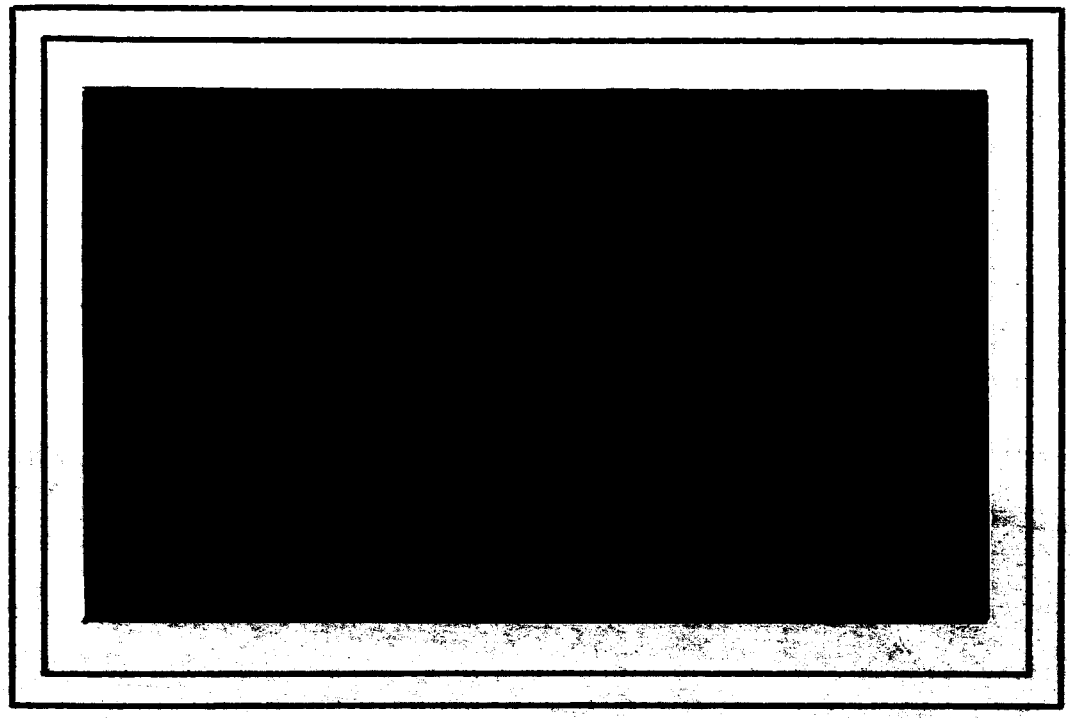


UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

Technical Report TR-67-40
NsG-398

DIGITAL ANALOG SIMULATION TECHNIQUES

Yaohan Chu

February 15, 1967

ABSTRACT

Digital analog simulation is the use of a general purpose digital computer to solve engineering and scientific problems or to simulate continuous models that are commonly done on an analog computer. Digital analog simulation is not new, but it has become more and more accepted during recent years. This paper presents some techniques in digital analog simulation. There are two types of techniques: those developed for analog computers but can also be applied to digital simulation, and those developed for digital analog simulation by provision in the simulation language and its compiler. In this paper, the recent language Mimic is chosen to be used as the simulation language.

Digital analog simulation techniques are shown by presenting the Mimic programs of several problems and their results. Six problems are chosen: (1) solution of a non-linear differential equation, (2) computation of a complex transcendental expression, (3) generation of Bessel functions of the first kind, (4) solution of a partial differential equation, (5) double integration of a function, and (6) simulation of a non-linear feedback control system.

The first problem is the well-known van der Pol's equation. This problem is to introduce the Mimic language to the readers who may not know this simulation language. The Mimic program is explained, and the formats and some features of the language are pointed out. The limit cycle and stability of the solution of this problem can be observed from the readily obtained time-response plot and phase-plane plot. The scaling and labeling of the plots are done automatically.

The second problem is computation of a complex transcendental expression which describes the antenna radiation pattern of a four-element binomial array. This problem is chosen to show computation for an engineering application by Mimic instead of solution of a differential equation.

The third problem is the generation of Bessel functions of the first kind by using the differential relations of these functions. This approach requires a successful way to handle the initial conditions. Mimic language can accomplish this readily.

The fourth problem is a partial differential equation describing a one-dimensional heat flow. As developed for analog computer solution, the partial differential equation is converted by finite difference to a system of ordinary differential equations. A larger number of these differential equations gives a more accurate result, and digital analog simulation can achieve this readily. The Mimic program here illustrates the simultaneous solution of a system of twenty differential equations.

The fifth problem is to find the volume of a sphere by using a numerical approach of double integration developed for the analog computer. Since the exact value of the sphere is known, the exact value is used for comparison with the result from this approach. The result shows the great accuracy of the double integration obtainable by this approach. The Mimic program here shows the technique to switch out the integrators, each at a specified condition.

The sixth problem is the simulation of a simple relay servo with four different kinds of relays: simple relay, relay with dead space, relay with hysteresis, and relay with both dead space and hysteresis. The dynamic responses using these relays

are compared with that of a linear servo. The Mimic program shows how these kinds of relays are simulated.

Many advantages in using digital analog simulation are now being recognized. The above Mimic programs and their results have shown the advantages of greater accuracy, no need of scaling, and no hardware setup. Plots are available with automatic scaling and labeling. Other advantages are feasibility of function generation of more than two variables, availability of many mathematical functions normally provided in a digital computer facility, and better computer utilization. In addition, the cost of digital analog simulation has become very reasonable. In conclusion, the use of digital analog simulation will be rapidly increased as more and better digital computers become available.

CONTENTS

Digital Analog Simulation Techniques

Yaohan Chu
University of Maryland
College Park, Md. U.S.A.
February 15, 1967

Digital analog simulation is the use of a general-purpose digital computer to solve engineering and scientific problems or to simulate continuous models in a manner similar to that used on an analog computer or a hybrid computer. Digital analog simulation is not new. R. G. Selfridge (1) first pioneered the simulation of a differential analyzer on a general-purpose digital computer (IBM 701) in 1955, but the computer then was terribly slow, hardware floating-point arithmetic was not available, and programming technique was primitive. Since then, many digital analog simulation languages and programs have been developed (2,3), including DYSAC (4), DAS (5), MIDAS (6), PACTOLUS (7),MADBLOC (8), MIMIC (9), DES-1 (10), and DSL/90 (11). Furthermore, digital computer has also been attempted to generate details for setting up the analog computer (12); this effort has recently been reported successful. However, the wide usage of digital-analog simulation appeared to begin when the difital computer became successful to give check solutions for analog simulations. This success has been sparkled by the MIDAS which owed its birth in 1963 to the DAS. The MIDAS has received wide acceptance in universities, research organizations, simulation laboratories, and governmental facilities. Because of some important limitations of MIDAS, a much improved version called MIMIC has been developed in 1965 to succeed the MIDAS. This MIMIC language is used here as the simulation language.

1. MIMIC Language and Compiler

MIMIC (19) is a block-oriented language. It has a set of reserved words such as T, DT, INT, and TRUE which carry special meaningsto MIMIC compiler. MIMIC is expressed in statements such as constant statement, comment statement, computation statement, and integration statement. A MIMIC program is a sequence of statements with each statement punched on one card.

Each MIMIC statement, except the comment statement, may have three fields: LCV (logical control variable) name field (card column 2-7), result name field (col. 10-15), expression field (col. 19-72). (The comment statement, a special case, is indicated by the presence of a character in column 1.) In the block concept, the result name is the output from the block, and the expression shows the inputs and the operation to be performed on the inputs. In the expression field, simple algebraic expressions using operators +, -, *, and / (add, subtract, multiply, and divide) are allowed as well as nesting of expressions by using pairs of parentheses. The LCV name has two values: TRUE and FALSE. If a statement has a LCV, the statement will be executed only when its LCV has a value of TRUE.

The Mimic compiler is written essentially in Fortran. It has a varying step-size, fourth-order Runge-Kutta integration subroutine. It allows multiple runs when some constants are designated as parameters. It has a sort subroutine which permits the computer to behave as if it were a parallel machine. The compiler translates a Mimic program directly into a machine language program and then initiates the execution of the machine language program. It also provides some diagnostic aid for debugging of programs. The compiler is relatively simple that a sophiscated user could understand the details of the compiler well and, if a need should arise, modify the compiler to suit his particular need. Lately, an on-line **ver-**

sion is being developed for use with a cathode-ray-tube display console (13).

This paper presents some techniques in digital analog simulation by using the Mimic language. There are two types of techniques: those developed for analog computers but may now be equally applied to digital analog simulation, and those developed in the digital analog simulation language and compiler. These techniques are shown subsequently by presenting the Mimic programs and the results. Six problems are chosen: solution of a non-linear differential equation, computation of a complex transcendental expression, generation of Bessel functions of the first kind, solution of a partial differential equation, double integration of a function, and simulation of a non-linear feedback control system.

2. Van der Pol's Equation

Many physical phenomena can be described by nonlinear differential equations which are often impractical if not impossible to solve by hand. Van der Pol (14) showed that the behavior of an electronic oscillator could be described by a nonlinear differential equation of the form,

$$\ddot{x} - A(1 - x^2)\dot{x} + B\,x = 0 \qquad (1)$$

where $\dot{x}$ and $\ddot{x}$ are the first and second derivatives of x in time t, and A and B are positive constants. A system characterized by this equation exhibits a limit cycle (oscillation of fixed amplitude and period), which can be clearly observed on the phase plane (a plot of $\dot{x}$ vs x). The limit cycle is due to the coefficient $A(1 - x^2)$ which can become positive, zero or negative when the absolute value of x becomes less than, equal to, or larger than one. At a proper value of x, the oscillation becomes stable.

In order to compute equation (1), one must give the values of constant B and parameter A. A is now taken as a parameter as this equation is to be solved twice, and each time the value of A is different while the value of B remains the same. Initial values of x and $\dot{x}$, (i.e. $x(0)$ and $\dot{x}(0)$ are also taken as parameters as their values are different for each solution. Furthermore, the time at which the computation is terminated must be given, and the plots, if any, must also be specified. These values and specifications are all shown in Table 1.

Table 1, Data for computation of equation (1)

	case (a)	Case (b)
constant, B	1	1
parameter, A	.2	1.5
initial condition,		
$x(0)$	1	3
$\dot{x}(0)$	0	0
termination,	compute until t=50 sec.	
plot,	x vs t and x vs x	

Once the equation and the data are given, we are ready to write a Mimic program which is shown in figure 1. In figure 1, the first five lines give five cards which are the control cards of the Mimic program. The last five lines also give five control cards to initiate a plot routine for off-line plotting. Since these control cards vary with each computer installation, they will not be further shown or discussed.

In a Mimic program proper, if there is a character (such as an asterisk in figure 1) in the first position of the line (or the card), the line (or the card) is a comment card for reference purpose. Therefore, the next three lines and subsequent lines which begin with an asterisk in figure 1 are comment cards. Comment cards will also not be further mentioned either.

The Mimic program in figure 1 is now explained as follows. We write a CON statement for the constant B,

```
CON (B)
```

where CON is a reserved word of the Mimic compiler, and there is a data card associated with this statement. For the three above-mentioned parameters, we write the PAR statement,

```
PAR (A, XO, 1DXO)
```

where XO and 1DXO represent $x(0)$ and $\dot{x}(0)$ respectively. PAR is also a reserved word and one data card for each solution is associated with this statement. Let 2DX, 1DX, and X represent $\ddot{x}$, $\dot{x}$, and x respectively, and move all the terms of equation (1) except the first to the right side of the equal sign. We then have the following statement:

```
2DX = A*1DX-A*1DX*X*X-B*X
```

to represent equation (1). And 1DX and X are obtained by integrating 2DX and 1DX respectively as shown by the following two INT statements,

```
1DX = INT(2DX, 1DX0)
```

and

```
X   = INT(1DX, X0)
```

where INT is a reserved word to represent integrator. To terminate the computation when the time exceeds 50 seconds, we use the FIN statement,

```
FIN(T, 50.)
```

where T is a reserved word representing the independent variable time. Note that the decimal point is required in the number 50., as a decimal point must be present for each number. To obtain a table for the result from the printer, we use the OUT statement,

```
OUT(T,X,1DX)
```

where the first argument T is the independent variable time. To provide headings for each column of the table, we use the HDR statement,

```
HDR(TIME,X,XDOT)
```

where TIME, X, andXDOT are the chosen words for the headings. To obtain the two required plots, we use two PLO statements,

```
PLO(T,X)
```

and

```
PLO(X,1DX)
```

where T and X represents the abscissa of the plots. Finally, we use the END statement to terminate the program. Note the three lines of numbers after the END statement. These three lines are three data cards: the first for the CON statement and the last two for the PAR statements. The CON and PAR statements at the beginning and the END statement and the data cards at the end of the Mimic program proper must be placed in this order; other statements between them can be placed at any order as the Mimic compiler has a sort routine to order these statements into a proper sequence.

As mentioned before, equation (1) and the data in Table 1 represent the data and the specification of the problem for computation. Compare and note the closeness between the Mimic program in figure 1 and the specificiation of the program. This closeness between the programming language and the user's language may be a measure of simplicity of the programming language and degree of the required programming effort.

The results of the solutions are shown in figures 2 through 6. Figure 2 shows portion of the table of results due to the OUT statement, where the values of the parameters are also shown. The time responses for the two solutions are shown in figures 3 and 4, and the phase plane plots are shown in figures 5 and 6. The scaling and labeling of these plots are done automatically by the plot routine in the Mimic Compiler. The phase plane plots show that the system has a stable limit cycle since the paths converge to a single closed path in the plane, and the time response plots show the oscillations reach a fixed amplitude and period. For the solution where the value of A is smaller, the limit cycle is nearly circular (fig. 5), and the oscillation is nearly sinusoidal (fig. 3). For the solution with a larger value of A, the limit cycle is no longer circular (fig. 6), and the oscil-

lation is not quite sinusoidal (fig. 4).

The statement in fig. 1 except the comment statement consists of one or two fields. Statements such as CON, PAR, FIN, HDR, OUT, PLO, and END have only the expression field. The three algebraic statments in fig. 1 have an additional Result Name field. As mentioned before, there is a third field, called the Logical Control Variable Name field. A statement with a logical control variable becomes a conditional statement. An example of using the logical control variable is now shown.

In using an OUT statement to obtain a table output, the time interval to print each entry of the table is automatically chosen by Mimic compiler to be 0.1 second, if it is not specified. This time interval is called DT, a reserved word. For example, no DT is specified in the program in fig. 1; thus, the difference between two adjacent numbers in the first column of fig. 1 is 0.1 second. However, the value of DT can be specified by using a CON or PAR statement or by an equal statement. (see fig. 7) The above situation also applies to PLO statement; in this case, it is the time interval between two adjacent plotting points.

Now, the logical control variable can be used to give different values of DT during different time periods. For example, one may require more entries or points during the initial period of a transient response, less entries or points as the response approaching a steady state, and a few entries or points as the response arriving at a steady state. As an example, the Mimic program in fig. 1 is rewritten so that DT is 0.1 second when t is less than one second. DT is 1 second when t is equal to or larger than 1 second but less than 10 seconds. And DT is 10 seconds when t is equal to or larger than 10 seconds. The rewritten program is shown in fig. 7 where logical control variables

D1, D2, and D3 control three values of DT. These logical control variables are determined by logical relations among variables C1, C2, and C3 which represent the time at less than 1 second, at less than 10 seconds, and at larger than or equal to 10 seconds respectively. These relations and representations are specified by FSW, AND, and NOT statements as shown in fig. 7. The table output of this program is shown in fig. 8, where there are only 29 yet adequate entries instead of possibly a table with pages of entries.

```
$EXECUTE        IBJOB
$ID   CHU  *305/65/020*3M*
$IBJOB          FIOCS
$IBLDR MIMIC    LIBE
$DATA
***PROBLEM NO.1***SOLUTION OF VAN DER POL,S EQUATION
*
                    CON(B)
                    PAR(A,X0,1DX0)
*
          2DX     = A*1DX-A*1DX*X*X-B*X
          1DX     = INT(2DX,1DX0)
          X       = INT(1DX,X0)
*
                    FIN(T,30.)
*
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
*
                    PLO(T,X)
                    PLO(X,1DX)
*
                    END
1.0
0.2          1.0          0.0
1.5          3.0          0.0
$EXECUTE        IBJOB
$ID   CHU  *305/65/020*3M*
$IBJOB          GO,NOSOURCE,FIOCS
$IBLDR MMPLOT   LIBE
```

Figure 1, Mimic Program for Problem No. 1

A	XO	1DXO
2.00000E-01	1.00000E 00	0.

