ANALYSIS: STATE EQUATION GENERATION AND TERMINAL MODELS
Final Report, 2 Feb. 1976 - 2 Apr. 1978 (University of South Florida) Unclas
123 p HC A06/MF A01 CSCL 09C G3/33 23572
Grant No. NSG-3096

COMPUTERIZED POWER SUPPLY ANALYSIS: STATE EQUATION GENERATION AND TERMINAL MODELS

Final Report
for the period from 2/2/76 to 4/2/78

University of South Florida
College of Engineering
Electrical and Electronic Systems Department

S. J. Garrett
Principal Investigator
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Analytical Background</td>
<td>3</td>
</tr>
<tr>
<td>3. USER's Manual for PROSE</td>
<td>12</td>
</tr>
<tr>
<td>4. Program Description</td>
<td>18</td>
</tr>
<tr>
<td>5. Examples</td>
<td>21</td>
</tr>
<tr>
<td>6. Summary and Conclusions</td>
<td>33</td>
</tr>
<tr>
<td>Appendix Listing of FORTRAN Programs</td>
<td>34</td>
</tr>
</tbody>
</table>
1. Introduction

To aid engineers that design power supply systems, special analytical tools are being introduced. These tools include integration routines that start with the description of a power supply in state-equation form and yield analytical results that include transient performance, steady-state equilibrium stability, and the response of the power supply to spurious inputs (audio susceptibility).

The Department of Electrical Engineering at the University of South Florida has worked to develop two analysis tools that can be used with the state-equation analysis package.

The first tool uses a computer program that works with the SUPER*SCEPTRE circuit analysis program and prints the state equations for an electrical network. The state equations developed automatically by the computer program are used to develop an algorithm for reducing the number of state variables required to describe an electrical network. In this way a second tool is obtained in which the order of the network is reduced and a simpler terminal model is obtained. The price of the reduction in order is a reduction in the accuracy with which the model represents the system.

At the conclusion of the first year, the computer program that converts the circuit description of an electrical network to a set of state equations was completed.

In the second year the computer program was modified to accept a larger class of circuits including circuits with mutual inductance. The work on reducing the number of state variables required to describe an electrical network was completed and is described in this report.

The first task that was completed was the development of a computer program that would print the state-equations for an electrical network. The analytical background for the state equations of an electrical network is presented in Chapter 2. From the analysis in Chapter 2, the generality of the approach can be seen together with the reason for the presentation of data as matrices which are inverted in the state equations. The analytical basis for reducing the number of state variables using eigenvalue sensitivities concludes Chapter 2.
In Chapter 3 a description of the steps required to get a set of state equations is given. Chapter 3 is a user's manual for the program to PROduce State Equations (PROSE program). The user begins with an electrical network. By following the directions in Chapter 3, he can obtain the state equations for the network. Additional features of the program are also described in Chapter 3.

The description of the programs as written in FORTRAN is given in Chapter 4 and a complete listing of the program appears in the Appendix.

Several examples are given in Chapter 5 that illustrate the use of the PROSE program in obtaining the state equations for an electrical circuit.

A summary of the work accomplished under this grant appears together with recommendations for continued research in Chapter 6.
2. State Equations

The standard form of the state equations for an electrical or mechanical system is

\[ \dot{x} = f(x, u, t) \]
\[ y = g(x, u, t) \]

where \( x \) is an \( n \times 1 \) state vector,
\( y \) is an \( m \times 1 \) output vector,
\( u \) is a \( p \times 1 \) input vector,
\( t \) is time,
\( f \) is the state transition function and
\( g \) is output function.

When the system is linear, the equations can be written as

\[ \dot{x} = Ax + Bu \]
\[ y = Cx + Du \]

where \( A \) is \( n \times n \) transition matrix,
\( B \) is \( n \times p \)
\( C \) is \( m \times n \)
\( D \) is \( m \times p \)

The matrices \( A, B, C, \) and \( D \) may be functions of time. In certain nonlinear problems the matrix representation is possible. The elements of the matrix are given as an algebraic expression in terms of circuit components such as resistors, inductors and capacitors. The value of a component may depend upon a voltage or current (the element may be nonlinear).

The state equations for an electrical circuit can be derived by the following steps:

The circuit diagram is the starting point. The model for each circuit device is obtain as a collection of two terminal elements.

