IMPLEMENTING A FUNCTION LIBRARY FOR NONLINEAR DYNAMIC SYSTEMS

By

Tsun-Yu Chou

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

1983

ABSTRACT

IMPLEMENTING A FUNCTION LIBRARY FOR NONLINEAR DYNAMIC SYSTEMS

By

Tsun-Yu Chou

ENPORT-5 is a bond-graph based interactive simulation program with extensive graphic ability that allows the user to control and analyze the simulation of physical systems effectively. To extend this ability to the nonlinear domain an immediate need is an explicit expression of the nonlinear constitutive equations that can be defined by the user and modified easily and efficiently. An interactive program, NOLMOD, has been written which is based on the core concept of a prestored function library. Certain efficiencies are available by using a pre-programmed library; flexibility must be achieved by other special methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

During the past decade the digital computer has come to play a very important role in system simulation. Many general purpose as well as special purpose modeling and simulation programs have been developed. One general requirement of such a program is that the user need not have extensive knowledge of computer programming and numerical techniques. The typical user can learn to operate such a program, if it is well-designed, within a very short period of time.

Many important engineering and physical systems are described by differential equations. Simulation programs provide a way of solving such differential equations and also provide graphical and tabular output that makes it very easy for the user to study the behavior of the systems. Selected parameters and functions can be entered to provide necessary information about the differential equations. They are constructed to define the relationships between dependent and independent variables and among dependent

variables. How to provide such information to a computer is a topic in which we are interested. Therefore several simulation programs are studied, including CSMP, SUPER-SCEPTRE and DIFFEQ. CSMP is a general purpose block-diagram-oriented program developed by IBM for simulation of physical and engineering systems [1-3]. SUPER-SCEPTRE is a special purpose program intended primarily for electrical systems[4]. DIFFEQ is a general equation-based program developed at Michigan State University for solving ordinary differential equations [5].

Patterns for entering data are very similar in both CSMP and SUPER-SCEPTRE. Control statements and data statements are used as input data. Control statements tell the system what kind of data is provided and how to handle it. Data statements give the value of parameters, the types of functions and other related information. Some commonly used functions are pre-programmed, and both programs also provide function-generating routines for special functions. Then the system, based on the data given, prepares the necessary subroutines for calculations. The user can also write his own subroutine, if he is familiar with computer programming. Then the program will compile and load the binary file and set up the run-time program. This is the approach taken in DIFFEQ. A FORTRAN subroutine is written describing the differential equations and the system provides the calculation and display capabilities.

The common point of all three programs is that they have to compile the function subroutines and combine them with the rest of the program. There are several disadvanteges of such a process. First of all, it take both time and disk space to do it. When a lot of users are active at one time and every user stores a binary file of the program on the disk, that can use up a lot of space. And the compiling and loading process takes computer time. An advantage, however, is that the run-time program can be saved for re-run with new data and execution control statements. Subsequent simulation jobs using the same model may avoid repeating the translating, compiling and linking process. Some careful planning is necessary, coupled with a sound background in FORTRAN, to ensure that the saved model has sufficient capacity for later uses. Since both CSMP and SUPER*SCEPTRE are used in batch mode, user has to submit the job all over again when the data are changed. But DIFFEQ is a fully interactive program in which changing of parameters can be done on-line. This feature enables the user to have direct interaction with the simulation. This is a great advantage in design, since it can be re-run relatively quickly until the user is satisfied with the results. But changing of a function type in DIFFEQ must be done by modifying the subroutine. Then the compiling process must be repeated.

ENPORT is an interactive program based on bond graph

theory for modeling and simulation of physical systems which has many of the desirable properties mentioned above [6-7]. However, at this time the ENPORT program can only handle linear or linearized models. Demand that ENPORT be able to handle nonlinear models is increasing. The purpose of this work is to develop a program that extends ENPORT into the nonlinear domain. For nonlinear problems functions are specified, instead of just constant parameters. Therefore our strategy is to set up a function library for such a purpose. The program NOLMOD was written to achieve this goal. The user can choose desired functions from the library and can modify these choices without leaving the program. To use this system the designer first runs ENPORT to read the bond graph as original data and to assign the causality. Then NOLMOD takes over the rest of the work. The program NOLMOD has been installed on the PRIME-750 computer of A. H. Case Center, College of Engineering, Michigan State University.

CHAPTER 2

EQUATION FORMULATION FOR NONLINEAR BOND GRAPHS

2.1 Bond Graph structure

One of the most important features of the bond graph method is the concept of fields and junction structures. These terms were defined by H. Paynter [8]. A multiport system is partitioned into a set of distinct, interconnected fields, namely, a source field, a storage field, a dissipation field, and a junction structure. By definition a junction structure is a collection of junction elements : 0- and 1- junctions, and transformers (TF) and gyrators (GY). In a standard bond graph every bond is connected to a junction element at least at one end. It is useful to classify the graph bonds into external bonds and internal bonds. A bond connected between a field element (C, I, R, SE, SF) and a junction element (0, 1, TF, GY) is called an external bond. An internal bond is a bond connected between two elements of the junction set. Furthermore, external bonds are classified into three groups according to the nature of the elements they adjoin : source field bonds (SE and SF), storage field bonds (C and I) and dissipation field

bonds (R). The equation structure implied by a bond graph is constructed based on these classifications. Figure 2.1 shows the bond graph field structure [6].

2.2 Key variables and causal considerations

Figure 2.2 shows the key variables (vectors) used for representing the input and output variables for each field. U is the input to the junction structure from the source field, V is the output from the junction structure to the source field; these are associated with the source field bonds. Z is the co-energy variable vector, X dot is the time derivative of the energy variables vector; these are associated with the storage field bonds. Do is the output from the dissipation field to the junction structure, Di is the input to the dissipation field from the junction structure; these are associated with the dissipation field bonds. Subscripts i and d stand for integral and derivative ports of the storage field, respectively.

Causal consideration is a very important aspect in the bond graph method. It provides all the information concerning input and output variables. Some multiports are strongly constrained to certain causality (SE and SF). For C- and I-fields, the constitutive laws are quite different for different causalities. Usually integral causalities are sought. Derivative causality arises for systems in which

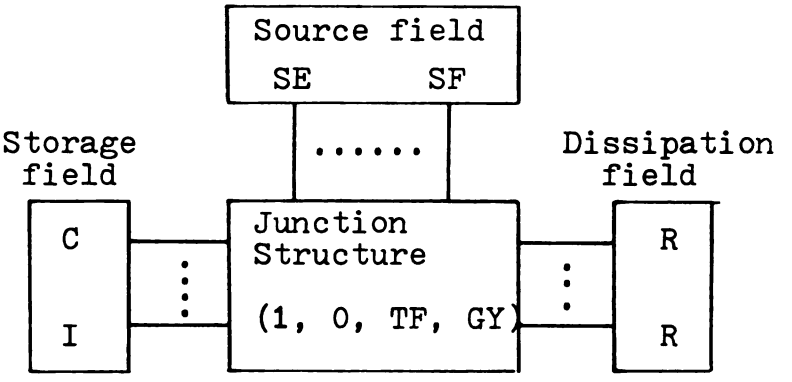

Figure 2.1 Basic fields of a multiport system.

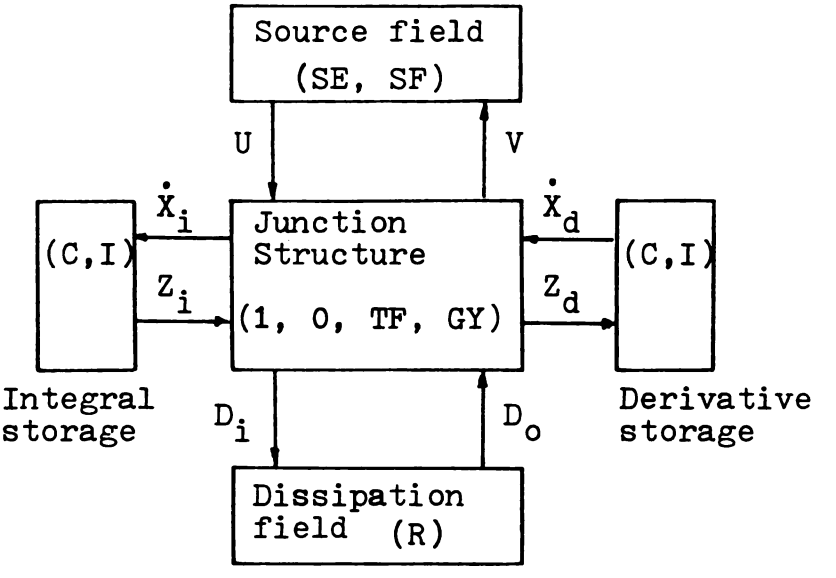

Figure 2.2 Identification of key vectors in a mixed causality system.

some storage elements are not dynamically independent. The energy variable of the derivative storage element port will be expressed algebraically in terms of other independent energy variables. Therefore any storage element ports with derivative causality do not contribute to the state variable set. For a nonlinear system it may be difficult to express one energy variable in terms of the others explicitly, since they might be coupled by nonlinear algebraic equations. This situation is typical of nonlinear mechanics, for example. The causality for the R-field usually is determined by the source and storage-field elements. Usually they are relatively indifferent to causality. However, in the nonlinear case some constitutive laws for the R-field are not uniquely determined for certain causalities. An example will be shown in Chapter 3.

2.3 System equations and state equations

First we shall look at the situation for linear systems, useful as a guide to our thinking in the nonlinear case. The field equations for linear systems are as follows:

Source field :

$$U = U(t) \tag{2.1}$$

Storage field :

$$Z_i = F_{ii} X_i + F_{id} X_d \tag{2.2a}$$

and

$$Z_d = F_{di} X_i + F_{dd} X_d \qquad (2.2b)$$

Dissipation field :

$$D_o = L D_i \qquad (2.3)$$

Junction structure equations :

$$\dot{X}_i = S_{11} Z_i + S_{12} \dot{X}_d + S_{13} D_o + S_{14} U \qquad (2.4a)$$

$$Z_d = S_{21} Z_i + S_{22} \dot{X}_d + S_{23} D_o + S_{24} U \qquad (2.4b)$$

$$D_i = S_{31} Z_i + S_{32} \dot{X}_d + S_{33} D_o + S_{34} U \qquad (2.4c)$$

$$V = S_{41} Z_i + S_{42} \dot{X}_d + S_{43} D_o + S_{44} U \qquad (2.4d)$$

The system equations above can be reduced to a single matrix state-space equation by some suitable manipulations. (We assume the required inverses exist.) Thus we arrive at

$$\dot{X}_i = A X_i + B U + E \dot{U} \qquad (2.5)$$

The E matrix only appear when we have mixed causality. One may expect that all entries in matrices A, B and E are real numbers which are derived from the parameters of the physical elements. Matrices A, B and E can be calculated from a causal bond graph after parameters are given. With

given X and U, X dot can be obtained easily. And by use of the equations (2.1), (2.2), (2.3) and (2.4), all system variables can be found.

For some nonlinear systems equations (2.2) and (2.3) are no longer valid. They will be in following form:

For the storage fields:

$$Z_i = \phi_{Fi}(X_i, X_d) \tag{2.6a}$$

$$Z_d = \phi_{Fd}(X_i, X_d) \tag{2.6b}$$

For the dissipation field:

$$D_o = \phi_L(D_i) \tag{2.7}$$

In this case state-space equations of the form:

$$\dot{X}_i = \phi(X_i, U) \tag{2.8}$$

may not be obtainable. System equations may be firmly coupled together. It may be very complex to reduce the system equations to a single state-space equation in the form of equation (2.8). Typical stumbling blocks are equations (2.6) and (2.7).

Therefore in this work we restrict the range of consideration to the case of:

(1) integral causality (hence $X_d = 0$)

(2) no implicit R-fields (hence $S_{33}=0$)

(3) constant moduli for TF and GY (hence S_{ij} have constant elements)

See Figure 2.3 for the key vectors in this case. Then the

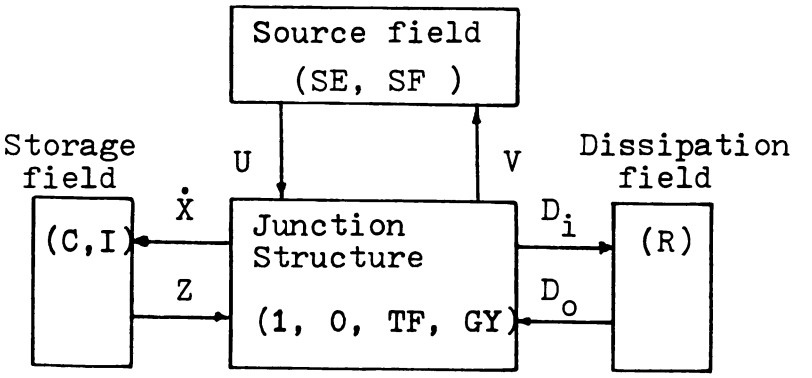

Figure 2.3 A multiport system having only integral causality.

field equations and junction structure equations will be simplified as follows:

$$Z = \phi_F(X) \qquad (2.9)$$

$$D_o = \phi_L(D_i) \qquad (2.7)$$

$$\dot{X} = S_{11} Z + S_{13} D_o + S_{14} U \qquad (2.10a)$$

$$D_i = S_{31} Z + S_{34} U \qquad (2.10b)$$

Rearrange the equations in the following sequence:

$$U = U(t) \qquad (2.1)$$

$$Z = \phi_F(X) \qquad (2.9)$$

$$D_i = S_{31} Z + S_{34} U \qquad (2.10b)$$

$$D_o = \phi_L(D_i) \qquad (2.7)$$

$$\dot{X} = S_{11} Z + S_{13} D_o + S_{14} U \qquad (2.10a)$$

With U and X given, then Z, Di, Do and X dot can be calculated step by step according to the above sequence [11]. Viewed as a total procedure this is equivalent to the explicit form given by equation (2.8).

2.4 Example of matrix formulation

An example is shown in this section to demonstrate the reduction procedures for both linear and nonlinear models. Figure 2.4 is a simple example of a spring-mass-damper model. An augmented bond graph is shown in Figure 2.5 which displays the fields and junction structure. The field equations for linear system are as follows:

Source field :

$$U = U(t) \qquad (2.1)$$

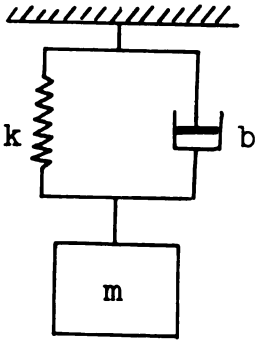

Figure 2.4a

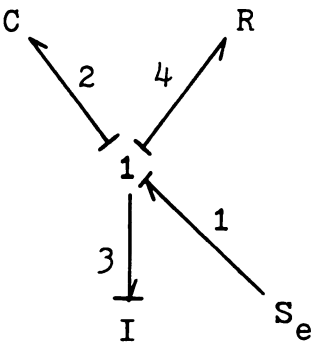

Figure 2.4b

Figure 2.4 (a) Spring-mass-damper model
(b) Bond graph model.

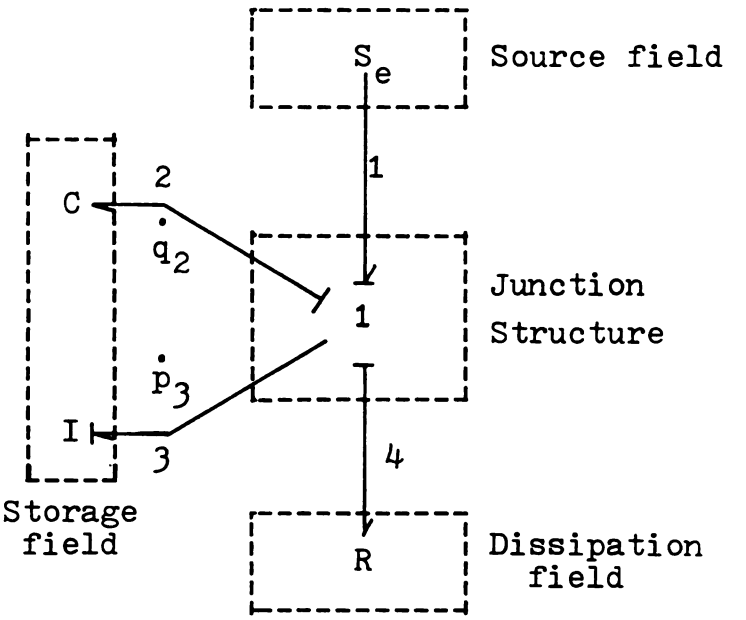

Figure 2.5 Augmented bond graph model.

Storage field :

$$Z = F \quad X \tag{2.11a}$$

$$\begin{bmatrix} F_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & 1/m \end{bmatrix} \begin{bmatrix} q_2 \\ P_3 \end{bmatrix} \tag{2.11b}$$

Dissipation field :

$$D_o = L \quad D_i \tag{2.12a}$$

$$\begin{bmatrix} F_4 \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} \begin{bmatrix} V_4 \end{bmatrix} \tag{2.12b}$$

Junction structure equations :

$$\begin{bmatrix} X \\ D_i \end{bmatrix} = \begin{bmatrix} S_{11} & S_{13} & S_{14} \\ S_{31} & S_{33} & S_{34} \end{bmatrix} \begin{bmatrix} Z \\ D_0 \\ U \end{bmatrix} \tag{2.13a}$$

$$\left[\begin{array}{c} q_2 \\ P_3 \\ \hline V_4 \end{array}\right] = \left[\begin{array}{cc:c:c} 0 & 1 & 0 & 0 \\ -1 & 0 & -1 & 1 \\ \hdashline 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} F_2 \\ V_3 \\ \hline F_4 \\ \hline F_1 \end{array}\right] \tag{2.13b}$$

The system equations can be reduced to a state-space equation:

$$\dot{X}_i = A X_i + B U \qquad (2.14)$$

Where :

$$A = \left[S_{11} + S_{13} L \left(I - S_{33} L \right)^{-1} S_{31} \right] F \qquad (2.15a)$$

$$B = \left[S_{14} + S_{13} L \left(I - S_{33} L \right)^{-1} S_{34} \right] F \qquad (2.15b)$$

Therefore, by matrix operations on equations 2.11a, 2.12b and 1.3b,

$$A = \begin{bmatrix} 0 & 1/m \\ -k & -b/m \end{bmatrix} \qquad (5.16a)$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad (5.16b)$$

If the storage and dissipation elements are nonlinear, then an explicit form of state-space equation (2.14) may not be obtainable. We must follow the procedures described in section 2.3 to obtain the implicit state-space equation. The field equations will be as follows:

Source field :

$$U = U(t) \qquad (2.1)$$

Storage field :

$$Z = \phi_F (X) \qquad (2.9)$$

$$F_2 = \phi_2(q_2) \qquad (2.17a)$$

$$V_3 = \phi_3(P_3) \qquad (2.17b)$$

Junction structure equations :

$$D_i = S_{31} Z + S_{34} U \qquad (2.10b)$$

$$D_i = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} F_2 \\ V_3 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} U \qquad (2.18)$$

Dissipation field :

$$D_o = \phi_L(D_i) \qquad (2.7)$$

$$F_4 = \phi_4(V_4) \qquad (2.19)$$

Implicit state-space equation :

$$\dot{X} = S_{11} Z + S_{13} D_o + S_{14} U \qquad (2.10a)$$

$$\begin{bmatrix} \dot{q}_2 \\ \dot{P}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} F_2 \\ V_3 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \begin{bmatrix} F_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} U \end{bmatrix} \qquad (2.20)$$

Values of F2, V3 and F4 should be obtained from equations (2.17), (2.18) and (2.19).

2.5 Computing implications

We now turn our attention to the practical question of how to implement equations (2.7) and (2.9) and the procedure for solving them. For equation (2.7) and (2.9), the nonlinear functions describe the relations between input and output variables on a port-by-port basis. In order for the engineer to supply such information to the program interactively, functions may be pre-programmed and parameter-type data given during interactive running. The most common engineering models involve one-port elements, in which case only one dependent variable is involved. For multi-port field elements the output set is a function of the input set of two or more variables. It is more difficult to define a set of library functions that can be used for describing a variety of multi-port elements. Therefore we restrict the bond graph to contain one-port field elements only. This will simplify the problem again.

A function library that contains commonly used functions and a function-generating routine are used to serve the purpose. The library can also be used to describe nonlinear transformer and gyrator element moduli, although they are not implemented at this stage of the work.

CHAPTER 3

EXAMPLES

3.1 Spring-mass-damper model

For better understanding of the procedures for simulating nonlinear systems by the bond graph method and NOLMOD program, the example of a spring-mass-damper system in Section 2.5 is used for demonstration. Commands and data listed below are given to ENPORT.

```
GRAPH
 SE 1 , C 2 , I 3 , R 4 , 1 1 2 3 4 .
CAUSAL
```

After causality has been assigned, NOLMOD will classify the basic equations according to types of elements and causal conditions.

```
INTEGRAL STORAGE ELEMENTS
     E 2 = FCN 2 (Q 2)
     F 3 = FCN 3 (P 3)

DISSIPATION ELEMENTS
```

E 4 = FCN 4 (F 4)

SOURCE ELEMENTS

E 1 = FCN (TIME)

Then the user can define the functions for the storage and dissipation constitutive equations. The user-defined functions are as follow :

FUNCTION 2 :

E 2 = 0.0000E-01 + 4.0000E+00 * Q 2

FUNCTION 3 :

P 3 = 0.0000E-01 + 6.0000E-01 * F 3

FUNCTION 4 :

E 4 = 8.0000E-01 * SIGN(F 4) * SQRT(ABS(F 4))

Following the causal conditions in Figure 4.5 the state equations will be :

$$\dot{Q}_2 = F_3 \qquad (3.1)$$

$$\dot{P}_3 = E_1 - E_2 - E_4 \qquad (3.2)$$

Then we can integrate equations (3.1) and (3.2) to obtain the displacement and momentum of the system. Now we may use DIFFEQ to simulate the system for demonstration. Figure 3.1 is the subroutine FCT used in DIFFEQ [4] and Figure 3.2 and 3.3 are time responses of the displacement and momentum.

```
      SUBROUTINE FCT(X,Y,DERY)
C
      REAL*4 DERY(20),Y(20),PA(20)
      REAL K,M
      COMMON/FCTCOM/PA
C*****************************************
C--- Y(1) = SPRING DEFLECTION
C--- Y(2) = MASS MOMENTUM
C
C---INITIAL CONDITIONS : Y(1)= 1.0
C                        Y(2)= 0.0
C
C---  E1 = WEIGHT OF MASS
C---  E2 = SPRING FORCE
C---  F3 = MASS DISPLACEMENT
C---  E4 = FRICTION FORCE
C
C---   K = SPRING CONSTANT
C---   M = MASS
C---   B = FRICTION COEFFICIENT
C*****************************************
C---DATA INPUT
C
      K=PA(1)
      M=PA(2)
      B=PA(3)
C
C---SOURCE FIELD EQUATION :
C
      E1 = M * 9.8
C
C---STORAGE FIELD EQUATIONS :
C
      E2 = K * Y(1)
      F3 = (1./M) * Y(2)
C
C---DISSIPATION FIELD EQUATION :
C
      E4 = B * F3 * ABS(F3)
C
C---STATE-SPACE EQUATIONS
C
      DERY(1) = F3
      DERY(2) = E1 - E2 - E4
C
      RETURN
      END
```

Figure 3.1 Subroutine FCT.

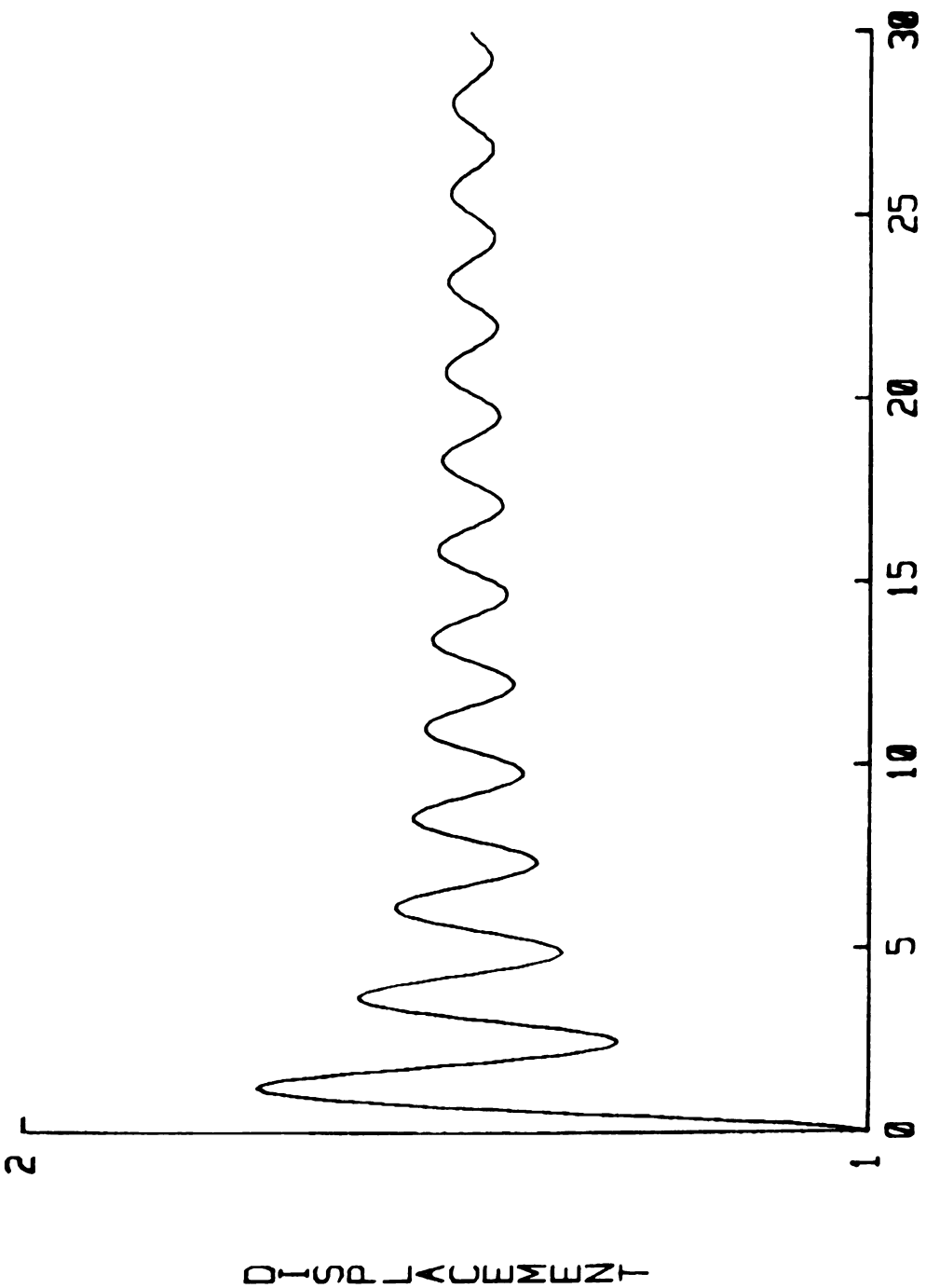

Figure 3.3 Momentum diagram.