TIME	X	XDOT
0.	1.00000E 00	0.
1.00000E-01	9.95004E-01	-9.98384E-02
2.00000E-01	9.80063E-01	-1.98748E-01
3.00000E-01	9.55313E-01	-2.95912E-01
4.00000E-01	9.20963E-01	-3.90628E-01
5.00000E-01	8.77289E-01	-4.82289E-01
6.00000E-01	8.24624E-01	-5.70361E-01
7.00000E-01	7.63353E-01	-6.54355E-01
8.00000E-01	6.93905E-01	-7.33794E-01
9.00000E-01	6.16762E-01	-8.08185E-01
1.00000E 00	5.32454E-01	-8.76987E-01
1.10000E 00	4.41571E-01	-9.39594E-01
1.20000E 00	3.44765E-01	-9.95310E-01
1.30000E 00	2.42765E-01	-1.04335E 00
1.40000E 00	1.36381E-01	-1.08283E 00
1.50000E 00	2.65155E-02	-1.11281E 00
1.60000E 00	-8.58305E-02	-1.13228E 00
1.70000E 00	-1.99558E-01	-1.14028E 00
1.80000E 00	-3.13473E-01	-1.13588E 00
1.90000E 00	-4.26296E-01	-1.11833E 00
2.00000E 00	-5.36682E-01	-1.08709E 00
2.10000E 00	-6.43250E-01	-1.04193E 00
2.20000E 00	-7.44609E-01	-9.82985E-01
2.30000E 00	-8.39404E-01	-9.10776E-01
2.40000E 00	-9.26353E-01	-8.26242E-01
2.50000E 00	-1.00428E 00	-7.30585E-01
2.60000E 00	-1.07218E 00	-6.25711E-01
2.70000E 00	-1.12917E 00	-5.13136E-01
2.80000E 00	-1.17461E 00	-3.94876E-01
2.90000E 00	-1.20802E 00	-2.72840E-01
3.00000E 00	-1.22912E 00	-1.48834E-01
3.10000E 00	-1.23778E 00	-2.44825E-02
3.20000E 00	-1.23405E 00	9.88180E-02
3.30000E 00	-1.21809E 00	2.19919E-01
3.40000E 00	-1.19017E 00	3.37918E-01
3.50000E 00	-1.15063E 00	4.52127E-01
3.60000E 00	-1.09989E 00	5.62041E-01
3.70000E 00	-1.03838E 00	6.67278E-01
3.80000E 00	-9.66596E-01	7.67533E-01
3.90000E 00	-8.85048E-01	8.62515E-01
4.00000E 00	-7.94279E-01	9.51898E-01
4.10000E 00	-6.94869E-01	1.03527E 00
4.20000E 00	-5.87444E-01	1.11208E 00
4.30000E 00	-4.72695E-01	1.18162E 00
4.40000E 00	-3.51392E-01	1.24299E 00
4.50000E 00	-2.24406E-01	1.29508E 00
4.60000E 00	-9.27277E-02	1.33661E 00
4.70000E 00	4.25155E-02	1.36613E 00
4.80000E 00	1.80043E-01	1.38214E 00
4.90000E 00	3.18443E-01	1.38313E 00
5.00000E 00	4.56129E-01	1.36775E 00
5.10000E 00	5.91412E-01	1.33494E 00
5.20000E 00	7.22514E-01	1.28407E 00
5.30000E 00	8.47621E-01	1.21507E 00

Fig. 2, Table Output of Problem No. 1

Fig. 3, Time response of van der Pol's equation for μ=0.2

FIGURE 1-1

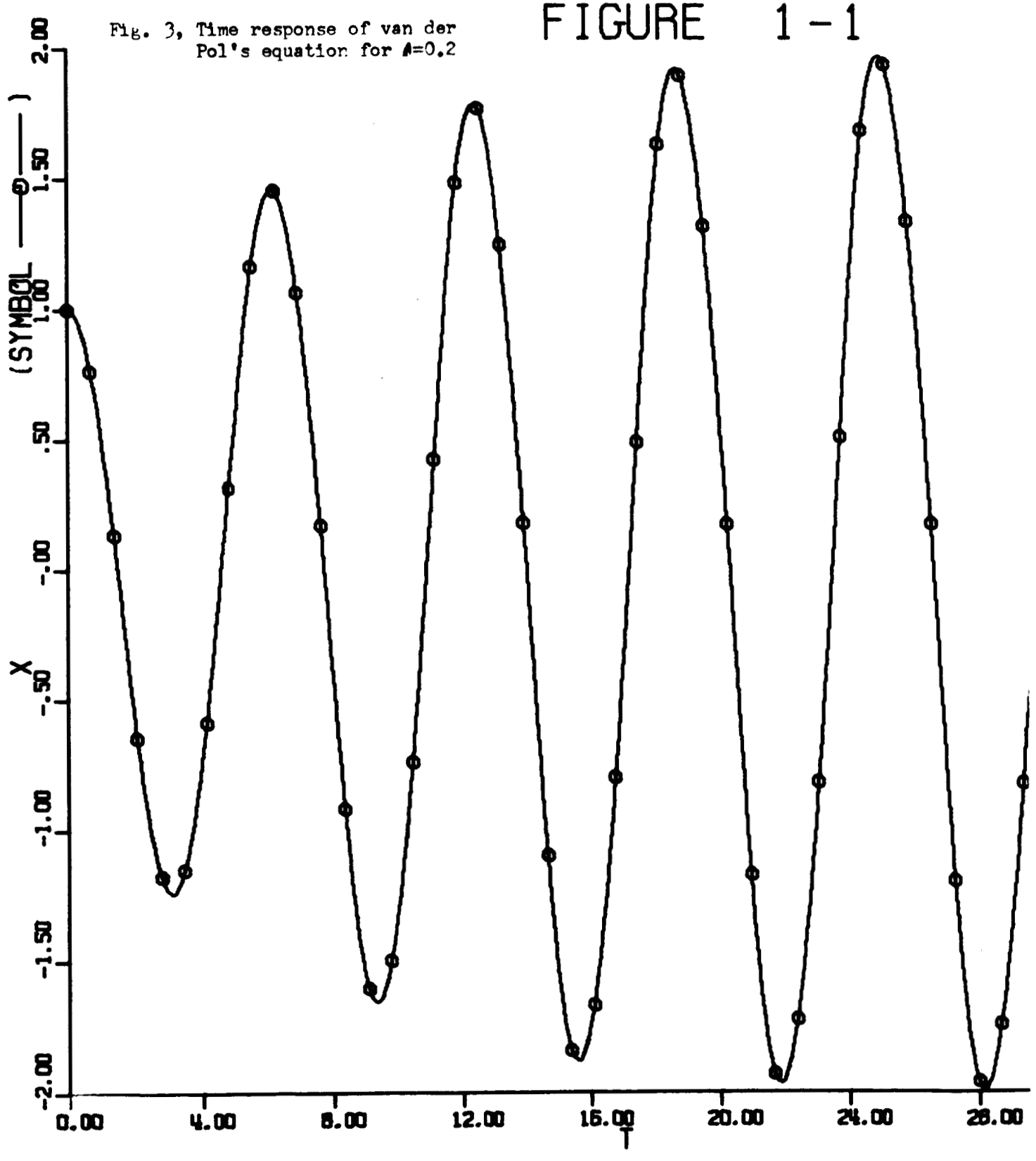
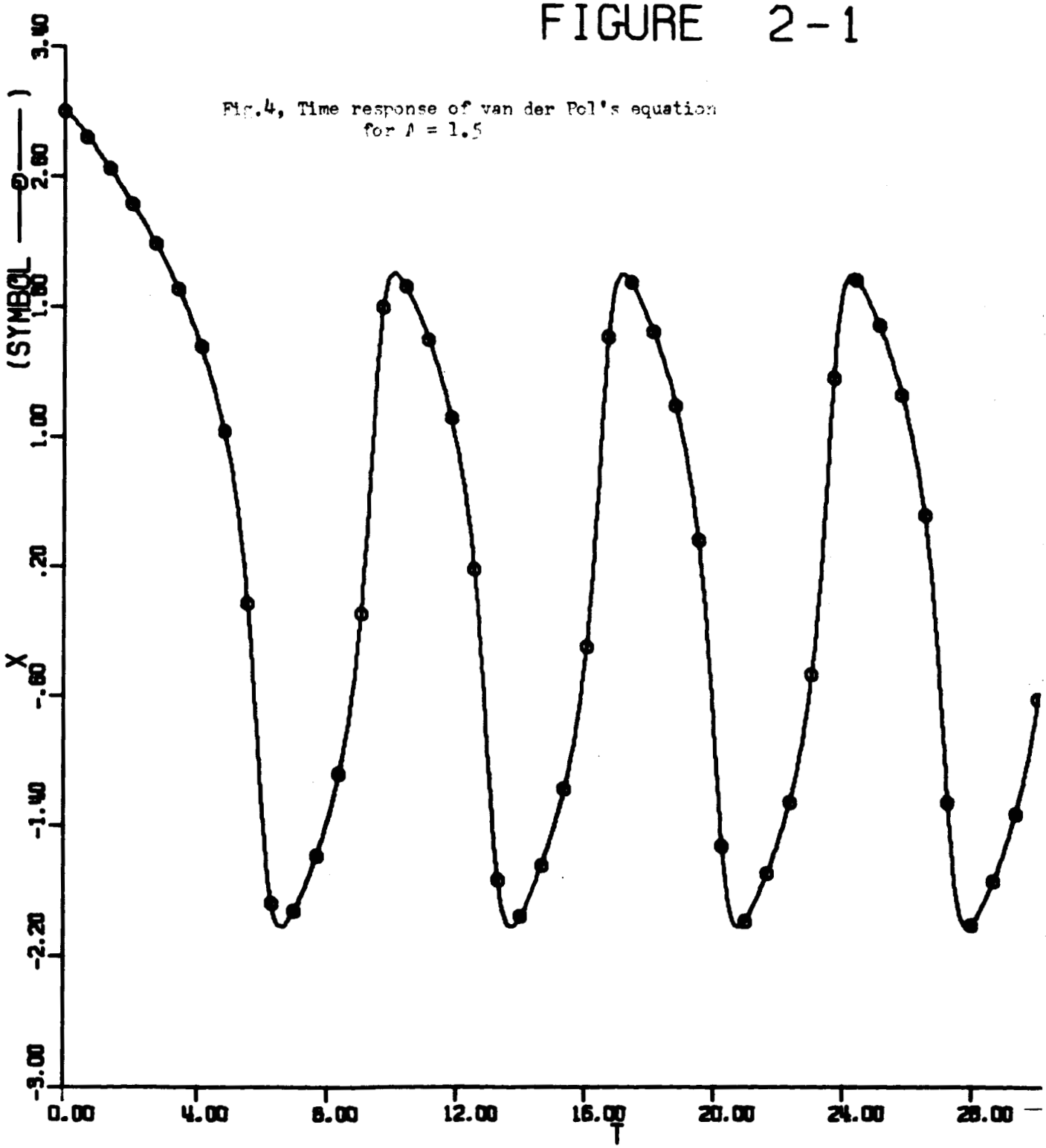