The final circuit diagram contains

a. capacitors
b. resistors
c. inductors
d. independent voltage sources
e. dependent voltage sources
f. independent current sources
g. dependent current sources
2. The graph of the circuit is used to obtain a tree with the tree branches selected in the following order:
   a. voltage sources
   b. capacitors
   c. resistors
   d. inductors

All voltage sources are selected before any capacitors are selected; capacitors are selected to add new nodes to the tree before any resistors are added; etc. The circuit elements which are not tree branches are called links.

3. The components are partitioned and listed in Table 1. The groups of components in the tree are listed across the top and the groups of components that are links are listed along the side.

<table>
<thead>
<tr>
<th></th>
<th>C4</th>
<th>R5</th>
<th>L6</th>
<th>E7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cl</td>
<td>F14</td>
<td>0</td>
<td>0</td>
<td>F17</td>
</tr>
<tr>
<td>R2</td>
<td>F24</td>
<td>F25</td>
<td>0</td>
<td>F27</td>
</tr>
<tr>
<td>L3</td>
<td>F34</td>
<td>F35</td>
<td>F36</td>
<td>F37</td>
</tr>
<tr>
<td>J8</td>
<td>F84</td>
<td>F85</td>
<td>F86</td>
<td>F87</td>
</tr>
</tbody>
</table>

where

Cl is a diagonal matrix of capacitor links  
R2 is a diagonal matrix of resistor links  
L3 is a matrix of inductor links  
J8 is a column matrix of current sources and

C4 is a diagonal matrix of capacitor tree branches  
R5 is a diagonal matrix of resistor tree branches  
L6 is a matrix of inductor tree branches  
E7 is a column matrix of voltage sources and

The symbols \( F_{ij} \) are matrices that shown the connections between the \( i \) links and the \( j \) tree branches.

\[
F_{ij} = [fpq] , \quad 1 \leq p \leq \text{number of links in class } p , \quad 1 \leq q \leq \text{number of tree branches in class } j
\]
where
\[
fpq = \begin{cases} 
1 & \text{if tree branch } q \text{ is in the loop formed by placing link } p \\
& \text{back in the tree and if } q \text{ is in the same direction in the} \\
& \text{loop as } p. \\
-1 & \text{if tree branch } q \text{ is in the opposite direction in the loop} \\
& \text{defined by } p. \\
0 & \text{if tree branch } q \text{ is not in the loop defined by } p.
\end{cases}
\]

The state variables are selected as the column matrix \( I_3 \).

**Inductor Current State Variables**

If mutual inductances are allowed, the equations describing the inductor currents are given as

\[
\begin{align*}
L_3 \dot{i}_3 + L_{36} \dot{i}_6 &= V_3 \\
L_{63} \dot{i}_3 + L_6 \dot{i}_6 &= V_6
\end{align*}
\]

where
- \( L_3 \) is the inductance matrix for the inductor links
- \( L_{36} \) are the inductance matrices for mutual inductance between link inductors and tree-branch inductors and
- \( L_6 \) is the inductance matrix for tree-branch inductor.

Using the connection equations

\[
I_6 = F^{T}_{36} I_3 + F^{T}_{86} J_8
\]

which may be differentiated to give

\[
\dot{I}_6 = F^{T}_{36} \dot{I}_3 + F^{T}_{86} \dot{J}_8
\]

Equation (4) may be substituted into Equation (1) to get

\[
\begin{bmatrix} L_3 + L_{36} F^{T}_{36} \end{bmatrix} \dot{I}_3 = V_3 - L_{36} F^{T}_{86} J_8
\]

The connection equations relating link voltages to tree branch voltages gives

\[
V_3 = -F_{34} V_4 - F_{35} V_5 - F_{36} V_6 - F_{37} E_7
\]

The inductor voltage matrix is removed by the following steps beginning with Equation (2) which is repeated next.

\[
\begin{align*}
V_6 &= L_{63} \dot{i}_3 + L_6 \dot{i}_6 \\
&= L_{63} \dot{i}_3 + L_6 \left( F^{T}_{36} \dot{i}_3 + F^{T}_{86} \dot{J}_8 \right)
\end{align*}
\]

The equation for \( \dot{i}_3 \) becomes

\[
\begin{bmatrix} L_3 + L_{36} F^{T}_{36} + F_{36} L_{63} + F_{36} L_6 F^{T}_{36} \end{bmatrix} \dot{I}_3
= -F_{35} V_4 - F_{35} V_5 - F_{37} E_7
\]
To reduce this equation so that only the state variables $I_3$ and $V_4$ together with independent sources $E_7$, $E_y$, $J_8$, and $J_x$ appear, the resistor voltages and currents must be obtained as a function of the state variables.