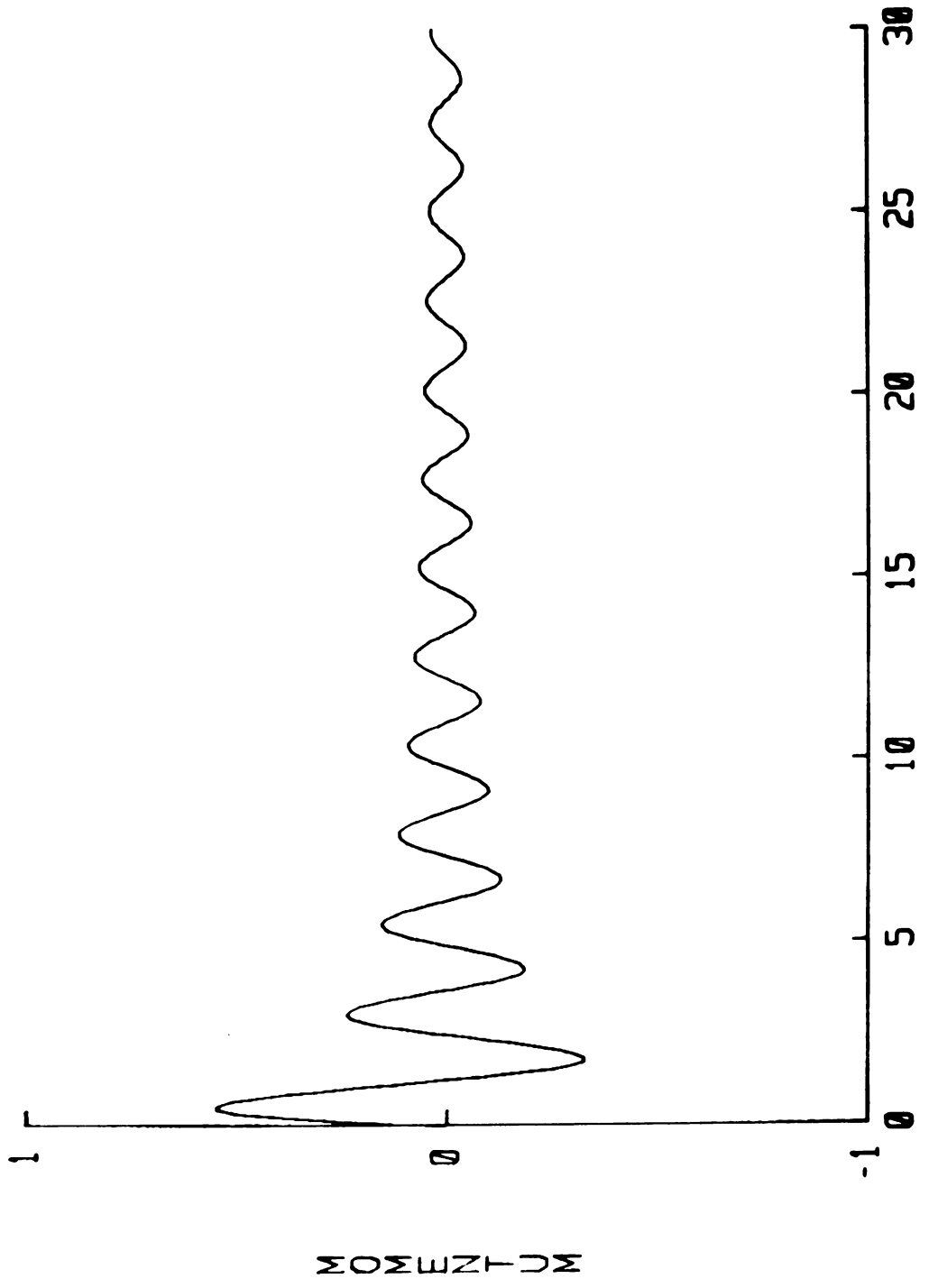

Figure 3.2 Displacement diagram.

3.2 Hydraulic system

Figure 3.1 shows a simple hydraulic system. A hydraulic motor is driven by a pump. The 2-block 4-way valve is in on position, a pressure relief valve is used to maintain system pressure and an adjustable throttle is used to control the flow through the hydraulic motor. The corresponding bond graph for such a hydraulic system is shown in Figure 3.2. Causality is uniquely assigned according to basic rules. In modeling the pipe line, attention is focused on the bent hose, since the pressure drop is significant at the bent compared to a straight line. Figure 3.3a shows a portion of bond graph for the bent hose which includes inertia, compliance and dissipation. Pressure drop in a bent tube is expressed by :

$$P = K \frac{\rho}{2} \left(\frac{Q}{A}\right)^2 \qquad (3.3)$$

Figure 3.3a shows the causality of the dissipation element Rh which indicates flow rate Q is input and pressure P is output. Equation 3.1 matches the causal condition of the R element, namely, pressure P (left-hand side) is given explicitly by flow rate Q (right-hand side).

If the inertia of the fluid is considered to be insignificant in the system, one might like to exclude the

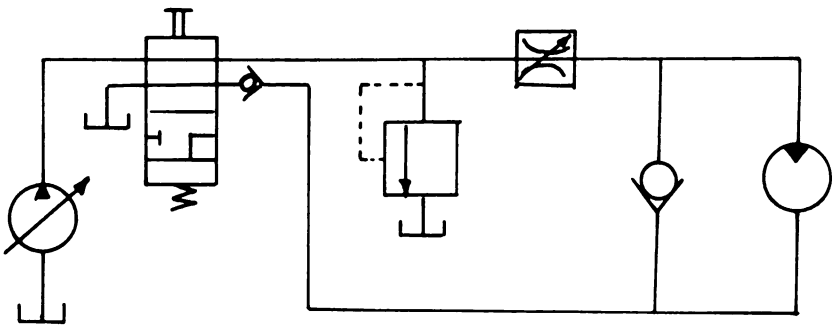

Figure 3.4 Hydraulic system.

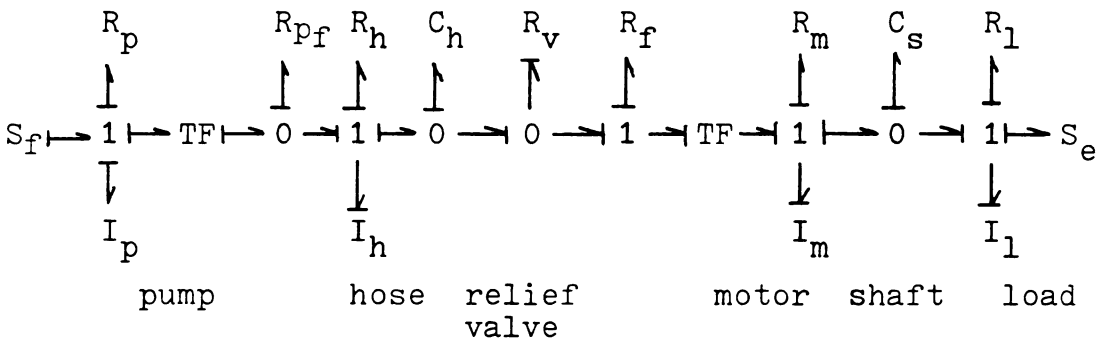

Figure 3.5 Bond graph of hydraulic system.

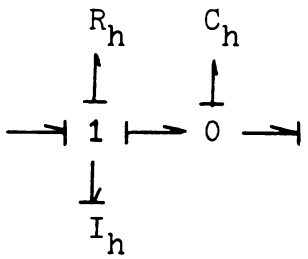

Figure 3.6a

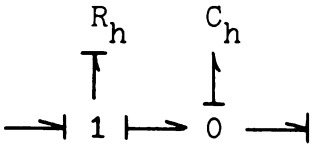

Figure 3.6b

Figure 3.6 (a) Bond graph of bent hose
(b) Modified bond graph of hose.

inertia from the model. Figure 3.3b shows the modified bond graph after the I element is removed. The causal condition of the Rh element is changed, while the rest of the system remains the same. Therefore the input and output relation is changed. Equation 3.1 can be inverted easily to fulfill the causal condition. For the user's convenience it is a good practice to have the simulation program invert the function when requested. The user can ask the program to invert the input-output relation of the function, if required, while supplying the same function as before. However, for some cases the inverse of a function might not exist. Therefore an alternative can be very helpful. An example of this situation is presented in the next section.

3.3 Dry friction

A vehicle can be described by a simple model consisting of masses, springs and a damper as shown in Figure 3.4. Here M is the mass of the vehicle and m is the mass of the tire. The spring function k models the tire. A bond graph model of such a system is expressed in Figure 3.5a, in which causalities are also assigned. Now if we assume the damper is not well lubricated, such that it experiences dry friction when the car is running. The relation between force and velocity for dry friction can be approximately expressed by the saturation function (Figure 3.5b). When the velocity reaches a certain value the friction force will

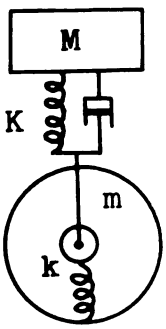

Figure 3.7 Dynamic model of a vehicle.

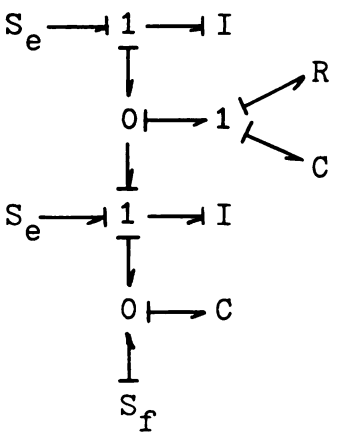

Figure 3.8a

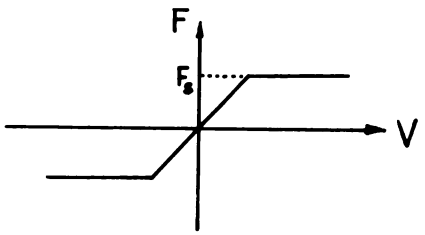

Figure 3.8b

Figure 3.8 (a) Bond graph of vehicle model
(b) Saturation function.

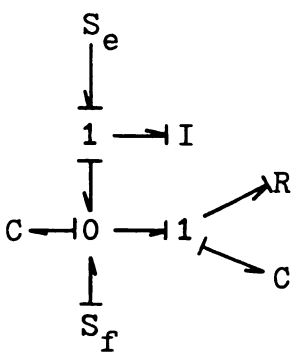

Figure 3.9a

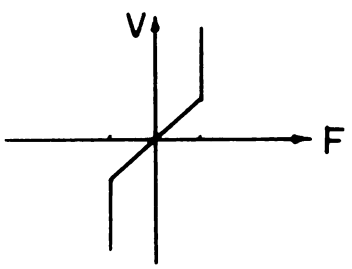

Figure 3.9b

Figure 3.9 (a) Modified bond graph of vehicle
(b) Inverse of saturation.

remain constant (Fs). According to the causality of the R element, velocity is the input to the dissipation element and force is output. The saturation function is suitable for defining the relation between force and velocity.

Often the mass of the tire can be neglected compared to the large mass of the vehicle. A modified bond graph model is shown in Figure 3.6a. The causal condition for the R element has changed. However, the inverse of the saturation function has an undefined region for F, which is now the input variable (Figure 3.6b). From the standpoint of computation serious difficulties might arise in such a system. Therefore an alternative might be necessary. One way of doing that is to define a very large finite value for the slope (instead of infinity).

The examples above show that causal conditions can change when the physical model of the system is modified. Nevertheless, we desire that the simulation program has the ability to handle it. It should not only provide a variety of functions but also be flexible in their causal implementation.

CHAPTER 4

DESIGN OF PROGRAM NOLMOD

4.1 Basic considerations

NOLMOD, which can be imbedded in the ENPORT-5 program, is written in standard FORTRAN V, which helps to make it readily portable. The program has been designed to allow easy implementation on a variety of computers. It has been developed as a completely separate module, which can be used as part of ENPORT-5 or used independently. When used with ENPORT, a new command has been added for invoking NOLMOD (see Appendix A for usage of program). Data are transmitted through common blocks, which can be passed through a memory or disk file interface between ENPORT and NOLMOD. Also it is user-oriented for easy operation.

In ENPORT, bond graphs are taken as input data, then power directions and causality are assigned. In linear models, the next step is to enter parameters of constitutive equations (i.e. for R, C and I, TF and GY). For the nonlinear case, functional relationships between input and output variables must be defined for the constitutive

relations. NOLMOD takes over the nonlinear part at this point. Since we currently restrict the nonlinear elements to be one-ports, only one input and one output variable is required for each element. The basic relation is:

output = function (input)

The most important information for defining such a relationship specifically is causality and the bond types (or element types). Such information is supplied by ENPORT. Based on such information, NOLMOD makes an explicit classification of the basic equations. Both integral and derivative causality, nonlinear 2-port -TF- and -GY-, and multi-port elements are classified within the program. But some classifications are reserved for future implementation. Attention is focussed below on one-port integral storage elements and dissipators.

4.2 Classification of equations

In the discussion below the following definitions are used:

E = generalized effort,

F = generalized flow,

P = generalized momentum,

and Q = genaralized displacement.

According to the different types of elements and causal conditions, equations are classified into the following

eight types:

1. Integral C elements :

$$Ei = FCN(Qi)$$

2. Integral I elements :

$$Fi = FCN(Pi)$$

3. Derivative C elements :

$$Qi = FCN(Ei)$$

4. Derivative I elements :

$$Pi = FCN(Fi)$$

5. R elements :

$$Fi = FCN(Ei)$$

or

$$Ei = FCN(Fi)$$

6. Source elements :

$$Ei = FCN(Time)$$

or

$$Fi = FCN(Time)$$

7. Transformers :

$$Ei = FCN(Ej), \quad Fj = FCN(Fi)$$

or

$$Fi = FCN(Fj), \quad Ej = FCN(Ei)$$

8. Gyrators :

$$Fi = FCN(Ej), \quad Fj = FCN(Ei)$$

or

$$Ei = FCN(Fj), \quad Ej = FCN(Fi)$$

where i and j are the particular bond indices.

In the current NOLMOD implementation source elements (type 6) are handled separately, since usually they are functions only of the indenpendent variable time. ENPORT already has an efficient procedure for handling source

elements, therefore NOLMOD excludes them. For the other types a function library has been built up.

Once the graph causality conditions are known, the NOLMOD program classifies the system-formatted equations into the types indicated in the last section. Input and output variables are defined for every constitutive equation. In some cases the user might have chosen a different input/output orientation for a particular element. Or the causal relation is changed due to modification of the physical model, as we mentioned in Chapter 3. Therefore, for the user's convenience, he (or she) can ask the program to invert the input/output orientation. The program accepts the user-defined equation form and inverts it to obtain the necessary data for the system-formatted equations. For functions that can not be inverted, the program will prompt the user with an error message and ask for further instructions.

4.3 Function library

Some commonly encountered functions are pre-programmed, available for user selection in defining input/outout relations. A list of the library functions is shown in Table 1. Details can be found in Appendix A. The user can check the properties of the function to see if it meets the required behavior of the model. When the user inverts the

Table 1. List of library functions

TYPE	DEFINITION
CONST	Y = C0
LINEAR	Y = C0 + C1*X
POLY	Y = C0 + C1*X + C2*X**2 + C3*X**3 + C4*X**4
SIN	Y = C1 * SIN(C2*X+C3) + C4
ASIN	Y = C1 * ASIN(C2*X+C3) + C4
COS	Y = C1 * COS(C2*X+C3) + C4
ACOS	Y = C1 * ACOS(C2*X+C3) + C4
DSIN	Y = C1 / SIN(C2*X+C3) + C4
DASIN	Y = C1 / ASIN(C2*X+C3) + C4
DCOS	Y = C1 / COS(C2*X+C3) + C4
DACOS	Y = C1 / ACOS(C2*X+C3) + C4
EXP	Y = C1 * EXP(C2*X+C3) + C4
ALOG	Y = C1 * ALOG(C2*X+C3) + C4
ABS2	Y = C1 * X * ABS(X)
ABSQRT	Y = C1 * SIGN(X) * SQRT(ABS(X))
RCPSQR	Y = C0 + C1 / (C2+X**2)
COULOM	Y = C1 * SIGN(X)
BRKPT	Y = Y1 + C1*(X-X1), X <=X1 Y = Y1 + C2*(X-X1), X1< X
SATUR	Y = Y1 , X <=X1 Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2 Y = Y2 , X2<=X

Table 1. (cont'd)

TYPE	DEFINITION	
ADJSAT	Y = Y1 + C1*(X-X1)	, X <=X1
	Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1)	, X1<X<X2
	Y = Y2 + C2*(X-X2)	, X2<=X
PWLIN	PIECEWISE LINEAR	
INTPLO	INTERPOLATION POLYNOMIAL	

input/output orientation, the user-defined function parameters can be read into the program. NOLMOD will invert the function and calculate the appropriate parameters for system-formatted equation. All the parameters are stored in the system-formatted form. Some function inverses are not available. Therefore it is user's responsibility to select other functions that are better suited to serve his (her) purpose.

4.4 Evaluation of functions

A program is provided for evaluation of the defined functions. Such evaluations are based on system-formatted equations (equations 2.7 and 2.8 of Chapter 2), which are determined by causality. Given the values of input variables, the values of output variables are obtained. This program will be used later to evaluate state-space

equations as described in Chapter 2. This is why the evaluation is based on system-formatted equations, not user-defined equations. (See Appendix A for its usage.)

4.5 Structure of NOLMOD

Figure 4.1 shows the basic components of the NOLMOD program. The subroutines enclosed in solid blocks refer to the major parts. Detailed calling sequences are shown in Figure 4.2. All important data are transferred through common blocks COMFILE and COMAREA, which are found in almost every subroutine. COMFILE is the common block set for ENPORT-5 and COMAREA is the common block set for NOLMOD. In general only certain control data are transferred through arguments of calling statements. To have an efficient way for handling the data from the terminal, two subroutines -CMREAD and CMNCPR- are used as the free-reader part of the program. All the data are entered as string variables through CMREAD. Then commands will go through CMNCPR for mapping onto the command list. Constant data go through a decoding process.

In order for NOLMOD to execute successfully certain data must be supplied by ENPORT or by a saved file of ENPORT. Those data include the number of elements (NEL), number of bonds (NBD), element type list (IELLST), types of bonds (IBT), pointer list for NBIMX (NPTR), and the list of

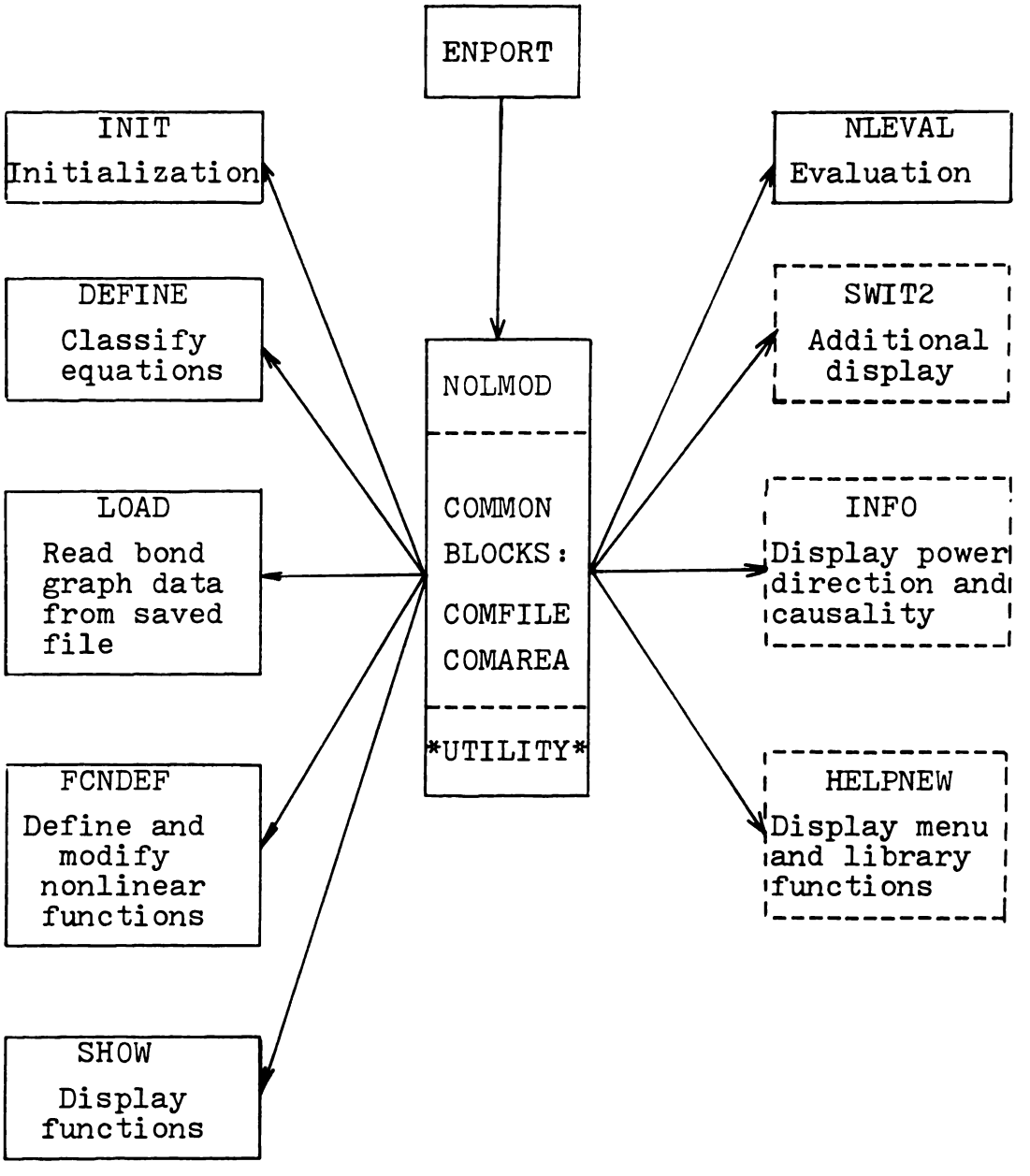

Figure 4.1 Basic components of NOLMOD.

