statements, the value corresponding to the jth argument must be punched in the jth data field of the data card ($1 \leq j \leq 6$).

Data is punched in FORTRAN floating point format E12.0, i.e.

(i) possible prefixed minus sign.

(ii) explicit decimal point.

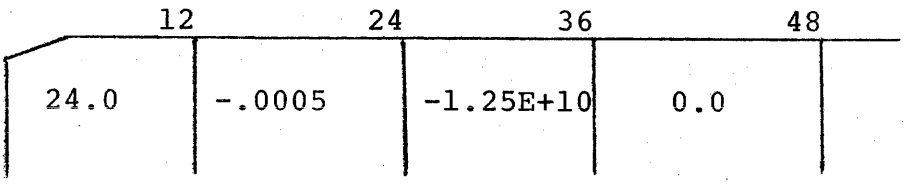
(iii) possible postfixed exponent of forms E±dd, E±d, Edd, Ed where d is a digit.

(iv) no more than 12 characters per number.

eg. -1.72E7 means -1.72×10^7

Note: An E-style number must be right justified in its data field.

For example, for CON(A,B,C,D) a data card might be



X. ERROR MESSAGES

Only the following errors will be detected and identified by the processor and will cause deletion of the execution phase.

(i) Variable names which have not been assigned a value.

(ii) Algebraic loops.

(iii) A BSP card and no ESP card.

(iv) A CSP card not followed immediately by an RSP card.

(v) Use of a function not contained in Table 1.

In addition, the usual execution time F-IV error messages might appear.

XI. SPECIAL POINTS AND SUGGESTIONS

1. Integration

(i) A maximum of 95 integrators is allowed.

(ii) If DTMAX, DTMIN, DT are not specified in the program, the following values are assumed:

DT = DTMAX = 0.1

DTMIN = 0.

(iii) The processor uses a variable step, high order Runge-Kutta routine for integration. The step size may range from DTMIN to DTMAX. Setting DTMIN equal to DTMAX results in fixed step length integration.

(iv) Use of hybrid functions FLF, MMV, TAS, ZOH require that DTMAX = DTMIN.

(v) The integration routine will automatically vary the step size in order to satisfy an error criterion. Thus, problems involving discontinuities generated by function switches, limiters, random number generators, etc. can cause trouble as the routine could continue to reduce the step size indefinitely, if a positive DTMIN is not specified, in order to integrate across discontinuities.

It is impossible to give a rule for selecting a DTMIN but the following approach is suggested.

Try DTMIN = 10^{-4}DT and compare the results with another run with DTMIN one tenth the first value. If the results agree well enough, neither value inhibited the integration routine to any appreciable extent. If the results differ significantly, decrease DTMIN and rerun. Always try to use the largest value of DTMIN consistent with satisfactory results in order to decrease computer time.

2. Machine Fault Conditions

A diagnostic of the following form will be returned every time a machine fault occurs as follows:-

ERROR IN MIMIC CALLED FROM XXXXX YYYYY

Where XXXXX is a machine address

and YYYYY is a. exponent overflow,
b. divide fault, or
c. arithmetic overflow.

3. Control of Printed Output

The processor outputs 60 lines per page on the printer. Thus if a DT = 10^{-6} were selected for integration over a range as small as $0 \leq T \leq 6$ about 10^5 pages of output would be printed. It might be possible that only the final values for T = 6 are of interest. The following are suggestions for controlling the volume of output:

(i) Specify DTMAX to control integration and specify DT>>DTMAX to control output.

(ii) Use logical control variables in the LCV field of OUT statements to produce selected output.

For example:

```
        :
CONOUT   FSW(T-B,FALSE,TRUE,TRUE)
        :
        :
CONOUT           OUT(T,X)
        :
```

will produce output only when $T \geq B$.

(iii) Use more than one FIN statement to terminate the run. For example, terminate if some condition, say stable values of a variable, has been achieved or total time has elapsed.

4. Errors in Programs

Section X details the only source language errors detected by the processor. Thus a programmer should take care in keypunching since the processor will evaluate expressions which are syntactically incorrect or outright meaningless. For example the processor will produce a value for 2.*TRUE.

In compiling a program, the processor reduces all complex expressions to a sequence of calculations using the basic elements of Table I and lists this simplified program below the input program. A perusal of this simplified listing will often detect errors in the original program.

XII. SAMPLE CONTROL CARDS FROM BATCH

7/8 JOB,<JOB #>,<USER #>,<LISTING ID>

7/8 TIME = <time limit>

7/8 MFBLKS = <File Block limit>

7/8 *MIMIC,<PARAMETER STRING>

MIMIC Program

Constants

Constant Functions

.
.
.
.

Data card - number of parameter sets (FORMAT(I4))

Parameter functions

Parameters

.
.
.