**Resistor Links and Tree Branches**

The resistor voltages and currents can be expressed by the following matrix equations.

\[
\begin{bmatrix}
R_2 & 0 \\
0 & G_5
\end{bmatrix}
\begin{bmatrix}
I_2 \\
V_5
\end{bmatrix}
=
\begin{bmatrix}
I_3 \\
V_4
\end{bmatrix}
+
\begin{bmatrix}
0 & -F_{25} \\
F_{25} & 0
\end{bmatrix}
\begin{bmatrix}
I_3 \\
V_4
\end{bmatrix}
+
\begin{bmatrix}
0 & -F_{27} \\
F_{27} & 0
\end{bmatrix}
\begin{bmatrix}
J_8 \\
E_7
\end{bmatrix}
\]

Equation (9) can be solved as

\[
\begin{bmatrix}
I_2 \\
V_5
\end{bmatrix}
=
\begin{bmatrix}
R_2 & F_{25} \\
-F_{25} & G_5
\end{bmatrix}^{-1}
\begin{bmatrix}
0 & -F_{24} \\
F_{24} & 0
\end{bmatrix}
\begin{bmatrix}
I_3 \\
V_4
\end{bmatrix}
+
\begin{bmatrix}
0 & -F_{27} \\
F_{27} & 0
\end{bmatrix}
\begin{bmatrix}
J_8 \\
E_7
\end{bmatrix}
\]

(10)

If the matrix

\[
M_R^{-1} = \begin{bmatrix}
R_2 & F_{25} \\
-F_{25} & G_5
\end{bmatrix}
\]

is partitioned as

\[
M_R^{-1} = \begin{bmatrix}
m_1 & m_2 \\
m_3 & m_4
\end{bmatrix}
\]

then

\[
M_R^{-1} = \begin{bmatrix}
R_2 & F_{25} \\
-F_{25} & G_5
\end{bmatrix}^{-1} = \begin{bmatrix}
m_1 R_2 - m_2 F_{25} & m_1 F_{25} + m_2 G_5 \\
m_3 R_2 - m_4 F_{25} & m_3 F_{25} + m_4 G_5
\end{bmatrix} = \begin{bmatrix}
U & 0 \\
0 & U
\end{bmatrix}
\]

(11)

and

\[
\begin{bmatrix}
R_2 & F_{25} \\
-F_{25} & G_5
\end{bmatrix}^{-1} = \begin{bmatrix}
R_2 m_1 + F_{25} m_3 & R_2 m_2 + F_{25} m_4 \\
-F_{25} m_1 + G_5 m_3 & -F_{25} m_2 + G_5 m_4
\end{bmatrix} = \begin{bmatrix}
U & 0 \\
0 & U
\end{bmatrix}
\]

(12)

where $U$ is a unit matrix of the appropriate size. The solution for the matrix $M_R^{-1}$ is obtained from Equation (11) as
\[ m_1 = \left[ R_2 + F_{25} R_5 \right]^{-1} \]

\[ m_2 = -m_1 F_{25} R_5 \]

\[ m_3 = R_5^{T} m_1 \]

\[ m_4 = \left[ U - R_5^{T} m_1 F_{25} \right] R_5 \]

Beginning with Equation (12) yields another expression for \( m_1, m_2, m_3, \) and \( m_4 \) as

\[ m_4 = \left[ F_{25}^T G_2 F_{25} + G_5 \right]^{-1} \]

\[ m_3 = m_4 F_{25} \]

\[ m_2 = -G_2 F_{25} m_4 \]

\[ m_1 = G_2 \left[ U - G_2 F_{25} m_4 F_{25} \right] \]

Solving Equation (10) for \( V_5 \) yields

\[ V_5 = m_4 F_{35} V_3 - m_3 F_{24} V_4 - m_3 F_{27} E_7 + m_4 F_{85} J_8 \]  \hspace{1cm} (15)