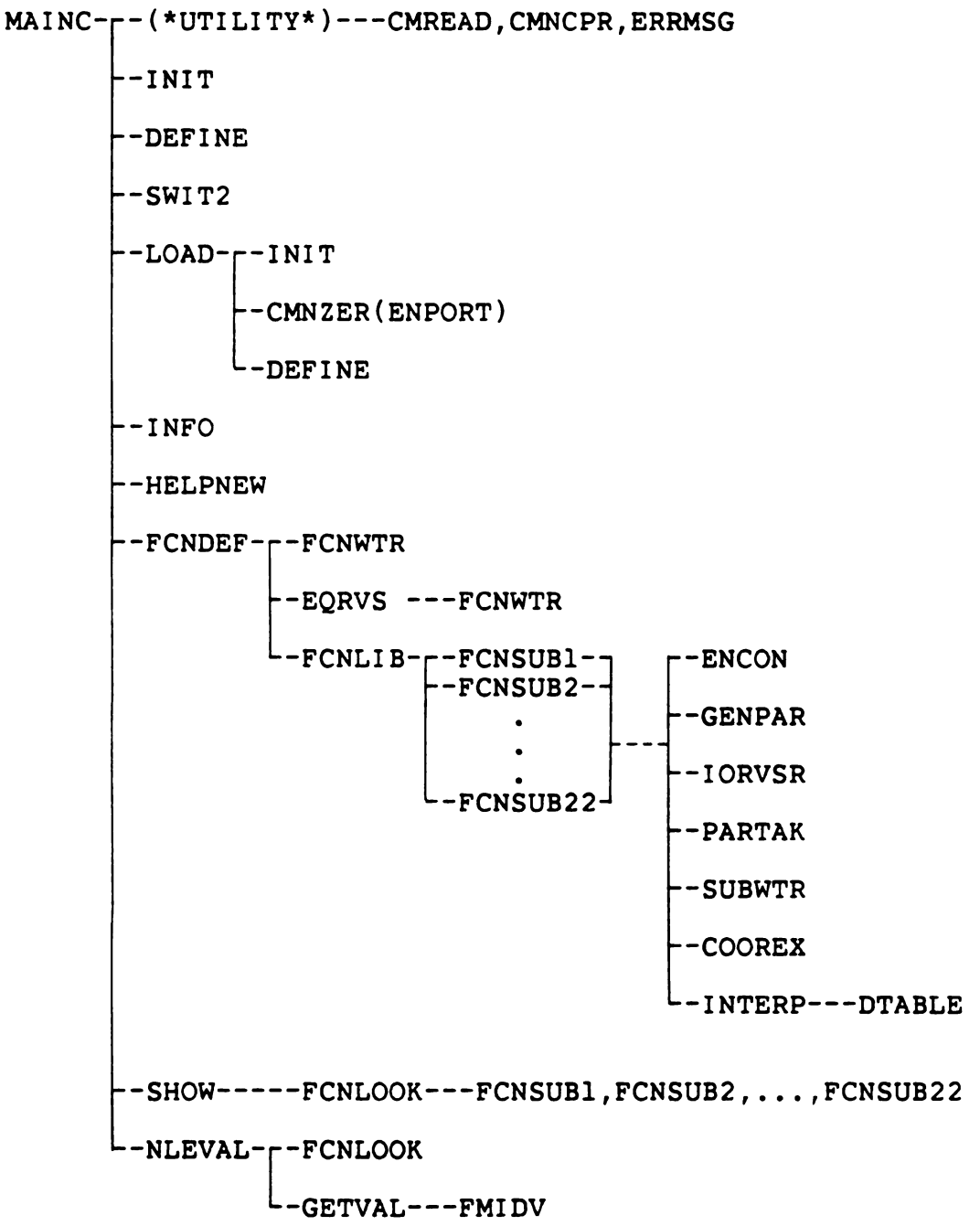

Figure 4.2. Subroutine calling structure for NOLMOD

bonds incident to each element (NBIMX). (Details regarding usage can be found in subroutines RFADBG and CLASS of the ENPORT program.)

Before reading the bond graph data into NOLMOD, INIT should be used to set the NOLMOD into neutral status. LOAD reads the bond graph data from a saved file. It will initialize both ENPORT and NOLMOD before reading the data. Then subroutine DEFINE classifies the basic equations, as described in section 4.2. Subroutine FCNDEF and subsequent subroutines shown in Figure 4.2 perform operations of setting up equations, providing inverse functions, modifying the functions, and verifying the inputs. SHOW is used to display functions in system-formatted form or in user-defined form. And NLEVAL provides on-line evaluation of the functions as defined by the user. A block diagram in Figure 4.3 shows the structure of function library. The interested reader can refer to the source code listed in Appendix C for further details.

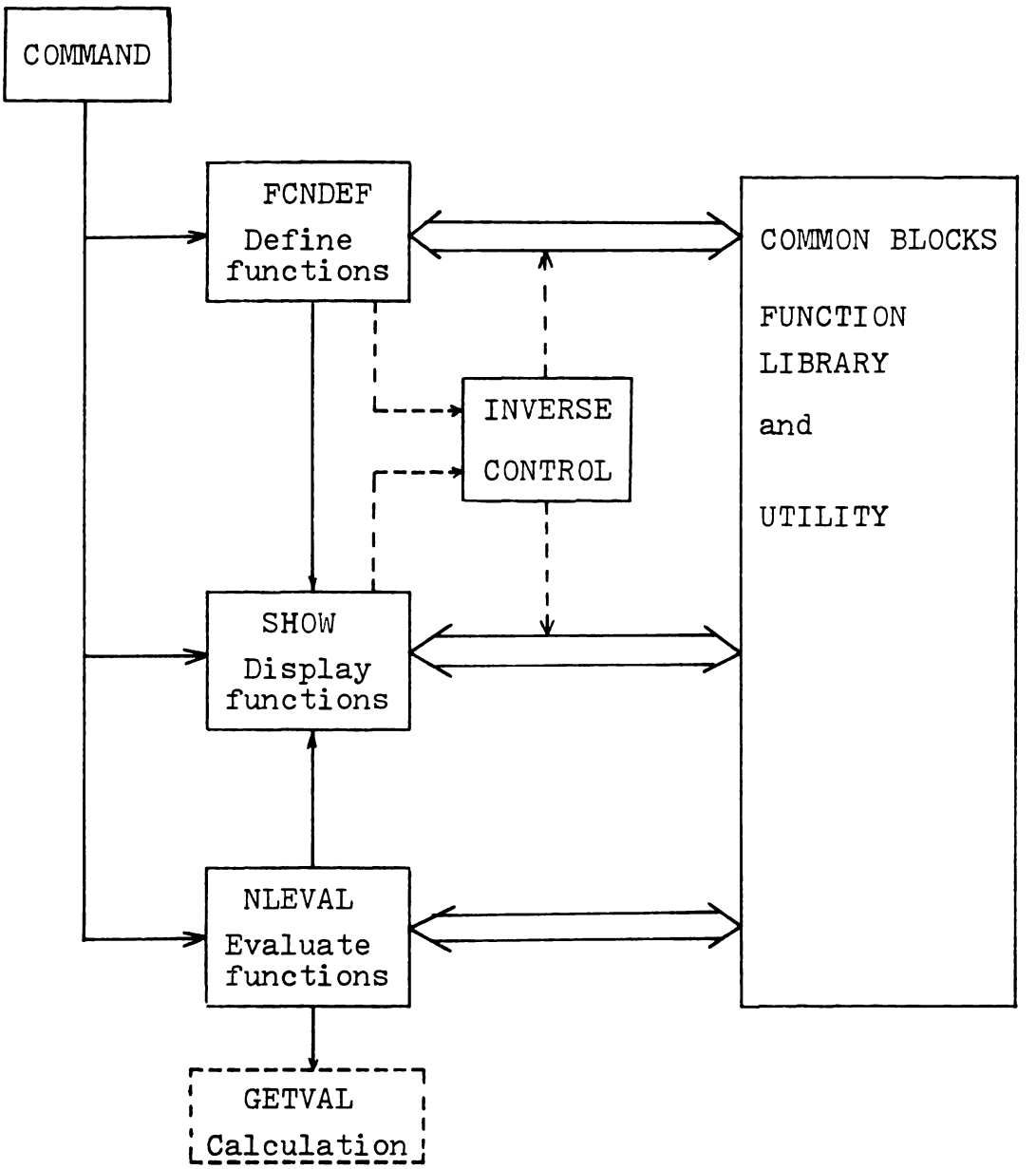

Figure 4.3 Data flow for function library.

CHAPTER 5

CONCLUSION

5.1 Conclusion

For the engineering user who wishes to simulate bond graph models in an interactive environment the ENPORT program has proven useful. However, the need to include nonlinear models has led to the design of a library-based approach for specifying nonlinear constitutive equations. Implemented as both a separate program (NOLMOD) and as a feature of ENPORT, it allows the user to choose from a variety of function types and to set the parameters of the functions in the causal form he (or she) prefers. NOLMOD then inverts the function to match the system causal requirements, if that is necessary. Cases where an inverse does not exist are identified and the decision control is returned to the user.

A manual for the use of NOLMOD has been developed. It defines the function library, the command set, and gives examples of use. Since the program is designed to be used interactively, it is menu-based. On-line error diagnostics

are provided. Anyone familiar with bond graph modeling in general, and ENPORT usage in particular, will find it easy to use NOLMOD.

5.2 Limitations and future work

The functions collected in the library are not sufficient to cover all the different needs that arise in practice. It is expected that the function library will be modified to meet additional needs from time to time. In the future some special functions might be included to fulfill the requirements of different engineering fields. Although the program is well structured for ease of implementation, a programmer is still needed to add new functions to the library. Appendix B describes the procedure to follow.

As mentioned in the previous chapters, at this stage NOLMOD bond graphs are subject to the following restrictions:

1. Storage and dissipation elements are one-ports.
2. Storage elements have integral causality.
3. Transformers and gyrators have constant moduli.

However, non-linear models often are more complicated. A common situation is the need for a modulated transformer, -MTF-, or modulated gyrator, -MGY-. Since the current program can not handle such problems, further implementations are necessary. The recommended sequence of

implementation is as follows :

1. Nonlinear transformers and gyrators.
2. Multi-port storages with integral causality.
3. Multi-port storages with derivative causality.

LIST OF REFERENCES

LIST OF REFERENCES

1. Frank H. Speckhart, Walter L. Green, A guide to using CSMP - The Continuous System Modeling Program, Prentice-Hall, 1976.

2. Continuous System Modeling Program III (CSMP III) Program Reference Manual, SH19-7001-3, IBM, 1975.

3. System/360 Continuous System Modeling Program User's manual, GH20-0367-4, IBM, 1972.

4. DIFFEQ User's Manual, A. H. Case Center, College of Engineering, Michigan State University, 1982.

5. James C. Bowers, John E. O'Reilly and Gary A. Show, SUPER-SCEPTRE User's manual, University of South Florida, Tampa, Florida, 1975.

6. Dean Karnopp and Ronald C. Rosenberg, System Dynamics: A Unified Approach, John Wiley & Sons, 1975.

7. Ronald C. Rosenberg, ENPORT-5 USER'S MANUAL, A. H. Case Center, College of Engineering, Michigan State University, 1981.

8. H. M. Paynter, Analysis and Design of Engineering Systems, M.I.T. Press, Cambridge, 1960.

9. S. H. Crandall, D. C. Karnopp, E. F. Kurtz, and D. C. Pridmore-Brown, Dynamics of Mechanical and Electro mechanical Systems, McGraw-Hill, 1968.

10. L. F. Godbout, Jr. and D. Jordan, On State Equation Descriptions of Linear Differential Systems, ASME Journal of Dynamic systems, Measurement, and Control, Vol. 97, Dec. 1975.

11. R. C. Rosenberg, State-Space Formulation for Bond Graph Models of Multiport Sysyems, ASME Journal of Dynamic systems, Measurement, and Control, Vol. 93, March 1971.

12. Brice Carnahan, H. A. Luther, James O. Wilkes, Applied Numerical Methods, John Wiley & Sons, 1969.

13. Herbert E. Merritt, Hydraulic Control Systems, John Wiley & Sons, 1967.

14. Dean Karnopp, Using bond graphs in nonlinear simulation problems, ASME Journal of Dynamic systems, Measurement, and Control, Vol. 103, Dec. 1981.

APPENDICES

APPENDIX A

HOW TO USE NOLMOD

APPENDIX A

HOW TO USE NOLMOD

A.1 Introduction

NOLMOD is an interactive program for specifying nonlinear constitutive equations that arise in the study of nonlinear dynamic systems using bond graphs.

The program can be invoked from ENPORT by using the command 'FUNCTION'. Available commands include : LOAD, LIST, CHANGE, USER, SYSTEM, HELP and RETURN. Each command can be abbreviated to the smallest set of characters that maintains its uniqueness. After you enter the command 'FUNCTION', NOLMOD will print out a function menu and ask for a choice.

```
    COMMAND$
   FUNCTION

    FUNCTION MENU :
    ---------------
    LOAD  : LOAD GRAPH FROM FILE

    LIST  : LIST OF FUNCTIONS TO BE SPECIFIED

    CHANGE: CHANGE FUNCTION DEFINITION

    USER  : LOOK AT USER DEFINED FUNCTIONS
```

SYSTEM: LOOK AT SYSTEM FORMAT FUNCTIONS

HELP : HELP ABOUT FUNCTION

RETURN: RETURN TO MAIN MENU

ENTER OPTION : (LO, LI, C, U, S, H, R):

There are three additional commands that do not appear in the function menu. These are INFO, DISP and EVAL. The set is used if you wish to develop more understanding of the program operation. They are used for helping the programmer doing further implementation. Usage of the commands is described in next section in alphabetical order. The normal processing sequence is : (DISP), LOAD, (INFO), LIST, (HELP), CHANGE, USER or SYSTEM, (EVAL), RETURN. The commands shown in parentheses are optional.

A.2 Commands and usage

1. CHANGE command

The CHANGE command lets you define or change the function type for each equation by choosing functions from the function library. The default function type is LINEAR. All the functions have been assigned default parameters. The default value is shown in parentheses (). If you accept the default value , just hit the return key. Otherwise, enter the desired value. For a YES/NO question, if you type in the wrong characters the system will treat it as NO. The name of the function also can be abbreviated. You can terminate the operation of CHANGE at the first three stages by entering 'X' in response to the

question. Entering a zero '0' for function index and entering 'QUIT' for a function type will also terminate the CHANGE process. Once you have been asked for function parameters, you must continue the process. No termination can be done. To leave quicker just accept the defaults. The rules cited also hold for commands USER and SYSTEM. For example:

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
C

CHANGE ALL ? (NO):


ENTER FUNCTION INDEX :
1

FUNCTION   1  NOW DEFINED AS :

     F 1 = LINEAR (P 1)

REVERSE INPUT AND OUTPUT ? (NO):
Y

FUNCTION   1  NOW DEFINED AS :

     P 1 = LINEAR (F 1)

FUNCTION TYPE ?
SIN

          P 1 = C1 * SIN( C2 * F 1 + C3 ) + C4

VALUE OF C1 ? (  1.0000 ):
2
VALUE OF C2 ? (  1.0000 ):
3
VALUE OF C3 ? (  0.0000 ):
-4
VALUE OF C4 ? (  0.0000 ):
0.31

VERIFY ? (NO):
Y

   P 1 = 2.0000E+00 * SIN ( 3.0000E+00 * F 1 + -4.0000E+00 )
       + 3.1000E-01

CHANGE ANOTHER FUNCTION ? (YES):
```

```
YES

ENTER FUNCTION INDEX :
X

ENTER OPTION : (LO, LI, C, U, S, H, R ):
```

2. DISP command

This command turns on an extra print-switch that enables you to view detailed information. It might help you to understand the internal operation better.

3. EVAL command

This command is used to evaluate the functions that were defined previously by the CHANGE operation. You can ask for a function index to be evaluated and set the numerical value of the input variable of that particular function. Note that all functions appear in system-formatted form.

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
EVAL

GIVE FUNCTION NUMBER :
1

     F 1 =  3.3333E-01 * ASIN(  5.0000E-01 * P 1 + -1.5500E-01 )
         +  1.3333E+00

GIVE THE VALUE OF INPUT VARIABLE :
0.5

VALUE OF OUTPUT VARIABLE IS :   1.3650E+00


EVALUATE ANOTHER FUNCTION ? (YES):
N
```

4. FILE command

This is an ENPORT command used for saving data for later use. (See ENPORT-5 User's Manual for a detailed description of ENPORT commands.) Useful option for NOLMOD is:

FILE

GRAPH filename

The file name must not have more than four characters. The saved file can be read back by NOLMOD's LOAD command or ENPORT's LOAD command, which depends upon where you would like to restart it. However NOLMOD only accepts a saved GRAPH file.

5. HELP command

The HELP command provides the list of different library functions that you can use to define the functions. Detail descriptions will be presented in the next section.

6. INFO command

The INFO command prints additional information concerning the causality and power direction of the bond graph.

7. LIST command

The LIST command prints out system-formatted equations at the terminal. Input and output variables for C and I ports are expressed by energy and co-energy variables: E, F, P and Q. Causal relationships are incorporated. The input variable is on right of the equation, the output variable is on the left side. Each function is labeled for identification. The number after FCN is the function index. For example:

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
LIST

INTEGRAL STORAGE ELEMENTS

     E 3 = FCN 3 (Q 3)

     F 1 = FCN 1 (P 1)

DISSIPATION ELEMENTS

     E 5 = FCN 5 (F 5)

SOURCE ELEMENTS

     E 2 = FCN ( TIME )

     F 4 = FCN ( TIME )
```

8. LOAD command

LOAD reads graph information from a previously saved file containing a causal bond graph. Such a file can be created by the ENPORT command FILE (see FILE command). If the user already has bond graph data in ENPORT, the LOAD command should not be used. Every time

the LOAD command is used common blocks will be initialized for accepting new data, existing data will be destroyed. The User will be asked for a file name after the LOAD command is given. If the file is loaded successfully a message will shown on the screen. For example :

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
LOAD

GIVE FILE NAME :
SAMPLE

*** GRAPH FILE LOADING COMPLETED ***
```

9. RETURN command

The RETURN command put you back under ENPORT COMMAND$ control.

10. SYSTEM command

The SYSTEM command lists the system-formatted function for each equation on the terminal. For example:

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
S

LOOK AT ALL ? (NO):

ENTER FUNCTION INDEX :
1

 FUNCTION  1 :

   F 1 = 3.3333E-01 * ASIN( 5.0000E-01 * P 1 + -1.5500E-01 )
       + 1.3333E+00

LOOK AT ANOTHER FUNCTION ? (YES):
NO
```

11. USER command

The USER command lists the user-defined function for each equation at the terminal. For example:

```
ENTER OPTION : (LO, LI, C, U, S, H, R ):
U

LOOK AT ALL ? (NO):

ENTER FUNCTION INDEX :
1

 FUNCTION  1 :

   P 1 = 2.0000E+00 * SIN ( 3.0000E+00 * F 1 + -4.0000E+00 )
       + 3.1000E-01

LOOK AT ANOTHER FUNCTION ? (YES):
N
```

A.3 Function library

The function library is the heart of the program. The user may choose desired functions from the library. Diagrams of several functions are shown in Figures A.1 to Figure A.9. Meanings of the parameter data are explained by the diagrams. Available functions are described below, where Y is the output, X is the input, and Ci are parameters of the function. A table arranged in alphabetical order is shown in Table 2.

1. CONST Y = C0

2. LINEAR Y = C0 + C1*X

3. POLY Y = C0 + C1*X + C2*X**2 + C3*X**3 + C4*X**4

4. SIN Y = C1 * SIN(C2*X+C3) + C4

5. COS Y = C1 * COS(C2*X+C3) + C4

6. ASIN Y = C1 * ASIN(C2*X+C3) + C4

7. ACOS Y = C1 * ACOS(C2*X+C3) + C4

8. DSIN Y = C1 / SIN(C2*X+C3) + C4

9. DCOS Y = C1 / COS(C2*X+C3) + C4

10. DASIN Y = C1 / ASIN(C2*X+C3) + C4

11. DACOS Y = C1 / ACOS(C2*X+C3) + C4

12. EXP Y = C1 * EXP(C2*X+C3) + C4

13. ALOG Y = C1 * ALOG(C2*X+C3) + C4

14. ABS2 Y = C1 * X * ABS(X) (Fig. A.1)

15. ABSQRT Y = C1 * SIGN(X) * SQRT(ABS(X)) (Fig. A.2)

16. RCPSQR Y = C0 + C1 / (1+(X/C2)**2) (Fig. A.3)

17. COULOM Y = C1 * SIGN(X)

Coulomb function. (Fig. A.4)

18. BRKPT Y = Y1 + C1*(X-X1), X <=X1
Y = Y1 + C2*(X-X1), X1< X

Select one data point and define two slopes. (Fig. A.5)

19. SATUR Y = Y1 , X <=X1
Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2
Y = Y2 , X2<=X

Saturation function. (Fig. A.6)

20. ADJSAT Y = Y1 + C1*(X-X1) , X <=X1
Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2
Y = Y2 + C2*(X-X2) , X2<=X

Select two data points and define two slopes. (Fig. A.7) This function has great flexibility for approximating various special functions. For example, it can be used as an inverse of function 18 by defining very large slopes at both ends. (Fig. A.8)

21. PWLIN PIECEWISE LINEAR

Linear interpolation between data points. Maximum number of data point is set to 20. User also defines the slopes leading to the first point and from the last point.

22. INTPLO INTERPOLATION POLYNOMIAL

Function 21 and 22 act as arbitrary function generators, and both are based on Newton's interpolation polynomial method. Function 21 is linear interpolation, and function 22 is used as a high-degree interpolation polynomial. The degree of interpolation can be 1, 2, 3,... up to 19. Usually a degree of two or three is commonly used. End slopes are not supplied. The maximum number of data points is 20. (Fig. A.9) The center part of the interpolation results is more accurate than those at the ends. Therefore it is helpful to extend the data range to be wider than the range of interest.

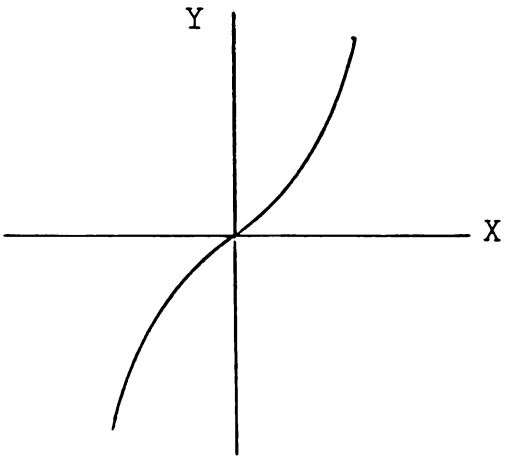

Figure A.1 Function ABS2

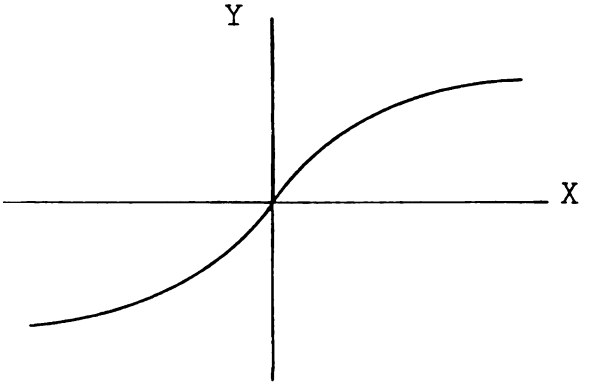

Figure A.2 Function ABSQRT

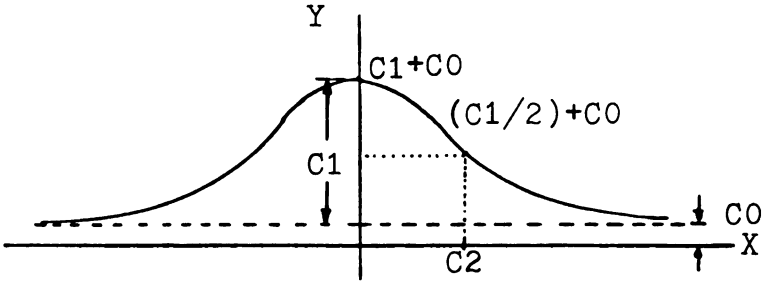

Figure A.3 Function RCPSQR

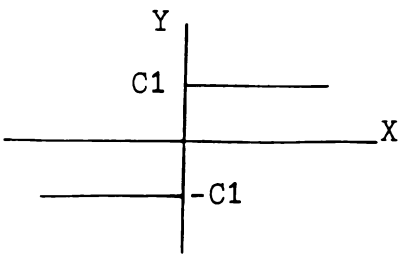

Figure A.4 Function COULOM

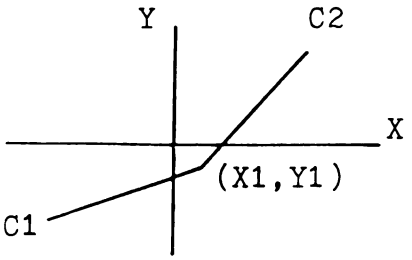

Figure A.5 Function BRKPT

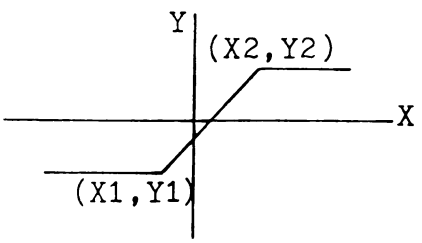

Figure A.6 Function SATUR

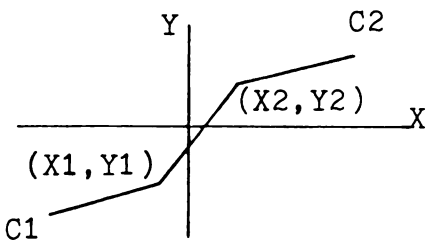

Figure A.7 Function ADJSAT

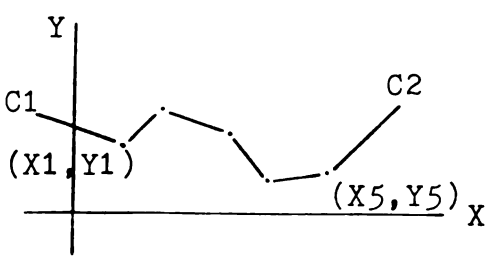

Figure A.8 Function PWLIN

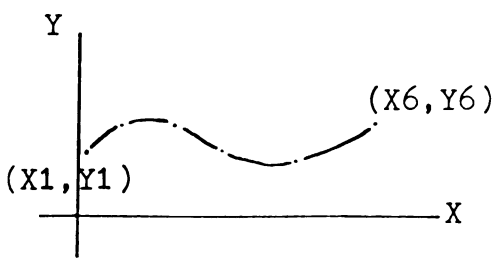

Figure A.9 Function INTPLO

Table 2. List of functions in alphabetic order

TYPE	DEFINITION
ABS2	Y = C1 * X * ABS(X)
ABSQRT	Y = C1 * SIGN(X) * SQRT(ABS(X))
ACOS	Y = C1 * ACOS(C2*X+C3) + C4
ADJSAT	Y = Y1 + C1*(X-X1) , X <=X1 Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2 Y = Y2 + C2*(X-X2) , X2<=X
ALOG	Y = C1 * ALOG(C2*X+C3) + C4
ASIN	Y = C1 * ASIN(C2*X+C3) + C4
BRKPT	Y = Y1 + C1*(X-X1), X <=X1 Y = Y1 + C2*(X-X1), X1< X
CONST	Y = C0
COS	Y = C1 * COS(C2*X+C3) + C4
COULOM	Y = C1 * SIGN(X)
DACOS	Y = C1 / ACOS(C2*X+C3) + C4
DASIN	Y = C1 / ASIN(C2*X+C3) + C4
DCOS	Y = C1 / COS(C2*X+C3) + C4
DSIN	Y = C1 / SIN(C2*X+C3) + C4
EXP	Y = C1 * EXP(C2*X+C3) + C4
INTPLO	INTERPOLATION POLYNOMIAL
LINEAR	Y = C0 + C1*X
POLY	Y = C0 + C1*X + C2*X**2 + C3*X**3 + C4*X**4
PWLIN	PIECEWISE LINEAR

Table 2. (cont'd)

TYPE	DEFINITION	
RCPSQR	Y = C0 + C1 / (C2+X**2)	
SATUR	Y = Y1	, X <=X1
	Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1)	, X1<X<X2
	Y = Y2	, X2<=X
SIN	Y = C1 * SIN(C2*X+C3) + C4	

APPENDIX B

IMPLEMENTATION OF A NEW FUNCTION

APPENDIX B

IMPLEMENTATION OF A NEW FUNTION

If a new function will be added to the library, changes must be made to the following subroutines : FCNLIB, FCNLOOK, GETVAL and HELPNEW. Source code for these subroutines can be found in Appendix C. Also a new subroutine for the function is needed. Let's do an example to show how to inplement a new function in the library. Suppose RCPSQR is a new function to be added and there already are 21 functions in the library. (Description of function RCPSQR can be found in Appendix A.) Therefore the new one will be function number 22. This number will serve as the identifier of the function type. The following modifications should be made.

1. Subroutine FCNLIB

There are three important variables to be defined, namely, IFCNTP, FCNSET and PARDF. Variable IFCNTP stores the function type number. Therefore set IFCNTP(I)=22 for new function RCPSQR. FCNSET is an array storing the name of the functions, therefore add the new name at the 22nd position of the list. (Data statement is in subroutine FCNLIB.) PARDF is a 30-by-10 matrix which stores the default values of the

parameters of the functions. Hence, each function has ten storage spaces available for its parameters. Unused spaces are filled with zeros. The number of row stands for the function type number. RCPSQR is the 22nd function in the library; therefore row 22 stores the default parameters of the function RCPSQR. Set a new row of appropriate default parameters for the new function. At last add a call statement for the new subroutine:

```
   ELSE IF(CMATCH(IFOUND).EQ.'RCPSQR')THEN
     IFCNTP(I)=22
     CALL FCNSUB22(I)
```

2. Subroutine FCNLOOK

Just add two line of statement like these:

```
   ELSE IF (IFCNTP(I).EQ.22) THEN
     CALL FCNSUB22(I)
```

3. Subroutine GETVAL

A few statements for calculating the output value (Y) are needed. Add lines like these:

```
   ELSE IF(NFCNTP.EQ.22)THEN
     YOUTP=EQPAR(NOEQ,1)+EQPAR(NOEQ,2) / (1+(XINP/EQPAR(NOEQ,3))**2)
     RETURN
```

where YOUTP is the output variable, XINP is the input variable and the EQPAR's are the parameters of the function. The index NOEQ is the equation number.

4. Subroutine HELPNEW

This subroutine prints library functions on the terminal when HELP command is used. Add a line of print statement to describe the new function.

```
PRINT *,'RCPSQR      Y = C0 + C1 / (C2+X**2)'
```

5. New subroutine FCNSUB22

The subroutine performs the functions of reading parameters and display of the equation. The very first thing that the subroutine should do is to check whether the user asked for inversion or not. If yes, and the function inversion is not available, a warning message should be displayed on the screen. The user should find another alternative to fulfill his requirements. If the inversion is available, some processes should be done which depend on the nature of the function itself. The reader can refer to other function subroutines of the program. Subroutine GENCON is used to accept input of constant parameters C's. And subroutine GENPAR is used for reading a pair of (X,Y) coordinates. Descriptions of the arguments for both subroutines can be found in the source code in Appendix C.

APPENDIC C

SOURCE CODE FOR NOLMOD

APPENDIX C

SOURCE CODE FOR NOLMOD

Subroutines in this appendix appear in the following sequence :

1. MAINC	17. PARTAK	33. FCNSUB11
2. INIT	18. SUBWTR	34. FCNSUB12
3. INFO	19. IOPVSR	35. FCNSUB13
4. HELPNEW	20. GENPAR	36. FCNSUB14
5. SWIT2	21. GENCON	37. FCNSUB15
6. LOAD	22. COOREX	38. FCNSUB16
7. CMREAD	23. FCNSUB1	39. FCNSUB17
8. CMNCPR	24. FCNSUB2	40. FCNSUB18
9. ERRMSG	25. FCNSUB3	41. FCNSUB19
10. DEFINE	26. FCNSUB4	42. FCNSUB20
11. FCNDEF	27. FCNSUB5	43. INTERP
12. EQRVS	28. FCNSUB6	44. DTABLE
13. SHOW	29. FCNSUB7	45. FMIDV
14. FCNWTR	30. FCNSUB8	46. FCNSUB22
15. FCNLIB	31. FCNSUB9	47. NLEVAL
16. FCNLOOK	32. FCNSUB10	48. GETVAL

49. COMFILE and COMAREA (common blocks)

```
C---MAINC-----------------------------------------------------
C
C               MAIN DRIVER
C
      SUBROUTINE MAINC
C
C-----WRITTEN BY: TSUN-YU CHOU, JUNE 1982, MICHIGAN STATE U.
C-----LATEST UPDATE: FEB. 1983, TYC.
C
C     SOURCE PROGRAM FOR NONLINEAR SIMULATION OF DYNAMIC SYSTEMS
C     BASED ON BOND GRAH THEORY.   THIS PROGRAM CAN BE USED
C     INDEPENDENTLY OR FUNCTION AS A MODULE OF ENPORT PROGRAM.
C     (CALLED BY ENPORT BY THE COMAND "FUNCTION")
C
C     OPTIONS CONTROL :
C          OPT1 : (LOGIN, HELP )
C                 (DEFINE, CHANGE, LOOK )
C          OPT2 : (L, U, S, N, X, Q)
C          OPT3 : (LINEAR, INTPLO )
C
C-----COMMON BLOCKS-----
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      DATA CMSET1/'LOAD  ','LIST  ','USER  ','SYSTEM','CHANGE',
     +            'HELP  ','ZMND07','ZMND08','ZMND09','RETURN',
     +            'ZMND11','ZMND12','INFO  ','DISP  ','EVAL  ',
     +            'ZMND16','ZMND17','ZMND18','ZMND19','ZMND20',
     +            'ZMND21','ZMND22','ZMND23','ZMND24','ZMND25',
     +            'ZMND26','ZMND27','ZMND28','ZMND29','ZMND30'/
C-------------------------------------------------------------
      DATA ILUN,JLUN/1,1/
      IPNT(1)=0
      CALL INIT
      CALL DEFINE
C-------------------------------------------------------------
10    OPT1='LOGIN '
      CALL HELPNEW
100   WRITE (JLUN,9000)
9000  FORMAT(/,'ENTER OPTION : (LO, LI, C, U, S, H, R ): ')
      CALL CMREAD
      IF (NOCMN .EQ. 0) GO TO 10
      CALL CMNCPR(CMSET1,30)
      IF(IFOUND .EQ. 0) THEN
          CALL ERRMSG(1)
          GO TO 10
      ENDIF
      IF(IFOUND .GT. 1) THEN
          CALL ERRMSG(2)
          GO TO 10
      ENDIF
C
C-----PROCESS-----
```

```
C
C
C---LOAD----------
      IF (CMATCH(IFOUND) .EQ. 'LOAD') THEN
      CALL LOAD
C
C---LIST----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'LIST')THEN
      OPT2='L'
      CALL DEFINE
C
C---USER----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'USER')THEN
      OPT2='U'
      CALL SHOW
C
C---SYSTEM----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'SYSTEM')THEN
      OPT2='S'
      CALL SHOW
C
C---CHANGE----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'CHANGE')THEN
      OPT2='U'
      CALL FCNDEF
C
C---HELP----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'HELP') THEN
      OPT1='HELP  '
      CALL HELPNEW
C
C---RETURN----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'RETURN' ) THEN
C     CALL EXIT
      PRINT *,' '
      RETURN
C
C---INFO----------
      ELSE IF(CMATCH(IFOUND) .EQ.'INFO')THEN
      CALL INFO
C
C---DISP  ----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'DISP') THEN
      CALL SWIT2
C
C---EVAL----------
      ELSE IF (CMATCH(IFOUND) .EQ. 'EVAL')THEN
      CALL NLEVAL
C
C---OTHERWISE----------
C
      ELSE
          CALL ERRMSG(3)
```

```
C
      ENDIF
C
C-------------------------------------------------------------
      IF (NOCMN .LT. 100) GO TO 100
C
      STOP
      END
C
C---INIT------------------------------------------------------
C
      SUBROUTINE INIT
C
C     INITIALIZATION
C
$INSERT COMAREA
C-------------------------------------------------------------
      OPT2='U'
C
      DO 10 I=1,50
      IFCNTP(I)=2
      RVS(I)=0.
      DO 10 J=1,10
      EQPAR(I,J)=PARDF(2,J)
10    CONTINUE
C
      RETURN
      END
C
C---INFO------------------------------------------------------
C
      SUBROUTINE INFO
C
C     PROVIDE ADDITIONAL INFORMATION ABOUT CAUSALITY AND POWER.
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      DATA INFSET/'CAUSAL','POWER ','HOME  ','SWIT04','SWIT05',
     +            'SWIT06','SWIT07','SWIT08','SWIT09','SWIT10',
     +            'SWIT11','SWIT12','SWIT13','SWIT14','SWIT15',
     +            'SWIT16','SWIT17','SWIT18','SWIT19','SWIT20'/
C-------------------------------------------------------------
10    WRITE(JLUN,20)
20    FORMAT(/,'INFO : CAUSAL, POWER, HOME ?'/)
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 10
      CALL CMNCPR(INFSET,20)
C
      IF(IFOUND.EQ.0)THEN
        CALL ERRMSG(1)
        GOTO 10
      ELSE IF(IFOUND.GT.1)THEN
        CALL ERRMSG(2)
```

```
        GOTO 10
      ENDIF
C
C-----CAUSAL---------
C
      IF(CMATCH(IFOUND) .EQ.'CAUSAL') THEN
C
        NBD2=NBD*2
        DO 50 K=1,NBD2
        IF(ICMX(K).GE.5) NBIMX(K)=-NBIMX(K)
50      CONTINUE
        WRITE (JLUN,5010)
        NP1=1
        DO 52 I= 1,NEL
        NP2=NPTR(I+1)-1
        WRITE (JLUN,5020) I,IELNAM(I),(NBIMX(J),J=NP1,NP2)
52      NP1=NP2+1
C
        DO 54 K=1,NBD2
54      NBIMX(K)=IABS(NBIMX(K))
        GOTO 10
C
5010    FORMAT(1X/' THE BOND GRAPH STRUCTURE WITH CAUSALITY IS'
     +  ,/,3X,'(+ = STROKE END, - = NON STROKE END)'/)
5020    FORMAT(2X,I3,2X,A2,2X,4(I5I4/9X))
C
C-----POWER------------
C
      ELSE IF(CMATCH(IFOUND).EQ.'POWER')THEN
C
        WRITE (JLUN,1000)
        N2=0
101     N1=N2+1
        N2=NBD
        IF ((N2-N1).GT.9) N2=N1+9
         WRITE(JLUN,1010) (N,N=N1,N2)
         WRITE(JLUN,1020) (IELNAM(IBMX(N,1)),N=N1,N2)
         WRITE(JLUN,1030) (IELNAM(IBMX(N,2)),N=N1,N2)
         IF (N2.LT.NBD) GOTO 101
         GOTO 10
C
1000  FORMAT(1X/' THE POWER DIRECTIONS')
1010  FORMAT(1X/' BOND',2X,I2,9(3X,I2))
1020  FORMAT(' FROM',2X,A4,9(1X,A4))
1030  FORMAT('   TO',2X,A4,9(1X,A4))
C
C-----HOME----------
C
      ELSE IF(CMATCH(IFOUND).EQ.'HOME')THEN
        RETURN
C
C
C-----OTHERWISE---------
```

```
C
      ELSE
       CALL ERRMSG(3)
        GOTO 10
C
      ENDIF
C
      RETURN
      END
C
C---HELPNEW-----------------------------------------------------
C
      SUBROUTINE HELPNEW
C
C    OPT1 ( HELP, LOGIN )
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C
      IF(OPT1.EQ.'HELP   ')GOTO 20
C
      PRINT *,' '
      PRINT *,'FUNCTION MENU :'
      PRINT *,'---------------'
      PRINT *,' '
      PRINT *,'LOAD   : LOAD GRAPH FROM FILE'
      PRINT *,'LIST   : LIST OF FUNCTIONS TO BE SPECIFIED'
      PRINT *,'CHANGE: CHANGE FUNCTION DEFINITION'
      PRINT *,'USER   : LOOK AT USER DEFINED FUNCTIONS'
      PRINT *,'SYSTEM: LOOK AT SYSTEM FORMAT FUNCTIONS'
      PRINT *,'HELP   : HELP ABOUT FUNCTION'
      PRINT *,'RETURN: RETURN TO MAIN MENU'
      PRINT *,' '
C
      IF(OPT1.EQ.'LOGIN ')RETURN
C
20    PRINT *,' '
      PRINT *,'WHEN USING COMMANDS  "CHANGE",  "USER", "SYSTEM",'
      PRINT *,'YOU ALWAYS CAN TERMINATE THE PROCESS BY ENTER "X"'
      PRINT *,'FOR THE FOLLOWING QUESTIONS :'
      PRINT *,'    (1) CHANGE ALL?   OR   LOOK AT ALL?  '
      PRINT *,'    (2) ENTER FUNCTION INDEX? '
      PRINT *,'    (3) REVERSE INPUT AND OUTPUT?'
      PRINT *,'ENTER "0" FOR FUNCTION INDEX AND ENTER'
      PRINT *,'"QUIT" FOR FUNCTION TYPE ALSO TERMINATE'
      PRINT *,'THE PROCESS AND PUT YOU BACK TO MAIN MENU.'
      PRINT *,' '
      PRINT *,'HIT RETURN KEY TO CONTINUE.'
      PRINT *,'(IF YOU ARE USING TEKTRONIX TERMINAL,'
      PRINT *,' PLEASE PAGE THE SCREEN FIRST.)'
      CALL CMREAD
       PRINT *,'FUNCTION LIBRARY :'
       PRINT *,' '
```

```
PRINT *,'TYPE        DEFINITION'
PRINT *,'-------------------------------------------------'
PRINT *,'CONST       Y = CO '
PRINT *,' '
PRINT *,'LINEAR      Y = CO + C1*X'
PRINT *,' '
PRINT *,'POLY        Y = CO + C1*X + C2*X**2 + C3*X**3 + C4*X**4'
PRINT *,' '
PRINT *,'SIN         Y = C1 * SIN(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'ASIN        Y = C1 * ASIN(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'COS         Y = C1 * COS(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'ACOS        Y = C1 * ACOS(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'DSIN        Y = C1 / SIN(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'DASIN       Y = C1 / ASIN(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'DCOS        Y = C1 / COS(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'DACOS       Y = C1 / ACOS(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'EXP         Y = C1 * EXP(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'ALOG        Y = C1 * ALOG(C2*X+C3) + C4'
PRINT *,' '
PRINT *,'*** MORE, PAGE AND HIT (RETURN) KEY.'
CALL CMREAD
PRINT *,' '
PRINT *,'TYPE        DEFINITION'
PRINT *,'-------------------------------------------------'
PRINT *,'ABS2        Y = C1 * X * ABS(X)'
PRINT *,' '
PRINT *,'ABSQRT      Y = C1 * SIGN(X) * SQRT(ABS(X))'
PRINT *,' '
PRINT *,'RCPSQR      Y = CO + C1 / (C2+X**2)'
PRINT *,' '
PRINT *,'COULOM      Y = C1 * SIGN(X)'
PRINT *,' '
PRINT *,'BRKPT       Y = Y1 + C1*(X-X1),    X <=X1'
PRINT *,'            Y = Y1 + C2*(X-X1),    X1< X'
PRINT *,' '
PRINT *,'SATUR       Y = Y1                                , X <=X1'
PRINT *,'            Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2'
PRINT *,'            Y = Y2                                , X2<=X'
PRINT *,' '
PRINT *,'ADJSAT      Y = Y1 + C1*(X-X1)                    , X <=X1'
PRINT *,'            Y = Y1 + ((Y1-Y2)/(X1-X2))*(X-X1), X1<X<X2'
PRINT *,'            Y = Y2 + C2*(X-X2)                    , X2<=X'
PRINT *,' '
PRINT *,'PWLIN       PIECEWISE LINEAR'
```

```
       PRINT *,' '
       PRINT *,'INTPLO      INTERPOLATION POLYNOMIAL'
C
      RETURN
      END
C
C---SWIT2-----------------------------------------------------
C
      SUBROUTINE SWIT2
C
C     SWITCH ROUTINE-- FOR ADDITION INFORMATION DISPLAY.
C     ONLY 1 SWITCH IS USED (IPNT(1)) RIGHT NOW.  UP TO 10
C     SWITCHS CAN BE USED FOR FUTURE EXPANSION.
C     1=ON, 0=OFF.
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      ITRANS=IPNT(1)
      IF(ITRANS.EQ.0) THEN
        IPNT(1)=1
      ELSE IF(ITRANS.EQ.1) THEN
        IPNT(1)=0
      ENDIF
C
      RETURN
      END
C
C---LOAD------------------------------------------------------
C
      SUBROUTINE LOAD
C
C-----WRITTEN BY : TSUN-YU CHOU,  JULY 1982
C
C-----LOAD DATA FROM SAVED FILE FOR FURTHER PROCESS.(BY SUB DEFINE)
C     DATA FILE CONTAIN NECESSARY GRAPH DATA SAVED BY
C     ENPORT COMMAND :
C                      FILE
C                      GRAPH filaname
C
      LOGICAL       EXIST, OPENED
      CHARACTER*80  FILNAM
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C
C-----INITIALIZE THE PARAMETER LIST.
C
      CALL INIT
      CALL CMNZER
C
C-----READ FILE NAME.
C
```

```
10     WRITE(JLUN,1000)
       CALL CMREAD
       IF(NOCMN.EQ.0)GOTO10
       IF(CMN.EQ.':') RETURN
       FILNAM=CMN
C
C-----CHECK IF FILE EXIST.
C
       INQUIRE(FILE=FILNAM,EXIST=EXIST)
       IF(EXIST .EQV..FALSE.)THEN
        WRITE(JLUN,1001)
        RETURN
C
C-----FILE EXIST, SEARCH FOR UNOPENED UNIT.
C
       ELSE
        NUNIT=5
30      INQUIRE(UNIT=NUNIT,OPENED=OPENED)
C
         IF(OPENED.EQV..TRUE.)THEN
          NUNIT=NUNIT+1
          GOTO 30
         ELSE
          OPEN(NUNIT,FILE=FILNAM)
         ENDIF
C
       ENDIF
C
C-----HEADING INFO.
C
       READ(NUNIT,1010,ERR=100,END=100) NAMEF
       READ(NUNIT,1010,ERR=100,END=100) (HEAD(I),I=1,20)
       READ(NUNIT,1040,ERR=100,END=100) ILUN,JLUN
C
C-----GRAPH DATA BLOCK.
C
       READ(NUNIT,1010,ERR=100,END=100) KWORD
       READ(NUNIT,1040,ERR=100,END=100) NEL,NBD
       READ(NUNIT,1010,ERR=100,END=100) (IELNAM(I),I=1,NEL)
       READ(NUNIT,1020,ERR=100,END=100) (IELLST(I),I=1,NEL)
       I2=2*NBD
       READ(NUNIT,1020,ERR=100,END=100) (NBIMX(I),I=1,I2)
       READ(NUNIT,1020,ERR=100,END=100) ((IBMX(I,J),J=1,2),I=1,NBD)
       I2=NEL+1
       READ(NUNIT,1020,ERR=100,END=100) (NPTR(I),I=1,I2)
C
C-----PARAMETERS
C
       I2=NEL+1
       READ(NUNIT,1010,ERR=100,END=100) KWORD
       READ(NUNIT,1020,ERR=100,END=100) (IPTR(I),I=1,I2),NPAR,IPFLG
       READ(NUNIT,1050,ERR=100,END=100) (PAR(I),I=1,NPAR)
C
```

```
C-----CAUSALITY
C
      READ(NUNIT,1010,ERR=100,END=100) KWORD
      I2=2*NBD
      READ(NUNIT,1020,ERR=100,END=100) (ICMX(I),I=1,I2)
      READ(NUNIT,1020,ERR=100,END=100) (IBEQ(I),I=1,NBD)
      READ(NUNIT,1020,ERR=100,END=100) (IBT(I),I=1,NBD)
      READ(NUNIT,1040,ERR=100,END=100) NBEX,NFI,NFD,NFL,NFS,NBIN
C
      CLOSE(NUNIT,STATUS='KEEP')
C-------------------------------------------------------------
C-----ECHO OF READING
C
      IF (IPNT(1).EQ.0) GOTO 50
C
      WRITE(JLUN,1010) NAMEF
      WRITE(JLUN,1010) (HEAD(I),I=1,20)
      WRITE(JLUN,1040) ILUN,JLUN
C
      WRITE(JLUN,2010)
      WRITE(JLUN,1040) NEL,NBD
      WRITE(JLUN,1010) (IELNAM(I),I=1,NEL)
      WRITE(JLUN,1020) (IELLST(I),I=1,NEL)
      I2=2*NBD
      WRITE(JLUN,1020) (NBIMX(I),I=1,I2)
      WRITE(JLUN,1020) ((IBMX(I,J),J=1,2),I=1,NBD)
      I2=NEL+1
      WRITE(JLUN,1020) (NPTR(I),I=1,I2)
C
C-----PARAMETERS
C
      I2=NEL+1
      WRITE(JLUN,2020)
      WRITE(JLUN,1020) (IPTR(I),I=1,I2),NPAR,IPFLG
      WRITE(JLUN,1050) (PAR(I),I=1,NPAR)
C
C-----CAUSALITY
C
      WRITE(JLUN,2030)
      I2=2*NBD
      WRITE(JLUN,1020) (ICMX(I),I=1,I2)
      WRITE(JLUN,1020) (IBEQ(I),I=1,NBD)
      WRITE(JLUN,1020) (IBT(I),I=1,NBD)
      WRITE(JLUN,1040) NBEX,NFI,NFD,NFL,NFS,NBIN
C
50    WRITE(JLUN,900)
C
      CALL DEFINE
C
      RETURN
C
100   WRITE(JLUN,990) FILNAM
      CLOSE(NUNIT,STATUS='KEEP')
```

```
      RETURN
C
C-----FORMAT----------
C
900   FORMAT(/,'*** GRAPH FILE LOADING COMPLETED ***')
990   FORMAT(/,'OOPS ] YOU ARE LOADING THE WRONG FILE : ',A80)
1000  FORMAT(/,'GIVE FILE NAME :')
1001  FORMAT(/,'FILE NOT EXIST')
C
1010  FORMAT(20A4)
1020  FORMAT(2013)
1040  FORMAT(614)
1050  FORMAT(5E12.5)
2010  FORMAT('GRAPH')
2020  FORMAT('PARAMETERS')
2030  FORMAT('CAUSAL')
C
      END
C
C---CMREAD-------------------------------------------------------
C
      SUBROUTINE CMREAD
C
C-----WRITTEN BY : TSUN-YU CHOU,     JUNE 1982
C
C-----FREE FIELD COMMAND READER
C
$INSERT COMFILE
$INSERT COMAREA
C
      INTEGER BLANK
      LOGICAL FOUND
      DATA    BLANK/160/
C-------------------------------------------------------------
100   CONTINUE
      READ(JLUN,9000) CINPUT
      FPOCMN=0
      LPOCMN=0
      FOUND= .FALSE.
C
      DO 500 I=1,80
      IF( ICHAR( CINPUT(I:I) ) .EQ. BLANK) THEN
        IF(FPOCMN .EQ. 0) THEN
          GO TO 500
        ELSE IF(FPOCMN .NE. 0) THEN
          LPOCMN=I-1
          NOCMN=1
          GO TO 800
        ENDIF
      ELSE
        IF(FOUND .EQV.  .TRUE.) GO TO 500
          FPOCMN=I
          FOUND= .TRUE.
```