7/8 LOGOFF

Note if plotted output is required, EQUIP a unit to a plotter; label it and have a PLOT = LUN PARAMETER on the control card.

```
7
8 JOB,<JOB #>, <USER #> , <LISTING ID>

7
8 TIME =  <TIME LIMIT>

7
8 MFBLKS = <FILE BLOCK LIMIT>

7
8 *MIMIC, <PARAMETER STRING>

      MIMIC PROGRAM
      CONSTANTS
      CONSTANT FUNCTIONS

7
8 *MIMICLOAD, <PARAMETER STRING>

      DATA CARD - Number of parameter sets (FORMAT 14)
      PARAMETER FUNCTIONS
      PARAMETERS

7
8 LOGOFF
```

XIII. SAMPLE CONTROL STATEMENTS FROM TELETYPE

```
#<JOB#>,<USER#>
#TIME=<TIME LIMIT>
#MFBLKS=<FILE BLOCK limit>
#*MIMIC,I=NAME,<PARAMETER STRING>
#LOGOFF
```

XIV. CONTROL CARD PARAMETERS

Options on the call to the compiler are separated by commas and each may be any number of letters long, and may be followed by an =XX where XX is some logical unit number. For *MIMIC the options currently available are:

ASSEMBLY — produces an assembly listing on the requested unit. If no unit is specified, the assembly listing appears on the same unit as the source listing (Logical unit 61 if neither the LIST or ASSEMBLY options have specified a unit).

CONSTANT — produces a listing of the constants defined on CON or CFN cards. If no unit is specified, listing defaults to other units in the same order as the FUNCTION listing.

FUNCTION — produces the SORTED FUNCTION LISTING GENERATED by the compilier. If no unit is specified use the assembly unit if specified or default to the list unit if specified or USE 61.

INPUT — specifies the unit or file from which the source program is to be read. (Logical unit 60 is assumed if no unit is specified or if the option is not present.) The compiler will automatically recognize and unblock EDIT and COSY compressed records in addition to standard Hollerith card images. As an additional feature, the name of a saved file may be used in place of a logical unit number on this option. MIMIC will rewind the input unit before reading if a name was specified as the input.

LIST — specifies that a listing of the source is to be written out on the specified unit (Logical unit 61 if no unit was specified.)

PLOT — produces the plot as specified on PLO cards. (If PLO cards are present and the plot option is not used the plot will be suppressed.) Assumes Logical unit 55 if none specified. (However it is a good idea to equip a unit to a plotter before starting a run and labeling the unit.)

OUTPUT — the unit to list the output generated by the source program. Logical unit 61 is assumed if not specified.

RUN — indicates a unit (Logical unit 56 if none is specified) on which binary output is to be generated. The RUN unit is released initially by the compiler, and on compiling the last program element, is automatically rewound and loaded and execution is initiated if no serious errors were encountered in compilation.

XCT — indicates a unit (Logical unit 56 if none is specified) on which binary output is to be generated. MIMIC releases this unit before compiling.

If the RUN option is not present and either the plot or output option is present they are ignored. If the source program input and the data for the source program are on different Logical units the RUN cannot be used.

For *MIMICLOAD the options currently available are:

INPUT — specifies the unit or file from which the data for the source program is to be read. (Logical unit 60 is assumed if no unit is specified or if the option is not present.) The compiler will automatically recognize and unblock EDIT and COSY compressed records in addition to standard Hollerith card images. As an additional feature, the name of a saved file may be used in place of a logical unit number on this option. The compiler will rewind the input unit before reading.

OUTPUT — the unit to list the output generated by the source program. Logical unit 61 is assumed if not specified.

PLOT — produces the plot as specified on PLO cards. (If PLO cards are present and the plot option is not used the plot will be suppressed.) Assumes Logical unit 55 if none specified. (However it is a good idea to equip a unit to a plotter before starting a run and labeling the unit.)

XCT — unit on which to find binary output from *MIMIC. If no unit specified assumes Logical unit 56.

NOTE: *MIMICLOAD is a loader that loads binary output produced by *MIMIC and then starts execution.

SAMPLE COMPILER CALLS

```
*MIMIC,R,L
*MIMIC,X,I=10,L,F=5
*MIMICLOAD,X,J=1,O=45,P=13
*MIMIC,R,L,A,F,C,P
```

APPENDIX I

Example 1: The equations for a bouncing ball dropped from rest at height h are $\ddot{x} = -g, \dot{x}(0) = 0, x(0) = h$ where $x(t)$ represents the height at time t. But whenever $x = 0$, the velocity is changed in direction and decreases in magnitude by proportion e, the coefficient of elasticity.

A possible job is

```
1 2              10          19   24
                             CON (TSTOP, DTMIN)
                             PAR (E,H)
                 ZEROHT      FSW(X,TRUE,TRUE,FALSE)
ZEROHT BECOMES TRUE WHEN X REACHES ZERO AND IS FALSE FOR X POSITIVE
                 XDOT        INT(-32.2,0.)
 ZEROHT          XDOT        -E*XDOT
                 X           INT(XDOT,H)
                             HDR (TIME,SPEED,HEIGHT)
                             HDR
                             OUT (T,XDOT,X)
                             FIN (T,TSTOP)
                             END
10.0                         1.0E-4
 0.09                       .15E+02
  .8              20.0
  .5              25.0
```

The use of logical variable ZEROHT ensures that we account for elasticity and direction only at times when $x \leq 0$. (< used should the integration routine slightly overshoot to negative x)

Example 2: Illustration of subprograms and precomputation of constants.