Equation (15) is substituted into Equation (8) and solved to give the solution for state variable \( I_3 \) as

\[ I_3 = M_3^{-1} \left[ \left[ -F_{35} m_4 F_{25}^T I_3 \right] - \left[ F_{34} - F_{35} m_3 F_{24} \right] V_4 - \left[ F_{37} - F_{35} m_3 F_{27} \right] E_7 
- F_{35} m_4 F_{85} J_8 - \left[ 1.36 F_{36}^T + F_{36} I_6 F_{85}^T \right] J_8 \right] \]  \hspace{1cm} (16)

**Capacitor Voltages**

The voltage state equation can be obtained beginning with the capacitor equations

\[ C_1 V_1 = I_1 \]

\[ C_4 V_4 = I_4 \]  \hspace{1cm} (17)

The solution begins with the second equation in (17)

\[ C_4 V_4 = I_4 = F_{14}^T I_1 + F_{24}^T I_2 + F_{34}^T I_3 + F_{84}^T J_8 \]  \hspace{1cm} (18)

The variables \( I_1 \) and \( I_2 \) must be eliminated in the solution. Beginning with \( I_1 \)

\[ C_1^{-1} I_1 = \dot{V}_1 = \left[ -F_{14} V_4 - F_{17} E_7 \right] \]  \hspace{1cm} (19)
\begin{align*}
\dot{11}^{-1}11 &= -F_{14} [C4^{-1}j + F_{24} T_{12} + F_{34} T_{13} + J8] - F_{16} E7 \\
\text{Denoting } 11 &= M^{-1} \{ -F_{14} S4 [F_{24} T_{12} + F_{23} T_{13} + F_{34} T_{84} J8] - F_{17} E7 \}
\text{where } M &= [S1 + F_{14} S4 F_{14}]
\text{Solving Equation (10) for } 12 \\
12 &= -m [F_{24} V4 + F_{24} E7] + m2 [F_{35} I3 + F_{35} T_{86}]
\text{When Equations (21) and (22) are substituted into Equation (18) and the result simplified, the result is}
\dot{V4} = S4 \{ [-F_{14} M^{-1} S4] + U_{44} \} [F_{24} m2F_{35} + F_{34} T_{13}]
+ [F_{14} M^{-1} S4 F_{24} T_{4} - F_{24} m2F_{24} V4]
+ [-F_{14} M^{-1} S4 F_{24} - U_{44} ] [F_{35} + F_{24} m2F_{85} ] J8
+ [F_{14} M^{-1} S4 - U_{44} ] F_{24} m2F_{27} E7
\dot{V4} = -F_{14} M^{-1} S4 F_{24} T_{17} E7
\text{Combining Equation (16) and Equation (23) gives the state equations.}
\begin{bmatrix}
L3 \\
V4
\end{bmatrix}
= \begin{bmatrix}
M & 0 \\
0 & C4
\end{bmatrix}
\begin{bmatrix}
-F_{35} m4F_{35} T \\
-F_{37} + F_{35} m3F_{27}
\end{bmatrix}
\begin{bmatrix}
I3 \\
J8
\end{bmatrix}
+ \begin{bmatrix}
U_{44} - F_{14} m4F_{14} S4 F_{24} T_{17} + F_{24} T_{35} F_{14} S4 F_{24} T_{35} J8 \\
U_{44} - F_{14} m4F_{14} S4 F_{24} T_{17} + F_{24} T_{35} F_{14} S4 F_{24} T_{35} J8
\end{bmatrix}
\begin{bmatrix}
S4 \\
E7
\end{bmatrix}
\begin{bmatrix}
-L36F_{35} T - F_{35} m6F_{35} T \\
0
\end{bmatrix}
\begin{bmatrix}
J8 \\
E7
\end{bmatrix}
+ \begin{bmatrix}
0 \\
-F_{14} m4F_{14} F_{17}
\end{bmatrix}
\begin{bmatrix}
J8 \\
E7
\end{bmatrix}
\end{align*}
The state equations are produced as the output of the preprocessor computer program known as SCEPTRE I. These state equations are the input to the computer program SCEPTRE II which performs the integration required to obtain a transient solution.

Notice that Equation (24) appears in matrix form even though each element may depend on time or on another voltage or current that can be obtained at times when the state variables are known.