FIGURE 2-1

Fig.4, Time response of van der Pol's equation for $\Lambda = 1.5$



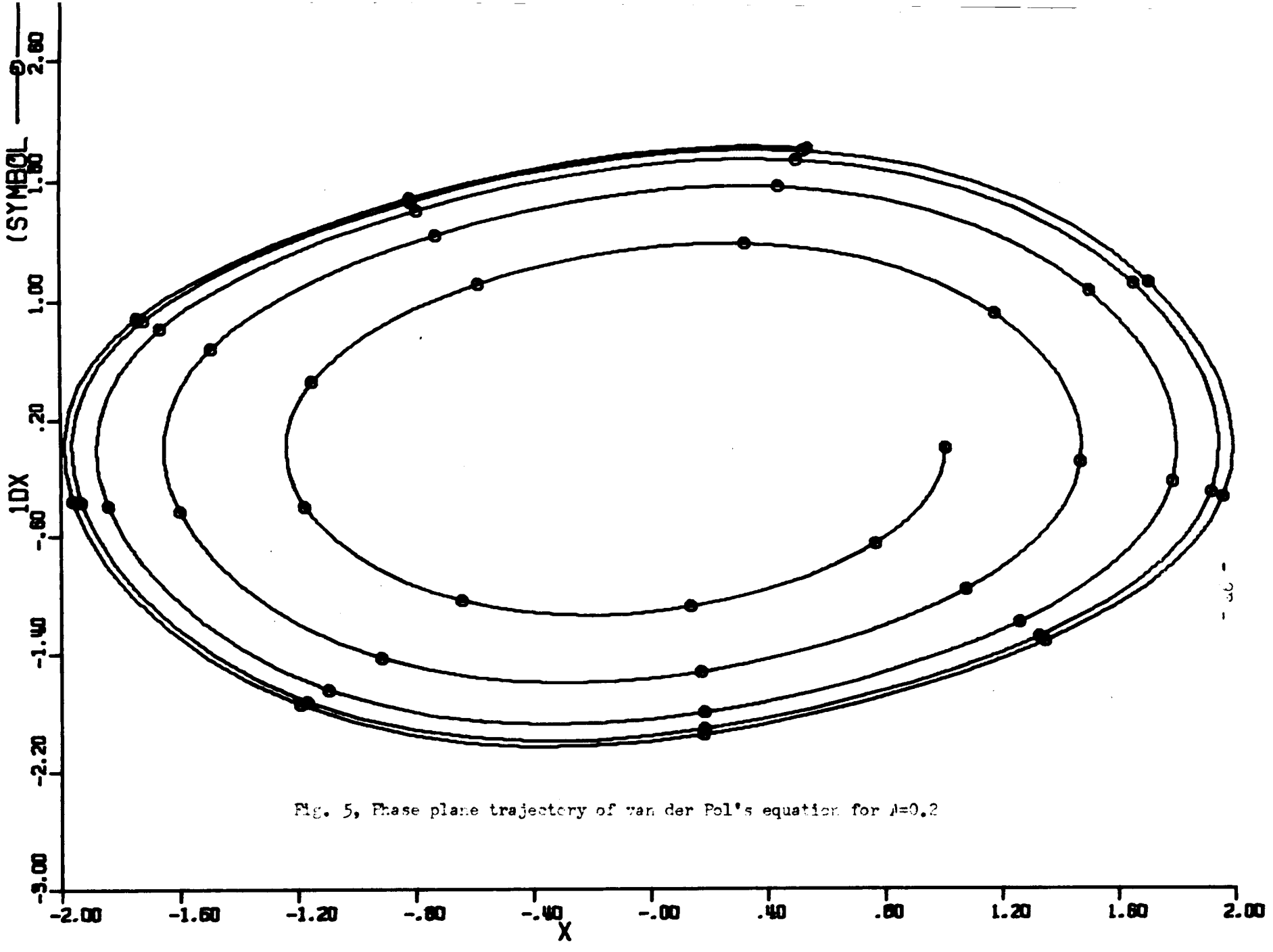


Fig. 5, Phase plane trajectory of van der Pol's equation for μ=0.2

FIGURE 2-2

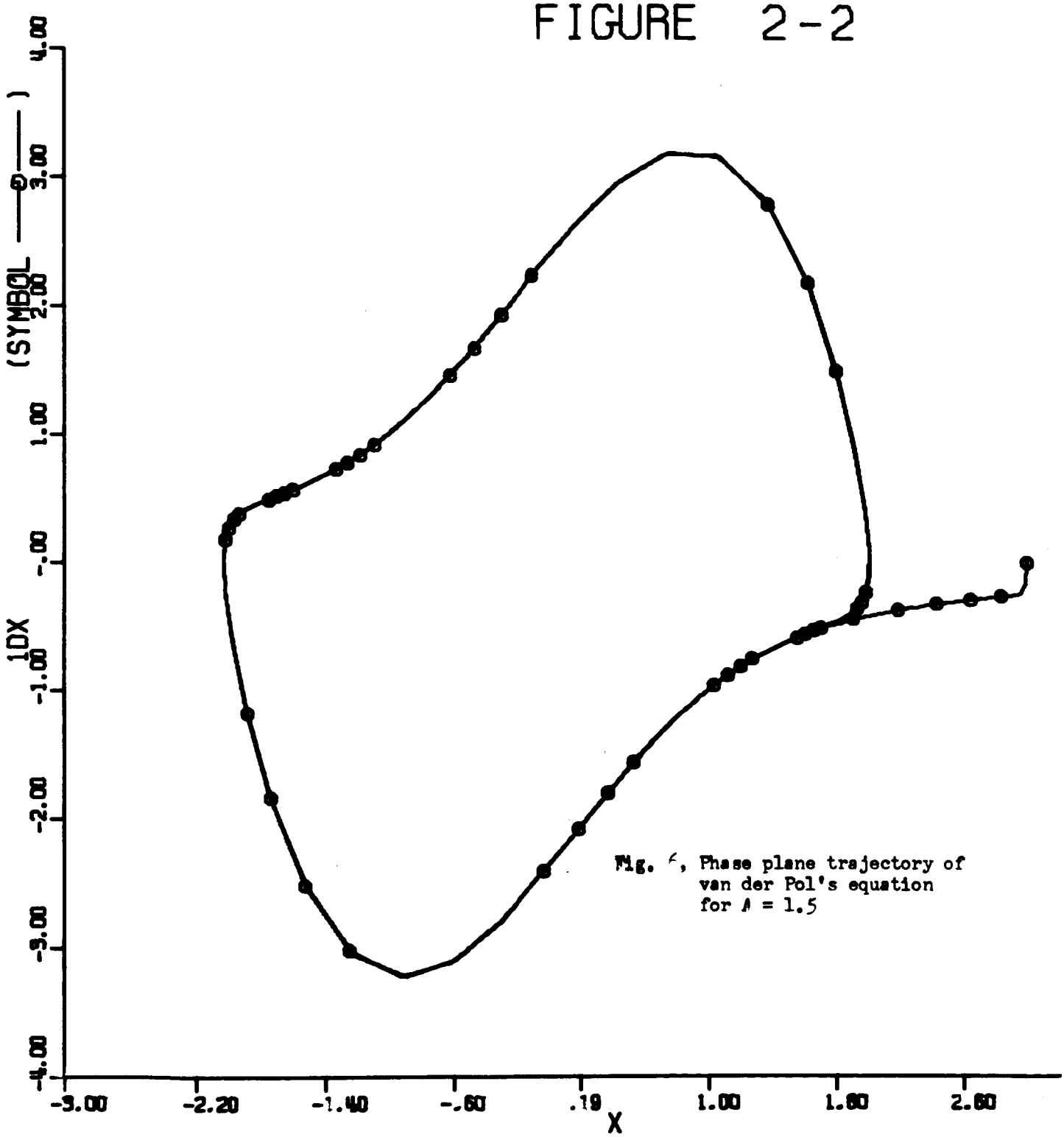


Fig. 6, Phase plane trajectory of van der Pol's equation for $A = 1.5$

```
***PROBLEM NO.1A**SOLUTION OF VAN DER POL,S EQUATION
*
                    CON(B)
                    PAR(A,X0,1DX0)
*
 D1       DT       = .1
 D2       DT       = 1.
 D3       DT       = 10.
*
          D1       = C1
          D2       = AND(C2,NOT(C1))
          D3       = AND(C3,NOT(C1),NOT(C2))
*
          C1       = FSW(T-1., TRUE,FALSE,FALSE)
          C2       = FSW(T-10.,TRUE,FALSE,FALSE)
          C3       = NOT(C2)
*
          2DX      = A*1DX-A*1DX*X*X-B*X
          1DX      = INT(2DX,1DX0)
          X        = INT(1DX,X0)
*
                    FIN(T,100.)
*
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
*
                    END
1.0
0.2          1.0          0.0
1.5          3.0          0.0
```

Figure 7, Mimic Program for Problem No. 1A

A	XO	DXO
2.00000E-01	1.00000E 00	0.

TIME	X	XDOT
0.	1.00000E 00	0.
1.00000E-01	9.95004E-01	-9.98384E-02
2.00000E-01	9.80063E-01	-1.98748E-01
3.00000E-01	9.55313E-01	-2.95912E-01
4.00000E-01	9.20963E-01	-3.90628E-01
5.00000E-01	8.77289E-01	-4.82289E-01
6.00000E-01	8.24624E-01	-5.70361E-01
7.00000E-01	7.63353E-01	-6.54355E-01
8.00000E-01	6.93905E-01	-7.33794E-01
9.00000E-01	6.16762E-01	-8.08185E-01
1.00000E 00	5.32454E-01	-8.76987E-01
2.00000E 00	-5.36682E-01	-1.08709E 00
3.00000E 00	-1.22912E 00	-1.48837E-01
4.00000E 00	-7.94283E-01	9.51898E-01
5.00000E 00	4.56126E-01	1.36776E 00
6.00000E 00	1.42421E 00	3.47235E-01
7.00000E 00	1.07102E 00	-9.70920E-01
8.00000E 00	-2.99046E-01	-1.60171E 00
9.00000E 00	-1.55026E 00	-5.94221E-01
1.00000E 01	-1.33287E 00	9.28230E-01
2.00000E 01	7.25612E-01	-1.69192E 00
3.00000E 01	3.22739E-01	2.03610E 00
4.00000E 01	-1.37422E 00	-1.60221E 00
5.00000E 01	1.95387E 00	4.46530E-01
6.00000E 01	-1.89276E 00	5.08337E-01
7.00000E 01	1.34421E 00	-1.33927E 00
8.00000E 01	-4.49562E-01	1.87068E 00
9.00000E 01	-6.51746E-01	-2.01458E 00
1.00000E 02	1.60941E 00	1.30475E 00

Fig. 8, Table output for Problem No. 1A

3. Antenna Radiation Pattern

The computation of the radiation pattern of an antenna (14) is often a tedious task. This problem is to show that **MIMIC** can perform such a computation and plot radiation pattern just as well as solving a differential equation.

A four-element binomial array with a given element spacings is the chosen antenna array. It is required to compute radiation pattern of this array where the current intensities follow a binomial distribution. The radiation intensity of this array is given by,

$$E(O) = I_1 \exp\left[j2\pi(d_1/\lambda)\sin\theta\right] + I_2 \exp\left[j2\pi(d_2/\lambda)\sin\theta\right] + I_3 \exp\left[j2\pi(d_3/\lambda)\sin\theta\right] + I_4 \exp\left[j2\pi(d_4/\lambda)\sin\theta\right] \quad (2)$$

when λ is the wavelength, θ is the angle between a direction line and the normal to the array plane d's are element spacings from an arbitrary reference and I_1 is the current intensity in the ith element. The real part R_1 of the $E(\theta)$ is,

$$R_1 = I_1C_1 + I_2C_2 + I_3C_3 + I_4C_4 \quad (3)$$

and the imaginary part R_2 is,

$$R_2 = 1_1S_1 + 1_2S_2 + I_3S_3 + I_4S_4 \quad (4)$$

where $C_i = \cos\left[2\pi(d_i/\lambda)\sin\theta\right]$ (5)

and $S_i = \sin\left[2\pi(d_i/\lambda)\sin\theta\right]$ (6)

and the magnitue R of the $E(\theta)$ is,

$$R = \sqrt{R_1^2 + R_2^2} \quad (7)$$

The radiation power PWR in db is,

$$PWR = 20 \log_{10} R \qquad (8)$$

Table 2, Data for computation of antenna radiation pattern

constants,	$I_1 = 1$ $I_2 = 2$ $I_3 = 2$ $I_4 = 1$
parameters,	$d_1/\lambda = 0$ $d_2/\lambda = 1$ $d_3/\lambda = 2$ $d_4/\lambda = 3$
termination,	compute from θ=0 to θ=180^o
plot,	PWR vs θ

A Mimic program for computing the magnitue and power of the radiation by using equations (2) through (8) and the data in Table 2 is shown in fig. 9. Equivalent symbols are shown in Table 3 for those in the program that differ from those in the equations. Notice in the program the use of SIN, COS, SQR, and LOG statements for obtaining sine, cosie, square-root, and logarithm of an argument. Also notice the algebraic expressions which are limited to operators of +, -, *, and / (addition, subtraction, multiplication and division respectively). The radiation pattern is shown in the plot of fig. 10 for an angle from 0 to 180 degrees.

Table 3. Equivalent symbols

Program symbols	equation symbols
Ii	I_i
Di	d_i/λ
Ci	$\cos(2\pi(d_i/\lambda)\sin\theta)$
Si	$\sin(2\pi(d_i/\lambda)\sin\theta)$
THETA	θ in radians
T	θ in degrees
L	$\pi\sin\theta$

```
***PROBLEM NO.2***COMPUTATION OF ANTENNA RADIATION PATTERN
*
                    CON(I1,I2,I3,I4)
                    PAR(D1,D2,D3,D4)
*
          THETA   = T*3.1416/180.0
          L       = 6.2832*SIN(THETA)
*
          C1      = COS(D1*L)
          C2      = COS(D2*L)
          C3      = COS(D3*L)
          C4      = COS(D4*L)
*
          S1      = SIN(D1*L)
          S2      = SIN(D2*L)
          S3      = SIN(D3*L)
          S4      = SIN(D4*L)
*
***REAL PART OF RADIATION INTENSITY
          R1      = (I1*C1+I2*C2+I3*C3+I4*C4)/6.0
*
***IMAGINARY PART OF RADIATION INTENSITY
          R2      = (I1*S1+I2*S2+I3*S3+I4*S4)/6.0
*
***MAGNITUDE OF RADIATION INTENSITY
          R       = SQR(R1*R1+R2*R2)
*
***RADIATION POWER IN DB
          PWR     = 20.0*LOG(R,10.0)
*
                    FIN(T,180.0)
                    HDR(T,R,PWR)
                    HDR
                    OUT(T,R,PWR)
                    PLO(T,PWR)
                    END
1.0          2.0          2.0          1.0
0.0          1.0          2.0          3.0
```

Figure 9, Mimic Program for Problem No. 2

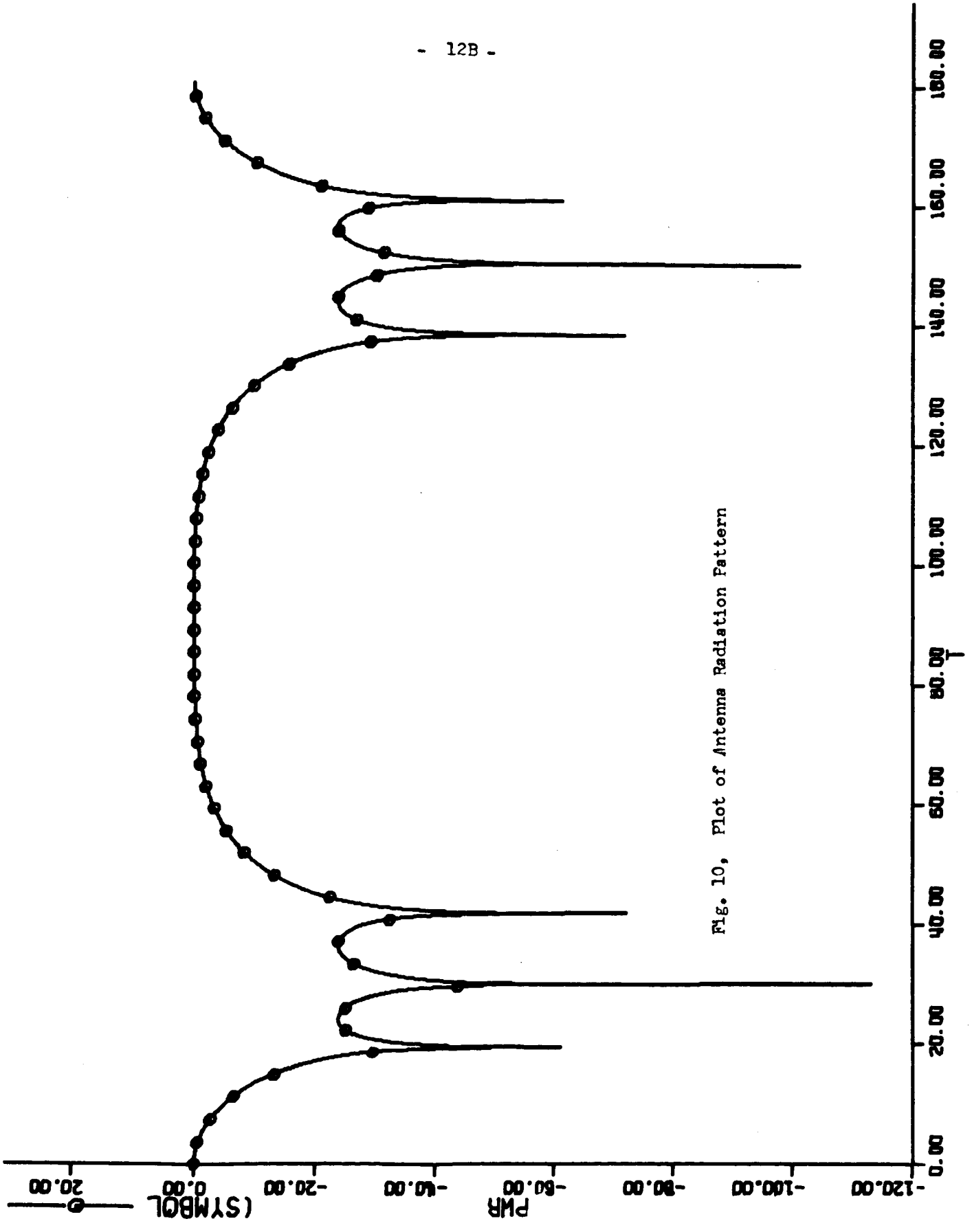


Fig. 10, Plot of Antenna Radiation Pattern

4. Generation of Bessel Functions of the First Kind

It is well known that the Bessel function of the first kind $J_n(t)$ may be defined as a solution of the following second-order deferential equation (15).

$$\ddot{J}_n + \frac{1}{t}\dot{J}_n + \left(1 - \frac{n^2}{t^2}\right) J_n = 0 \qquad (9)$$

where n is the order of Bessel function. The solution of this equation is a well-known series with infinite terms. This series can give the initial conditions of Bessel functions of nth and lower orders which are listed below,

$$\begin{aligned} &J_0(0) = 1, && \dot{J}_0(0) = 0 \\ &J_1(0) = 0, && \dot{J}_1(0) = 0.5 \\ &J_2(0) = J_3(0) = \ldots = 0, && \dot{J}_2(0) = \dot{J}_3(0) = \ldots = 0 \end{aligned} \qquad (10)$$

In the case of n equal to 0, it is possible to generate $J_0(t)$ by solving equation (9) on an analog computer (16,17). For analog generation of Bessel functions of first and higher orders, it is at best inaccurate if impossible because division of the independent variable t at t equal to 0 is required in equation (9). Since initial conditions of (n+1) derivatives of Bessel function of nth order are available, it has been shown (18) that, if equation 9 is deferentiated n times, solution of the resulting (n+2)th order deferential equation can generate Bessel function of nth or lower order with the use of the available (n+2) initial conditions.

```
***PROBLEM NO.3***GENERATION OF BESSEL FUNCTIONS OF THE FIRST KIND
*
          J0      = INT(-J1,1.0)
          J1      = INT(J0-K0,0.)
          J2      = INT(J1-K1,0.)
          J3      = INT(J2-K2,0.)
          J4      = INT(J3-K3,0.)
*
          K0      = J1/T
          K1      = 2.*J2/T
          K2      = 3.*J3/T
          K3      = 4.*J4/T
*
                    FIN(T,16.)
                    HDR(TIME,J0,J1,J2,J3,J4)
                    HDR
                    OUT(T,J0,J1,J2,J3,J4)
                    OUT
                    PLO(T,J0,J1,J2,J3,J4,)
                    END
```

Fig. 11, Mimic Program for Problem No. 3

Another possible and simple method is based on the use of the following known recurrence realtion (17),

$$\dot{J}_n(t) = J_{n-1}(t) - \frac{n}{t} J_n(t) \qquad (11)$$

Which permits generation of J_n if J_{n-1} is available. For example, if n is equal to 4, the following set of equations can be obtained from equation (11):

$$\begin{aligned}
\dot{J}_0(t) &= -J_1(t) \\
\dot{J}_1(t) &= J_0(t) - \frac{1}{t} J_1(t) \\
\dot{J}_2(t) &= J_1(t) - \frac{2}{t} J_2(t) \\
\dot{J}_3(t) &= J_2(t) - \frac{3}{t} J_3(t) \\
\dot{J}_4(t) &= J_3(t) - \frac{4}{t} J_4(t)
\end{aligned} \qquad (12)$$

Solution of this set of first-order deferential equations can give Bessel functions of 4th and lower orders.