```
      ENDIF
500   CONTINUE
C
      IF((FPOCMN .EQ. 0 ) .AND. (LPOCMN .EQ. 0) ) THEN
        NOCMN=0
        RETURN
      ELSE IF((FPOCMN .NE. 0) .AND. (LPOCMN .EQ. 0) ) THEN
        WRITE(JLUN,9001)
        NOCMN=0
        GO TO 900
      ENDIF
C
800   CONTINUE
      CMN=CINPUT(FPOCMN:LPOCMN)
C
C----- FORMAT
9000  FORMAT(A80)
9001  FORMAT(//,'YOUR COMNAND EXCEED 80 COLUMN ')
C
900   RETURN
      END
C
C---CMNCPR-----------------------------------------------------
C
      SUBROUTINE CMNCPR(CMNSET,NOCMAP)
C
C---WRITTEN BY : TSUN-YU CHOU,     JUNE  1982
C
C      THIS SUBROUTINE MAPPING THE INPUT COMMAND TO THE
C      COMMAND LIST, COMPARING AND ACCEPTING ABBRIVATED
C      COMMAND.
C
      CHARACTER*6 CMNSET(100)
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      NOWORD=LPOCMN-FPOCMN+1
      IFOUND=0
C
      DO 200 I=1,NOCMAP
C
      DO 100 J=1,NOWORD
      IF ( CMNSET(I)(J:J) .NE. CMN(J:J) ) GO TO 200
100   CONTINUE
C
      IFOUND=IFOUND+1
      CMATCH(IFOUND)=CMNSET(I)
C
200   CONTINUE
C
      RETURN
      END
```

```
C
C-----ERRMSG----------------------------------------------------
C
      SUBROUTINE ERRMSG(IE)
C
C     THIS SUBROUTINE PRINT OUT (ERROR) MESSAGE IN THE TERMINAL.
C
$INSERT COMFILE
$INSERT COMAREA
C---------------------------------------------------------------
      GOTO(10,20,30,40,50,60,70,80,90,100),IE
C
10    WRITE(JLUN,1001)CMN
      RETURN
C
20    WRITE(JLUN,1002)(CMATCH(K),K=1,IFOUND)
      RETURN
C
30    WRITE(JLUN,1003)CMN
      RETURN
C
40    WRITE(JLUN,1004)CMN
      RETURN
C
50    WRITE(JLUN,1005)
      RETURN
C
60    WRITE(JLUN,1006)
      RETURN
C
70    WRITE(JLUN,1007)
      RETURN
C
80    WRITE(JLUN,1008)
      RETURN
C
90    WRITE(JLUN,1009)
      RETURN
C
100   RETURN
C
C-----FORMAT------------
C
1001  FORMAT(/,' ERROR] NO SUCH COMMAND : ',A80)
1002  FORMAT(/,' ERROR] COMMANDS MIXED UP',/,(A6,'?'))
1003  FORMAT(/,' SORRY] THIS COMMAND CANNOT USE RIGHT NOW : ',A80)
1004  FORMAT(/,' ERROR] EXPECTING NUMBER, YOUR INPUT IS : ',A80/)
1005  FORMAT(/,' ERROR] BAD PARAMETER. ',/)
1006  FORMAT(/,' YOU MAY NOT REVERSE CAUSALITY FOR THIS FUNCTION',
     +       /,' EITHER CHANGE THE FUNCTION OR ACCEPT THE CAUSALITY.')
1007  FORMAT(/,'VERIFY ? (NO):')
1008  FORMAT(/,' ERROR] INVERSE OF ZERO SLOPE WILL BE INFINITY.',/)
1009  FORMAT(/,' ERROR] SLOPE CAN NOT BE INFINITY.'/)
```

```
C
      END
C
C---DEFINE----------------------------------------------------
C
      SUBROUTINE DEFINE
C
C-----WRITTEN BY : TSUN-YU CHOU,  JULY 1982
C
C---PURPOSE: DEFINE THE CONSTITUTIVE EQUATIONS OF BG.
C
C-----VARIABLES:
C     IBT(I) --- CLASSIFY TYPES OF BOND
C                POSITIVE-- E INTO FIELD ELEMENT
C                NEGATIVE-- F INTO FIELD ELEMENT
C
C            TYPES OF BONDS    IBT
C              EXTERNAL
C                INDEPENDENT    1
C                DEPENDENT      2
C                R              3
C                SOURCE         4
C              INTERNAL         5
C
C     IEELST --- TYPES OF ELEMENTS.
C         TYPES OF ELEMENTS
C                C              1
C                I              2
C                R              3
C                SE             4
C                SF             5
C                0              6
C                1              7
C                TF             8
C                GY             9
C                MACRO          10
C
C
C     NEQTP --- TYPES OF EQUATIONS.
C               TYPE        EQUATION
C              ----------------------------
C                  1      E1 = FCN ( Q1 )
C
C                  2      F1 = FCN ( P1 )
C
C                  3      Q1 = FCN ( E1 )
C
C                  4      P1 = FCN ( F1 )
C
C                  5      E1 = FCN ( F1 )
C
C                  6      F1 = FCN ( E1 )
C
```

```
C                7       E1 = FCN (TIME)
C
C                8       F1 = FCN (TIME)
C
C               12       E1 = FCN ( E2 )
C                         F2 = FCN ( F1 )
C
C               13       F1 = FCN ( F2 )
C                         E2 = FCN ( E1 )
C
C               14       E1 = FCN ( F2 )
C                         E2 = FCN ( F1 )
C
C               15       F1 = FCN ( E2 )
C                         F2 = FCN ( E1 )
C
C      NFCN --- STORE THE FUNCTION IDENTIFIER.
C      NOIO --- TOTAL NUMBER OF I/O VARIABLES.
C      IEQ2S(NEQS,1) --- OUTPUT PORT IDENTIFIER.
C      IEQ2S(NEQS,2) --- INPUT  PORT IDENTIFIER.
C      CHIO(NEQS,1/3) --- OUTPUT VARIABLE.
C      CHIO(NEQS,2/4) --- INPUT  VARIABLE.
C
C-----COMMON BLOCK
C
      DIMENSION IGNORE(50)
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C-----CHECKING FOR HEADING.
C
      OPT1='DEFINE'
      IF(IPNT(1).EQ.1) OPT2='L'
      IBTC=0
      IBTI=0
      IBTR=0
      IBTS=0
      DO 5 I=1,NBD
      IBTABS=IABS(IBT(I))
      IF(IBTABS.EQ.1) IBTC=IBTC+1
      IF(IBTABS.EQ.2) IBTI=IBTI+1
      IF(IBTABS.EQ.3) IBTR=IBTR+1
      IF(IBTABS.EQ.4) IBTS=IBTS+1
5     CONTINUE
C
C-----CHECKING FOR ONE-PORT C, I, R ELEMENTS.
C
      DO 20 I=1,NBD
      IGNORE(I)=1
20    CONTINUE
C
      NP1=1
```

```
      DO 30 I=1,NEL
      NP2=NPTR(I+1)-1
      IF((IELLST(I).LE.3).AND.(NP1.NE.NP2))GOTO 25
      GOTO 30
25    DO 27 K=NP1,NP2
      IG=IABS(NBIMX(K))
27    IGNORE(IG)=0
30    NP1=NP2+1
C
C-----PRINT OUT HEADING AND SYMBOLIC EQUATIONS FOR
C-----ONE-PORT C, I, R ELEMENTS AND SOURCE ELEMENTS SE AND SF.
C
      NEQS=0
C
      DO 200 I=1,4
      IF(OPT2.EQ.'L')THEN
        IF((I .EQ. 1).AND.(IBTC.GT.0)) WRITE(JLUN,9001)
        IF((I .EQ. 2).AND.(IBTI.GT.0)) WRITE(JLUN,9002)
        IF((I .EQ. 3).AND.(IBTR.GT.0)) WRITE(JLUN,9003)
        IF((I .EQ. 4).AND.(IBTS.GT.0)) WRITE(JLUN,9004)
      ENDIF
      ICAUS=-1*I
C
50    DO 100 J=1,NBD
      IF(IBT(J) .NE. ICAUS) GOTO100
      IF((IBT(J).EQ. 1).AND.(IGNORE(J).EQ.1)) GOTO 65
      IF((IBT(J).EQ.-1).AND.(IGNORE(J).EQ.1)) GOTO 60
      IF((IBT(J).EQ. 2).AND.(IGNORE(J).EQ.1)) GOTO 70
      IF((IBT(J).EQ.-2).AND.(IGNORE(J).EQ.1)) GOTO 75
      IF((IBT(J).EQ. 3).AND.(IGNORE(J).EQ.1)) GOTO 85
      IF((IBT(J).EQ.-3).AND.(IGNORE(J).EQ.1)) GOTO 80
      IF((IBT(J).EQ. 4).AND.(IGNORE(J).EQ.1)) GOTO 95
      IF((IBT(J).EQ.-4).AND.(IGNORE(J).EQ.1)) GOTO 90
C
      GOTO 100
C
C-----INTEGRATE C-ELEMENT
60      NEQS=NEQS+1
        NFCN(NEQS)=J
        NEQTP(NEQS)=1
        IEQ2S(NEQS,1)=J
        IEQ2S(NEQS,2)=J
        NOIO(NEQS)=2
        CHIO(NEQS,1)='E'
        CHIO(NEQS,2)='Q'
         IF(OPT2.EQ.'L')THEN
            CALL FCNWTR(NEQS)
         ENDIF
        GOTO 100
C
C-----INTEGRATE I-ELEMENT
65      NEQS=NEQS+1
        NFCN(NEQS)=J
```

```
          NEQTP(NEQS)=2
          IEQ2S(NEQS,1)=J
          IEQ2S(NEQS,2)=J
          NOIO(NEQS)=2
          CHIO(NEQS,1)='F'
          CHIO(NEQS,2)='P'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
          GOTO 100
C
C-----DERIVATIVE C-ELEMENT
70        NEQS=NEQS+1
          NFCN(NEQS)=J
          NEQTP(NEQS)=3
          IEQ2S(NEQS,1)=J
          IEQ2S(NEQS,2)=J
          NOIO(NEQS)=2
          CHIO(NEQS,1)='Q'
          CHIO(NEQS,2)='E'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
          GOTO 100
C
C-----DERIVATIVE I-ELEMENT
75        NEQS=NEQS+1
          NFCN(NEQS)=J
          NEQTP(NEQS)=4
          IEQ2S(NEQS,1)=J
          IEQ2S(NEQS,2)=J
          NOIO(NEQS)=2
          CHIO(NEQS,1)='P'
          CHIO(NEQS,2)='F'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
          GOTO 100
C
C-----FLOW INPUT TO R ELEMENT
80        NEQS=NEQS+1
          NFCN(NEQS)=J
          NEQTP(NEQS)=5
          IEQ2S(NEQS,1)=J
          IEQ2S(NEQS,2)=J
          NOIO(NEQS)=2
          CHIO(NEQS,1)='E'
          CHIO(NEQS,2)='F'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
          GOTO 100
C
```

```
C-----EFFORT INPUT TO R ELEMENT
85       NEQS=NEQS+1
         NFCN(NEQS)=J
         NEQTP(NEQS)=6
         IEQ2S(NEQS,1)=J
         IEQ2S(NEQS,2)=J
         NOIO(NEQS)=2
         CHIO(NEQS,1)='F'
         CHIO(NEQS,2)='E'
          IF(OPT2.EQ.'L')THEN
             CALL FCNWTR(NEQS)
          ENDIF
         GOTO 100
C
C-----SOURCE ELEMENT SE
90     IF(OPT2.EQ.'L')THEN
         WRITE(JLUN,9010) J
       ENDIF
         GOTO 100
C
C-----SOURCE ELEMENT SF
95     IF(OPT2.EQ.'L')THEN
         WRITE(JLUN,9020) J
       ENDIF
C
100    CONTINUE
C
       IF(ICAUS .GT. 0)GOTO 200
       ICAUS=I
       GOTO 50
200    CONTINUE
C
C-----SYMBOLIC EQUATIONS FOR MULTI-PORT ELEMENTS
C
       NP1=1
       DO 700 I=1,NEL
       NP2=NPTR(I+1)-1
       IF(NP1 .EQ. NP2)GOTO 700
       NP=NP2-NP1+1
       GOTO(610,620,630,700,700,700,700,700,700,700),IELLST(I)
C
C-----MULTI-PORT C-FIELD
C
610    IF(OPT2.EQ.'L') WRITE(JLUN,8100)NP
        DO 611 J=1,NP
          IF(OPT2.EQ.'L')THEN
           WRITE(JLUN,8110) NBIMX(J),NBIMX(J),(NBIMX(K),K=NP1,NP2)
          ENDIF
        NEQS=NEQS+1
        NFCN(NEQS)=NBIMX(J)
        NEQTP(NEQS)=9
611     CONTINUE
       GOTO 700
```

```
C
C-----MULTI-PORT I-FIELD
C
620    IF(OPT2.EQ.'L') WRITE(JLUN,8200)NP
       DO 621 J=1,NP
         IF(OPT2.EQ.'L')THEN
          WRITE(JLUN,8210) NBIMX(J),NBIMX(J),(NBIMX(K),K=NP1,NP2)
         ENDIF
       NEQS=NEQ+1
       NFCN(NEQS)=NBIMX(J)
       NEQTP(NEQS)=10
621    CONTINUE
      GOTO 700
C
C-----MULTI-PORT R-FIELD
C
630    IF(OPT2.EQ.'L') WRITE(JLUN,8300)NP
       DO 631 J=1,NP
         IF(OPT2.EQ.'L')THEN
          WRITE(JLUN,8310) NBIMX(J),NBIMX(J),(NBIMX(K),K=NP1,NP2)
         ENDIF
       NEQS=NEQS+1
       NFCN(NEQS)=NBIMX(J)
       NEQTP(NEQS)=11
631    CONTINUE
C
700    NP1=NP2+1
C
C-----SYMBOLIC EQUATIONS FOR TF AND GY.
C
       ISLT=8
690    NP1=1
       DO 800 I=1,NEL
       NP2=NPTR(I+1)-1
       IF(NP1 .EQ. NP2)GOTO 800
       NP=NP2-NP1+1
       IF((IELLST(I) .EQ. 8) .AND. (ISLT .EQ. 8))GOTO710
       IF((IELLST(I) .EQ. 9) .AND. (ISLT .EQ. 9))GOTO720
       GOTO 800
C
C-----TRANSFORMER
C
710    CONTINUE
       IF(ICMX(NP1) .LE. 3)GOTO 711
C
       IF(OPT2.EQ.'L') WRITE(JLUN,8800)
        NEQS=NEQS+1
        NFCN(NEQS)=NBIMX(NP1)
        NEQTP(NEQS)=12
        IEQ2S(NEQS,1)=NBIMX(NP1)
        IEQ2S(NEQS,2)=NBIMX(NP2)
         NOIO(NEQS)=4
         CHIO(NEQS,1)='E'
```

```
          CHIO(NEQS,2)='E'
          CHIO(NEQS,3)='F'
          CHIO(NEQS,4)='F'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
        GOTO 800
C
711     IF(OPT2.EQ.'L') WRITE(JLUN,8800)
         NEQS=NEQS+1
         NFCN(NEQS)=NBIMX(NP1)
         NEQTP(NEQS)=13
         IEQ2S(NEQS,1)=NBIMX(NP1)
         IEQ2S(NEQS,2)=NBIMX(NP2)
          NOIO(NEQS)=4
          CHIO(NEQS,1)='F'
          CHIO(NEQS,2)='F'
          CHIO(NEQS,3)='E'
          CHIO(NEQS,4)='E'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
        GOTO 800
C
C-----GYRATOR
C
720     CONTINUE
        IF(ICMX(NP1) .LE. 3)GOTO 721
C
        IF(OPT2.EQ.'L') WRITE(JLUN,8900)
         NEQS=NEQS+1
         NFCN(NEQS)=NBIMX(NP1)
         NEQTP(NEQS)=14
         IEQ2S(NEQS,1)=NBIMX(NP1)
         IEQ2S(NEQS,2)=NBIMX(NP2)
          NOIO(NEQS)=4
          CHIO(NEQS,1)='E'
          CHIO(NEQS,2)='F'
          CHIO(NEQS,3)='E'
          CHIO(NEQS,4)='F'
           IF(OPT2.EQ.'L')THEN
              CALL FCNWTR(NEQS)
           ENDIF
        GOTO 800
C
721     IF(OPT2.EQ.'L') WRITE(JLUN,8900)
         NEQS=NEQS+1
         NFCN(NEQS)=NBIMX(NP1)
         NEQTP(NEQS)=15
         IEQ2S(NEQS,1)=NBIMX(NP1)
         IEQ2S(NEQS,2)=NBIMX(NP2)
          NOIO(NEQS)=4
          CHIO(NEQS,1)='F'
```

```
        CHIO(NEQS,2)='E'
        CHIO(NEQS,3)='F'
        CHIO(NEQS,4)='E'
         IF(OPT2.EQ.'L')THEN
            CALL FCNWTR(NEQS)
         ENDIF
      GOTO 800
C
800   NP1=NP2+1
      IF(ISLT .EQ. 9)GOTO 810
      ISLT=9
      GOTO 690
C
810   CONTINUE
C
      IF (IPNT(1).EQ.0) GOTO 1001
C
      WRITE(JLUN,9100)
      DO 1000 I=1,NEQS
      WRITE(JLUN,9200) I,NFCN(I),NEQTP(I)
1000  CONTINUE
C
1001  OPT2='U'
      RETURN
C
C--- FORMATS
C
C-----CLASSIFIED HEADING
C
9001  FORMAT(//,25HINTEGRAL STORAGE ELEMENTS)
9002  FORMAT(//,27HDERIVATIVE STORAGE ELEMENTS)
9003  FORMAT(//,20HDISSIPATION ELEMENTS)
9004  FORMAT(//,15HSOURCE ELEMENTS)
C
C-----SE AND SF
C
9010  FORMAT(/,5X,'E',I2,' = FCN ( TIME )')
9020  FORMAT(/,5X,'F',I2,' = FCN ( TIME )')
C
C-----MULTI-PORT ELEMENTS
C
8100  FORMAT(//,I2,14H-PORT  C-FIELD )
8110  FORMAT(/,5X,'E',I2,' = FCN',I2,' (Q',I2,(', Q',I2),')'  )
C
8200  FORMAT(//,I2,14H-PORT  I-FIELD )
8210  FORMAT(/,5X,'F',I2,' = FCN',I2,' (P',I2,(', P',I2),')' )
C
8300  FORMAT(//,I2,14H-PORT  R-FIELD )
8310  FORMAT(/,5X,'E',I2,' = FCN',I2,' (F',I2,(', F',I2),')'  )
C
C-----TF AND GY
C
8800  FORMAT(//,11HTRANSFORMER)
```

```
8900  FORMAT(//,7HGYRATOR)
C-----
9100  FORMAT(/5X,3HNO.,5X,3HFCN,5X,4HTYPE)
9200  FORMAT(5X,I2,6X,I2,7X,I2)
C
      END
C
C
C---FCNDEF-----------------------------------------------------
C
      SUBROUTINE FCNDEF
C
C-----WRITTEN BY : TSUN-YU CHOU,  JULY 1982
C
C     OPT2 = N ---> WHEN FUNCTION INVERSE NOT ABAILABLE.
C     OPT2 = Q ---> USER ENTER QUIT FOR FUNCTION TYPE.
C     OPT2 = X ---> USER ENTER X TO TERMINATE THE OPERATION.
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      OPT1='CHANGE'
C
125   WRITE(JLUN,126)
126   FORMAT(/,'CHANGE ALL ? (NO):')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 127
      IF(CMN.EQ.'X') RETURN
      IF((CMN.EQ.'Y').OR.(CMN.EQ.'YES'))GOTO 200
C
127   WRITE(JLUN,128)
128   FORMAT(/,'ENTER FUNCTION INDEX :')
      CALL CMREAD
C
      IF(CMN.EQ.'X')RETURN
      IF(NOCMN.EQ.0)GOTO 127
      DECODE(80,*,CMN,ERR=140)MFCN
      IF  (MFCN.LT.0) THEN
        PRINT *,'ERROR] INCORRECT FUNCTION NUMBER'
        GOTO 127
      ELSE IF(MFCN.EQ.0) THEN
        RETURN
      ENDIF
C
      DO 130 JJ=1,NEQS
      IF(IEQ2S(JJ,1) .EQ. MFCN) THEN
        NOF=JJ
        GOTO 150
      ENDIF
130   CONTINUE
C
140   PRINT *,'ERROR] BAD FUNCTION LABEL'
      GOTO 127
C
```

```
150   WRITE(JLUN,155)IEQ2S(NOF,1)
155   FORMAT(/,'FUNCTION  ',I2,'  NOW DEFINED AS :')
      CALL FCNWTR(NOF)
      CALL EQRVS(NOF)
        IF(OPT2.EQ.'X')THEN
         OPT2=' '
         RETURN
        ENDIF
160   WRITE(JLUN,165)
165   FORMAT(/,'FUNCTION TYPE ?')
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        CMN=FCNSET(IFCNTP(NOF))
        FPOCMN=1
        LPOCMN=6
      ENDIF
      CALL CMNCPR(FCNSET,30)
C
        IF (IFOUND.EQ.0)THEN
          WRITE(JLUN,9001)CMN
          GOTO 160
        ELSE IF(IFOUND.GT.1)THEN
          WRITE(JLUN,9002)(CMATCH(K),K=1,IFOUND)
          GOTO 160
        ENDIF
C
      CALL FCNLIB(NOF)
      IF(OPT2.EQ.'N')THEN
        OPT2='U'
        RVS(NOF)=0.
        GOTO 150
      ELSE IF(OPT2.EQ.'Q')THEN
        OPT2='U'
        RETURN
      ENDIF
C
      GOTO 400
C
C---CHANGE ALL
200   CONTINUE
      DO 300 NOF=1,NEQS
C
210   WRITE(JLUN,215)IEQ2S(NOF,1)
215   FORMAT(/,'FUNCTION  ',I2,'  NOW DEFINED AS :')
      CALL FCNWTR(NOF)
      CALL EQRVS(NOF)
        IF(OPT2.EQ.'X')THEN
         OPT2=' '
         RETURN
        ENDIF
220   WRITE(JLUN,225)
225   FORMAT(/,'FUNCTION TYPE ?')
      CALL CMREAD
```

```
      IF(NOCMN.EQ.0)CMN=FCNSET(IFCNTP(NOF))
      CALL CMNCPR(FCNSET,30)
        IF (IFOUND.EQ.0)THEN
          WRITE(JLUN,9001)CMN
          GOTO 220
        ELSE IF(IFOUND.GT.1)THEN
          WRITE(JLUN,9002)(CMATCH(K),K=1,IFOUND)
          GOTO 220
        ENDIF
C
      CALL FCNLIB(NOF)
      IF(OPT2.EQ.'N')THEN
        OPT2=' '
        RVS(NOF)=0.
        GOTO 210
      ELSE IF(OPT2.EQ.'Q')THEN
        OPT2='U'
        RETURN
      ENDIF
C
300   CONTINUE
C
      RETURN
C----------
400   CONTINUE
C
405   WRITE(JLUN,410)
410   FORMAT(/,'CHANGE ANOTHER FUNCTION ? (YES):')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 127
      IF((CMN.EQ.'Y').OR.(CMN.EQ.'YES'))GOTO 127
      RETURN
C-----FORMAT-----
C
9001  FORMAT(/,' ERROR ] NO SUCH FUNCTION :  ',A80)
9002  FORMAT(/,' ERROR ] FUNCTIONS MIXED UP',/,(A6,'  ?'))
C
       END
C
C---EQRVS-----------------------------------------------------
C
      SUBROUTINE EQRVS(K)
C
C-----WRITTEN BY : TSUN-YU CHOU,    SEPT. 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(IPNT(1).EQ.1)WRITE(JLUN,150)RVS(K)
      WRITE(JLUN,101)
101   FORMAT(/,'REVERSE INPUT AND OUTPUT ? (NO):')
C
      CALL CMREAD
```

```
      IF(NOCMN.EQ.0)RETURN
      IF((CMN.EQ.'Y').OR.(CMN.EQ.'YES'))GOTO 100
      IF(CMN.EQ.'X')OPT2='X'
      RETURN
C
100   WRITE(JLUN,110) IEQ2S(K,1)
110   FORMAT(/,'FUNCTION  ',I2,'  NOW DEFINED AS :')
C
      RR=RVS(K)
      OPT2='U'
      IF(RR.EQ.0.)THEN
       RVS(K)=1.
      ELSE IF(RR.EQ.1)THEN
       RVS(K)=0.
      ENDIF
C
      CALL FCNWTR(K)
150   FORMAT(/,'*** RVS = ',I1,' ***')
C
      RETURN
      END
C
C---SHOW----------------------------------------------------
C
      SUBROUTINE SHOW
C
C-----WRITTEN BY : TSUN-YU CHOU,  JULY 1982
C
C     DISPLAY USER-DEFINED FUNCTIONS
C     AND SYSTEM-FORMATED FUNCTIONS
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      OPT1='LOOK  '
C
425   WRITE(JLUN,426)
426   FORMAT(/,'LOOK AT ALL ? (NO):')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO430
      IF(CMN.EQ.'X') RETURN
      IF((CMN.EQ.'Y').OR.(CMN.EQ.'YES'))GOTO 560
C
430   WRITE(JLUN,432)
432   FORMAT(/,'ENTER FUNCTION INDEX :')
      CALL CMREAD
C
      IF(CMN.EQ.'X')RETURN
      IF(NOCMN.EQ.0)GOTO 430
      DECODE(80,*,CMN,ERR=540)MFCN
      IF (MFCN.LT.0) THEN
        PRINT *,'ERROR] INCORRECT FUNCTION NUMBER'
        GOTO 430
```