The state equations are calculated by processing the output of SCEPTRE I to obtain matrix equations in the form similar to Equation (24). While this equation is non-linear, it can be linearized around a quiescent value as specified by initial conditions for the state variables \[
\begin{bmatrix}
\mathbf{I}_3 \\
\mathbf{V}_4
\end{bmatrix}
\].

The linearized state equations are

\[
\begin{bmatrix}
\mathbf{I}_3 \\
\mathbf{V}_4
\end{bmatrix} = A \begin{bmatrix}
\mathbf{I}_3 \\
\mathbf{V}_4
\end{bmatrix} + B \begin{bmatrix}
\mathbf{J}_8 \\
\mathbf{E}_7
\end{bmatrix} + E \begin{bmatrix}
\mathbf{J}_8 \\
\mathbf{E}_7
\end{bmatrix}
\]

The eigenvalues of this equation are eigenvalues of the constant matrix, A.

In the representation of any physical system a compromise is made in choosing the order of the mathematical model. The order may be order of the differential equation that describes the system or may be the number of poles in the transfer function for a linear system. In any physical system that is modeled with lumped parameters, a decision must be made on how many energy storage elements (capacitors, inductors) are included in the model. More energy storage elements increase accuracy and computational difficulty. The number of independent energy storage elements is equal to the number of state variables required.

As the number of state variables is increased the accuracy can be increased. However, in choosing the equivalent circuit for a physical system such as a power supply, some energy storage elements may be selected that do not affect the accuracy of the model significantly. A simpler description that did not reduce accuracy would give a better model.

In a state variable description of a physical system, when an independent energy storage element is discarded, the number of state variables is reduced by one. Energy storage elements that are incidental to the accurate description of a physical system usually contribute to only the shortest time constants or the largest eigenvalues of the A matrix.
The eigenvalues of the A matrix are obtained using a subroutine. The sensitivity is calculated for each parameter by increasing the parameter by 1% and calculating the eigenvalue. For example the sensitivity to a resistor is given by

\[ S^p_R = \frac{R}{p} \frac{\Delta p}{\Delta R} = (100) \frac{\Delta p}{p} \]  

(27)

where \( p \) is the eigenvalue,
\( \Delta p \) is the change in the eigenvalue
\( \Delta R \) is the change in resistor and
\( R \) is the resistor value.

The sensitivities can be used to reduce the number of state variables. First the eigenvalue with the largest magnitude is selected. The sensitivities of this eigenvalue to each parameter is scanned. The circuit parameter (resistance, inductance, or capacitance) to which the largest eigenvalue has a sensitivity with a magnitude within 3 per cent of unity is selected. If the selected circuit parameter does not affect other eigenvalues, it is removed from the network. It is set equal to zero if the sensitivity is negative; if the sensitivity is positive, the circuit parameter is set to infinity.

The desire is to force the eigenvalue to negative infinity. From the equation

\[ \begin{align*}
\frac{\Delta p}{p} &= \sum R S^p_R \frac{\Delta R}{R} + \sum C S^p_C \frac{\Delta C}{C} + \sum L S^p_L \frac{\Delta L}{L}
\end{align*} \]

(28)

it can be inferred that the per cent change in an eigenvalue can be maximized by selecting a circuit element, \( \xi \), with the largest sensitivity and maximizing

\[ S^p_{\xi} \frac{\Delta \xi}{\xi} \]

(29)

for that term.

In every case the eigenvalue will be moved away from the origin toward infinity. If it is moved to infinity, a state variable is eliminated by the removal of a circuit element. The correlation of eigenvalue going to infinity and state variable disappearing can be observed from the characteristic equation:

\[ |S I - A| = S^n + a_{n-1} S^{n-1} + \ldots + a_1 S + a_0 \]

(30)

In factored form the characteristic equation becomes

\[ \kappa \left( \frac{S}{p_1} + 1 \right) \left( \frac{S}{p_2} + 1 \right) \ldots \left( \frac{S}{p_n} + 1 \right) \]

(31)
Obviously one term disappears if \( p_i \to \infty \) for the largest eigenvalue \( p_i \).

If not moved to infinity, the eigenvalue is made larger in magnitude and the process can be repeated so that other circuit elements are deleted and the eigenvalue is moved to infinity.