A Mimic program which solves this set of equations is shown in fig. 11 by using initial conditions (10). The problem of dividing zero by zero in equations (12) at t equal to 0 is handled by the compiler as/initial conditions which are specified in the INT statements are used for the first iteration. A plot showing Bessel functions of the first kind for n equal to 0 through 4 is shown in fig. 12. Notice that all J's have the initial values of zero except J_0 which has an initial value of 1.

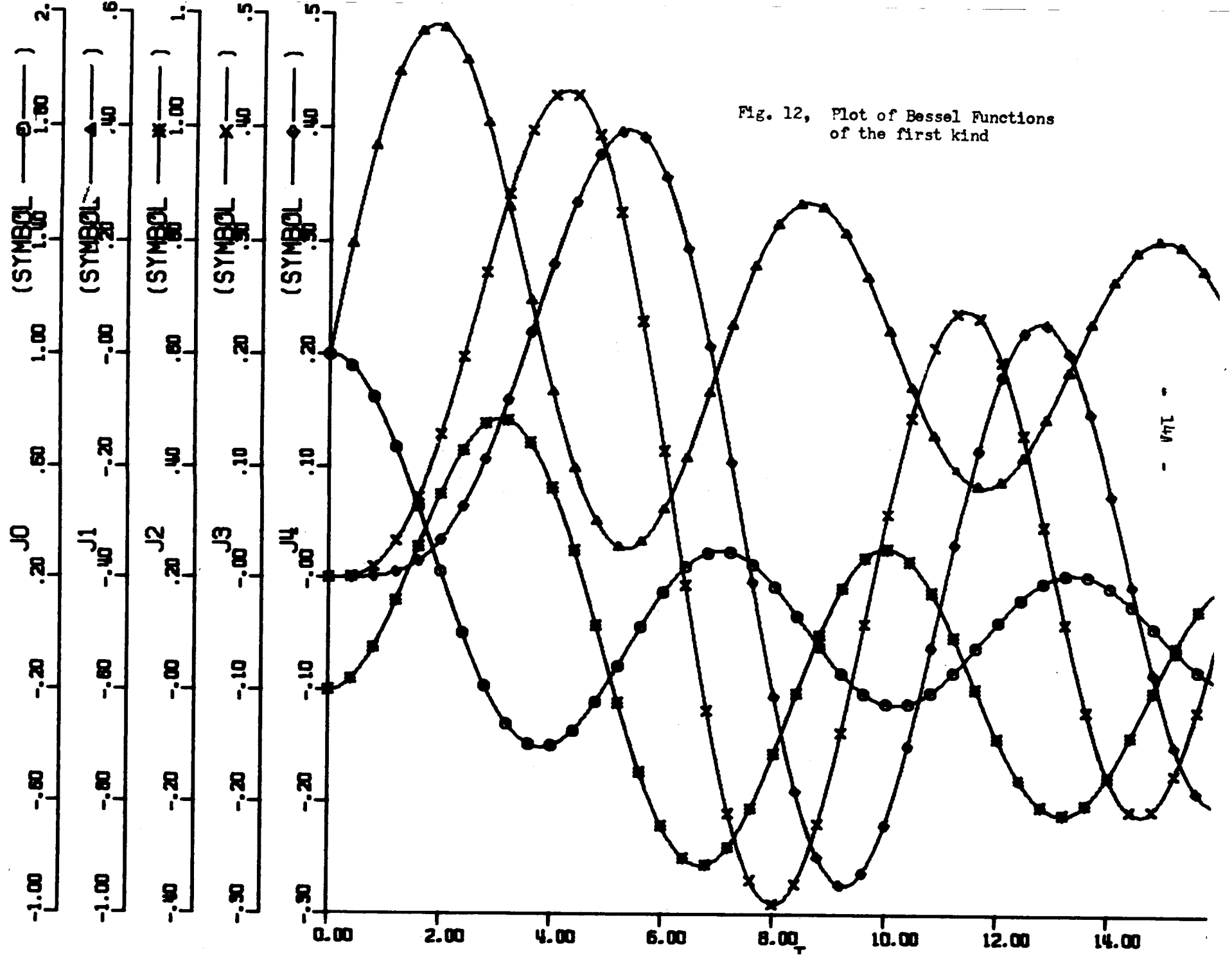


Fig. 12, Plot of Bessel Functions of the first kind

5. One-dimensional Heat Flow Equation

Analog computer has been used to solve partial differential equation. Consider the one-dimensional heat flow in a long, thin, uniform rod which is insulated on all sides except one end (say, left end of the rod). As shown elsewhere (18-21), the temperature distribution, as a function of time t, of the rod can be described by the following one-dimensional partial differential equation which is parabolic,

$$\frac{\partial T}{\partial t} = \frac{k}{c \cdot p} \frac{\partial^2 T}{\partial x^2} \qquad (13)$$

where x is the distance measured from the left end, T the temperature, k the thermal conductivity, c the specific heat, and p the density of the rod. Both x and T are independent variables.

Since both x and T are the independent variables, it is not possible to simulate the partial differential equation exactly on an analog computer as time is the only independent variable available in an analog computer. By using finite difference to approximate the partial derivative in x, equation (13) can be changed into the following system of first-order ordinary differential equations. Let the rod of length L be divided into 20 equal increments of length Δx. We then have,

$$\begin{aligned}
T_0 &= f(t) \\
dT_1/dt &= \mathcal{Z}(T_2 - 2T_1 + T_0) \\
dT_2/dt &= \mathcal{Z}(T_3 - 2T_2 + T_1) \\
&\text{---------- (14)} \\
dT_{19}/dt &= \mathcal{Z}(T_{20} - 2T_{19} + T_{18}) \\
dT_{20}/dt &= \mathcal{Z}(2T_{19} - 2T_{20})
\end{aligned}$$

where $\mathcal{Z} = k/cp(\Delta x)^2$ (15)

and $\Delta x = L/20$ (16)

Function f(t) is the temperature applied at the uninsulated end of the rod, and T_i's are the temperatures at the ends of each increment. The above set of ordinary differential equations provides an approximate solution when they are solved simultaneously. And this solution describes the temperature as a function of time at finite increments of distance along the rod.

Table 4. Data for one-dimensional heat flow simulation

Qunatity	Value
L	1 ft.
k	2.4 btu/min-ft-^{o}F
c	0.2 btu/lb-^{o}F
p	150 lb/ft^3
$T_i(0)$	0^{o}F
f(t)	120^{o}F

To simulate the heat flow in the rod, we assume that the rod is initially at 0^{o}F and a heat source at 120^{o}F is applied at t equal to 0 to its left end. Chosen values of other constants are shown in Table 4. The Mimic program is shown in fig. 13 which consists essentially of the 20 INT statements. Fig. 14 is the transient response where the temperature at the ends of 5th increment, 10th increment, 15th increment, and the right end of the rod are shown.

```
***PROBLEM NO.4***ONE-DIMENSIONAL HEAT FLOW
*
          L       = 1.
          K       = 2.4
          C       = 0.2
          P       = 150.
          DELTAX  = 1./20.
          Z       = K/(C*P*DELTAX*DELTAX)
          IC      = 0.
*
          T0      = 120.
          T1      = INT(Z*(T2 -2.*T1 +T0 ),IC)
          T2      = INT(Z*(T3 -2.*T2 +T1 ),IC)
          T3      = INT(Z*(T4 -2.*T3 +T2 ),IC)
          T4      = INT(Z*(T5 -2.*T4 +T3 ),IC)
          T5      = INT(Z*(T6 -2.*T5 +T4 ),IC)
*
          T6      = INT(Z*(T7 -2.*T6 +T5 ),IC)
          T7      = INT(Z*(T8 -2.*T7 +T6 ),IC)
          T8      = INT(Z*(T9 -2.*T8 +T7 ),IC)
          T9      = INT(Z*(T10-2.*T9 +T8 ),IC)
          T10     = INT(Z*(T11-2.*T10+T9 ),IC)
*
          T11     = INT(Z*(T12-2.*T11+T10),IC)
          T12     = INT(Z*(T13-2.*T12+T11),IC)
          T13     = INT(Z*(T14-2.*T13+T12),IC)
          T14     = INT(Z*(T15-2.*T14+T13),IC)
          T15     = INT(Z*(T16-2.*T15+T14),IC)
*
          T16     = INT(Z*(T17-2.*T16+T15),IC)
          T17     = INT(Z*(T18-2.*T17+T16),IC)
          T18     = INT(Z*(T19-2.*T18+T17),IC)
          T19     = INT(Z*(T20-2.*T19+T18),IC)
          T20     = INT(Z*(2.*T19-2.0*T20),IC)
*
                    FIN(T,20.)
*
                    HDR(TIME,T0,T5,T10,T15,T20)
                    HDR
                    OUT(T,T0,T5,T10,T15,T20)
                    PLO(T,T0,T5,T10,T15,T20)
*
                    END
```

Fig. 13, Mimic Program for Problem No. 4

FIGURE

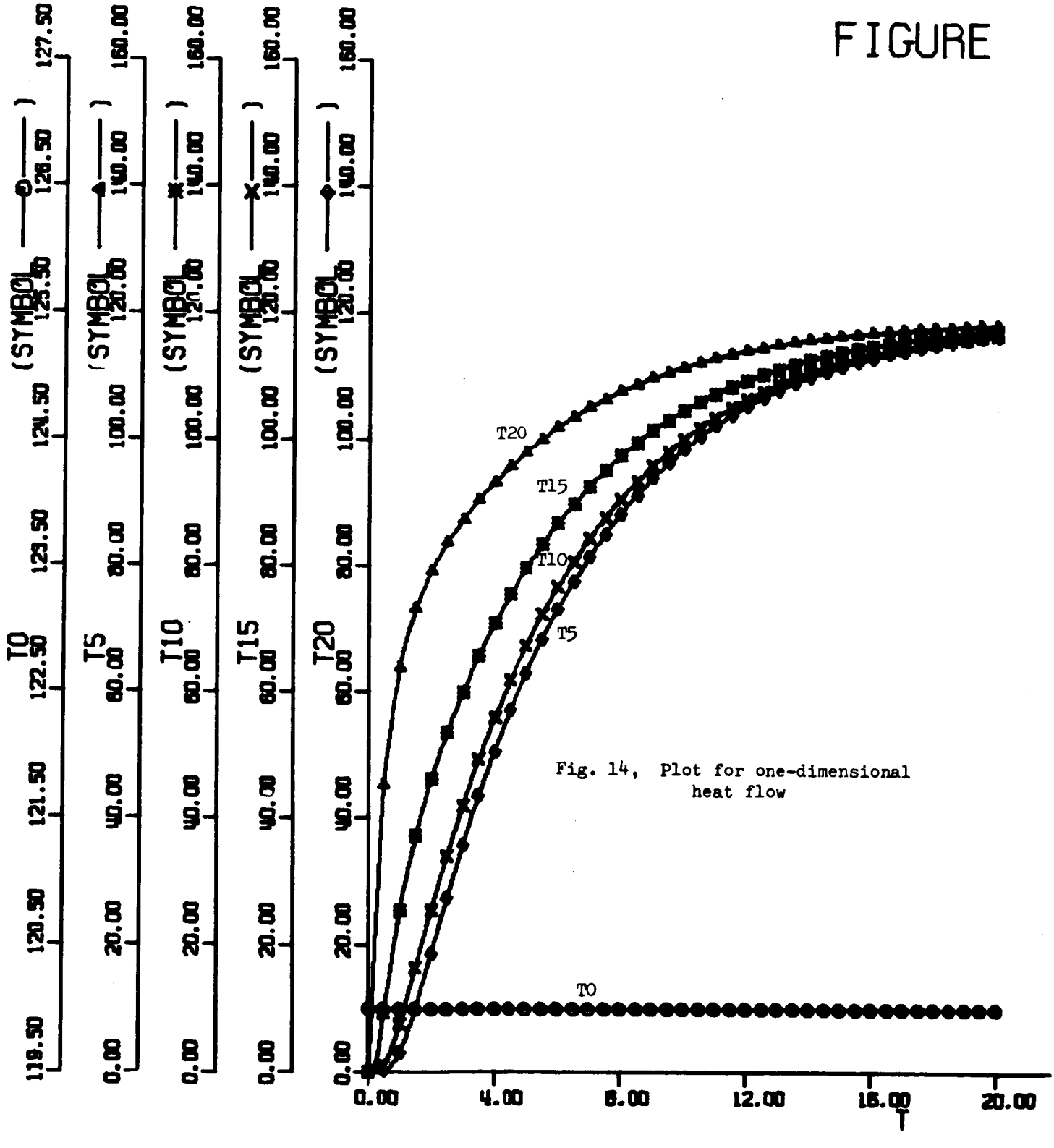


Fig. 14, Plot for one-dimensional heat flow

6. Double integration of a function

As mentioned, there exists only one independent variable in an electronic analog computer. Double integration of a function implies that there exist two independent variables. For example, the volume of a sphere with a unity radius, V, can be expressed as below,

$$V = \int_{-1}^{1} \int_{0}^{1-x^2} 2\pi y \, dy \, dx \qquad (17)$$

where x and y are the distance along X and Y coordinates whose origin is located at the center of the sphere. This integration can be analytically evaluated and the exact volume of the sphere is $4\pi/3$.

The above expression can be rewritten into,

$$V = 4\pi \int_{0}^{1} Y_i \, dx \qquad (18)$$

where

$$Y_i = \int_{0}^{\sqrt{1-x_i^2}} y \, dy \qquad (19)$$

Since both x and y are the independent variables, it is not possible to perform both integrations on an analog computer. An alternative method has been reported (22). Another alternative is to have one of the two integrations performed by a numerical method; this is the approach taken here.

Expression (18) is to be integrated numerically. Let the radius along the X axis be divided into six equal increments. Let h be the increment, which is then equal to 1/6. Various numerical methods of integration are available.

Table 5. Numerical integration formulas and results

Method	Integration formula	Result
Exact	$V = 4\pi/3$	4.18880
Euler	$V = (y_o+y_1+y_2+y_3+y_4+y_5)(4\pi h)$	4.68316
Simpson	$V = (y_o 4y_1+2y_2+4y_3+2y_4+4y_5+y_6)(4\pi h/3)$	4.18858
Weddle	$V = (y_o+5y_1+y_2+6y_3+y_4+5y_5+y_6)(4\pi)(3h/10)$	4.18858

Three integration methods (23-25) are selected and their formulas are shown in Table 5. In evaluating expression (17), the values of x_i must be known, and then the integration limits y_i are computed and shown below,

$$
\begin{aligned}
x_0 &= 0h = 0, & y_0 &= \sqrt{1-x_0^2} = 1 \\
x_1 &= 1h = 1/6, & y_1 &= \sqrt{35}/6 \\
x_2 &= 2h = 2/6, & y_2 &= \sqrt{32}/6 \\
x_3 &= 3h = 3/6, & y_3 &= \sqrt{27}/6 \\
x_4 &= 4h = 4/6, & y_4 &= \sqrt{20}/6 \\
x_5 &= 5h = 5/6, & y_5 &= \sqrt{11}/6
\end{aligned}
\qquad (20)
$$

The Mimic program for evaluating the expressions (18) and (19) is shown in fig. 15 when T represents variable y. Notice that the integration Y_i's are terminated by making the integrands DY_i equal to 0 at the proper terminating time T_i by means of FSW statements. Logical control variable S is to make the computation of volume done only when variable T is at the terminating value of unity, as there is no need to compute the volume at any other time. The

results are also shown in Table 5. Notice that Simpson and Weddle integration formulas give a result accurate to five digits. And better accuracy can be readily obtained by dividing the radius into more increments.

```
***PROBLEM NO.5***DOUBLE INTEGRATION OF A FUNCTION
*
          DT       = .01
*
          Y0       = INT(DY0,0.)
          Y1       = INT(DY1,0.)
          Y2       = INT(DY2,0.)
          Y3       = INT(DY3,0.)
          Y4       = INT(DY4,0.)
          Y5       = INT(DY5,0.)
          Y6       = 0.
*
***FUNCTION SWITCHES DYI AND S
          DY0      = FSW(T-T0,T,T,0.)
          DY1      = FSW(T-T1,T,T,0.)
          DY2      = FSW(T-T2,T,T,0.)
          DY3      = FSW(T-T3,T,T,0.)
          DY4      = FSW(T-T4,T,T,0.)
          DY5      = FSW(T-T5,T,T,0.)
          S        = FSW(T-.99999,FALSE,TRUE,TRUE)
*
***TIME TO TERMINATE THE INTEGRATORS
          T0       = 1.
          T1       = SQR(35.)/6.
          T2       = SQR(32.)/6.
          T3       = SQR(27.)/6.
          T4       = SQR(20.)/6.
          T5       = SQR(11.)/6.
*
***EXACT VALUE
 S        VEXACT   = 3.1416*4./3.
*
***USE EULER'S RULE FOR N = 6
 S        X        = (Y0+Y1+Y2+Y3+Y4+Y5+Y6)*4.*3.1416/6.
*
***USE SIMPSON'S RULE FOR N = 6
 S        V1       = 4.*(Y1+Y3+Y5)
 S        V2       = 2.*(Y2+Y4)
 S        V        = (Y0+V1+V2+Y6)*4.*3.1416/18.
*
***USE WEDDLE'S RULE FOR N = 6
 S        W1       = 5.*(Y1+Y5)
 S        W2       = Y0+Y2+Y4+Y6
 S        W        = (W2+W1+6.*Y3)*4.*3.1416/20.
***TERMINATION
                   FIN(T,1.)
***OUTPUT
                   HDR(T,VEXACT,X,V,W)
                   HDR
 S                 OUT(T,VEXACT,X,V,W)
                   END
```

Fig. 15, Mimic Program for Problem No. 5

7. Simulation of Relay Servos

A relay servo (26-28) is a feedback control system in which the corrective power to the motor or load is applied discontinuously. Because the corrective power is operated at full power, the relay servo responses rapidly. The simplicity and economy of a relay servo is attractive, provided that the requirements in static accuracy, stability, and transient performance can be met.