```
      ELSE IF(MFCN.EQ.0) THEN
        RETURN
      ENDIF
C
525   DO 530 JJ=1,NEQS
      IF(IEQ2S(JJ,1) .EQ. MFCN) THEN
        NOF=JJ
        GOTO 550
      ENDIF
530   CONTINUE
540   PRINT *,'ERROR] BAD FUNCTION LABEL'
      GOTO 430
C
550   WRITE(JLUN,9003) IEQ2S(NOF,1)
      CALL FCNLOOK(NOF)
      GOTO 600
C
C-----LOOK AT ALL FUNCTIONS.
560   CONTINUE
      DO 570 NOF=1,NEQS
      WRITE(JLUN,9003) IEQ2S(NOF,1)
      CALL FCNLOOK(NOF)
570   CONTINUE
C
      RETURN
C----------
600   CONTINUE
C
605   WRITE(JLUN,610)
610   FORMAT(/,'LOOK AT ANOTHER FUNCTION ? (YES):')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 430
      IF((CMN.EQ.'Y').OR.(CMN.EQ.'YES'))GOTO 430
      RETURN
C
C-----FORMAT-----
C
9001  FORMAT(/,' ERROR ] NO SUCH FUNCTION :  ',A80)
9002  FORMAT(/,' ERROR ] FUNCTIONS MIXED UP',/,(A6,'  ?'))
9003  FORMAT(//,' FUNCTION ',I2,' :')
C
       END
C
C
C---FCNWTR-----------------------------------------------------
C
      SUBROUTINE FCNWTR(K)
C
C---WRITTEN BY : TSUN-YU CHOU,    AUGUST 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
```

```
      NNEQTP=NEQTP(K)
C
      IF(NNEQTP .LE. 6) THEN
       IF(RVS(K).EQ.0)THEN
        IF(OPT1.EQ.'DEFINE')THEN
         WRITE(JLUN,9010)CHIO(K,1),NFCN(K),NFCN(K),
     +                   CHIO(K,2),NFCN(K)
        ELSE IF(OPT1.EQ.'CHANGE')THEN
         WRITE(JLUN,9020)CHIO(K,1),NFCN(K),FCNSET(IFCNTP(K)),
     +                   CHIO(K,2),NFCN(K)
        ENDIF
C
       ELSE IF(RVS(K).EQ.1)THEN
         WRITE(JLUN,9020)CHIO(K,2),NFCN(K),FCNSET(IFCNTP(K)),
     +                   CHIO(K,1),NFCN(K)
       ENDIF
C
C
C
      ELSE IF(NNEQTP .GE. 12) THEN
        IF(RVS(K).EQ.0) THEN
         IF(OPT1.EQ.'DEFINE')THEN
          WRITE(JLUN,8810)CHIO(K,1),IEQ2S(K,1),IEQ2S(K,1),
     +                    CHIO(K,2),IEQ2S(K,2),
     +                    CHIO(K,3),IEQ2S(K,2),IEQ2S(K,1),
     +                    CHIO(K,4),IEQ2S(K,1)
         ELSE IF(OPT1.EQ.'CHANGE')THEN
          WRITE(JLUN,8820)CHIO(K,1),IEQ2S(K,1),FCNSET(IFCNTP(K)),
     +                    CHIO(K,2),IEQ2S(K,2),
     +                    CHIO(K,3),IEQ2S(K,2),FCNSET(IFCNTP(K)),
     +                    CHIO(K,4),IEQ2S(K,1)
         ENDIF
C
        ELSE IF(RVS(K).EQ.1) THEN
          WRITE(JLUN,8820)CHIO(K,2),IEQ2S(K,2),FCNSET(IFCNTP(K)),
     +                    CHIO(K,1),IEQ2S(K,1),
     +                    CHIO(K,4),IEQ2S(K,1),FCNSET(IFCNTP(K)),
     +                    CHIO(K,3),IEQ2S(K,2)
C
        ENDIF
C
      ENDIF
C
C-----FORMAT----------
C
9010  FORMAT(/,5X,A1,I2,' = FCN',I2,' (',A1,I2,')' )
9020  FORMAT(/,5X,A1,I2,' = ',A6,' (',A1,I2,')')
8810  FORMAT(/,5X,A1,I2,' = FCN',I2,' (',A1,I2,')',/,
     +         5X,A1,I2,' = FCN',I2,' (',A1,I2,')' )
8820  FORMAT(/,5X,A1,I2,' = ',A6,' (',A1,I2,')',
     +       /,5X,A1,I2,' = ',A6,' (',A1,I2,')')
C
      RETURN
```

```
      END
C
C---FCNLIB-----------------------------------------------------
C
      SUBROUTINE FCNLIB(I)
C
C-----WRITTEN BY : TSUN-YU CHOU,   SEPT. 1982
C
$INSERT COMFILE
$INSERT COMAREA
C
C
C     VARIABLE : IFCNTP --- STORE FUNCTION TYPE BELOW.
C
C ***** ALL FUNCTION INDICES USED IN OTHER SUBROUTINES ARE
C ***** ACCORDING TO THE FOLLOWING TYPE NUMBER.
C
C      TYPE                    DEFINITION
C     ------------------------------------------------------------
C       01        CONST       Y=CO
C       02        LINEAR      Y=CO+C1*X
C       03        SIN         Y=C1*SIN(C2*X+C3) + C4
C       04        COS         Y=C1*COS(C2*X+C3) + C4
C       05        ASIN        Y=C1*ASIN(C2*X+C3) + C4
C
C       06        ACOS        Y=C1*ACOS(C2*X+C3) + C4
C       07        ABS2        Y=C1*X*ABS(X)
C       08        ABSQRT      Y=C1*SIGN(X)*SQRT(ABS(X))
C       09        EXP         Y=C1*EXP(C2*X+C3) + C4
C       10        ALOG        Y=C1*ALOG(C2*X+C3) + C4
C
C       11        DSIN        Y=C1/SIN(C2*X+C3) + C4
C       12        DCOS        Y=C1/COS(C2*X+C3) + C4
C       13        DASIN       Y=C1/ASIN(C2*X+C3) + C4
C       14        DACOS       Y=C1/ACOS(C2*X+C3) + C4
C       15        POLY        Y=CO+C1*X+C2*X**2+C3*X**3+C4*X**4
C       16        COLUMB      Y=C1* SIGN(X)
C
C       17        BRKPT       Y=Y1+C1*(X-X1)  , X <=X1
C                             Y=Y1+C2*(X-X1)  , X1< X
C
C       18        SATUR       Y=Y1  , X <= X1
C                             Y=Y1+((Y2-Y1)/(X2-X1))*(X-X1)  , X1<X<X2
C                             Y=Y2  , X2 <= X
C
C       19        ADJSAT      Y=Y1+C1*(X-X1)  , X <= X1
C                             Y=Y1+((Y2-Y1)/(X2-X1))*(X-X1)  , X1<X<X2
C                             Y=Y2+C2*(X-X2)  , X2<= X
C
C       20        PWLIN       PIECEWISE LINEAR
C
C       21        INTPLO      INTERPOLATION POLYNOMIAL
C
```

```
C        22        RCPSQR       Y=C0 + C1/(1+(X/C2)**2)
C
C
       DATA FCNSET/'CONST ','LINEAR','SIN   ','COS   ','ASIN  ',
     +             'ACOS  ','ABS2  ','ABSQRT','EXP   ','ALOG  ',
     +             'DSIN  ','DCOS  ','DASIN ','DACOS ','POLY  ',
     +             'COULOM','BRKPT ','SATUR ','ADJSAT','PWLIN ',
     +             'INTPL0','RCPSQR','SET023','SET024','SET025',
     +             'SET026','SET027','SET028','SET029','QUIT  '/
C
C
       DATA ((PARDF(I,J),J=1,10),I=1,30)
     +            / 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
C--6
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
C--11
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
C--16
     +              1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
     +             -1.,-1., 1., 1., 0., 0., 0., 0., 0., 0.,
     +             -1.,-1., 1., 1., 1., 1., 0., 0., 0., 0.,
     +              5., 1., 1., 1., 0., 0., 0., 0., 0., 0.,
C--21
     +              5., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
C--26
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     +              0., 0., 0., 0., 0., 0., 0., 0., 0., 0./
C-------------------------------------------------------------
C
       IF(CMATCH(IFOUND) .EQ.'CONST')THEN
         IFCNTP(I)=1
         CALL FCNSUB1(I)
C
```

```
      ELSE IF(CMATCH(IFOUND) .EQ.'LINEAR')THEN
        IFCNTP(I)=2
        CALL FCNSUB2(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'SIN')THEN
        IFCNTP(I)=3
        CALL FCNSUB3(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'COS')THEN
        IFCNTP(I)=4
        CALL FCNSUB4(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ASIN')THEN
        IFCNTP(I)=5
        CALL FCNSUB5(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ACOS')THEN
        IFCNTP(I)=6
        CALL FCNSUB6(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ABS2')THEN
        IFCNTP(I)=7
        CALL FCNSUB7(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ABSQRT')THEN
        IFCNTP(I)=8
        CALL FCNSUB8(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'EXP')THEN
        IFCNTP(I)=9
        CALL FCNSUB9(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ALOG')THEN
        IFCNTP(I)=10
        CALL FCNSUB10(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'DSIN')THEN
        IFCNTP(I)=11
        CALL FCNSUB11(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'DCOS')THEN
        IFCNTP(I)=12
        CALL FCNSUB12(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'DASIN')THEN
        IFCNTP(I)=13
        CALL FCNSUB13(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'DACOS')THEN
        IFCNTP(I)=14
        CALL FCNSUB14(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'POLY')THEN
```

```
        IFCNTP(I)=15
        CALL FCNSUB15(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'COULOM')THEN
        IFCNTP(I)=16
        CALL FCNSUB16(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'BRKPT')THEN
        IFCNTP(I)=17
        CALL FCNSUB17(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'SATUR')THEN
        IFCNTP(I)=18
        CALL FCNSUB18(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'ADJSAT')THEN
        IFCNTP(I)=19
        CALL FCNSUB19(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'PWLIN')THEN
        IFCNTP(I)=20
        OPT3='LINEAR'
        CALL FCNSUB20(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'INTPLO')THEN
        IFCNTP(I)=21
        OPT3='INTPLO'
        CALL FCNSUB20(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ.'RCPSQR')THEN
        IFCNTP(I)=22
        CALL FCNSUB22(I)
C
      ELSE IF(CMATCH(IFOUND) .EQ. 'QUIT')THEN
        OPT2='Q'
        RETURN
C
      ENDIF
C
      RETURN
      END
C
C
C---FCNLOOK-----------------------------------------------------
C
      SUBROUTINE FCNLOOK(I)
C
C-----WRITTEN BY : TSUN-YU CHOU,  SEPT. 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C
```

```
      IF(IFCNTP(I).EQ.1) THEN
        CALL FCNSUB1(I)
C
      ELSE IF(IFCNTP(I).EQ.2) THEN
        CALL FCNSUB2(I)
C
      ELSE IF(IFCNTP(I).EQ.3) THEN
        CALL FCNSUB3(I)
C
      ELSE IF(IFCNTP(I).EQ.4) THEN
        CALL FCNSUB4(I)
C
      ELSE IF(IFCNTP(I).EQ.5) THEN
        CALL FCNSUB5(I)
C
      ELSE IF(IFCNTP(I).EQ.6) THEN
        CALL FCNSUB6(I)
C
      ELSE IF(IFCNTP(I).EQ.7) THEN
        CALL FCNSUB7(I)
C
      ELSE IF(IFCNTP(I).EQ.8) THEN
        CALL FCNSUB8(I)
C
      ELSE IF(IFCNTP(I).EQ.9) THEN
        CALL FCNSUB9(I)
C
      ELSE IF(IFCNTP(I).EQ.10) THEN
        CALL FCNSUB10(I)
C
      ELSE IF(IFCNTP(I).EQ.11) THEN
        CALL FCNSUB11(I)
C
      ELSE IF(IFCNTP(I).EQ.12) THEN
        CALL FCNSUB12(I)
C
      ELSE IF(IFCNTP(I).EQ.13) THEN
        CALL FCNSUB13(I)
C
      ELSE IF(IFCNTP(I).EQ.14) THEN
        CALL FCNSUB14(I)
C
      ELSE IF(IFCNTP(I).EQ.15) THEN
        CALL FCNSUB15(I)
C
      ELSE IF(IFCNTP(I).EQ.16) THEN
        CALL FCNSUB16(I)
C
      ELSE IF(IFCNTP(I).EQ.17) THEN
        CALL FCNSUB17(I)
C
      ELSE IF(IFCNTP(I).EQ.18) THEN
        CALL FCNSUB18(I)
```

```
C
      ELSE IF(IFCNTP(I).EQ.19) THEN
        CALL FCNSUB19(I)
C
      ELSE IF(IFCNTP(I).EQ.20) THEN
        CALL FCNSUB20(I)
C
      ELSE IF(IFCNTP(I).EQ.21) THEN
        CALL FCNSUB20(I)
C
      ELSE IF(IFCNTP(I).EQ.22) THEN
        CALL FCNSUB22(I)
C
      ENDIF
C
      RETURN
      END
C
C---PARTAK-----------------------------------------------------
C
      SUBROUTINE PARTAK(LIBN,K)
C
C-----WRITTEN BY: TSUN-YU CHOU, OCT. 1982
C
C     SUBROUTINE 'PARTAK' AND 'SUBWTR' CALLED BY FCNSUB'S
C     (3, 4, 5, 6, 9, 10, 11, 12, 13, 14)
C     FOR ENTER PARAMETERS AND VERIFY THE VALUES.
C     ARGUMENTS : LIBN---ROW NUMBER OF 'PARDF'.(FUNCTION TYPE INDEX)
C                 K -----EQUATION INDEX. EX: FCN 2<--(2 IS INDEX)
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C
10    NP=1
100   WRITE(JLUN,1002)PARDF(LIBN,1)
1002  FORMAT(/,'VALUE OF C1 ? (',F8.4,' ):')
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        C1=PARDF(LIBN,1)
        GOTO 15
      ENDIF
      DECODE(80,*,CMN,ERR=110)CONST1
      C1=CONST1
15    IF(C1.EQ.0) THEN
        CALL ERRMSG(5)
        GOTO 10
      ENDIF
C
20    NP=2
200   WRITE(JLUN,2002)PARDF(LIBN,2)
2002  FORMAT('VALUE OF C2 ? (',F8.4,' ):')
      CALL CMREAD
```

```
      IF (NOCMN.EQ.0) THEN
        C2=PARDF (LIBN,2)
        GOTO 25
      ENDIF
      DECODE (80,*,CMN,ERR=110) CONST2
      C2=CONST2
25    IF (C2.EQ.0)  THEN
        CALL ERRMSG (5)
        GOTO 20
      ENDIF
C
30    NP=3
300   WRITE (JLUN,3002) PARDF (LIBN,3)
3002  FORMAT ('VALUE OF C3 ? (',F8.4,' ):')
      CALL CMREAD
      IF (NOCMN.EQ.0) THEN
        C3=PARDF (LIBN,3)
        GOTO 40
      ENDIF
      DECODE (80,*,CMN,ERR=110) CONST3
      C3=CONST3
C
40    NP=4
400   WRITE (JLUN,4002) PARDF (LIBN,4)
4002  FORMAT ('VALUE OF C4 ? (',F8.4,' ):')
      CALL CMREAD
      IF (NOCMN.EQ.0) THEN
        C4=PARDF (LIBN,4)
        GOTO 50
      ENDIF
      DECODE (80,*,CMN,ERR=110) CONST4
      C4=CONST4
C
50    CONTINUE
      IF (RVS (K) .EQ.1)  THEN
        IF (LIBN.LE.10)  THEN
          EQPAR (K,1) =1./C2
          EQPAR (K,2) =1./C1
          EQPAR (K,3) =-C4/C1
          EQPAR (K,4) =-C3/C2
        ELSE IF ((LIBN.GE.11) .AND. (LIBN.LE.14)) THEN
          EQPAR (K,1) =1./C2
          EQPAR (K,2) =C1
          EQPAR (K,3) =C4
          EQPAR (K,4) =-C3/C2
        ENDIF
      ELSE
        EQPAR (K,1) =C1
        EQPAR (K,2) =C2
        EQPAR (K,3) =C3
        EQPAR (K,4) =C4
      ENDIF
C
```

```
      RETURN
C
110   CALL ERRMSG(4)
      GOTO(10,20,30,40),NP
C
      END
C
C---SUBWTR-----------------------------------------------------
C
      SUBROUTINE SUBWTR(LIBN,K)
C
$INSERT COMFILE
$INSERT COMAREA
      CHARACTER*6 FTYPE(14),RFTYPE(14)
      COMMON /SP/ FTYPE,RFTYPE
      DATA RFTYPE /'      ','      ','* ASIN','* ACOS',
     +             '* SIN ','* COS ','      ','      ',
     +             '* ALOG','* EXP ','/ ASIN','/ ACOS',
     +             '/ SIN ','/ COS '/
C
C-------------------------------------------------------------
C
60    IF(RVS(K).EQ.1) THEN
C
        IF(LIBN.LE.10)THEN
         TT1=1./EQPAR(K,2)
         TT2=1./EQPAR(K,1)
         TT3=-EQPAR(K,4)*TT2
         TT4=-EQPAR(K,3)*TT1
        ELSE IF(LIBN.GE.11)THEN
         TT1=EQPAR(K,2)
         TT2=1./EQPAR(K,1)
         TT3=-EQPAR(K,4)*TT2
         TT4=EQPAR(K,3)
        ENDIF
C
       IF(OPT2.EQ.'U') THEN
        WRITE(JLUN,1003)CHIO(K,2),IEQ2S(K,2),TT1,
     +                  FTYPE(IFCNTP(K)),TT2,
     +                  CHIO(K,1),IEQ2S(K,1),TT3,TT4
C
           IF(NOIO(K).EQ.4) THEN
             WRITE(JLUN,1003)CHIO(K,4),IEQ2S(K,1),TT1,
     +                       FTYPE(IFCNTP(K)),TT2,
     +                       CHIO(K,3),IEQ2S(K,2),TT3,TT4
           ENDIF
C
       ELSE IF(OPT2.EQ.'S') THEN
        WRITE(JLUN,1005)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),
     +                  RFTYPE(IFCNTP(K)),EQPAR(K,2),
     +                  CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),EQPAR(K,4)
C
           IF(NOIO(K).EQ.4) THEN
```

```
            WRITE(JLUN,1005) CHIO(K,3),IEQ2S(K,2),EQPAR(K,1),
     +             RFTYPE(IFCNTP(K)),
     +             EQPAR(K,2),CHIO(K,4),IEQ2S(K,1),EQPAR(K,3),
     +             EQPAR(K,4)
          ENDIF
C
       ENDIF
      ELSE
        WRITE(JLUN,1003) CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),
     +                   FTYPE(IFCNTP(K)),EQPAR(K,2),
     +                   CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),EQPAR(K,4)
C
          IF(NOIO(K).EQ.4) THEN
            WRITE(JLUN,1003) CHIO(K,3),IEQ2S(K,2),EQPAR(K,1),
     +             FTYPE(IFCNTP(K)),
     +             EQPAR(K,2),CHIO(K,4),IEQ2S(K,1),EQPAR(K,3),
     +             EQPAR(K,4)
          ENDIF
C
      ENDIF
C
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' ',A6,'(',1PE12.4,
     +   ' * ',A1,I2,' +',1PE12.4,' )',/,9X,'+',1PE12.4)
1005  FORMAT(/,5X,A1,I2,' =',1PE12.4,' ',A6,'(',1PE12.4,
     +   ' * ',A1,I2,' +',1PE12.4,' )',/,9X,'+',1PE12.4)
C
      RETURN
      END
C
C---IORVSR-----------------------------------------------
C
      SUBROUTINE IORVSR(K,I5)
C
C-----WRITTEN BY: TSUN-YU CHOU, DEC. 1982
C     ARGUMENTS : K ----EQUATION INDEX.
C                 I5----FUNCTION TYPE INDEX.
C
$INSERT COMFILE
$INSERT COMAREA
C
      CHARACTER*6 FTYPE(14),RFTYPE(14)
      COMMON /SP/ FTYPE,RFTYPE
      DATA FTYPE/'      ','      ','* SIN ','* COS ',
     +           '* ASIN','* ACOS','      ','      ',
     +           '* EXP ','* ALOG','/ SIN ','/ COS ',
     +           '/ ASIN','/ ACOS'/
C
C-------------------------------------------------------------
      IF(RVS(K).EQ.1)THEN
        I1=2
        I2=1
        I3=4
        I4=3
```

```
      ELSE
        I1=1
        I2=2
        I3=3
        I4=4
      ENDIF
C
      IF(I5.EQ.7)THEN
        WRITE(JLUN,1002)CHIO(K,I1),IEQ2S(K,I1),CHIO(K,I2),
     +             IEQ2S(K,I2),CHIO(K,I2),IEQ2S(K,I2)
        IF(NOIO(K).EQ.4) THEN
        WRITE(JLUN,1002)CHIO(K,I3),IEQ2S(K,I2),CHIO(K,I4),
     +             IEQ2S(K,I1),CHIO(K,I4),IEQ2S(K,I1)
        ENDIF
C
      ELSE IF(I5.EQ.8)THEN
        WRITE(JLUN,1003)CHIO(K,I1),IEQ2S(K,I1),CHIO(K,I2),
     +             IEQ2S(K,I2),CHIO(K,I2),IEQ2S(K,I2)
        IF(NOIO(K).EQ.4) THEN
        WRITE(JLUN,1003)CHIO(K,I3),IEQ2S(K,I2),CHIO(K,I4),
     +             IEQ2S(K,I1),CHIO(K,I4),IEQ2S(K,I1)
        ENDIF
C
      ELSE
        WRITE(JLUN,1001)CHIO(K,I1),IEQ2S(K,I1),FTYPE(I5),
     +                  CHIO(K,I2),IEQ2S(K,I2)
        IF(NOIO(K).EQ.4) THEN
        WRITE(JLUN,1001)CHIO(K,I3),IEQ2S(K,I2),FTYPE(I5),
     +                  CHIO(K,I4),IEQ2S(K,I1)
        ENDIF
C
      ENDIF
C
1001  FORMAT(/,5X,A1,I2,' = C1 ',A6,' ( C2 * ',A1,I2,
     +        ' + C3 ) + C4')
1002  FORMAT(/,5X,A1,I2,' = C1 * ',A1,I2,' * ABS( ',A1,I2,' )')
1003  FORMAT(/,5X,A1,I2,' = C1 * SIGN( ',A1,I2,' ) * SQRT( ',
     +          'ABS( ',A1,I2,' ) )' )
C
      RETURN
      END
C
C---GENPAR-----------------------------------------------------
C
      SUBROUTINE GENPAR(K,KDF,KPO,NCOOR)
C     THIS SUBROUTINE USED FOR ENTER PAIR OF X,Y COORDINATE.
C     ARGUMENTS : K ----- EQUATION INDEX.
C                 KDF---- ROW NUMBER OF 'PARDF'.
C                 KPO---- COLUMN NUMBER OF 'PARDF'.
C                 NCOOR-- COORDINATE INDEX. EX: (X1,Y1)
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
```

```
10    NP=1
      WRITE(JLUN,1002)NCOOR,PARDF(KDF,KPO)
1002  FORMAT(/,'VALUE OF X',I1,' ? (',F8.4,' ):')
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        EQPAR(K,KPO)=PARDF(KDF,KPO)
        GOTO 20
      ENDIF
      DECODE(80,*,CMN,ERR=110)CONST1
      EQPAR(K,KPO)=CONST1
C
20    NP=2
      KPO2=KPO+1
      WRITE(JLUN,2002)NCOOR,PARDF(KDF,KPO2)
2002  FORMAT('VALUE OF Y',I1,' ? (',F8.4,' ):')
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        EQPAR(K,KPO2)=PARDF(KDF,KPO2)
        GOTO 30
      ENDIF
      DECODE(80,*,CMN,ERR=110)CONST2
      EQPAR(K,KPO2)=CONST2
C
30    CONTINUE
C
      RETURN
C
110   CALL ERRMSG(4)
      GOTO (10,20),NP
C
      END
C
C---GENCON----------------------------------------------------
C
      SUBROUTINE GENCON(K,KDF,KPO,NCONS)
C     THIS SUBROUTINE USED FOR ENTER CONSTANT Ci.
C     ARGUMENTS : K ----- EQUATION INDEX.
C                 KDF---- ROW NUMBER OF 'PARDF'.
C                 KPO---- COLUMN NUMBER OF 'PARDF'.
C                 NCONS---CONSTANT INDEX. EX: C1, C2
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
10    WRITE(JLUN,1002)NCONS,PARDF(KDF,KPO)
1002  FORMAT(/,'VALUE OF C',I1,' ? (',F8.4,' ):')
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        EQPAR(K,KPO)=PARDF(KDF,KPO)
        GOTO 20
      ENDIF
      DECODE(80,*,CMN,ERR=110)CONST1
      EQPAR(K,KPO)=CONST1
C
```