By considering the characteristic equation again it can be seen that

\[
|sI - A| = s^n + a_{n-1}s^{n-1} + \ldots + a_0
\]

where

\[
a_0 = p_1 \cdots p_n.
\]

That is, \( a_0 \) is the product of the \( n \) eigenvalues. If the derivative of \( a_0 \) is taken with respect to a circuit element \( \xi \) where \( \xi \) is an \( R, L, \) or \( C \), the result is

\[
\frac{a_0}{\xi} = \sum_{i=1}^{n} \frac{p_i \cdots p_{i-1} p_{i+1} \cdots p_n}{p_i} \frac{\partial p_i}{\partial \xi}
\]

A little algebra yields

\[
S_{\xi} = \frac{\xi}{a_0} \frac{a_0}{\xi} = \sum_{i=1}^{n} \frac{\xi}{p_i} \frac{\partial p_i}{\partial \xi} = \sum_{i=1}^{n} \frac{p_i}{s_{\xi}}.
\]

For linearized state equations the constant \( a_0 \) has been observed to have the form

\[
a_0 = \frac{(C_1 + C_2) R_1 R_2}{C_1 C_2 L_1 L_2 (R_1 + R_2)} (+ \text{ other terms})
\]

Clearly

\[
|S_{\xi}^a| \leq 1
\]

for this case.

In words, the sum of the pole sensitivities to a parameter is less than or equal to 1. For this reason in most cases of interest if the sensitivity of an eigenvalue is nearly equal to one, the sensitivity of other eigenvalues will be very small. Thus, the circuit element is deleted will only cause one eigenvalue to have an appreciable change in magnitude.

The sensitivity calculations are made in the PROSE program for the eigenvalue with the maximum absolute value. If the magnitude of the sensitivity is within three per cent of unity for a circuit parameter, the circuit parameter can be removed from the circuit. In the examples in Chapter 5 the computer output shows how the circuit elements can be removed.

3.1 Introduction

The user's manual for the computer program to PROduce State Equations is given in this section. The PROSE program can be used to derive the state equations for any electrical network that can be analyzed with the SCEP* TRE circuit analysis program and compute eigenvalue sensitivities with the following restrictions:

1. Each circuit component can be modeled by two-terminal elements which can be represented as either a voltage source, a current source, a resistor, a capacitor, or an inductor.
2. No FORTRAN Subroutines are allowed.
3. No linear, secondary, or primary dependent sources are permitted. (In place of the SCEP* TRE dependent sources, voltage or current sources can be defined by expressions which can give a dependent source).

The PROSE program can be used by following these steps:

Step 1. A circuit diagram is drawn for the circuit for which the state equations are desired. Each component in the original circuit has been represented as a collection of two terminal elements in the circuit diagram.

Step 2. The circuit diagram is translated into a listing for the SUPER*SCEPTRE circuit analysis program.*

Step 3. Input sources are requested under OUTPUTS subheading.

Step 4. After the RUN CONTROLS heading, the instruction EXECUTE SETUP PHASE ONLY is inserted.

Step 5. The control cards are added and the complete program is entered.

The PROSE program responds by outputting the state equations and a state transition matrix for a linear set of equations. In addition the PROSE program computes a set of "sensitivities" of the eigenvalues of the transition matrix that result from a change in each of the circuit elements.

* The format for the computer input can be found in one of the following:
3.2 Example

To fix ideas consider as an example obtaining the state equations for the circuit in Figure 1.

![Circuit Diagram](image)

Fig. 1. Boost Power Supply Average Model TD Interval Model

The input description for PROSE use is:

```
//JOB card for 240 k byte region
//STEP1 EXEC PROSE
//INPUT DD *
CIRCUIT DESCRIPTION
BOOST POWER SUPPLY
AVERAGE MODEL
TD INTERVAL
ELEMENTS
EG,G-1=1
RL,1-2=1
L,2-G=2
C,3-G=3
RC,4-3=4
R,4-G=5
OUTPUTS
EG
RUN CONTROLS
EXECUTE SETUP PHASE ONLY
STOP TIME=5
END
```

Fig. 2. PROSE Input Listing for Circuit of Figure 1
The PROSE system output is condensed below.

\[
DL = \frac{-IL*RL+EG}{L}
\]
\[
DC = \frac{-VC}{(R+RC)/(C)}
\]