The characteristic of the relay is very important in determining the stability and accuracy of the servo. Four most common types of relay characteristics are shown in figure 16. Note that, in these characteristics, there are no more than three corrective levels: a positive maximum, zero, and a negative maximum.

Four relay servos are simulated to illustrate the technique in simulating nonlinear control system. Each of these four servos is a linear, second-order servo, but each is incorporated with one of the four types of relays. For comparison purpose, the linear servo is also simulated. These five cases are listed in Table 6. The block diagram of the relay servos

Table 6 Five Simulated Servos

Case	Description	Mimic Program	Transient Response	Phase Trajectory
6(a)	linear second-order servo	fig. 18	fig. 19	fig. 20
6(b)	servo with a simple relay	fig. 21	fig. 22	fig. 23
6(c)	servo with a dead space in the relay	fig. 24	fig. 25	fig. 26
6(d)	servo with hysteresis in the relay	fig. 27	fig. 28	fig. 29
6(e)	servo with both dead space and hysteresis in the relay	fig. 30	fig. 31	fig. 32

Table 7. Values of Constants and Parameters for Simulation

Case	Constants					Parameters	
	A	B	V	D	H	IDXO	XO
6(a)	2	.5	-	-	-	0	1.5
6(b)	2	.5	1	-	-	0	1.5
6(c)	2	.5	1	.2	-	0	1.5
6(d)	2	.5	1	.2	-	0	1.5
6(e)	2	.5	1	.2	.1	0	1.5

is shown in fig. 17. Two relations are apparent from the block diagram.

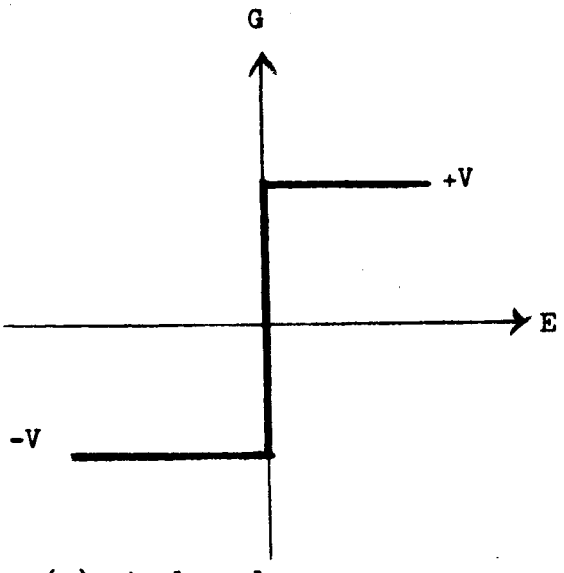
$$E = Y - X \tag{21}$$

$$\frac{X}{G} = \frac{A}{A(BS+1)} \tag{22}$$

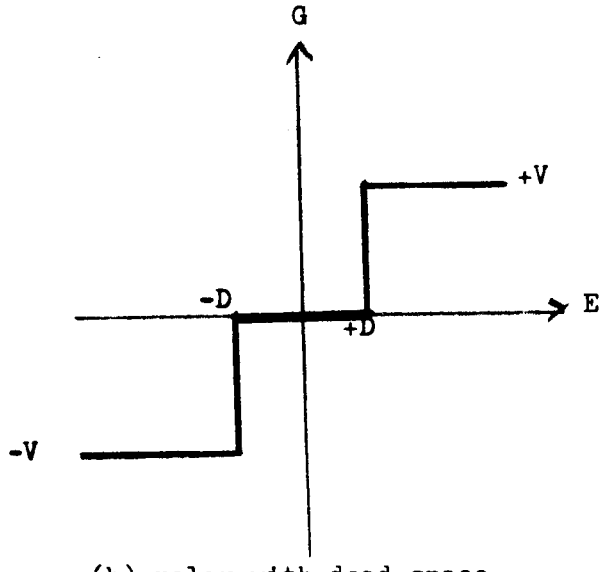
where X and Y are respectively the input and output of the servo, and E (error signal) and G are respectively the input and output of the relay. A is the gain of the loop, without the relay, and B the time constant of the motor. The above transfer function X/G can be rewritten into,

$$S^2X = - SX/B + G \cdot A/B \tag{23}$$

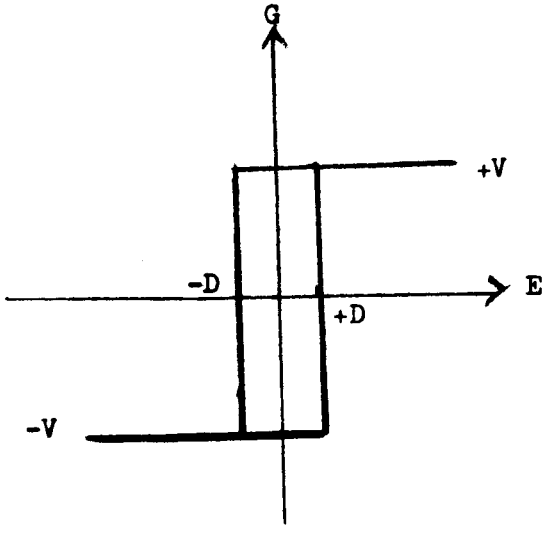
For each of these cases, input Y is taken to be zero, but output X is initially displace; thus, X is equal to -E. The transient response of the output of these servos and the phase trajectories are plotted. The values of the constants and parameters for five cases are chosen and listed in Table 7. where constants A and B are defined in equation (22) and constants V, D, and H are



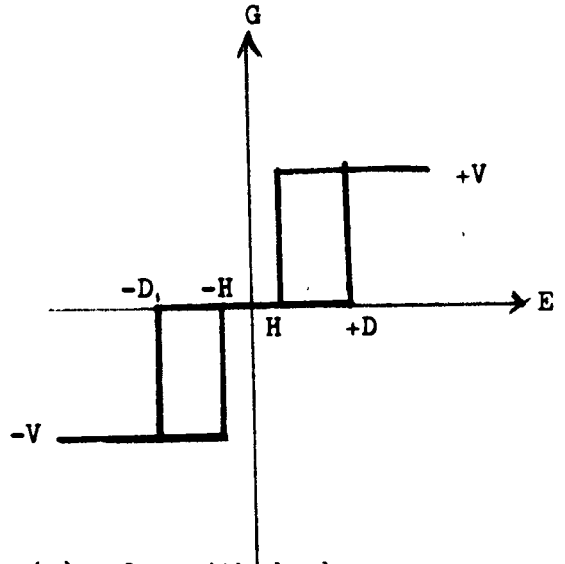
(a) simple relay



(b) relay with dead space



(c) relay with hysteresis



(d) relay with dead space and hysteresis

Fig. 16, Characteristics of four relays

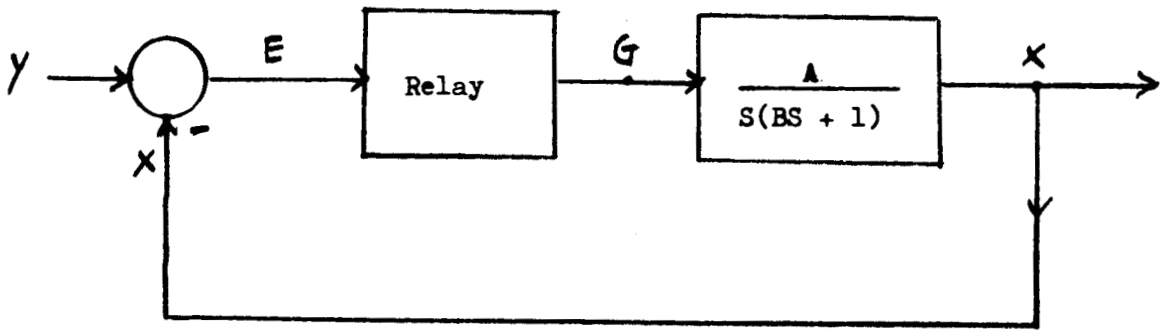


Fig. 12, Bloack diagram of the relay servo

defined in fig. 17. Simulation is terminated when time reaches five seconds.

(a) Linear servo

The linear, second-order servo is represented by the above equations (21) and (23) in addition to the relation G=E. This relation, however, is redundant, but it enables one to use the same variable names for all five cases.

The Mimic program for the linear servo is shown in fig. 18. The transient or error response is shown in fig. 19 where it exhibits the commonly-desired, slightly-underdamped response. The phase trajectory is shown in fig. 20. It is a continuous curve, which begins at X equal to 1.5 and $\dot{X}$ equal to 0. Section AB of the trajectory shows the overshoot portion and BO the undershoot portion. The trajectory approaches the origin exponentially in time.

```
*
***PROBLEM NO. 6A***LINEAR SERVO SIMULATION
*
                    CON(A,B)
                    PAR(1DX0,X0)
*
          DT        = 0.01
*
          2DX       = -1DX/B+G*A/B
          1DX       = INT(2DX,1DX0)
          X         = INT(1DX,X0)
          E         = Y-X
          Y         = 0.0
*
          G         = E
*
                    FIN(T,5.0)
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
                    PLO(T,X)
                    PLO(X,1DX)
                    END
2.0           0.5
0.0           1.5
```

Figure 18, Mimic Program for Problem No.6A

FIGURE 1-1

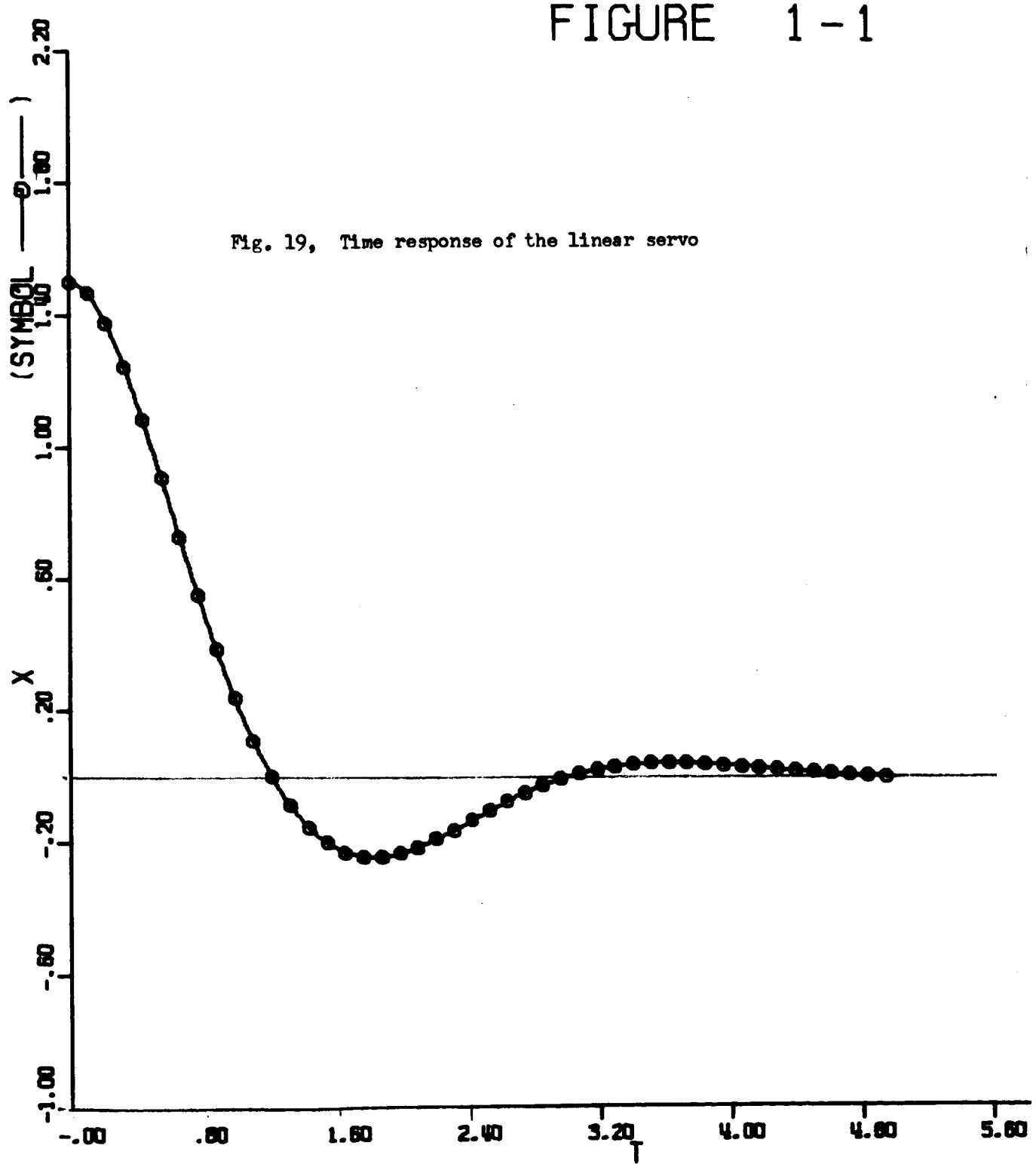
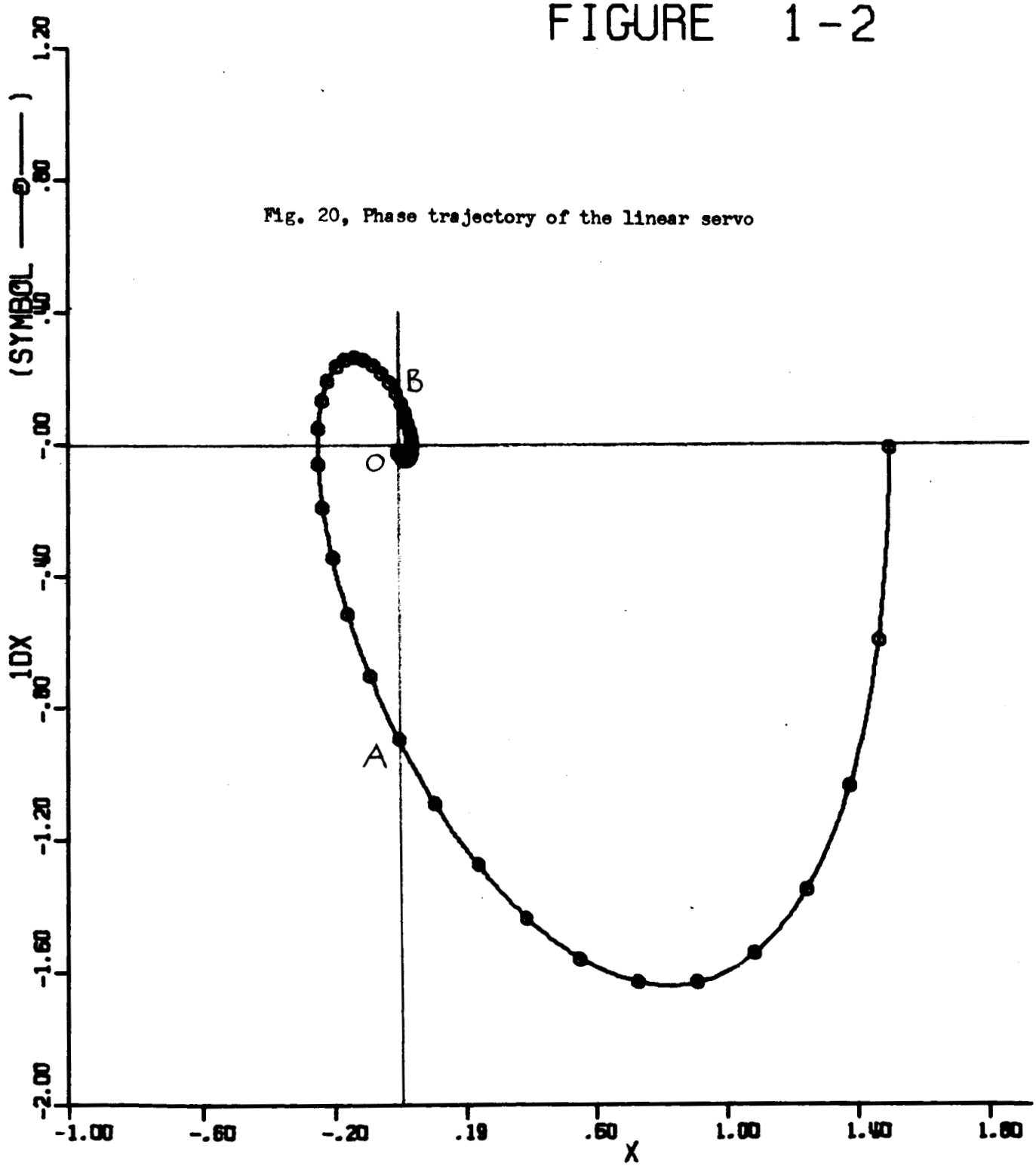