Evaluate the complex-valued variable

$$z = P(s)/q$$

where $P(s) = us^2 - vs + w = (us-v)s + w$

$u = e^{i\alpha}$

$v = \log(\sin\beta)$

$w = \alpha\cos\beta + i\beta\sin\alpha$

$q = \alpha + i\beta t$

$s = t(\alpha + i\beta)$

for t = 0 (.2) 2. (α, β are real numbers)

Since u, v, w remain constant for all t, we will compute them only once when t = 0 by using a logical control variable which has the value TRUE only when T = 0.

Subroutines COMPAS, COMPMD will be defined as follows:

COMPAS - number of inputs = 5

- inputs 1, 2, 3, 4 are real and imaginary parts of 2 complex numbers

- input 5 is a logical-valued argument
  - if its value is TRUE, COMPAS adds
  - if its value is FALSE, COMPAS subtracts

- number of outputs = 2

- the real and imaginary parts of the result.

COMPMD - number of inputs = 5

- inputs 1, 2, 3, 4 are real and imaginary parts of 2 complex numbers

- input 5 is a logical-valued argument
  - if its value is TRUE, COMPMD multiplies
  - if its value is FALSE, COMPMD divides

- number of outputs = 3

  - outputs 1, 2 are real and imaginary parts of the result

  - output 3 - a logical-valued variable
    - its value is FALSE only if the input divisor was zero for a complex division. (Outputs 1, 2 = 0 in this case) Otherwise it has the value TRUE.

The program:

```
1 2          10        19
|  |         |         |
|  |         |         CON(ALPHA,BETA)
|  DEFINE  SUBPROGRAM  COMPAS
|  |         COMPAS    BSP (RA,IA,RB,IB,LIN)
|  |         SIGN      LSW(LIN,1.,-1.)
|  |         RC        RA+SIGN*RB
|  |         IC        IA+SIGN*IB
|  |         COMPAS    ESP (RC,IC)
|  DEFINE  SUBPROGRAM  COMPMD
|  |         COMPMD    BSP (R1,I1,R2,I2,LOGIN)
|  |         SIGN1     LSW(LOGIN,1.,-1.)
|  |         R3        R1*R2-SIGN*I1*I2
|  |         I3        SIGN1*R1*I2+I1*R2
|  |         NOTL      NOT(LOGIN)
|  |         LOUT      TRUE
|  |         ZERO      FALSE
|  |         |         |
|  |         |         |
```

```
1 2          10         19
  NOTL       DENOM      R2*R2+I2*I2
  NOTL       ZERO       FSW(DENOM,TRUE,FALSE,TRUE)
  ZERO       R3         R3/DENOM
  ZERO       I3         I3/DENOM
  NOTL       LOUT       ZERO
             COMPMD     ESP(R3,I3,LOUT)
PRECOMPUTE   U,V,W,      DT
             ZEROT      FSW(T,TRUE,TRUE,FALSE)
  ZEROT      RU         COS(ALPHA)
  ZEROT      IU         SIN(ALPHA)
  ZEROT      RV         LOG(SIN(BETA))
  ZEROT      IV         0.
  ZEROT      RW         ALPHA*COS(BETA)
  ZEROT      IW         BETA*SIN(ALPHA)
  ZEROT      DT         0.2
CALCULATION OF Z
             COMPMD     CSP(RU,IU,T*ALPHA,T*BETA,TRUE)
                        RSP(RP,IP,LOGVAL)
             COMPAS     CSP(RP,IP,RV,IV,FALSE)
                        RSP(RQ,IQ)
             COMPMD     CSP(RQ,IQ,T*ALPHA,T*BETA,TRUE)
                        RSP(RS,IS,LOGVAL)
             COMPAS     CSP(RS,IS,RW,IW,TRUE)
                        RSP(RT,IT)
             COMPMD     CSP(RT,IT,ALPHA*T,BETA*T,FALSE)
                        RSP(RZ,IZ,LOGVAL)

LOGVAL                  OUT(T,RZ,IZ)
                        FIN(T,2.)
                        END
```

LOGVAL is used as an LCV on the OUT statement so no results are printed in case a zero division was attempted in COMPMD.

APPENDIX II

Solution of a Partial Differential Equation by Finite Approximations[1]

The equation for the concentration C (r,t) of a liquid component diffusing radially out of a solid particle cylinder of radius R is

$$\frac{\partial C}{\partial t} = D\left[\frac{\partial^2 C}{\partial r^2} + \frac{1}{r}\frac{\partial C}{\partial r}\right]$$

D is the diffusivity.

There is an initial uniform concentration distribution C* throughout the cylinder and at time t = 0 we allow the material to commence diffusing by putting C(R,O) to zero and maintaining it at zero.