```
20     CONTINUE
       RETURN
C
110    CALL ERRMSG(4)
       GOTO 10
C
       END
C
C---COOREX----------------------------------------------------
C
       SUBROUTINE COOREX(K,IPO,JPO)
C      THIS SUBROUTINE EXCHANGE THE X-Y COORDINATE.
C      ARGUMENTS : K ----- EQUATION INDEX.
C                  IPO---- STORAGE POSITION OF X COORDINATE.
C                  JPO---- STORAGE POSITION OF Y COORDINATE.
$INSERT COMAREA
C-------------------------------------------------------------
       TEMP1=EQPAR(K,IPO)
       EQPAR(K,IPO)=EQPAR(K,JPO)
       EQPAR(K,JPO)=TEMP1
C
       RETURN
       END
C
C---FCNSUB1---------------------------------------------------
C
       SUBROUTINE FCNSUB1(K)
C
C-----FUNCTION TYPE : CONST    Y=CO
C
C-----WRITTEN BY : TSUN-YU CHOU,    AUGUST 1982
C-----MODIFIED : FEB. 1983.
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 200
C
       IF(RVS(K).EQ.1)THEN
         OPT2='N'
          CALL ERRMSG(6)
         RETURN
       ENDIF
C
       WRITE(JLUN,1001)CHIO(K,1),IEQ2S(K,1)
       IF(NOIO(K).EQ.4)  THEN
       WRITE(JLUN,1001)CHIO(K,3),IEQ2S(K,2)
       ENDIF
C
1001   FORMAT(/,5X,A1,I2,' = CO ')
C
100    CALL GENCON(K,1,1,0)
C
```

```
      IF(EQPAR(K,1).EQ.0)THEN
        CALL ERRMSG(5)
        GOTO 100
      ENDIF
C
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 200
      RETURN
C
200    WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1)
       IF(NOIO(K).EQ.4) THEN
         WRITE(JLUN,1003)CHIO(K,3),IEQ2S(K,2),EQPAR(K,1)
       ENDIF
C
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4)
C
      RETURN
      END
C
C---FCNSUB2-----------------------------------------------------
C
      SUBROUTINE FCNSUB2(K)
C
C-----FUNCTION TYPE : LINEAR      Y=C0 +C1*X
$INSERT COMFILE
$INSERT COMAREA
C---------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
C
      IF(RVS(K).EQ.1)THEN
        I1=2
        I2=1
        I3=4
        I4=3
      ELSE
        I1=1
        I2=2
        I3=3
        I4=4
      ENDIF
C
      WRITE(JLUN,1001)CHIO(K,I1),IEQ2S(K,I1),CHIO(K,I2),IEQ2S(K,I2)
      IF(NOIO(K).EQ.4) THEN
      WRITE(JLUN,1001)CHIO(K,I3),IEQ2S(K,I2),CHIO(K,I4),IEQ2S(K,I1)
      ENDIF
C
1001  FORMAT(/,5X,A1,I2,' = C0 + C1 * ',A1,I2)
C
      CALL GENCON(K,2,1,0)
C
10    CALL GENCON(K,2,2,1)
15    IF(EQPAR(K,2).EQ.0) THEN
```

```
        CALL ERRMSG(5)
        GOTO 10
      ENDIF
C
      IF(RVS(K).EQ.1) THEN
        EQPAR(K,2)=1./EQPAR(K,2)
        EQPAR(K,1)=-EQPAR(K,1)*EQPAR(K,2)
      ENDIF
C
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    IF(RVS(K).EQ.1 .AND. OPT2.EQ.'U')THEN
        TT2=1./EQPAR(K,2)
        TT1=-EQPAR(K,1)*TT2
C
        WRITE(JLUN,1003)CHIO(K,2),IEQ2S(K,2),TT1,TT2,
     +  CHIO(K,1),IEQ2S(K,1)
C
        IF(NOIO(K).EQ.4) THEN
          WRITE(JLUN,1003)CHIO(K,4),IEQ2S(K,1),TT1,TT2,
     +    CHIO(K,3),IEQ2S(K,2)
        ENDIF
C
       ELSE
          WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),
     +    EQPAR(K,2),CHIO(K,2),IEQ2S(K,2)
C
          IF(NOIO(K).EQ.4) THEN
            WRITE(JLUN,1003)CHIO(K,3),IEQ2S(K,2),EQPAR(K,1),
     +      EQPAR(K,2),CHIO(K,4),IEQ2S(K,1)
          ENDIF
C
       ENDIF
C
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' * ',A1,I2)
      RETURN
      END
C
C---FCNSUB3-----------------------------------------------
C
      SUBROUTINE FCNSUB3(K)
C
C-----FUNCTION TYPE : SIN
C     Y=C1*SIN(C2*X+C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,3)
      CALL PARTAK(3,K)
```

```
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(3,K)
      RETURN
      END
C
C---FCNSUB4-----------------------------------------------------
C
      SUBROUTINE FCNSUB4(K)
C
C-----FUNCTION TYPE : COS
C     Y=C1*COS(C2*X +C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,4)
      CALL PARTAK(4,K)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(4,K)
      RETURN
      END
C
C---FCNSUB5-----------------------------------------------------
C
      SUBROUTINE FCNSUB5(K)
C
C-----FUNCTION TYPE : ASIN  (SIN INVERSE)
C     Y=C1*ASIN(C2*X +C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,5)
      CALL PARTAK(5,K)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(5,K)
      RETURN
      END
C
C---FCNSUB6-----------------------------------------------------
C
```

```
      SUBROUTINE FCNSUB6(K)
C
C-----FUNCTION TYPE : ACOS
C     Y=C1*ACOS(C2*X +C3) +C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,6)
      CALL PARTAK(6,K)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(6,K)
      RETURN
      END
C
C---FCNSUB7-----------------------------------------------------
C
      SUBROUTINE FCNSUB7(K)
C
C-----FUNCTION TYPE : ABS2  (ABS SQUARE)
C     Y=C1*X*ABS(X)
C-----WRITTEN BY : TSUN-YU CHOU,    AUGUST 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 200
      CALL IORVSR(K,7)
100   CALL GENCON(K,7,1,1)
120   IF(EQPAR(K,1).LE.0)THEN
        CALL ERRMSG(5)
        GOTO 100
      ENDIF
C
      IF(RVS(K).EQ.1.)  EQPAR(K,1)=1./SQRT(EQPAR(K,1))
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 200
      RETURN
C
200   IF( (RVS(K).EQ.1).AND.(OPT2.EQ.'U') ) THEN
        TT=1./( EQPAR(K,1)*EQPAR(K,1) )
C
        WRITE(JLUN,1003)CHIO(K,2),IEQ2S(K,2),TT,CHIO(K,1),
     +                  IEQ2S(K,1),CHIO(K,1),IEQ2S(K,1)
        IF(NOIO(K).EQ.4) THEN
          WRITE(JLUN,1003)CHIO(K,4),IEQ2S(K,1),TT,CHIO(K,3),
     +                    IEQ2S(K,2),CHIO(K,3),IEQ2S(K,2)
        ENDIF
```

```
C
      ELSE
C
        WRITE(JLUN,1005) CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),CHIO(K,2),
     +                   IEQ2S(K,2),CHIO(K,2),IEQ2S(K,2)
        IF(NOIO(K).EQ.4) THEN
          WRITE(JLUN,1005) CHIO(K,3),IEQ2S(K,1),EQPAR(K,1),CHIO(K,4),
     +                     IEQ2S(K,1),CHIO(K,4),IEQ2S(K,1)
        ENDIF
      ENDIF
C
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' * ',A1,I2,' * ABS( ',
     +          A1,I2,' )' )
1005  FORMAT(/,5X,A1,I2,' =',1PE12.4,' * SIGN( ',A1,I2,' )',
     +          ' * SQRT( ABS( ',A1,I2,' ) )' )
C
      RETURN
      END
C
C---FCNSUB8-----------------------------------------------------
C
      SUBROUTINE FCNSUB8(K)
C
C-----FUNCTION TYPE : ABSQRT  (ABS SQRUARE ROOT)
C     Y=C1* SIGN(X) * SQRT( ABS(X) )
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 200
      CALL IORVSR(K,8)
100   CALL GENCON(K,8,1,1)
120   IF(EQPAR(K,1).LE.0)THEN
        CALL ERRMSG(5)
        GOTO 100
      ENDIF
C
      IF(RVS(K).EQ.1.) EQPAR(K,1)=1./(EQPAR(K,1)*EQPAR(K,1))
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 200
      RETURN
C
200   IF( (RVS(K).EQ.1).AND.(OPT2.EQ.'U') ) THEN
      TT=1./SQRT(EQPAR(K,1))
C
        WRITE(JLUN,1003) CHIO(K,2),IEQ2S(K,2),TT,CHIO(K,1),
     +                   IEQ2S(K,1),CHIO(K,1),IEQ2S(K,1)
        IF(NOIO(K).EQ.4) THEN
          WRITE(JLUN,1003) CHIO(K,4),IEQ2S(K,1),TT,CHIO(K,3),
     +                     IEQ2S(K,2),CHIO(K,3),IEQ2S(K,2)
        ENDIF
C
      ELSE
```

```
C
         WRITE(JLUN,1005) CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),CHIO(K,2),
     +                    IEQ2S(K,2),CHIO(K,2),IEQ2S(K,2)
         IF(NOIO(K).EQ.4) THEN
           WRITE(JLUN,1005) CHIO(K,3),IEQ2S(K,1),EQPAR(K,1),CHIO(K,4),
     +                      IEQ2S(K,1),CHIO(K,4),IEQ2S(K,1)
         ENDIF
C
       ENDIF
C
1003   FORMAT(/,5X,A1,I2,' =',1PE12.4,' * SIGN( ',A1,I2,' )',
     +          '* SQRT( ABS( ',A1,I2,' ) )' )
1005   FORMAT(/,5X,A1,I2,' =',1PE12.4,' * SIGN( ',A1,I2,' )',
     +           ' * ( ',A1,I2,' ) ** 2')
C
       RETURN
       END
C
C---FCNSUB9-----------------------------------------------------
C
       SUBROUTINE FCNSUB9(K)
C
C-----FUNCTION TYPE : EXP
C      Y=C1*EXP(C2X+C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 60
       CALL IORVSR(K,9)
       CALL PARTAK(9,K)
       CALL ERRMSG(7)
       CALL CMREAD
       IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
       RETURN
C
60     CALL SUBWTR(9,K)
       RETURN
       END
C
C---FCNSUB10----------------------------------------------------
C
       SUBROUTINE FCNSUB10(K)
C
C-----FUNCTION TYPE : ALOG
C      Y=C1*ALOG(C2*X+C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 60
       CALL IORVSR(K,10)
       CALL PARTAK(10,K)
       CALL ERRMSG(7)
       CALL CMREAD
```

```
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(10,K)
      RETURN
      END
C
C---FCNSUB11-----------------------------------------------------
C
      SUBROUTINE FCNSUB11(K)
C
C-----FUNCTION TYPE : DSIN
C     Y=C1/SIN(C2*X + C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,11)
      CALL PARTAK(11,K)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(11,K)
      RETURN
      END
C
C---FCNSUB12-----------------------------------------------------
C
      SUBROUTINE FCNSUB12(K)
C
C-----FUNCTION TYPE : DCOS
C     Y=C1/COS(C2*X + C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      CALL IORVSR(K,12)
      CALL PARTAK(12,K)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CALL SUBWTR(12,K)
      RETURN
      END
C
C---FCNSUB13-----------------------------------------------------
C
      SUBROUTINE FCNSUB13(K)
C
```

```
C-----FUNCTION TYPE : DASIN
C      Y=C1/ASIN(C2*X + C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 60
       CALL IORVSR(K,13)
       CALL PARTAK(13,K)
       CALL ERRMSG(7)
       CALL CMREAD
       IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
       RETURN
C
60     CALL SUBWTR(13,K)
       RETURN
       END
C
C---FCNSUB14--------------------------------------------------
C
       SUBROUTINE FCNSUB14(K)
C
C-----FUNCTION TYPE : DACOS
C      Y=C1/ACOS(C2*X + C3) + C4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 60
       CALL IORVSR(K,14)
       CALL PARTAK(14,K)
       CALL ERRMSG(7)
       CALL CMREAD
       IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
       RETURN
C
60     CALL SUBWTR(14,K)
       RETURN
       END
C
C---FCNSUB15--------------------------------------------------
C
       SUBROUTINE FCNSUB15(K)
C
C-----FUNCTION TYPE : POLY
C      Y=C0 + C1*X + C2*X**2 + C3*X**3 + C4*X**4
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
       IF(OPT1.EQ.'LOOK  ') GOTO 60
       IF(RVS(K).EQ.1)THEN
         OPT2='N'
          CALL ERRMSG(6)
         RETURN
       ENDIF
```

```
C
      WRITE(JLUN,1001)CHIO(K,1),IEQ2S(K,1),CHIO(K,2),IEQ2S(K,2),
     +  CHIO(K,2),IEQ2S(K,2),CHIO(K,2),IEQ2S(K,2),
     +  CHIO(K,2),IEQ2S(K,2)
1001  FORMAT(/,5X,A1,I2,' = C0 + C1*',A1,I2,' + C2*(',A1,I2,
     +   ')**2 + C3*('A1,I2,')**3 + C4*(',A1,I2,')**4',/)
C
      CALL GENCON(K,15,1,0)
      CALL GENCON(K,15,2,1)
      CALL GENCON(K,15,3,2)
      CALL GENCON(K,15,4,3)
      CALL GENCON(K,15,5,4)
C
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),EQPAR(K,2),
     +   CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),
     +   CHIO(K,2),IEQ2S(K,2),EQPAR(K,4),
     +   CHIO(K,2),IEQ2S(K,2),EQPAR(K,5),CHIO(K,2),IEQ2S(K,2)
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' *',A1,I2,
     +    ' +',1PE12.4,' *(',A1,I2,')**2',//,9X,
     +    '+',1PE12.4,' *(',A1,I2,')**3 +',
     +    1PE12.4,' *(',A1,I2,')**4')
C
      RETURN
      END
C
C---FCNSUB16-----------------------------------------------------
C
      SUBROUTINE FCNSUB16(K)
C
C-----FUNCTION TYPE : COLUMB
C
C     Y = C1 * SIGN(X)
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 50
      IF(RVS(K).EQ.1)THEN
        OPT2='N'
         CALL ERRMSG(6)
        RETURN
      ENDIF
C
      WRITE(JLUN,1001)CHIO(K,1),IEQ2S(K,1),CHIO(K,2),IEQ2S(K,2)
1001  FORMAT(/,5X,A1,I2,' = C1 * SIGN(',A1,I2,')',/)
C
      CALL GENCON(K,16,1,1)
      CALL ERRMSG(7)
```

```
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 50
      RETURN
C
50    WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),
     +   CHIO(K,2),IEQ2S(K,2)
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' * SIGN(',A1,I2,')' )
      RETURN
      END
C---FCNSUB17-----------------------------------------------
C
      SUBROUTINE FCNSUB17(K)
C
C-----FUNCTION TYPE :  BRKPT (BREAK POINT)
C       Y= Y1+C1*(X-X1), WHEN X <=X1
C       Y= Y1+C2*(X-X1), WHEN X1<X
C
C     EQPAR(K,1) STORE X1
C     EQPAR(K,2) STORE Y1
C     EQPAR(K,3) STORE C1
C     EQPAR(K,4) STORE C2
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 30
C
      WRITE(JLUN,1001)CHIO(K,2),IEQ2S(K,2),CHIO(K,1),IEQ2S(K,1)
1001  FORMAT(/,5X,'Y= Y1+C1*(X-X1), WHEN X <=X1',/,
     +          5X,'Y= Y1+C2*(X-X1), WHEN X1<X',//,
     +          25X,'   WHERE X = ',A1,I2,/,
     +          25X,'         Y = ',A1,I2,/)
C
      CALL GENPAR(K,17,1,1)
10    CALL GENCON(K,17,3,1)
      IF(RVS(K).EQ.1 .AND. EQPAR(K,3).EQ.0)THEN
        CALL ERRMSG(8)
        GO TO 10
      ENDIF
20    CALL GENCON(K,17,4,2)
      IF(RVS(K).EQ.1 .AND. EQPAR(K,4).EQ.0)THEN
        CALL ERRMSG(8)
        GO TO 20
      ENDIF
C
      IF(RVS(K).EQ.1)THEN
        CALL COOREX(K,1,2)
        EQPAR(K,3)=1./EQPAR(K,3)
        EQPAR(K,4)=1./EQPAR(K,4)
      ENDIF
C
      CALL ERRMSG(7)
      CALL CMREAD
```

```
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 30
      RETURN
C
30    CONTINUE
      IF(RVS(K).EQ.1 .AND. OPT2.EQ.'U')THEN
        TT1=EQPAR(K,2)
        TT2=EQPAR(K,1)
        TT3=1./EQPAR(K,3)
        TT4=1./EQPAR(K,4)
      WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),TT2,TT3,
     +       CHIO(K,2),IEQ2S(K,2),TT1,CHIO(K,2),IEQ2S(K,2),
     +       TT1,
     +       CHIO(K,1),IEQ2S(K,1),TT2,TT4,
     +       CHIO(K,2),IEQ2S(K,2),TT1,
     +       TT1,CHIO(K,2),IEQ2S(K,2)
C
      ELSE
      WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),EQPAR(K,3),
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),CHIO(K,2),IEQ2S(K,2),
     +       EQPAR(K,1),
     +       CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),EQPAR(K,4),
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +       EQPAR(K,1),CHIO(K,2),IEQ2S(K,2)
C
      ENDIF
C
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' * (',A1,I2,
     +        ' - ',1PE12.4,' ), ',/,35X,'WHEN ',A1,I2,' <= ',1PE12.4,
     +      //,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' * (',A1,I2,
     +        ' - ',1PE12.4,' ), ',/,35X,'WHEN ',1PE12.4,' < ',A1,I2)
C
      RETURN
      END
C
C---FCNSUB18-----------------------------------------------------
C
      SUBROUTINE FCNSUB18(K)
C
C-----FUNCTION TYPE : SATUR ( SATURATION )
C
C     Y = Y1                                , X  <= X1
C     Y = Y1+ ((Y2-Y1)/(X2-X1))*(X-X1)  , X1 < X < X2
C     Y = Y2                                , X2 <= X
C
C     STORAGE    PARDF      EQPAR
C        1         X1         X1
C        2         Y1         Y1
C        3         X2         X2
C        4         Y2         Y2
C
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------
```

```
      IF(OPT1.EQ.'LOOK  ') GOTO 60
      IF(RVS(K).EQ.1)THEN
        OPT2='N'
         CALL ERRMSG(6)
        RETURN
      ENDIF
C
      WRITE(JLUN,1001)CHIO(K,2),IEQ2S(K,2),CHIO(K,1),IEQ2S(K,1)
C
1001  FORMAT(/,5X,'Y = Y1                                  , X  <= X1',
     +       /,5X,'Y = Y1+ ((Y1-Y2)/(X1-X2))*(X-X1) , X1 < X < X2',
     +       /,5X,'Y = Y2                                  , X2 <= X'//,
     +     40X,' WHERE  X = ',A1,I2,/,
     +     40X,'        Y = ',A1,I2,/)
C
      CALL GENPAR(K,18,1,1)
      CALL GENPAR(K,18,3,2)
C
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 60
      RETURN
C
60    CONTINUE
      SLOPE=(EQPAR(K,4)-EQPAR(K,2))/(EQPAR(K,3)-EQPAR(K,1))
      WRITE(JLUN,1003)
     +      CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),
     +      CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +      CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),SLOPE,
     +      CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +      EQPAR(K,1),CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),
     +      CHIO(K,1),IEQ2S(K,1),EQPAR(K,4),
     +      EQPAR(K,3),CHIO(K,2),IEQ2S(K,2)
C
1003  FORMAT(
     +       /,5X,A1,I2,' =',1PE12.4,'       , WHEN ',A1,I2,' <=',
     +        1PE12.4,//,
     +       5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' * ( ',A1,I2,
     +      ' - ',1PE12.4,' )',//,29X,', WHEN ',1PE12.4,' < ',A1,I2,
     +      ' < ',1PE12.4,//,
     +       5X,A1,I2,' =',1PE12.4,'       , WHEN ',1PE12.4,' <=',
     +        A1,I2)
C
      RETURN
      END
C
C---FCNSUB19-----------------------------------------------
C
      SUBROUTINE FCNSUB19(K)
C
C-----FUNCTION TYPE : ADJUST
C
C      Y= Y1 + C1 * (X-X1)           , WHEN X  <= X1
```

```
C       Y=Y1+((Y2-Y1)/(X2-X1))*(X-X1)  , WHEN X1 < X < X2
C       Y= Y2 + C2 * (X-X2)            , WHEN X2 <= X
C
C     EQPAR(K,1)  STORE VALUE OF X1
C     EQPAR(K,2)  STORE VALUE OF Y1
C     EQPAR(K,3)  STORE VALUE OF X2
C     EQPAR(K,4)  STORE VALUE OF Y2
C     EQPAR(K,5)  STORE VALUE OF C1
C     EQPAR(K,6)  STORE VALUE OF C2
C
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 80
C
      WRITE(JLUN,1001)CHIO(K,2),IEQ2S(K,2),CHIO(K,1),IEQ2S(K,1)
1001  FORMAT(/,5X,'Y= Y1 + C1 * (X-X1)            , WHEN X  <= X1',
     +       /,5X,'Y=Y1+((Y2-Y1)/(X2-X1))*(X-X1)  , WHEN X1 < X < X2',
     +       /,5X,'Y= Y2 + C2 * (X-X2)            , WHEN X2 <= X ',//,
     +       40X,' WHERE  X = ',A1,I2,/,
     +       40X,'        Y = ',A1,I2,/)
C
5     CALL GENPAR(K,19,1,1)
      CALL GENPAR(K,19,3,2)
      XD=EQPAR(K,3)-EQPAR(K,1)
      YD=EQPAR(K,4)-EQPAR(K,2)
      IF(RVS(K).EQ.1 .AND. YD.EQ.0)THEN
        CALL ERRMSG(8)
        GO TO 5
      ENDIF
      IF(RVS(K).EQ.0 .AND. XD.EQ.0)THEN
        CALL ERRMSG(9)
        GO TO 5
      ENDIF
10    CALL GENCON(K,19,5,1)
      IF(RVS(K).EQ.1 .AND. EQPAR(K,5).EQ.0)THEN
        CALL ERRMSG(8)
        GO TO 10
      ENDIF
20    CALL GENCON(K,19,6,2)
      IF(RVS(K).EQ.1 .AND. EQPAR(K,6).EQ.0)THEN
        CALL ERRMSG(8)
        GO TO 20
      ENDIF
C
      IF(RVS(K).EQ.1)THEN
        CALL COOREX(K,1,2)
        CALL COOREX(K,3,4)
        EQPAR(K,5)=1./EQPAR(K,5)
        EQPAR(K,6)=1./EQPAR(K,6)
      ENDIF
C
      CALL ERRMSG(7)
```

```
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 80
      RETURN
C
80    CONTINUE
      IF(RVS(K).EQ.0)THEN
        SLOPE=(EQPAR(K,4)-EQPAR(K,2))/(EQPAR(K,3)-EQPAR(K,1))
      ELSE IF(RVS(K).EQ.1)THEN
        TT1=EQPAR(K,2)
        TT2=EQPAR(K,1)
        TT3=EQPAR(K,4)
        TT4=EQPAR(K,3)
        TT5=1./EQPAR(K,5)
        TT6=1./EQPAR(K,6)
        SLOPE=(EQPAR(K,3)-EQPAR(K,1))/(EQPAR(K,4)-EQPAR(K,2))
      ENDIF
C
      IF(RVS(K).EQ.1 .AND. OPT2.EQ.'U')THEN
      WRITE(JLUN,1003)
     +       CHIO(K,2),IEQ2S(K,2),TT2,TT5,
     +       CHIO(K,1),IEQ2S(K,1),TT1,
     +       CHIO(K,1),IEQ2S(K,1),TT1,
     +       CHIO(K,2),IEQ2S(K,2),TT2,SLOPE,
     +       CHIO(K,1),IEQ2S(K,1),TT1,
     +       TT1,CHIO(K,1),IEQ2S(K,1),TT3,
     +       CHIO(K,2),IEQ2S(K,2),TT4,TT6,
     +       CHIO(K,1),IEQ2S(K,1),TT3,
     +       TT3,CHIO(K,1),IEQ2S(K,1)
C
      ELSE
      WRITE(JLUN,1003)
     +       CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),EQPAR(K,5),
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +       CHIO(K,1),IEQ2S(K,1),EQPAR(K,2),SLOPE,
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,1),
     +       EQPAR(K,1),CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),
     +       CHIO(K,1),IEQ2S(K,1),EQPAR(K,4),EQPAR(K,6),
     +       CHIO(K,2),IEQ2S(K,2),EQPAR(K,3),
     +       EQPAR(K,3),CHIO(K,2),IEQ2S(K,2)
      ENDIF
1003  FORMAT(
     +        //,5X,A1,I2,' =',1PE12.4,' +',1PE12.4,' * ( ',A1,I2,
     +    ' - ',1PE12.4,')',//,33X,', WHEN ',A1,I2,' <=',1PE12.4,
     +      //,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' * ( ',A1,I2,
     +    ' - ',1PE12.4,')',//,33X,', WHEN ',1PE12.4,' < ',A1,I2,
     +      ' < ',1PE12.4,//,
     +         5X,A1,I2,' =',1PE12.4,' +',1PE12.4,' * ( ',A1,I2,
     +    ' - ',1PE12.4,')',//,33X,', WHEN ',1PE12.4,' <=',A1,I2)
      RETURN
      END
C
C---FCNSUB20-----------------------------------------------
```