Fig. 19, Time response of the linear servo

FIGURE 1-2

Fig. 20, Phase trajectory of the linear servo



(b) Simple relay servo

The simple relay servo is represented by equations (21) and (23) and the bang-bang characteristic shown in fig. 6(a), where exhibitis three power levels: +V, 0, and -V. This characteristic can be stated as below,

$$G = +V \quad \text{when} \quad E > 0$$

$$G = 0 \quad \text{when} \quad E = 0$$

and $$G = -V \quad \text{when} \quad E < 0$$

Condition E>0 can be expressed by using logical control variable E1 determined by the following FSW statement,

E1 = FSW(E,FALSE,FALSE,TRUE)

Which states that E1 is true whenever E>0; otherwise E1 is false. Similarly, condition E<0 can be expressed by using E2 determined by the following statement,

E2 = FSW(E,TRUE,FALSE,FALSE)

Condition E=0 is equivalent to that when both E1 and E2 are false; this can be expressed by using E3 determined by the following nesting statement,

E3 = NOT(TOR(E1,E2))

Where NOT and IOR denote logical-not and logical-or operations respectively. The above results in six statements which are shown in the Mimic program for itus servo in fig. 21.

The transient or error response is shown in fig. 22, and the phase trajectory in fig. 23. Since the corrective power reverses

instantaneously as the error signal goes through zero, the output X oscillates about the horizontal axis as shown in fig. 22. The oscillation ultimately damps to the origin of fig. 23, because of positive damping by the motor. The vertical axis in fig. 23 is the line where the switching of the relay occurs; this line divides the phase plane into a positive- and a negative- torque regions.

```
*
***PROBLEM NO.6B***SIMULATION OF A SIMPLE RELAY SERVO
*
                    CON(A,B,V)
                    PAR(1DX0,X0)
*
          DT        = 0.01
*
          2DX       = -1DX/B+G*A/B
          1DX       = INT(2DX,1DX0)
          X         = INT(1DX,X0)
          E         = Y-X
          Y         = 0.0
*
 E1       G         = V
 E3       G         = 0.
 E2       G         = -V
*
          E1        = FSW(E,FALSE,FALSE,TRUE)
          E2        = FSW(E,TRUE,FALSE,FALSE)
          E3        = NOT(IOR(E1,E2))
*
                    FIN(T,5.0)
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
                    PLO(T,X)
                    PLO(X,1DX)
                    END
2.0           0.5           1.0
0.0           1.5
```

Figure 21. Mimic Program for Problem No.6B

FIGURE 1-1

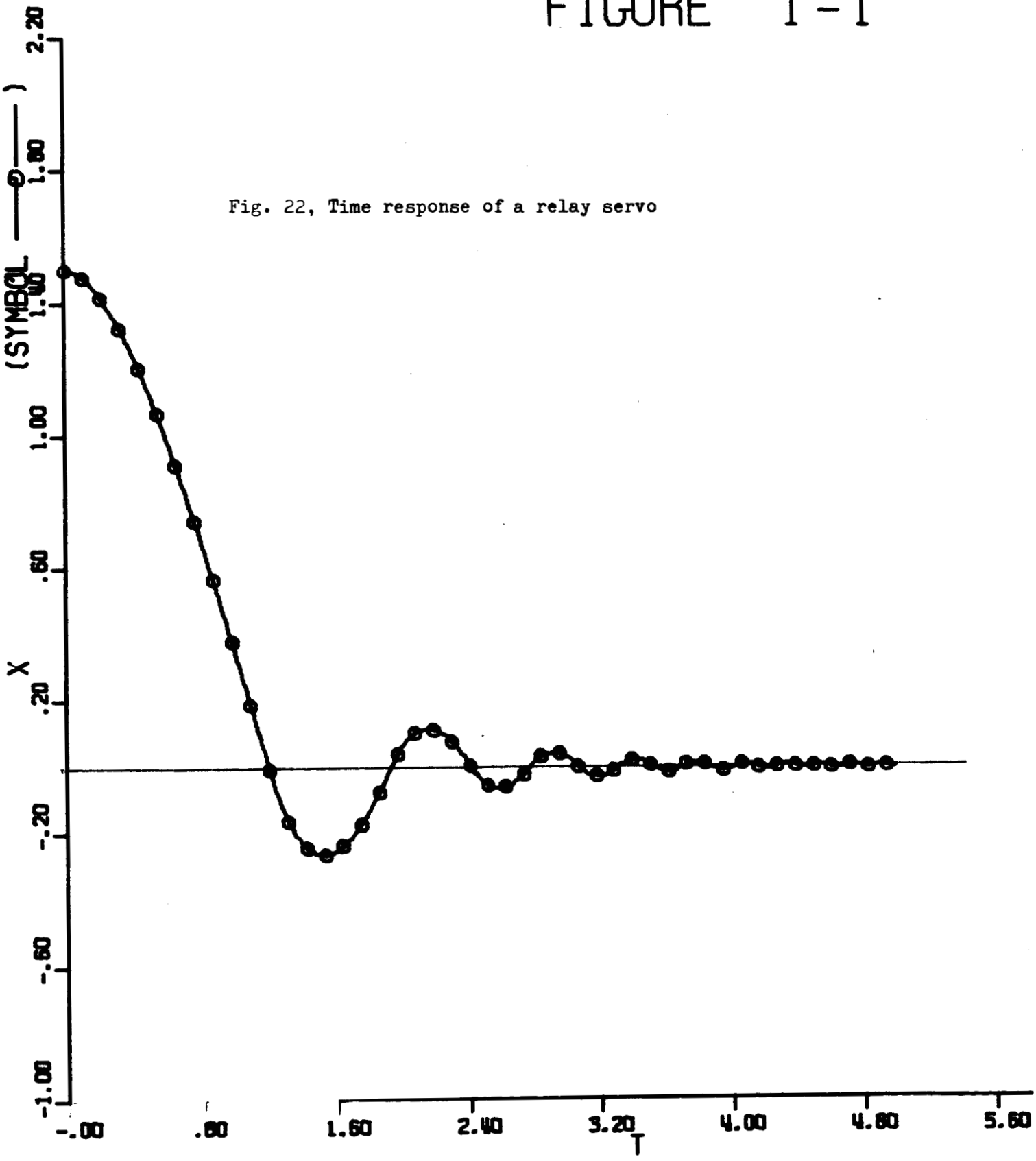


Fig. 22, Time response of a relay servo

FIGURE 1-2

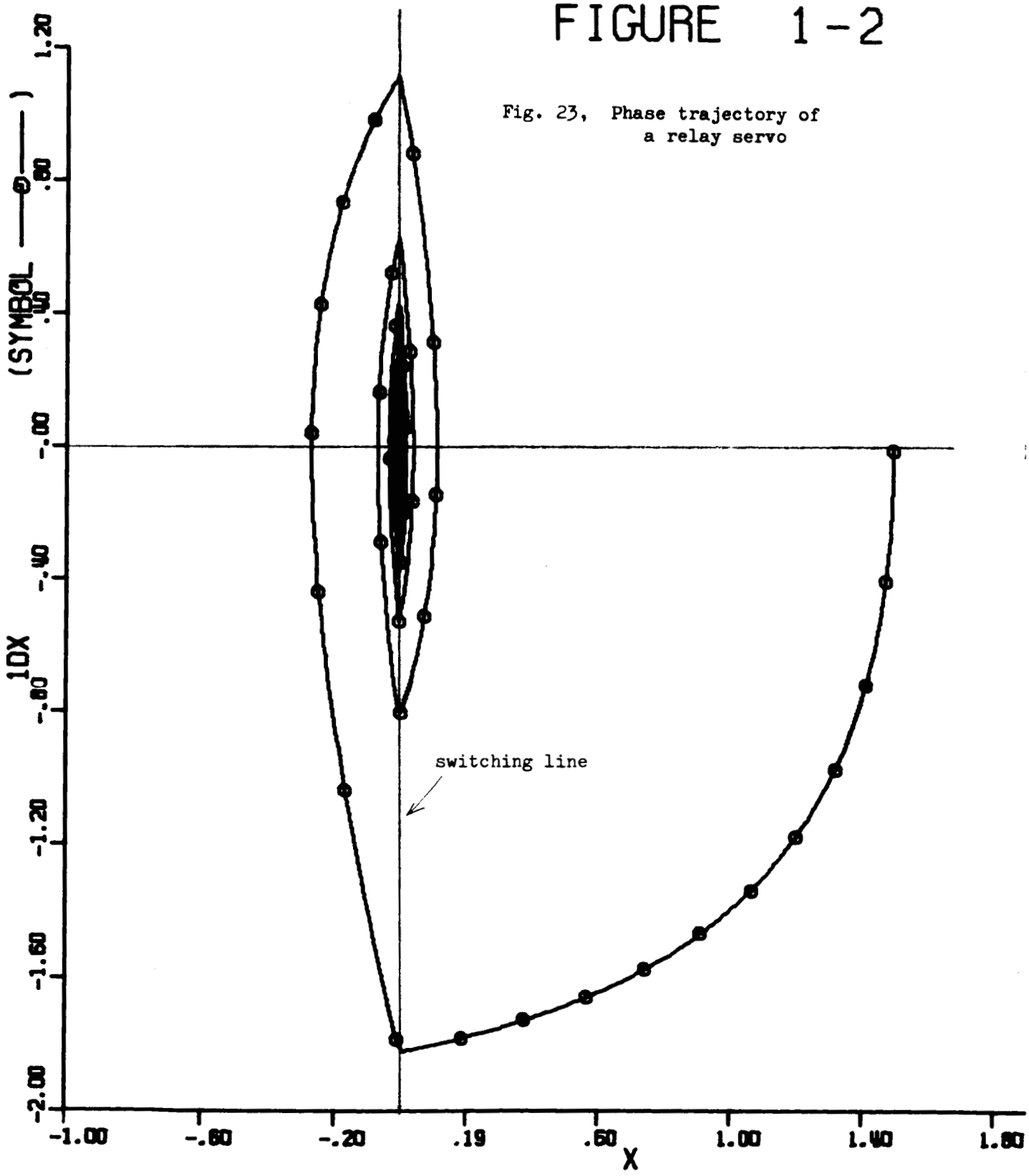


Fig. 23, Phase trajectory of a relay servo

(c) Relay servo with dead space

The relay in this servo has the dead-space characteristic shown in fig. 6(b), where there are also three power levels, +V, 0, and -V. This characteristic can be stated as below,

$G = +V$ when $E > +D$

$G = 0$

and $G = -V$ when $E < -D$

Similar to the simple relay servo, these conditions can be expressed by logical control variables E1, E2, and E3, determined by the following three statements,

E1 = FSW(E-D,FALSE,FALSE,TRUE)

E2 = FSW(E+D,TRUE,FALSE,FALSE)

and E3 = NOT(IOR(E1,E2))

The Mimic program for this servo is shown in fig. 24 where the above six statements are included. The transient or error response is shown in fig. 25. Output X approaches the value of D(i.e. 0.2); this limits the static accuracy. The phase trajectory is shown in fig. 26 where the relay switches at two vertical lines of X=-D and and X=+D. The phase plane is thus divided into three regions; the positive-, zero- and negative-torque regions. In the zero-torque region, the corrective power is not activited, and the load coasts across the dead-space with little change in speed. The zero-torque region helps stabilization, and the servo can come to rest and stay within the dead-space, instead of going into a limit cycle. However, here is a larger status error due to the dead space.

```
***PROBLEM NO. 6C***SERVO SIMULATION WITH DEAD SPACE IN THE RELAY
*
                    CON(A,B,V,D)
                    PAR(1DX0,X0)
*
          DT        = 0.01
*
          2DX       = -1DX/B+G*A/B
          1DX       = INT(2DX,1DX0)
          X         = INT(1DX,X0)
          E         = Y-X
          Y         = 0.0
*
 E1       G         = V
 E3       G         = 0.0
 E2       G         = -V
*
          E1        = FSW(E-D,FALSE,FALSE,TRUE)
          E2        = FSW(E+D,TRUE,FALSE,FALSE)
          E3        = NOT(IOR(E1,E2))
*
                    FIN(T,5.0)
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
                    PLO(T,X)
                    PLO(X,1DX)
                    END
2.0           0.5           1.0           0.2
0.0           1.5
```