The analytic solution to the problem is

$$C(r,t) = \sum_{i=1}^{\infty} \frac{2C^*}{\alpha_i J_1(\alpha_i)} J_0\left(\alpha_i \frac{r}{R}\right) \exp\left(-\frac{D\alpha_i^2}{R^2} t\right) \quad \text{II-1}$$

(See Reddick and Miller: Advanced Mathematics for Engineers, J. Wiley, p. 319, Eq. 21)

α_i are the positive zeros of $J_0(x)$

To acquire numerical values of C(r,t) using MIMIC we make the approximations

$$\frac{\partial C_n}{\partial r} \quad \frac{C_{n+1} - C_{n-1}}{2\Delta r}, \qquad \frac{\partial^2 C_n}{\partial r^2} = \frac{C_{n+1} - 2C_n + C_{n-1}}{(\Delta r)^2}$$

1. Contributed by Professor T. Fahidy, University of Waterloo

where $C_n = C(n\Delta r, t)$. The partial differential equation is turned into a set containing n ordinary differential equations. This is a conventional technique for analogue computers.

Now we have

$$\frac{dC_n}{dt} = \frac{D}{(\Delta r)^2}\left[\left(1 + \frac{1}{2n}\right) C_{n+1} - 2C_n + \left(1 - \frac{1}{2n}\right) C_{n-1}\right]$$

$n = 1, 2, 3, 4, \ldots$

for the concentration on concentric circles of radius $n\Delta r$.

Taking $C^* = 1$, $R = 0.5$, $r = 0.1$, $n = 5$, $D = 10^{-5}$ we get

$$\dot{C}_1 = 10^{-3}\,(1.5C_2 - 2C_1 + 0.5\,C_0)$$

$$\dot{C}_2 = 10^{-3}\,(1.25C_3 - 2C_2 + 0.75\,C_1)$$

$$\dot{C}_3 = 10^{-3}\,(1.167C_4 - 2C_3 + 0.833\,C_2)$$

$$\dot{C}_4 = 10^{-3}\,(1.125C_5 - 2C_4 + 0.875\,C_3) \quad (C_5 = 0 = \text{boundary value})$$

A MIMIC program to integrate this system of equations for C_j in t = 0 (10.) 200. along with all computer output follows.

Total 3300 time for this job was 1 minute 21 seconds. The value of C(0.4, 200) calculated from II-1 is 0.872488 in contrast with the MIMIC - computed value of 0.813412. The error (about 7% may be further reduced by considering smaller increments of Δr. This is a case where accuracy and program complexity have to be balanced.

MIMIC SOURCE-LANGUAGE PROGRAM

The Radial Diffusion Problem with 5 Increments

```
      CON(1A,2A,3A,2N)
      CON(1B,2B,3B,4B)
      CON(DT,DTMAX)
1C    INT(1A*2C-2N*1C+1B,1.)
2C    INT(2A*3C-2N*2C+2B*1C,1.)
3C    INT(3A*4C-2N*3C+3B*2C,1.)
4C    INT(-2N*4C+4B*3C,1.)
      FIN(T,200.)
      HDR(T,C1,C2,C3)
      HDR(,C4)
      HDR
      OUT(T,1C,2C,3C)
      OUT(,4C)
      END
```

FUNCTION-LANGUAGE PROGRAM GENERATED

IFN	LCV	RESULT	FTN	A	B	C	D	E	F
1			CON	1A	2A	3A	2N		
2			CON	1B	2B	3B	4B		
3			CON	DT	DTMAX				
4		103	MPY	1A	2C				
5		104	MPY	2N	1C				
6		105	SUB	103	104				
7		106	ADD	105	1B				
8		1C	INT	106	1.				
9		108	MPY	2A	3C				
10		109	MPY	2N	2C				
11		110	MPY	2B	1C				
12		111	SUB	108	109				
13		112	ADD	111	110				
14		2C	INT	112	1.				
15		114	MPY	3A	4C				
16		115	MPY	2N	3C				
17		116	MPY	3B	2C				
18		117	SUB	114	115				
19		118	ADD	117	116				
20		3C	INT	118	1.				
21		120	MPY	2N	4C				
22		121	MPY	4B	3C				
23		00	NEG	120					
24		122	ADD	00	121				
25		4C	INT	122	1.				
26			FIN	T	200.				
27			HDR	T	C1	C2	C3		
28			HDR		C4				
29			HDR						
30			OUT	T	1C	2C	3C		
31			OUT		4C				
32			END						