**STATE VARIABLE NAMES**

- IL
- VC

**MATRIX A**

\[
\begin{array}{cc}
-5.00000D-01 & -5.00000D-01 \\
0.0 & -3.70370D-02
\end{array}
\]

**INDEPENDENT SOURCE NAMES**

- EG

**MATRIX B**

\[
\begin{array}{cc}
-5.00000D-01 \\
-3.70370D-02
\end{array}
\]

**EIGENVALUES OF MATRIX A**

<table>
<thead>
<tr>
<th>REAL PART</th>
<th>IMAG. PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5.000000D-01</td>
<td>0.0</td>
</tr>
<tr>
<td>-3.7037037D-02</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Fig. 3. Summarized PROSE output for input listing of figure 2.

Figure 1 indicates the circuit whose state variable equations are computed. Each element is given a name and each node is labeled with an encircled character.

Figure 2 shows the IBM OS job control language (lines 1 through 3) to execute the system. Lines 4 through 14 in Figure 2 describe the circuit in standard SCEPTRE format. The OUTPUTS request (lines 15 & 16) alerts PROSE to regard EG as an independent source in its determination of the B matrix.

The RUN CONTROLS (lines 17 through 19) include the special line:

---

EXECUTE SETUP PHASE ONLY. Both this card and the STOP TIME = number
dummy value are required. The END card finishes the PROSE input description.

The output of Figure 3 indicates the state variable equations for
dL \frac{dIL}{dt} and dC \frac{dVC}{dt} as well as the state variable names (IL and VC), the
linear transition matrix A, the independent sources (EG), the input matrix
B, and the eigenvalues for the transition matrix.

The matrices are determined from the state variable equations and the
element values in the form: \dot{X} = AX + BU, where X is the vector of state
variables and U is the vector of independent sources.

Additional outputs include the sensitivities which will be further
described later in this document.

3.3.0 PROSE Language

The PROSE system input language is the standard SCEPTRE transient
solution language. For those unfamiliar with the SCEPTRE language, the
reference of note 1 must be consulted.

3.3.1 Descriptions

PROSE will allow all standard SCEPTRE descriptions; i.e. CIRCUIT,
MODEL, and RERUN. However, the rerun description is ignored and model
description is used only if models under it are called in the circuit
description.

3.3.2 Elements

All SCEPTRE elements (E,R,L,C,J) are allowed by PROSE. Element values
may be constants, expressions or tables. PROSE will allow defined para-
meters in expressions but will not allow FORTRAN function subroutines,
equations, secondary dependent sources, linear dependent sources, and
primary dependent current sources. Voltage and current source deriva-
tives are also allowed.

Models may be called as elements as in standard SCEPTRE, but model
elements must also conform to the above rules.

Since PROSE is transient analysis oriented, no AC sources nor ele-
ments with bounds are permitted.
3.3.3 Defined Parameters

Standard SCEPTRE transient oriented real defined parameters may be utilized.

3.3.4 Outputs

At least one output request is necessary for proper execution. The OUTPUT requests of voltage or current sources implies that these sources are independent sources and are used in the PROSE computation of the B matrix.

3.3.5 Initial Conditions

Initial conditions may be supplied and sometimes are required since PROSE does not compute a DC operation point prior to matrix computations. The initial conditions are used to set the system operating point when a linear set of state equations are computed. If the initial conditions are not supplied, all initial state variable values are assumed to be zero.

3.3.6 Functions

Only Tables are allowed under functions. PROSE does not permit equations.

3.3.7 Run Controls

PROSE requires the special run control: EXECUTE SETUP PHASE ONLY for execution of the state variable equation generator.

A dummy STOP TIME is also required.

For debugging purposes, PROSE allows the run control: WRITE SIMUL8 DATA

All other run controls are illegal.

3.3.8 End

An END card is required at the end of the input to PROSE

No other heading or subheading cards may be included.

3.4 Features

3.4.1 PROSE system output

The output from PROSE begins with a FORTRAN routine. Within the routine is a listing of the state variable equations. These equations are of the form:

\[ DL \text{ or } DC = f(\text{elements, } I_L, V_c) \]

where \( DL = \frac{dI_L}{dt} \) and \( DC = \frac{dV_c}{dt} \)

where \( I_L \) and \( V_c \) are the circuit state variables. Also, if any matrices appear in the state equations, the matrix elements will be listed and their inverse elements (elements of the inverted matrix) will be specified by a trailing "I" in the state equations.