```
C
      SUBROUTINE FCNSUB20(K)
C
C-----WRITTEN BY: TSUN-YU CHOU,  NOV.1982
C-----FUNCTION TYPE : PWLIN  AND INTPLO
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 50
      IF(RVS(K).EQ.1)THEN
        OPT2='X'
          CALL ERRMSG(6)
        RETURN
      ENDIF
C
      IF(OPT3.EQ.'LINEAR') THEN
        IL=1
        WRITE(JLUN,900)
        WRITE(JLUN,920)
      ELSE IF(OPT3.EQ.'INTPLO') THEN
        IL=2
        WRITE(JLUN,910)
        WRITE(JLUN,920)
      ENDIF
      CALL INTERP(K,IL)
C
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 50
      RETURN
C
50    CONTINUE
      N=EQPAR(K,1)
      IF(IFCNTP(K).EQ.20)  THEN
        IL=1
        WRITE(JLUN,900)
      ELSE IF(IFCNTP(K).EQ.21)  THEN
        IL=2
        WRITE(JLUN,910)
      ENDIF
      WRITE(JLUN,930)N
      DO 60 I=1,N
60    WRITE(JLUN,935)I,XCO(IL,I),YCO(IL,I)
      IF(IL.EQ.1)THEN
       WRITE(JLUN,940)EQPAR(K,3),EQPAR(K,4)
      ENDIF
C
C-----FORMAT-----
900   FORMAT(/,5X,'PIECEWISE LINEAR :' )
910   FORMAT(/,5X,'INTERPOLATION POLYNOMIAL :')
920   FORMAT(/,7X,'MAXIMUM NUMBER OF DATA POINTS IS 20',
     +       /,7X,'X COORDINATE MUST BE IN ASCENDING ORDER.')
```

```
930   FORMAT(/,5X,I2,' DATA POINTS BEEN INPUT, THERE ARE :',
     +       //,13X,'NO.',10X,'X',16X,'Y')
935   FORMAT(/,(10X,I5,5X,E12.4,5X,E12.4))
940   FORMAT(/,6X,'C1= ',F12.4,/,6X,'C2= ',F12.4)
      RETURN
      END
C
C
C-------------------------------------------------------------
C
      SUBROUTINE INTERP(K,IL)
C
C-----WRITTEN BY: TSUN-YU CHOU, OCT.1982
C
C     NEWTON'S DIVIDED-DIFFERENCE INTERPOLATION POLYNOMIAL.
C     CALLED BY SUBROUTINE FCNSUB20
C
C     EQPAR(K,1)  STORE NO. OF DATA POINTS
C     EQPAR(K,2)  STORE IDEG
C     EQPAR(20,3)  STORE SLOPE 1
C     EQPAR(20,4)  STORE SLOPE 2
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
10    WRITE(JLUN,900)
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        N=5
        EQPAR(K,1)=N
        GOTO 20
      ELSE
        DECODE(80,*,CMN,ERR=899)N
        EQPAR(K,1)=N
      ENDIF
C
20    M=N-1
      IF(OPT3.EQ.'LINEAR') THEN
        IDEG=1
      ELSE IF(OPT3.EQ.'INTPLO') THEN
        IDEG=M
      ENDIF
      EQPAR(K,2)=IDEG
C
      DO 100 I=1,N
30    WRITE(JLUN,920) I,I
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        XCO(IL,I)=I
      ELSE
        DECODE(80,*,CMN,ERR=40)CON
        XCO(IL,I)=CON
      ENDIF
```

```
C
      IF(I.EQ.1)GOTO 50
      DUM=XCO(IL,I)-XCO(IL,I-1)
      IF(DUM.LE.0.)THEN
        WRITE(JLUN,930)
        GOTO 10
      ELSE
        GOTO 50
      ENDIF
C
40    CALL ERRMSG(4)
      GOTO 30
C
50    WRITE(JLUN,940)  I,I
      CALL CMREAD
      IF(NOCMN.EQ.0)THEN
        YCO(IL,I)=I
      ELSE
        DECODE(80,*,CMN,ERR=60)CON1
        YCO(IL,I)=CON1
      ENDIF
      GOTO 100
60    CALL ERRMSG(4)
      GOTO 50
100   CONTINUE
C
      IF(IL.EQ.2)GOTO 125
C
      WRITE(JLUN,950)  XCO(IL,1)
      CALL GENCON(K,20,3,1)
      WRITE(JLUN,960)  XCO(IL,N)
      CALL GENCON(K,20,4,2)
C
125   CALL DTABLE(IL,N,M)
      RETURN
C
899   CALL ERRMSG(4)
      GOTO 10
C
C-----FORMAT-----
C
900   FORMAT(/,'ENTER NUMBER OF DATA POINTS (5):')
920   FORMAT(/,'ENTER COORDINATE X(',I2,'),   (',I2,'):')
930   FORMAT(/,' ERROR] X MUST ARRANGED IN ASCENDING ORDER,DO IT AGAIN')
940   FORMAT(/,'ENTER COORDINATE Y(',I2,'),   (',I2,'):')
950   FORMAT(/,'DY/DX FOR X LESS THAN ',E12.4,
     +     ' IS C1')
960   FORMAT(/,'DY/DX FOR X GREATER THAN ',E12.4,
     +     ' IS C2')
C
      END
C
C-------------------------------------------------------------
```

```
C
      SUBROUTINE DTABLE(IL,N,M)
C
C-----CALLED BY SUBROUTINE INTERP
C
      COMMON/EQ4/ XCO(2,20),YCO(2,20),TABLE(2,20,20)
C-------------------------------------------------------------
10    NM1=N-1
      DO 30 I=1,NM1
30    TABLE(IL,I,1)=( YCO(IL,I+1)-YCO(IL,I) )
     +              /( XCO(IL,I+1)-XCO(IL,I) )
      IF(M.LE.1)GOTO 100
C
      DO 50 J=2,M
      DO 50 I=J,NM1
      ISUB=I+1-J
      TABLE(IL,I,J)=(TABLE(IL,I,J-1)-TABLE(IL,I-1,J-1))
     +                     / (XCO(IL,I+1)-XCO(IL,ISUB) )
50    CONTINUE
C
100   CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
      FUNCTION FMIDV(IL,N,IDEG,XARG)
C
C-----CALLED BY SUBROUTINE NLEVAL, GETVAL
C
      COMMON/EQ4/ XCO(2,20),YCO(2,20),TABLE(2,20,20)
C-------------------------------------------------------------
      DO 100 I=1,N
      IF(I.EQ.N .OR. XARG.LE.XCO(IL,I)) GOTO 200
100   CONTINUE
C
200   MAX=I+IDEG/2
      IF(MAX.LE.IDEG) MAX=IDEG+1
      IF(MAX.GT.N) MAX=N
C
      YEST=TABLE(IL,MAX-1,IDEG)
      IF(IDEG.LE.1)GOTO 300
      IDEGM1=IDEG-1
      DO 250 I=1,IDEGM1
      ISUB1=MAX-I
      ISUB2=IDEG-I
250   YEST=YEST*(XARG-XCO(IL,ISUB1)) + TABLE(IL,ISUB1-1,ISUB2)
C
300   ISUB1=MAX-IDEG
      FMIDV=YEST*(XARG-XCO(IL,ISUB1))+YCO(IL,ISUB1)
C
      RETURN
      END
```

```
C
C---FCNSUB22-----------------------------------------------------
      SUBROUTINE FCNSUB22(K)
C
C-----FUNCTION TYPE : RCPSQR
C
C     Y = C0 + C1 / ( 1+(X/C2)**2 )
C
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------------
      IF(OPT1.EQ.'LOOK  ') GOTO 50
      IF(RVS(K).EQ.1)THEN
        OPT2='N'
         CALL ERRMSG(6)
        RETURN
      ENDIF
C
      WRITE(JLUN,1001)CHIO(K,1),IEQ2S(K,1),CHIO(K,2),IEQ2S(K,2)
1001  FORMAT(/,5X,A1,I2,' = C0 + C1 / ( 1 + (',A1,I2,' / C2)**2 )',/)
C
      CALL GENCON(K,22,1,0)
      CALL GENCON(K,22,2,1)
      CALL GENCON(K,22,3,2)
      CALL ERRMSG(7)
      CALL CMREAD
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 50
      RETURN
C
50    WRITE(JLUN,1003)CHIO(K,1),IEQ2S(K,1),EQPAR(K,1),EQPAR(K,2),
     +   CHIO(K,2),IEQ2S(K,2),EQPAR(K,3)
1003  FORMAT(/,5X,A1,I2,' =',1PE12.4,' + ',1PE12.4,' / ( 1 + (',
     +     A1,I2,' /',1PE12.4,')**2 )' )
      RETURN
      END
C
C---NLENAL-------------------------------------------------------
C
      SUBROUTINE NLEVAL
C
C-----WRITTEN BY : TSUN-YU CHOU,     AUGUST 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-----------------------------------------------------------------
C
20    WRITE(JLUN,21)
21    FORMAT(/,'GIVE FUNCTION NUMBER :')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 20
      DECODE(80,*,CMN,ERR=40)MFCN
      IF (MFCN.LT.0) THEN
        PRINT *,'ERROR] INCORRECT FUNCTION NUMBER'
```

```
        GOTO 20
      ELSE IF(MFCN.EQ.0)THEN
        RETURN
      ENDIF
C
      DO 30 JJ=1,NEQS
      IF(IEQ2S(JJ,1) .EQ. MFCN) THEN
        I=JJ
        GOTO 60
      ENDIF
30    CONTINUE
40    PRINT *,'ERROR] BAD FUNCTION LABEL.'
      GOTO 20
C
60    OPT1='LOOK  '
      OPT2='S'
C
      CALL FCNLOOK(I)
C
100   WRITE(JLUN,101)
101   FORMAT(/,'GIVE THE VALUE OF INPUT VARIABLE :')
      CALL CMREAD
      DECODE(80,*,CMN,ERR=120)XXX
C
      CALL GETVAL(XXX,Y,1,IFCNTP(I))
      WRITE(JLUN,110)Y
110   FORMAT(/,'VALUE OF OUTPUT VARIABLE IS : ',1PE12.4,/)
      GO TO 130
C
120   PRINT *,'ERROR, BAD DATA'
      GOTO 100
C
130   WRITE(JLUN,140)
140   FORMAT(/,'EVALUATE ANOTHER FUNCTION ? (YES):')
      CALL CMREAD
      IF(NOCMN.EQ.0)GOTO 20
      IF(CMN.EQ.'Y' .OR. CMN.EQ.'YES')GOTO 20
C
      RETURN
      END
C
C---GETVAL-----------------------------------------------------
C
      SUBROUTINE GETVAL(XINP,YOUTP,NOEQ,NFCNTP)
C
C-----WRITTEN BY : TSUN-YU CHOU,     AUGUST 1982
C
$INSERT COMFILE
$INSERT COMAREA
C-------------------------------------------------------------
C1
      IF(NFCNTP.EQ.1)THEN
        YOUTP=EQPAR(NOEQ,1)
```

```
        RETURN
C2
      ELSE IF (NFCNTP.EQ.2) THEN
        YOUTP=EQPAR(NOEQ,1) + EQPAR(NOEQ,2) * XINP
        RETURN
C3
      ELSE IF (NFCNTP.EQ.3) THEN
       IF (RVS(NOEQ) .EQ.0) THEN
        YOUTP=EQPAR(NOEQ,1) *SIN(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ELSE IF (RVS(NOEQ) .EQ.1) THEN
        YOUTP=EQPAR(NOEQ,1) *ASIN(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ENDIF
        RETURN
C4
      ELSE IF (NFCNTP.EQ.4) THEN
       IF (RVS(NOEQ) .EQ.0) THEN
        YOUTP=EQPAR(NOEQ,1) *COS(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ELSE IF (RVS(NOEQ) .EQ.1) THEN
        YOUTP=EQPAR(NOEQ,1) *ACOS(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ENDIF
        RETURN
C5
      ELSE IF (NFCNTP.EQ.5) THEN
       IF (RVS(NOEQ) .EQ.0) THEN
        YOUTP=EQPAR(NOEQ,1) *ASIN(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ELSE IF (RVS(NOEQ) .EQ.1) THEN
        YOUTP=EQPAR(NOEQ,1) *SIN(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ENDIF
        RETURN
C6
      ELSE IF (NFCNTP.EQ.6) THEN
       IF (RVS(NOEQ) .EQ.0) THEN
        YOUTP=EQPAR(NOEQ,1) *ACOS(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ELSE IF (RVS(NOEQ) .EQ.1) THEN
        YOUTP=EQPAR(NOEQ,1) *COS(EQPAR(NOEQ,2) *XINP+EQPAR(NOEQ,3))
     +       +EQPAR(NOEQ,4)
       ENDIF
        RETURN
C7
      ELSE IF (NFCNTP.EQ.7) THEN
       IF (RVS(NOEQ) .EQ.0) THEN
        YOUTP=EQPAR(NOEQ,1) * XINP * ABS(XINP)
       ELSE IF (RVS(NOEQ) .EQ.1) THEN
        YOUTP=EQPAR(NOEQ,1) *SIGN(1,XINP) *SQRT(ABS(XINP))
       ENDIF
        RETURN
```

```
C8
      ELSE IF (NFCNTP.EQ.8) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) *SIGN (1, XINP) *SQRT (ABS (XINP))
       ELSE IF (RVS (NOEQ) .EQ.1) THEN
        YOUTP=EQPAR (NOEQ, 1)  * XINP * ABS (XINP)
       ENDIF
        RETURN
C9
      ELSE IF (NFCNTP.EQ.9) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) *EXP (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3)  )
     +        +EQPAR (NOEQ, 4)
       ELSE IF (RVS (NOEQ) .EQ.1) THEN
        YOUTP=EQPAR (NOEQ, 1) *ALOG (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3))
     +        +EQPAR (NOEQ, 4)
       ELSE IF (RVS (NOEQ) .EQ.1) THEN
       ENDIF
        RETURN
C10
      ELSE IF (NFCNTP.EQ.10) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) *ALOG (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3))
     +        +EQPAR (NOEQ, 4)
       ELSE IF (RVS (NOEQ) .EQ.1) THEN
        YOUTP=EQPAR (NOEQ, 1) *EXP (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3)  )
     +        +EQPAR (NOEQ, 4)
       ENDIF
        RETURN
C11
      ELSE IF (NFCNTP.EQ.11) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) /SIN (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3))
     +        +EQPAR (NOEQ, 4)
       ELSE
        YOUTP=EQPAR (NOEQ, 1) *ASIN (EQPAR (NOEQ, 2) / (XINP-EQPAR (NOEQ, 3)))
     +        +EQPAR (NOEQ, 4)
       ENDIF
       RETURN
C12
      ELSE IF (NFCNTP.EQ.12) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) /COS (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3))
     +        +EQPAR (NOEQ, 4)
       ELSE
        YOUTP=EQPAR (NOEQ, 1) *ACOS (EQPAR (NOEQ, 2) / (XINP-EQPAR (NOEQ, 3)))
     +        +EQPAR (NOEQ, 4)
       ENDIF
       RETURN
C13
      ELSE IF (NFCNTP.EQ.13) THEN
       IF (RVS (NOEQ) .EQ.0) THEN
        YOUTP=EQPAR (NOEQ, 1) /ASIN (EQPAR (NOEQ, 2) *XINP+EQPAR (NOEQ, 3))
```

```
     +        +EQPAR(NOEQ,4)
        ELSE
         YOUTP=EQPAR(NOEQ,1)*SIN(EQPAR(NOEQ,2)/(XINP-EQPAR(NOEQ,3)))
     +        +EQPAR(NOEQ,4)
        ENDIF
        RETURN
C14
       ELSE IF(NFCNTP.EQ.14)THEN
        IF(RVS(NOEQ).EQ.0)THEN
         YOUTP=EQPAR(NOEQ,1)/ACOS(EQPAR(NOEQ,2)*XINP+EQPAR(NOEQ,3))
     +        +EQPAR(NOEQ,4)
        ELSE
         YOUTP=EQPAR(NOEQ,1)*COS(EQPAR(NOEQ,2)/(XINP-EQPAR(NOEQ,3)))
     +        +EQPAR(NOEQ,4)
        ENDIF
        RETURN
C15
       ELSE IF(NFCNTP.EQ.15)THEN
         YOUTP=EQPAR(NOEQ,1) +EQPAR(NOEQ,2)*XINP +EQPAR(NOEQ,3)*XINP**2
     +        +EQPAR(NOEQ,4)*XINP**3 + EQPAR(NOEQ,5)*XINP**4
         RETURN
C16
       ELSE IF(NFCNTP.EQ.16)THEN
         IF(XINP.EQ.0.)THEN
           YOUTP=0.
         ELSE
           YOUTP=SIGN(1,XINP) * EQPAR(NOEQ,1)
         ENDIF
         RETURN
C17
       ELSE IF(NFCNTP.EQ.17)THEN
         IF(XINP.LE.EQPAR(NOEQ,1))THEN
           YOUTP=EQPAR(NOEQ,2)+EQPAR(NOEQ,3)*(XINP-EQPAR(NOEQ,1))
         ELSE
           YOUTP=EQPAR(NOEQ,2)+EQPAR(NOEQ,4)*(XINP-EQPAR(NOEQ,1))
         ENDIF
         RETURN
C18
       ELSE IF(NFCNTP.EQ.18)THEN
         IF(XINP.LE.EQPAR(NOEQ,1)) THEN
           YOUTP=EQPAR(NOEQ,2)
         ELSE IF(XINP.GE.EQPAR(NOEQ,3))THEN
           YOUTP=EQPAR(NOEQ,4)
         ELSE
           YOUTP=EQPAR(NOEQ,2) + ( (EQPAR(NOEQ,4)-EQPAR(NOEQ,2))/
     +        (EQPAR(NOEQ,3)-EQPAR(NOEQ,1)))*(XINP-EQPAR(NOEQ,1))
         ENDIF
         RETURN
C19
       ELSE IF(NFCNTP.EQ.19)THEN
         IF(XINP.LE.EQPAR(NOEQ,1))THEN
           YOUTP=EQPAR(NOEQ,2)+EQPAR(NOEQ,5)*(XINP-EQPAR(NOEQ,1))
         ELSE IF(XINP.GE.EQPAR(NOEQ,3))THEN
```

```
          YOUTP=EQPAR(NOEQ,4)+EQPAR(NOEQ,6)*(XINP-EQPAR(NOEQ,3))
        ELSE
          YOUTP=EQPAR(NOEQ,2) + ((EQPAR(NOEQ,4)-EQPAR(NOEQ,2)) /
     +      (EQPAR(NOEQ,3)-EQPAR(NOEQ,1))) * (XINP-EQPAR(NOEQ,1))
        ENDIF
        RETURN
C20
      ELSE IF(NFCNTP.EQ.20)THEN
        IL=1
        NN=EQPAR(NOEQ,1)
        IDEG=EQPAR(NOEQ,2)
        IF((XINP.GE.XCO(IL,1)).AND.(XINP.LE.XCO(IL,NN)))THEN
          YOUTP=FMIDV(IL,NN,IDEG,XINP)
        ELSE IF(XINP.LT.XCO(IL,1))THEN
          YOUTP=YCO(IL,1)+EQPAR(NOEQ,3)*(XINP-XCO(IL,1))
        ELSE IF(XINP.GT.XCO(IL,NN))THEN
          YOUTP=YCO(IL,NN)+EQPAR(NOEQ,4)*(XINP-XCO(IL,NN))
        ENDIF
        RETURN
C21
      ELSE IF(NFCNTP.EQ.21)THEN
        IL=2
        NP=EQPAR(NOEQ,1)
        IDEG=EQPAR(NOEQ,2)
         PRINT *,'ORDER OF POLYNOMIAL CAN BE CHANGED'
100      PRINT *,'ENTER THE ORDER :'
         CALL CMREAD
         DECODE(80,*,CMN,ERR=110)N1
         IDEG=N1
         IF(IDEG.GE.NP)THEN
          WRITE(JLUN,120)NP
          GOTO 100
         ENDIF
        YOUTP=FMIDV(IL,NP,IDEG,XINP)
        RETURN
110      PRINT *,'ERROR, BAD DATA.'
         GOTO 100
120      FORMAT(/,'ORDER MUST BE LESS THAN ',I3)
C
C22
      ELSE IF(NFCNTP.EQ.22)THEN
        YOUTP=EQPAR(NOEQ,1)+ EQPAR(NOEQ,2) / (1+(XINP/EQPAR(NOEQ,3))**2)
        RETURN
C
C-------------------------------------------------------------
      ENDIF
C-------------------------------------------------------------
      RETURN
      END
C
C-------------------------------------------------------------
C---COMFILE
C
```

```
      INTEGER*4 IERRF,MCMAM
      CHARACTER*4 IALPHA,IELNAM(50),NTYP(9),HEAD(20)
      CHARACTER*1 LINE(72),LABEL(72)
      CHARACTER*1 IEPFQL(20),IXUDL(20),LBL1(1,5)
C
      COMMON /BK1/ ILUN,JLUN,IERRF,IPRLST(20),HEAD
      COMMON /BK2/ NREAD,LINE,LABEL,INT,REAL,IALPHA,KEY,NCOL,
     +             LI,MARGIN,NEXT,IST,ISP
      COMMON /BK3/ NEL,NBD,IELLST(50),IELNAM,NBIMX(100),
     +             IBMX(50,2),NPTR(51),MNEL,MNBD,NTYP
      COMMON /BK4/ NMCR,NMCPT,NSTD,MNMCR,MNMCPT,
     +             MCNAM(10),MPTRA(10),MPTRB(10),MBIMX(50)
      COMMON /BK5/ ICMX(100),MELMNT(40),MBND(40),MIT(40),NTEMP(40),
     +             MNSTKM,MNSTKN,MTOP,NTOP,LEVEL,IBLIS(13),ICCNOD(13),
     +             IBPNT(4),NODLIS(4),JPNT,KBOND,INODE,MAXBD,MAXND
      COMMON /BK6/ IBEQ(50),IBT(50),NBEX,NFI,NFD,NFL,NFS,NBIN
      COMMON /BK7/ PAR(100),IPTR(51),NPAR,MNPAR,IPFLG
      COMMON /BK8/ S(150),IPS(150),LENS,MAXS,
     +             T(150),IPT(150),LENT,MAXT,
     +             WK(100),IWK(100),LENWK,MAXWK
      COMMON /BK9/ FL(100),IPFL(100),LENFL,MAXFL,
     +             FS(100),IPFS(100),LENFS,MAXFS,
     +             W2(100),IW2(100),LENW2,MAXW2
      COMMON /BK10/ A(150),IPA(150),LENA,MAXA,
     +              B(100),IPB(100),LENB,MAXB,
     +              E(50),IPE(50),LENE,MAXE,
     +              IEPFQL,IBDL(20),IXUDL,IDXL(20),NREQ,MAXXUD
      COMMON /BK11/ RM(225),IPRM(225),LENRM,MAXRM,
     +              RN(225),IPRN(225),LENRN,MAXRN,
     +              XIN(15),X(15),IPX(15),LENX,MAXX,
     +              U(10),IPU(10),LENU,MAXU,
     +              DX(15),IPDX(15),LENDX,
     +              IUTYP(10),UPAR(10,10),NARGU(10),
     +              TIN,TFIN,DT,KLIM,DEL,NSAV,MAXRES,RES(101,20)
      COMMON /BK12/ T1,T2,DELTA,JX,JDX(5),LBL2(2,5),JDXMAX,
     +              YMAX(20),YMIN(20),YSCLO,YSCHI,LBL1
      COMMON /BK13/ WR(10),WI(10)
      COMMON /BK14/ DATAP(5,101)
C
C***END COMFILE
C-------------------------------------------------------
C
C-------------------------------------------------------
C---COMAREA
C
      CHARACTER*80    CINPUT,CMN
      CHARACTER*6     CMATCH(30),CMSET1(30),INFSET(20)
      CHARACTER*6     FCNSET(30),OPT1,OPT3
      CHARACTER*1     CH10(50,4),OPT2,OPT4
      INTEGER         FPOCMN
C
      COMMON/READER/ CINPUT,CMN,FPOCMN,LPOCMN,NOCMN,
     +               CMATCH,IFOUND
```

```
C
      COMMON /CM1/ CMSET1,INFSET,FCNSET
      COMMON /CTL/ IPNT(10),OPT1,OPT2,OPT3,OPT4
      COMMON /EQ1/ NEQS,NFCN(50),NEQTP(50),IEQ2S(50,2),
     +             CH10,NO10(50)
      COMMON /EQ2/ PARDF(30,10),EQPAR(50,10)
      COMMON /EQ3/ IFCNTP(50),RVS(50)
      COMMON /EQ4/ XCO(2,20),YCO(2,20),TABLE(2,20,20)
C
C***END COMAREA
C-------------------------------------------------------
```