FURTHER DIAGNOSTICS AND EXECUTION FOLLOW

1A	2A	3A	2N
1.50000E-03	1.25000E-03	1.16700E-03	2.00000E-03

1B	2B	3B	4B
5.00000E-04	7.50000E-04	8.33000E-04	8.75000E-04

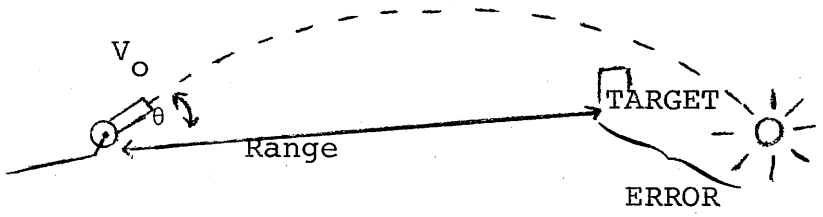
DT	DTMAX
1.00000E 01	1.00000E-01

T	C1	C2	C3
	C4		
0.	1.00000E 00	1.00000E 00	1.00000E 00
	1.00000E 00		
1.00000E 01	9.99998E-01	9.99998E-01	9.99934E-01
	9.88860E-01		
2.00000E 01	9.99997E-01	9.99995E-01	9.99742E-01
	9.77940E-01		
3.00000E 01	9.99994E-01	9.99989E-01	9.99429E-01
	9.67234E-01		
4.00000E 01	9.99992E-01	9.99978E-01	9.98999E-01
	9.56737E-01		
5.00000E 01	9.99990E-01	9.99962E-01	9.98458E-01
	9.46443E-01		
6.00000E 01	9.99987E-01	9.99938E-01	9.97810E-01
	9.36348E-01		
7.00000E 01	9.99986E-01	9.99907E-01	9.97058E-01
	9.26447E-01		
8.00000E 01	9.99984E-01	9.99866E-01	9.96208E-01
	9.16735E-01		
9.00000E 01	9.99980E-01	9.99814E-01	9.95263E-01
	9.07208E-01		
1.00000E 02	9.99976E-01	9.99752E-01	9.94228E-01
	8.97860E-01		
1.10000E 02	9.99972E-01	9.99677E-01	9.93105E-01
	8.88689E-01		
1.20000E 02	9.99965E-01	9.99590E-01	9.91899E-01
	8.79688E-01		
1.30000E 02	9.99958E-01	9.99488E-01	9.90613E-01
	8.70856E-01		
1.40000E 02	9.99949E-01	9.99372E-01	9.89250E-01
	8.62186E-01		
1.50000E 02	9.99939E-01	9.99241E-01	9.87815E-01
	8.53677E-01		
1.60000E 02	9.99927E-01	9.99095E-01	9.86309E-01
	8.45323E-01		
1.70000E 02	9.99912E-01	9.98932E-01	9.84736E-01
	8.37121E-01		
1.80000E 02	9.99896E-01	9.98752E-01	9.83099E-01
	8.29067E-01		
1.90000E 02	9.99877E-01	9.98555E-01	9.81400E-01
	8.21159E-01		
2.00000E 02	9.99855E-01	9.98341E-01	9.79643E-01
	8.13392E-01		

APPENDIX III

Example of Integrator Mode Control and Hybrid Functions[1]

This problem is designed to illustrate the use of individual mode control of the integrators and the use of certain other logical elements in the solution of problems which more conventionally might be solved on a hybrid analogue-digital computer installation. The problem chosen is that of hitting a target at a known distance with a projectile fired at a certain muzzle velocity and at some angle of elevation to the horizon.



Actual dynamic equations describing such a system are quite complex when all the aerodynamic forces on the projectile are considered. The method of solution illustrated by this problem is general enough to be used for the determination of the firing parameters in complex non-linear representations of the projectile equations. However, for the sake of computer time and so as not to obscure the method of solution by using an unnecessarily difficult equation set, the actual equations chosen have been simplified to the point of triviality.

---

1. Contributed by E. M. Hinchley

The equations (neglecting all forces except gravity) are:

$$\dot{x} = V_o \cos\theta \qquad x(0) = 0$$
$$\ddot{y} = -g \qquad y(0) = V_o \sin\theta$$
$$y(0) = 0$$

The desired point of impact is designated by x = RANGE and y is the variable representing the height of the projectile. V_o is the muzzle velocity, and θ is the angle of elevation (see sketch). For simplicity V_o will be assumed constant and θ will be the parameter we must determine. The equations of the projectile are so simplified that they are easily solved by direct integration. However, we are attempting to illustrate a more general method of solution- an iterative analog computer-like solution.

The equations are programmed in MIMIC as they would be for an analog computer solution. A value of θ is assumed and the analog solution is operated until $y \leq 0$ (and $x>0$). The value of x at y=0 represents the distance of the point of impact. The analog solution is placed in the HOLD mode while the error function x-RANGE is calculated, and a new value of θ equal to the previous θ plus K(x-RANGE) is determined. K is a carefully chosen constant which affects both the convergence properties and the stability of the solution. The analog solution is next RESET to the new initial conditions and then placed in OPERATE and the new point of impact is determined. The process is repeated until a point of sufficiently close to the desired RANGE is found.

The modes of the integrators are controlled by logical variables C and D according to this table:

D \ C	TRUE	FALSE
TRUE	OPERATE	HOLD
FALSE	RESET	OPERATE

Through the use of two function switches in the program, a logical variable B1 is obtained such that B1 is FALSE for $y \leq 0$, $x > 0$ and TRUE otherwise. Another logical variable S becomes

FALSE at the same time. It was decided arbitrarily that the solution should HOLD for 1 second after impact, and then RESET for 1 second after that. The timing was obtained through the use of monostable multivibrators (MMV). The logical variable H is TRUE when S is TRUE and becomes FALSE one second after S becomes FALSE. Logical variable 0 becomes FALSE 2 seconds after S becomes FALSE. The integrator control variables C and D are obtained by "ANDing" and "ORing" combinations of the above variables to give the proper mode cycling from OPERATE to HOLD to RESET to OPERATE. The logical functions are:

$$S = H \cdot 0 \cdot Bl + Bl \cdot \overline{0}$$
$$C = S \cdot 0 \cdot H + \overline{S} \cdot \overline{H}$$
$$D = H \cdot 0$$

$\cdot$ is AND
$+$ is OR
x is NOT x

Initially Bl is TRUE, and S is made TRUE, is assuring that H and 0 are TRUE, so that C and D are TRUE and the solution will OPERATE.