Figure 24, Mimic Program for Problem No. 6C

FIGURE 1-2

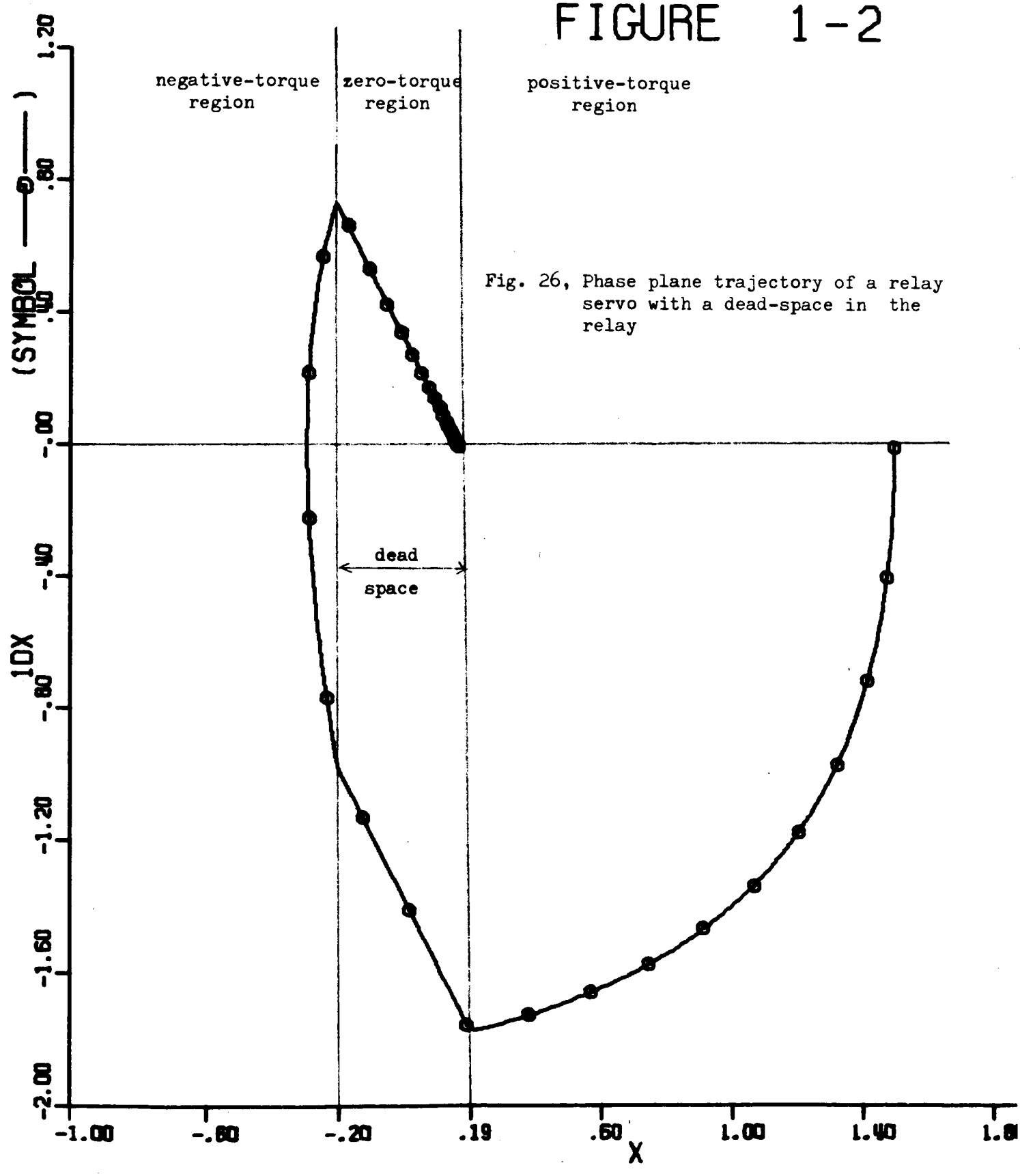


Fig. 26, Phase plane trajectory of a relay servo with a dead-space in the relay

FIGURE 1-1

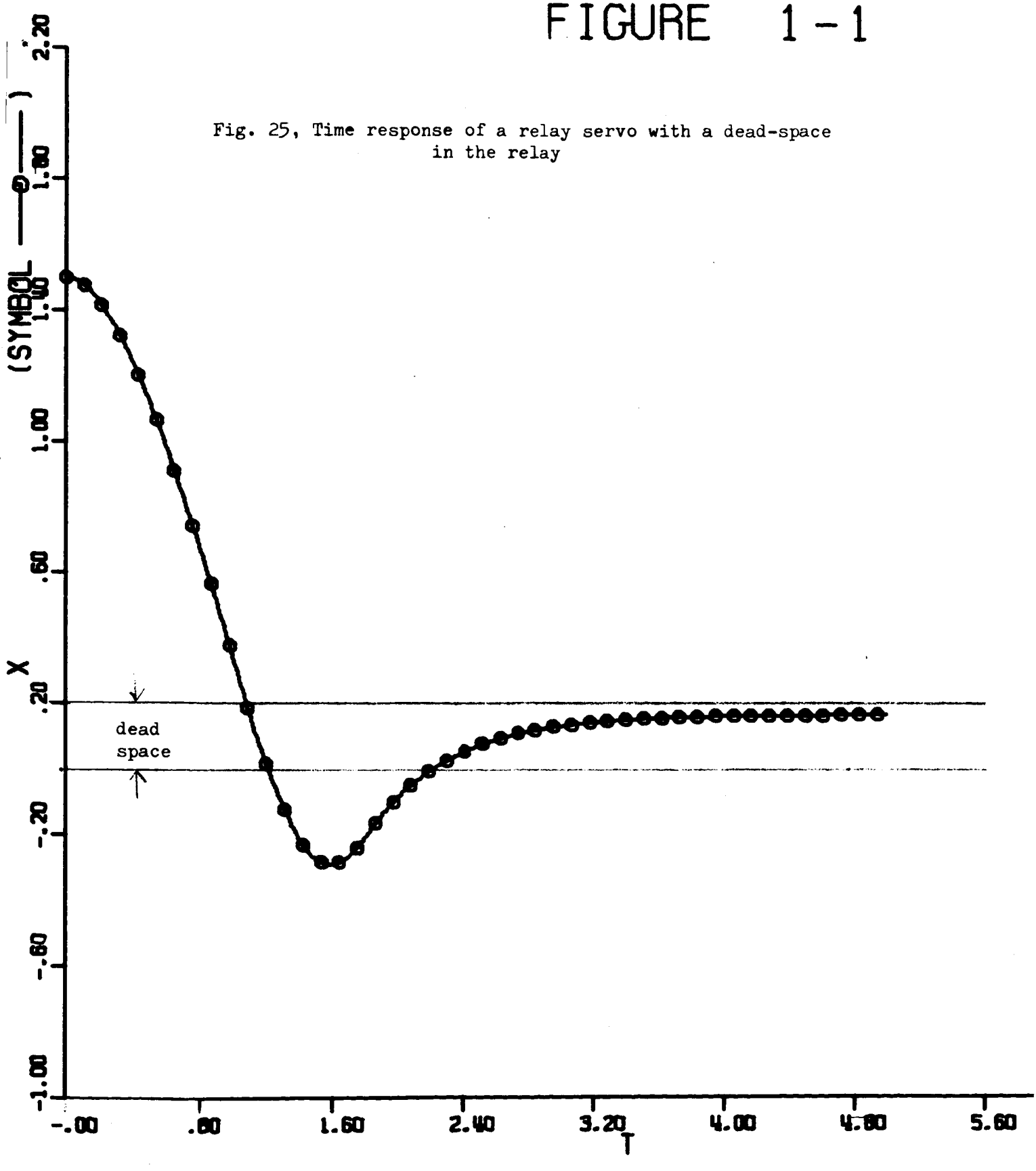


Fig. 25, Time response of a relay servo with a dead-space in the relay

(d) Relay servo with hysteresis

The relay in this servo has the hysteresis characteristic shown in fig. b(c). This characteristic can be described as follows. If G is initially +V, then G remains at +V if E > -D; otherwise, G becomes -V. If G is initially -V, then G remains at -V if E < +D; otherwise, G becomes +V.

Let H represent G equal to +V and -V when H is true and false respectively, then we have the following LSW statement,

G = LSW(H,+V,-V)

Let E1 be true when E > +D and E2 be true when E < -D, then we have,

E1 = FSW(E-D,FALSE,FALSE,TRUE)

E2 = FAS(E+D,TRUE,FALSE,FALSE)

Now to remember the initial status of G, a flip-flop represented by a FLF statement is employed. The FLF statement has three arguments, say, E1, E2, and R, where R denotes the initial status at time equal to 0(to be arbitrary chosen true here). Let H be the status of the flip-flop. The following FLF statement,

H = FLF(E1,E2,TRUE)

prescribes that H become true when E1 is true, become false when E2 is true, and remain unchanged when both E1 and E2 are false.

If G is initially at +V, H is true and remains true until E2 is true (this means until E < -D). If G is initially at -V, H is false and remains false until E1 is true (this means until E > +D). Both E1 and E2 are false when $-D \leqq E \leqq +D$, and that both E1 and E2 are

true is impossible. Thus, the above four statements prescribes the hysteresis characteristic in fig. 6(c). These statements are shown in the Mimic program for this servo in fig. 27.

The transient or error response is shown in fig. 28 and the phase trajectory in fig. 29. Since the hysteresis delays the switching operation, the corrective power is not reversed until the output X is past the desired zero point. As can be seen on the phase plane, the effect is destabilizing. There exists a limit cycle as shown in fig. 29.

```
***PROBLEM NO. 6D***SERVO SIMULATIONWITH HYSTERESIS IN THE RELAY
*
                    CON(A,B,V,D)
                    PAR(1DX0,X0)
*
          DT        = 0.01
*
          2DX       = -1DX/B+G*A/B
          1DX       = INT(2DX,1DX0)
          X         = INT(1DX,X0)
          E         = Y-X
          Y         = 0.0
*
          E1        = FSW(E-D,FALSE,FALSE,TRUE)
          E2        = FSW(E+D,TRUE,FALSE,FALSE)
          H         = FLF(E1,E2,TRUE)
          G         = LSW(H,V,-V)
*
                    FIN(T,5.0)
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
                    PLO(T,X)
                    PLO(X,1DX)
                    END
2.0           0.5           1.0           0.2
0.0           1.5
```

Figure 27, Mimic Program for Problem No. 6D

FIGURE 1-1

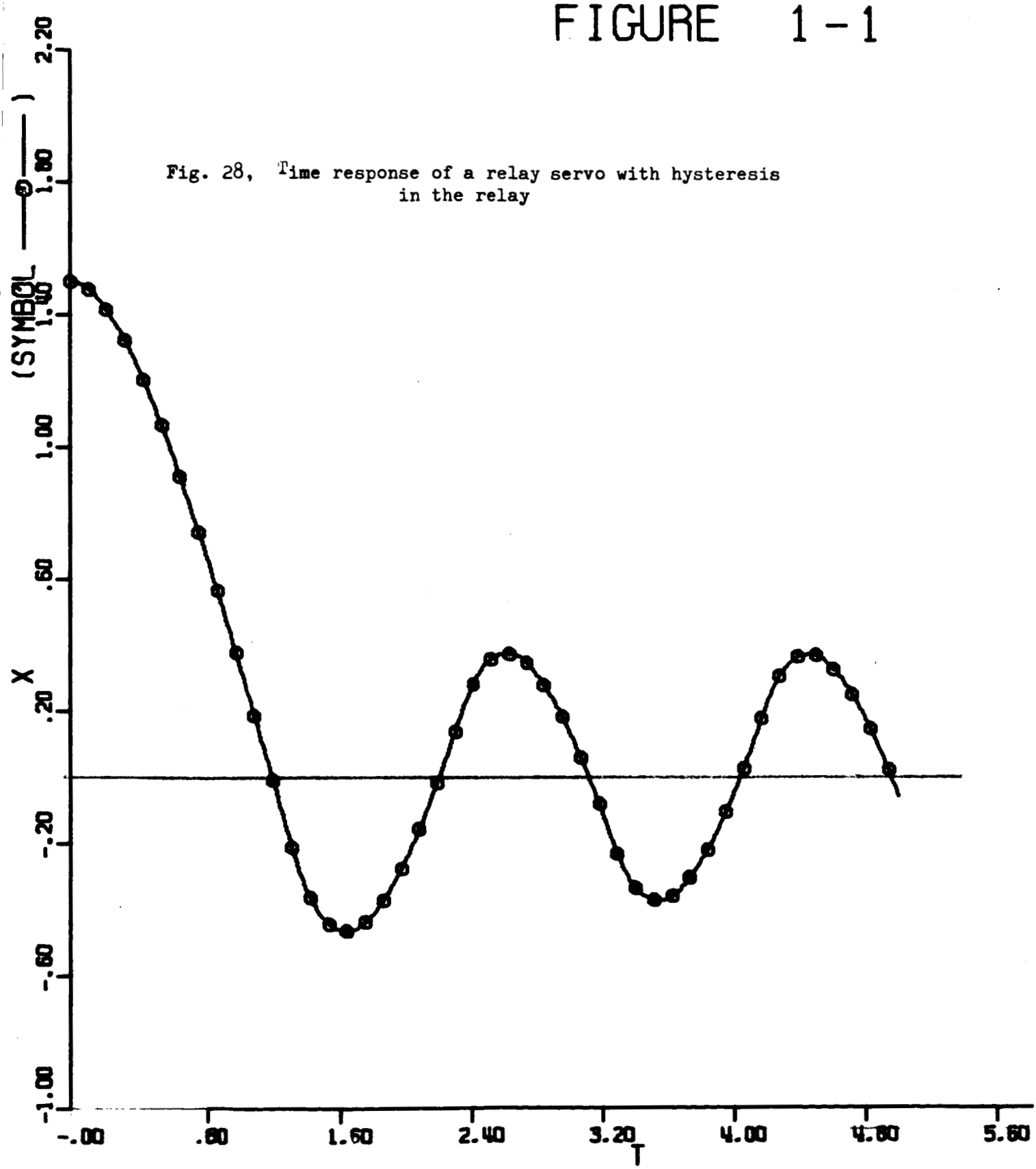
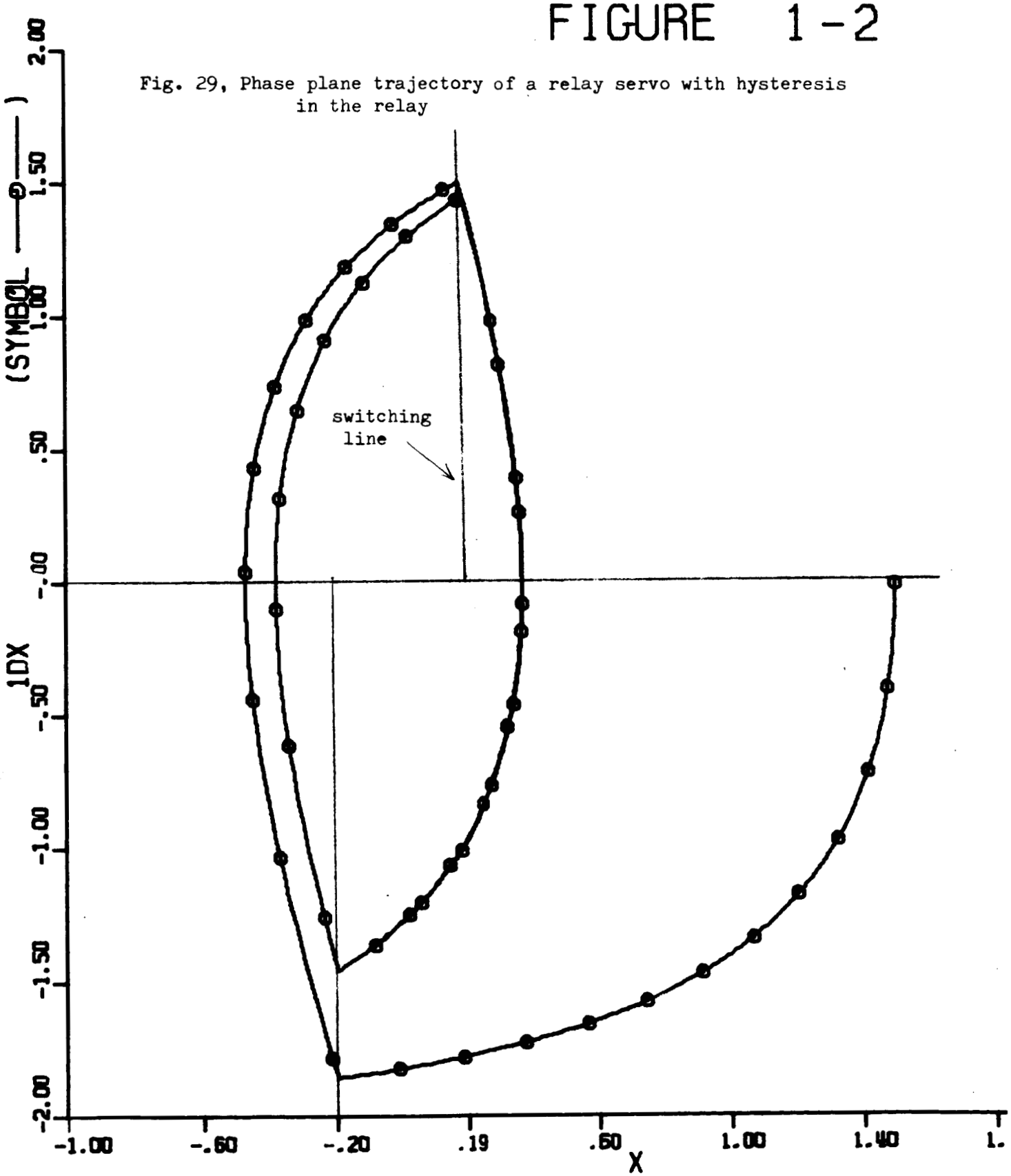


Fig. 28, Time response of a relay servo with hysteresis in the relay

FIGURE 1-2

Fig. 29, Phase plane trajectory of a relay servo with hysteresis in the relay



(e) Relay servo with both dead space and hysteresis

The relay in this servo has the characteristic shown in fig. 6(d). This characteristic can be described as follows. If G is initially +V, then G remains at +V if $E > +H$ and becomes 0 when $E \leq +H$. If G is initially -V, then G remains at -V if $E < -H$ and becomes 0 when $E \geq -H$. If G is initially 0, then G remains 0 if $-D \leq E \leq +D$, becomes +V if $E > +D$, and becomes -V if $E < -D$.