These logical functions were obtained through the use of a simple truth table. Some other simpler methods of controlling the integrator modes were tried, but since the 7040 is a serial machine, it was found that the simulation of parallel logic required a great deal of care.

The variable THETA (θ) is updated in the HOLD mode only through the use of a track and store device (TAS) and logical control variables. It was necessary to use equal integration intervals because of the logical devices being used. Despite the simplicity of the equations and the use of a fairly large integration interval, solution times were surprisingly large on the 7040. Convergence towards a solution was obtained. A program listing including comment cards follows:

```
PARAMETERS
                    PAR(RANGE,VO,THETAO,G,K,DT)
EQUAL INTEGRATION INTERVAL
          DTMIN     EQL(.01)
          DTMAX     EQL(DTMIN)
LOGICAL VARIABLE B1 BECOMES FALSE AT THE END OF EACH ITERATION
          B         FSW(Y,FALSE,FALSE,TRUE)
          B1        FSW(X,TRUE,TRUE,B)
TQ TRUE AT T=0, CONTROLS PRECOMPUTATION OF CONSTANTS
          TQ        FSW(T,FALSE,TRUE,FALSE)
          NTQ       COM(TQ)
LOGICAL VARIABLES FOR MODE CYCLING FOLLOW
 TQ       S         EQL(TRUE)
 NTQ      S         IOR(AND(H,O,B1),AND(B1,COM(O)))
          H         MMV(S,1.0)
          O         MMV(S,2.0)
          C         IOR(AND(S,O,H),AND(COM(S),COM(H)))
          CP        COM(C)
          D         AND(H,O)
NEXT 3 CARDS COMPUTE THETA,UPDATING IT IN THE HOLD MODE
          TH        TAS(THETA,C,THETAO)
 CP       THETA     EQL(TH+COR)
 TQ       THETA     EQL(THETAO)
ACTUAL PROJECTILE EQUATIONS ARE SOLVED BY THE NEXT 2 STATEMENTS
          X         INT(VO*COS(THETA),0.0,C,D)
INTEGRATE -G TWICE WITH RESPECT TO TIME
USING MODE CONTROLS
          Y         INT(INT(-G,VO*SIN(THETA),C,D),0.0,C,D)
ERROR FUNCTION AND A CORRECTION FOR THE NEXT VALUE OF THETA ARE FOUND
          ERR       SUB(X,RANGE)
          COR       MPY(K,ERR)
FINISH CONDITIONS
 CP                 FIN(100.0,ABS(ERR))
                    FIN(T,200.)
TITLES AND OUTPUT CARDS
                    HDR(T,X,Y,THETA)
                    HDR
                    OUT(T,X,Y,THETA)
NUMERICAL VALUES FOR PARAMETERS FOLLOW END CARD
                    END
2000.0       750.0       0.8          32.2         0.00001          0.2
```

APPENDIX IV

The Simulation of a Servo Control System

Figure 1 shows the control system to be simulated using three settings of the compensation network bandwidth denoted as

$$X1, X\frac{1}{3} \text{ and } X\frac{1}{10}.$$

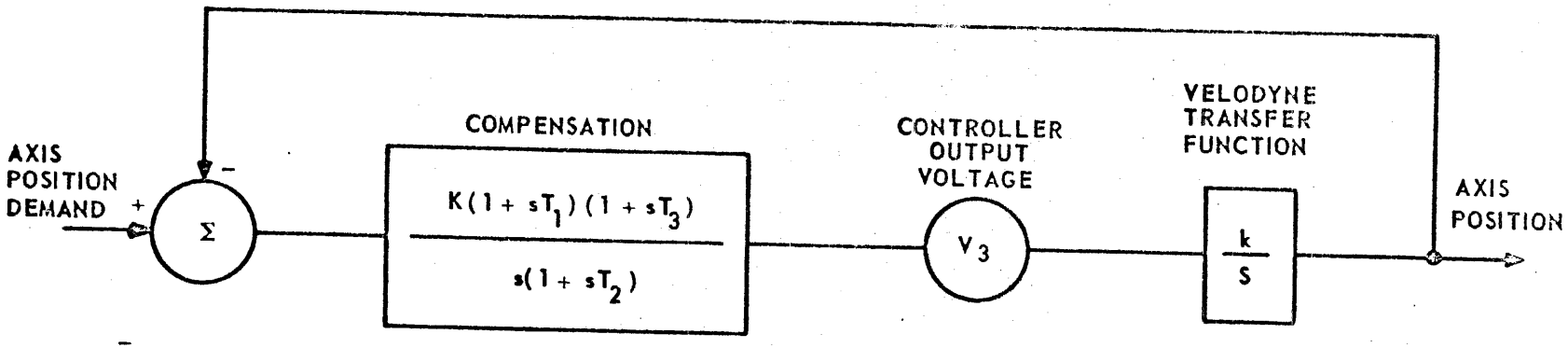
The transfer function

$$\frac{(1 + ST_1)(1 + ST_3)}{S(1 + ST_2)}$$

is reduced to

$$\frac{T_1T_3}{T_2} + \frac{1}{S} + \frac{T_1 + T_3 - \frac{T_1T_3}{T_2} - T_2}{1 + T_2S}$$

and the equivalent mimic elemental representation of the servo system is shown in Figure 2. Use is made of the first order transfer function, FTR, the integrator INT and the usual arithmetic elements. Listing of the source program is shown in Figure 3 and the plotted output Figure 4. Total computer run time on the CDC 3300/MASTER was 1 minute 38 seconds.



PARAMETER	X1		X 1/3		X 1/10		UNITS
	EL	AZ	EL	AZ	EL	AZ	
K	45.4	25	4.54	2.5	.454	.25	VOLTS/SECOND/DEGREE
k	.222	.400	.222	.400	.222	.400	DEGREES/SECOND/VOLT
T_1	15		47.4		150		SECONDS
T_2	165		522		1650		SECONDS
T_3	2.5		7.91		25		SECONDS

FIGURE 1 - COMPENSATED TYPE 2 POSITION LOOP

MIMIC EQUIVALENT OF FIGURE 1

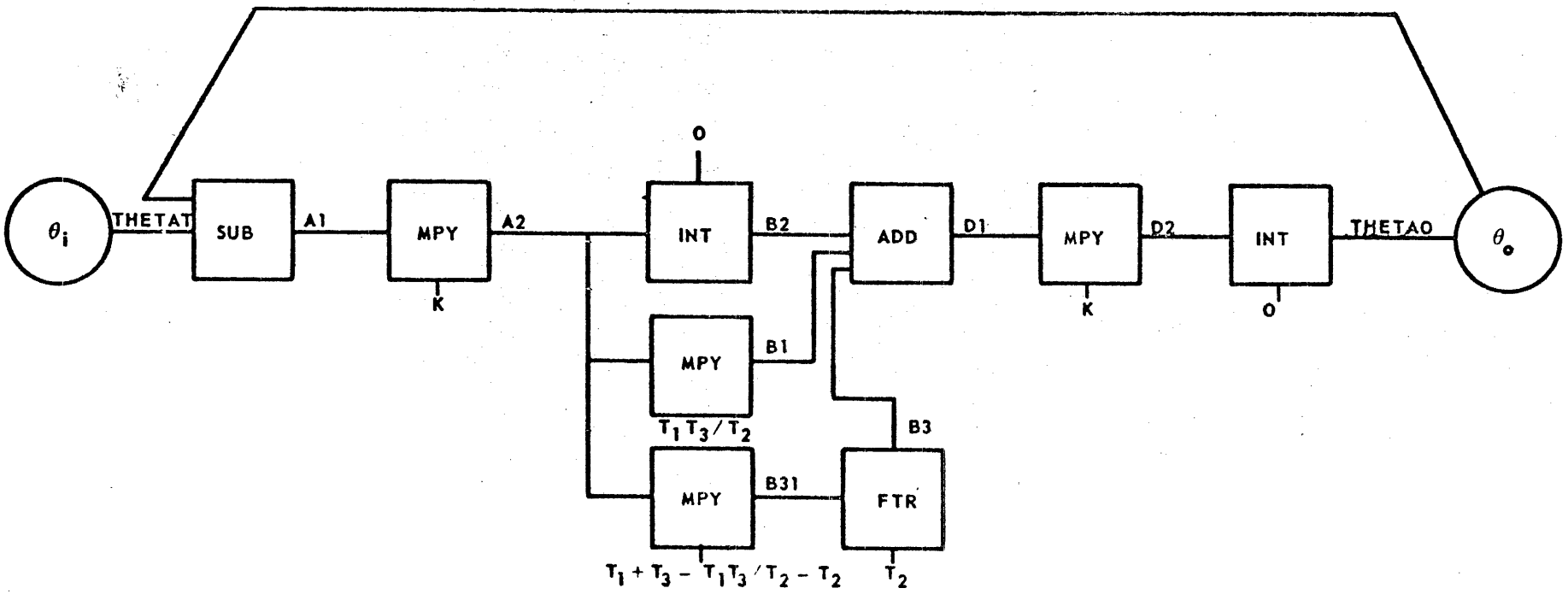
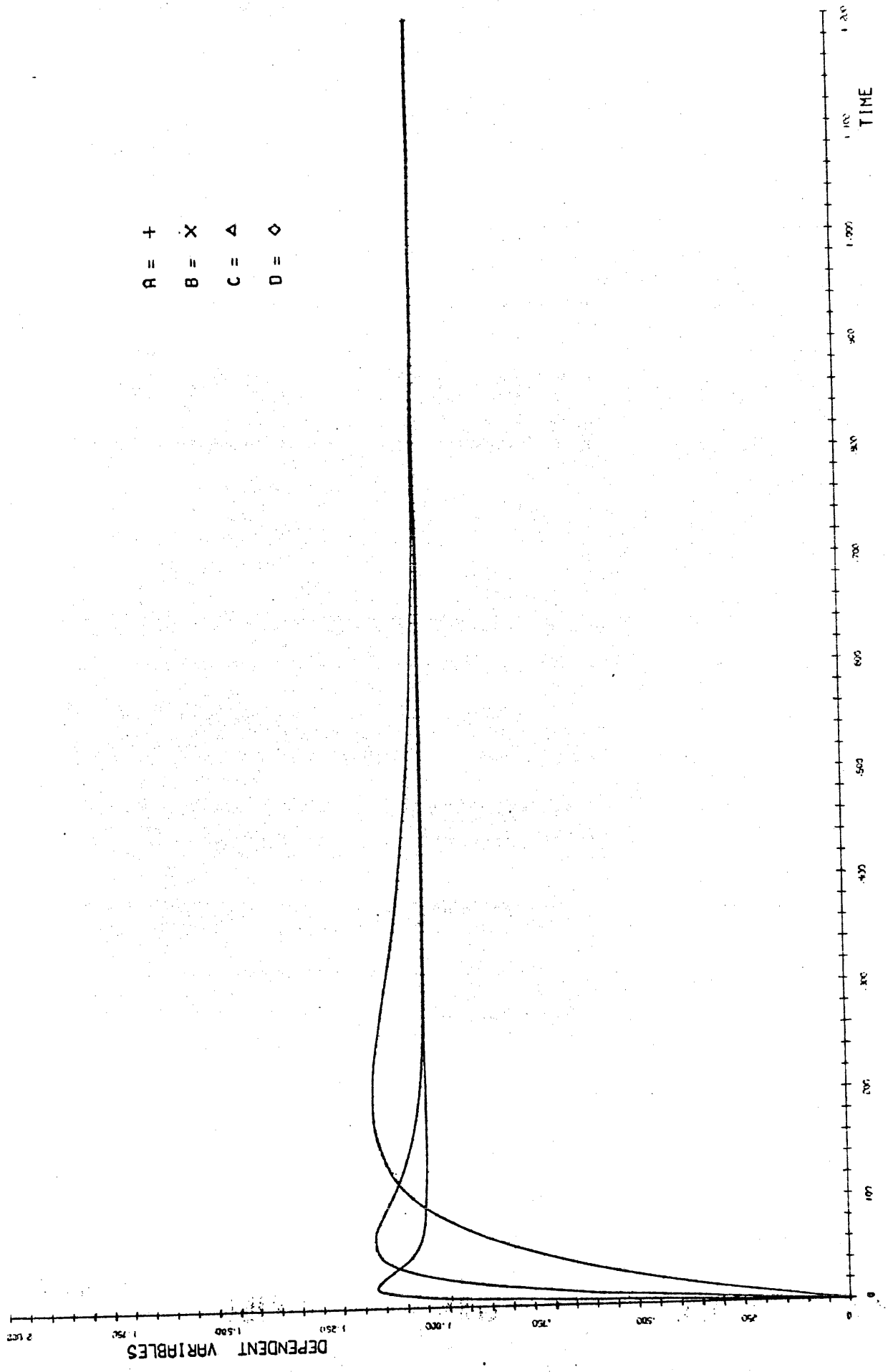