Let S1 be true when $E > D$, S2 be true when $E < -D$, R1 be true when $E \leq H$, and R2 be true when $E \geq -H$, then we have,

S1 = FSW(E-D,FALSE,FALSE,TRUE)

S2 = FSW(E+D,TRUE,FALSE,FALSE)

R1 = FSW(E-H,TRUE,TRUE,FALSE)

and R2 = FSW(E+H,FALSE,TRUE,TRUE)

Let L1 represent the status of a flip-flop so that L1 is true or false when S1 is true or when R1 is true, respectively; otherwise L1 remains unchanged. Let L2 represent the status of another flip-flop so that L2 is true or false when S2 is true or when R2 is true, respectively; otherwise L2 remains unchanged. Note the case that both S1 and R1 are true is impossible, neither is the case that both S2 and R2 are true. We then have,

L1 = FLF(S1,R1,TRUE)

and L2 = FLF(S2,R2,TRUE)

When L1 is true, G is equal to +V. When L2 is true, G is equal to -V. When both L1 and L2 are false, G is equal to 0. These can be described by using logical control variables E1, E2, and E3 defined by the following three statements,

E1 = LSW(L1,TRUE,FALSE)

E2 = LSW(L2,TRUE,FALSE)

and E3 = NOT(IOR(E1,E2))

There are altogether 12 statements which are shown in the Mimic program for this servo in fig. 30.

If G is initially at +V, L1 is true and remains true until R1 is true (this means until E≦ H). If G is initially at -V, L2 is true and remains true until R2 is true (this means until E ≧ -H). If G is initially 0, both L1 and L2 are false and remains false until S1 is true (this means until E> D) when G becomes +V, or until S2 is true (this means until E < -D) when G becomes -V. Thus, these statements prescribe the hysteresis/dead-space characteristic of the relay.

The transient or error response is shown in fig. 31 and the phase trajectory in fig. 32. Notice that the two verticals lines at which the switching of the relay occurs are off-set along the horizontal axis. Since hysteresis or dead space contributes destabilizing and static error, the existance of both gives a larger static error and a limit cycle with a larger amplitude than either alone.

```
***PROBLEM NO. 6E***SERVO SIMULATION WITH DEAD SPACE AND HYSTERESIS
*
                    CON(A,B,V,D,H)
                    PAR(1DX0,X0)
*
          DT        = 0.01
*
          2DX       = -1DX/B+G*A/B
          1DX       = INT(2DX,1DX0)
          X         = INT(1DX,X0)
          E         = Y-X
          Y         = 0.0
*
          S1        = FSW(E-D,FALSE,FALSE,TRUE)
          S2        = FSW(E+D,TRUE,FALSE,FALSE)
          R1        = FSW(E-H,TRUE,FALSE,FALSE)
          R2        = FSW(E+H,FALSE,FALSE,TRUE)
          L1        = FLF(S1,R1,TRUE)
          L2        = FLF(S2,R2,TRUE)
*
 E1       G         = V
 E3       G         = 0.0
 E2       G         = -V
*
          E1        = LSW(L1,TRUE,FALSE)
          E2        = LSW(L2,TRUE,FALSE)
          E3        = NOT(IOR(E1,E2))
*
                    FIN(T,5.0)
                    HDR(TIME,X,XDOT)
                    HDR
                    OUT(T,X,1DX)
                    PLO(T,X)
                    PLO(X,1DX)
                    END
2.0          0.5          1.0          0.2          0.1
0.0          1.5
```

Figure 30, Mimic Program for Problem No. 6E

FIGURE 1-1

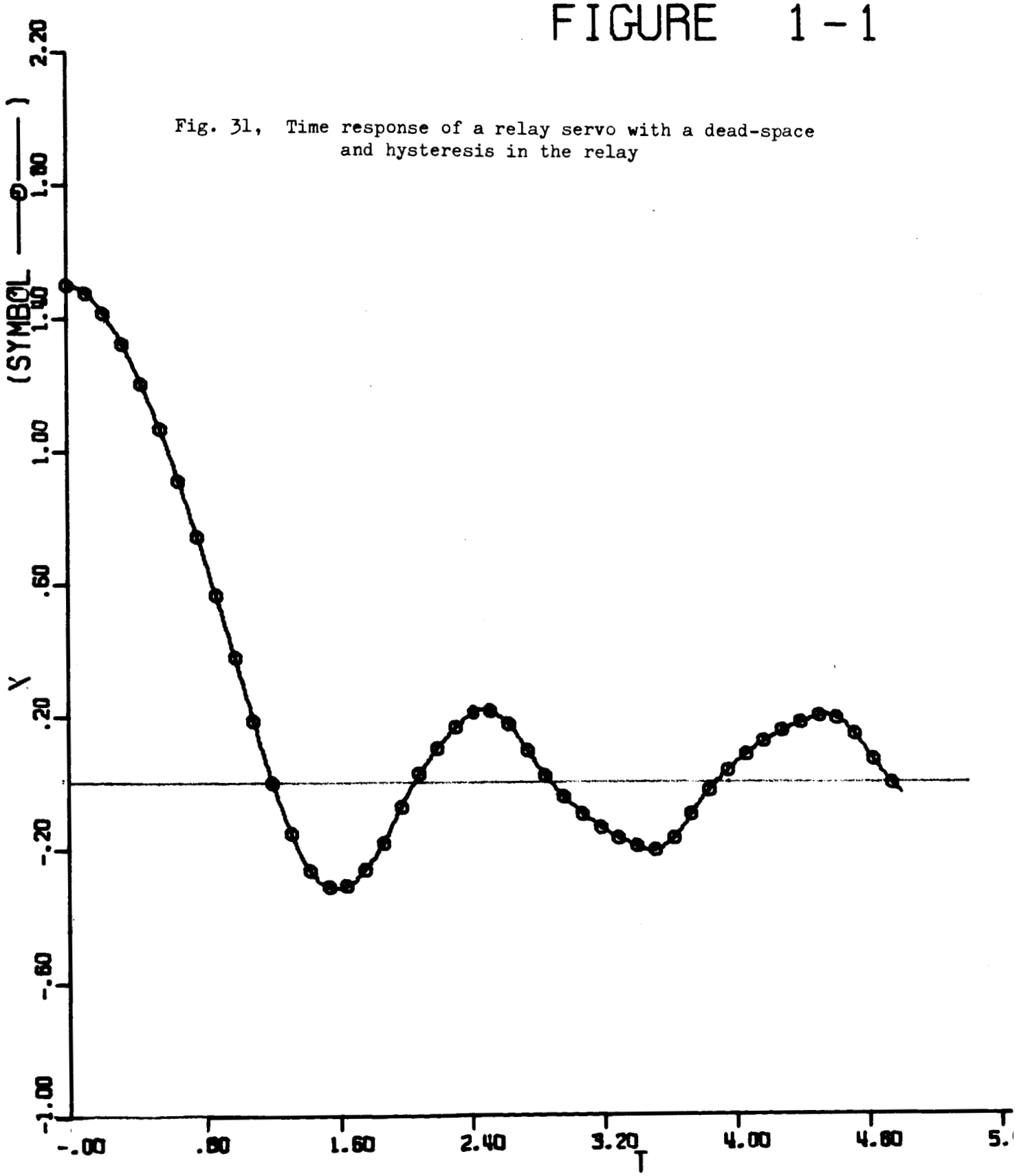


Fig. 31, Time response of a relay servo with a dead-space and hysteresis in the relay

FIGURE 1-2

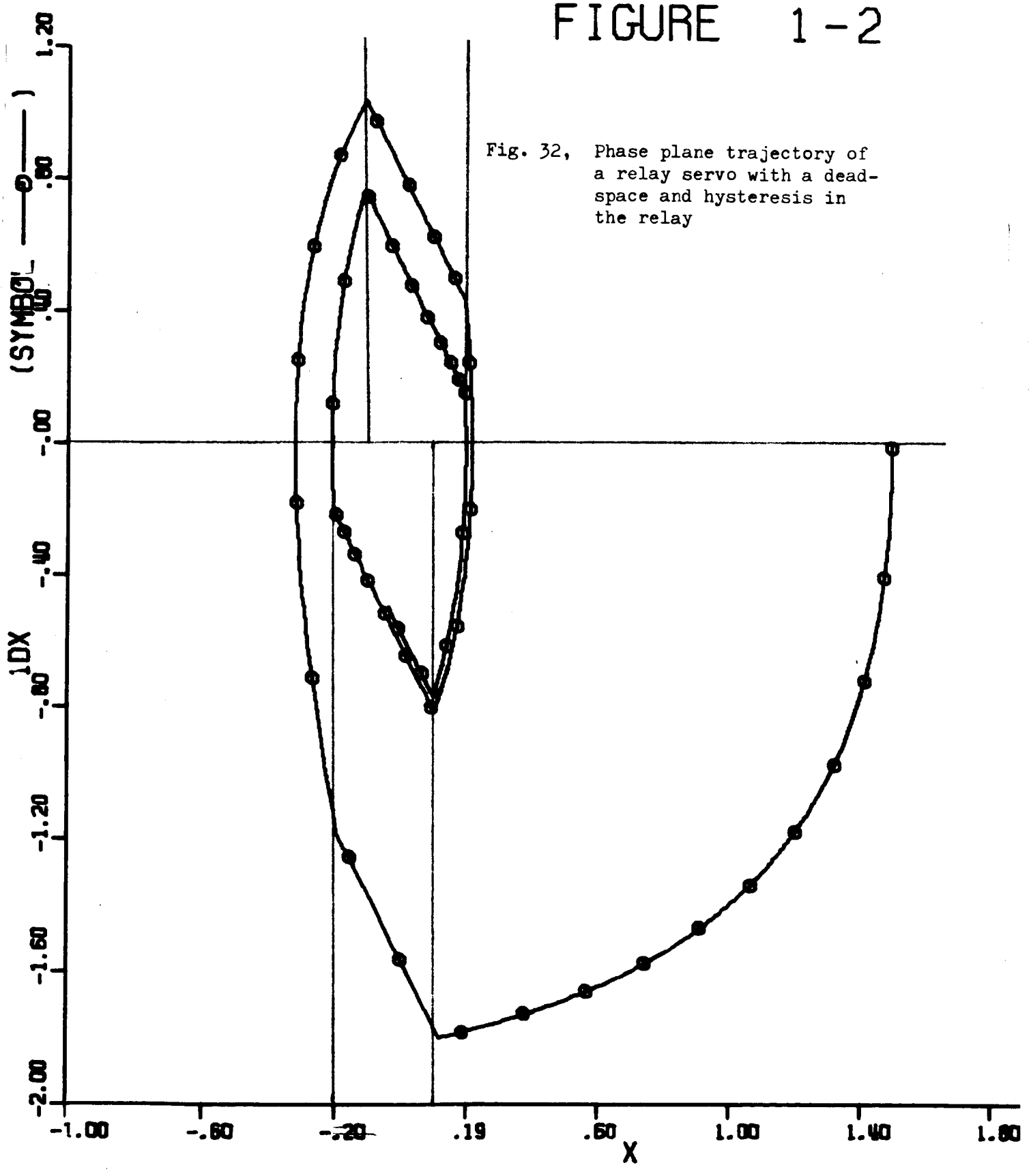


Fig. 32, Phase plane trajectory of a relay servo with a dead-space and hysteresis in the relay

(8) Conclusion

The above Mimic programs and their programs and their results have shown simplicity of the language and some techniques in simulation and computation of engineering and scientific problems. They have also shown the advantages of great accuracy, no need of scaling, and no hardware setup. Plots of curves are available with automatic scaling and labeling.

Consideration of cost of computation should not be neglected. The computer time for any of the above programs without plots ranges from about one-half minute to about two minutes on the IBM 7094 computer system at the University of Maryland; this variation depends to a great extent on the value of DT. With plots, the computer time is about doubled. The computer time at University of Maryland costs at about $5.00 per minute. The cost for one run of these programs with plots, thus ranges from $5.00 to $10.00. If one takes five runs to have the program debugged, then the cost of the computer time for one program ranges from $25.00 to $100.00.

There are other advantages of digital analog simulation that have been or are being recognized. These are feasibility and simplicity of function generation of more than two variables, availability of many mathematical functions normally provided in a digital computer facility better computer utilization, rapid switching from one simulation to another, simpler storage of sim-

ulation programs for reuse, and easier machine maintenance. Lastly, the use of cathode-ray-tube for one-line display as well as large-scale real-time simulation are becoming realities.

In conclusion, the use of digital analog simulation as a tool for engineering analysis and design will be rapidly increased as more and better digital computer systems become available.

9. References

(1) Selfridge, R. G., "Coding a General Purpose Digital Computer to Operate as a Differential Analyzer," Proceedings of Western Joint Computer Conference, 1955, pp. 82-84.

(2) Brennan, R. D., and R. N. Linebarger, "A Survey of Digital Simulation: Digital Analog Simulator Programs," Simulation, Vol. 3, No. 6, December, 1964 pp.

(3) Clancy, J. J. and M. S. Fineberg, "Digital Simulation Languages: A Critique and a Guide," Proceedings of the Fall Joint Computer Conference, 1965, Spartan Books, pp. 23-36

(4) Hurley, J. R., and J. J. Skiles, "Dysac, A Digitally Simulated Analog Computer", Proceedings of Spring Joint Computer Conference, 1963, Spartan Books, Inc., pp. 69-82.

(5) Gaskill, R. A., J. W. Harris and A. L. McKnight, "DAS - A Digital Analog Simulator", Proceedings of Spring Joint Computer Conference, 1963 pp.83-90.

(6) Petersen, H. E., F. J. Sansom, R. T. Hartnett, and L. M. Warshawsky, "Midas - How It Works and How It's Worked," Proceedings of Fall Joint Computer Conference, 1964, pp. 313-324.

(7) Brennan, R. D. and H. Sano, "Pactolus - A Digital Analog Simulator Program for IBM 1620," Proceedings of Fall Joint Computer Conference, 1964, pp. 299-312.

(8) Rideout, V. C. and L. Tavernini, "Madbloc, A Program for Digital Simulation of Hybrid Computer," Simulation, January, 1965, pp. 20-24.

(9) Petersen, H. E., and F. J. Sansom, "Mimic---A Digital Simulation Program," SESCA Internal Memo 65-12, U. S. Air Force, May, 1965.

(10) Levine, L., "The DES-1, A New Digital Computer for Solving Differential Equations," Simulation, April, 1965, pp. 264-276.

(11) Syn, W. M. and R. N. Linebarger, "DSL/90 - A Digital Simulation Program for Continuous System Modeling," Proceedings of the Spring Joint Computer Conference, 1966, Spartan Books, pp. 165-187.

(12) Green, C., H. D. Hoop and A. Debroux, "Apache - A Breakthrough in Analog Computing," IRE Transactions on Electronic Computers, October, 1962, pp. 699-706.

(13) Green, D. T., "OLDAS - An On-line Simulation Language," Naval Weapons Laboratory, Dahlgren, Virginia.

(14) B. Van der Pol, "On Relaxation Oscillations," Phil. Mag., ol. 17, No. 2 1926, p. 986.

(15) Irving, J. and M. Mullineux, "Mathematics in Physics and Engineering," Academic Press, 1959.

(16) Rogers, A. E., and T. W. Connolly, "Analog Computation in Engineering Design," McGraw-Hill Book Co., Inc. 1960

(17) Fifer, S., "Analog Computation," Vol. 4, McGraw-Hill Book Co., 1961.

(18) Bingulac, S. P., and E. A. Humo, "Analog Computer Generation of Bessel Functions of Arbitrary Order," IEEE Transactions on Electronic Computers, December, 1965, pp. 886-889.

(19) Haberman, C. M., "Engineering Systems Analysis," C. E. Merrill Books, Inc., 1965.

(20) Jackson, A. S., "Analog Computation," McGraw-Hill Book Co., 1960.

(21) Howe, R. M., "Solution of Partial Partial Deferential Equations" in H. D. Huskey and G. A. Korn, "Computer Handbook," McGraw-Hill Book Co., 1962.

(22) Hausner, A., "Multiple Integrals on a Non-repetitive Analog Computer" Proceedings of Spring Joint Computer Conference, 1963, pp. 205-212.

(23) Kopal, Z., "Numerical Analysis," John Wiley & Sons, Inc. 1955

(24) Henrici, P., "Discrete Variable Methods in Ordinary Deferential Equations," John Wiley & Sons, Inc. 1962.