FIGURE 2

TO RUN FROM BATCH THE DECK LOOKS LIKE:

```
000100  #<JOB#>,<USER#>,<USER ID>
000200  7
        8EQUIP,1=PLOT
000300  7
        8LABEL,1/<USER ID>
000400  7
        8#MIMIC,L,P=1,R
000500  THIS IS A SIMULATION OF A PRECISION ANTENNA
000600  CONTROL SYSTEM
000700                          CON(THETAI,DT)
000800                          PAR(T1,T2,T3,K1,K2)
000900          A1              SUB(THETAI,THETAO)
001000          A2              MPY(A1,K1)
001100          B1              MPY(A2.(T1*T3)/T2)
001200          B2              INT(A2,0.)
001300          B31             MPY(A2,T1+T3-T1*T3/T2-T2)
001400          B3              FTR(B31,T2)
001500          D1              ADD(B1,B2,B3)
001600          D2              MPY(D1,K2)
001700          THETAO          INT(D2,0.)
001800                          FIN(T,120.)
001900                          HDR(TIME,RESP)
002000                          HDR
002100                          OUT(T,THETAO)
002200                          PLO(2.,.25,THETAO)
002300                          END
002400  1.00000E 01 8.00000E-01
002500    3
002600  1.50000E 01 1.65000E 02 2.50000E 00 2.50000E 01 0.40000E 00
002700  4.74000E 01 5.22000E 02 7.91000E 00 2.50000E 00 0.40000E 00
002800  1.50000E 02 1.65000E 03 2.50000E 01 2.50000E-01 0.40000E 00
```

FIGURE 3



MIMIC PROGRAM OUTPUT