Following the subroutine output, the linear circuit sensitivity analysis results are given.
The state variable names are listed, followed by the state transition matrix \( A \). These are followed by the independent source names (if any) and the input matrix \( B \) (if any independent sources exist).

The eigenvalues of matrix \( A \) are then displayed followed by the sensitivity analysis. This analysis is computed by increasing each element value (except for independent sources) by 1% separately and recomputing the transition matrix eigenvalues.

PROSE lists the sensitivities of each element to the maximum eigenvalue and also gives an indication of elements which may be removed (either short or open circuited) to eliminate a state variable from the system.

3.4.2 Mechanical Systems

The standard SUPER*SCEPTRE mechanical description systems\(^2\) may be used by PROSE giving state variable equations for these systems. Combinations of circuit and mechanical descriptions may also be analyzed.

3.5 Limitations

3.5.1 No equations may be specified
3.5.2 No FORTRAN functions
3.5.3 No linear or secondary or primary dependent sources

3.6 Error Messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RUN INITIAL CONDITIONS ONLY specified</td>
</tr>
<tr>
<td>2</td>
<td>VECTOR EQUATIONS specified</td>
</tr>
<tr>
<td>3</td>
<td>RUN IC VIA IMPLICIT specified</td>
</tr>
<tr>
<td>4</td>
<td>RUN AC specified</td>
</tr>
<tr>
<td>5</td>
<td>( EY ) type elements exist (Linear dependent source)</td>
</tr>
<tr>
<td>6</td>
<td>( JX ) type elements exist (Linear dependent source)</td>
</tr>
<tr>
<td>7</td>
<td>( J8 ) type elements exist (Primary dependent source)</td>
</tr>
<tr>
<td>8</td>
<td>( J0 ) type elements exist (Secondary dependent source)</td>
</tr>
<tr>
<td>9</td>
<td>LOWER LIMIT TABLE specified</td>
</tr>
<tr>
<td>10</td>
<td>RUN MONTE CARLO specified</td>
</tr>
<tr>
<td>11</td>
<td>RUN OPTIMIZATION specified</td>
</tr>
<tr>
<td>12</td>
<td>RUN SENSITIVITY specified</td>
</tr>
<tr>
<td>13</td>
<td>RUN WORST CASE specified</td>
</tr>
<tr>
<td>14</td>
<td>EXECUTE SETUP PHASE ONLY not specified</td>
</tr>
<tr>
<td>15</td>
<td>SCEPTRE1 DTASAV dataset empty(^3)</td>
</tr>
<tr>
<td>16</td>
<td>SCEPTRE1 PGMSAV dataset empty(^3)</td>
</tr>
<tr>
<td>17</td>
<td>SCEPTRE1 PGMSAV dataset incorrect(^3)</td>
</tr>
<tr>
<td>18</td>
<td>No state variables exist</td>
</tr>
</tbody>
</table>

\(^2\) See User's Manual for SUPER*SCEPTRE, AD-A011-348
22 Pushdown stack overflow<sup>4,5</sup>
23 Invalid symbol encountered<sup>5</sup>
24 Invalid use of XTABLE format
25 Symbol divided incorrectly in FMMSAV<sup>5</sup>
26 More than 300 elements, tables, expressions, and defined parameters
27 Invalid symbol pushed into stack<sup>5</sup>
28 Invalid expression symbol pushed into stack<sup>5</sup>
29 Pushdown stack overflow<sup>4,5</sup> (same as 22)
30 A TABLE appeared incorrectly in an expression
31 Invalid use of subscript zero<sup>5</sup>
32 A state variable cannot be located<sup>5</sup>
33 Stack KSHUV overflow - possible computational delay
34 Passive element is variable - element set to zero
35 Matrix is too large for display

Probable SCEPTRE syntax error
Possibly a computational delay exists
Please notify author; those are system errors.
All errors without note 5 are user errors.
Note: References refer to AD catalog numbers at:
National Technical Information Services
U. S. Department of Commerce
Springfield, VA 22161

4.0 PROSE Principles of Operation

4.1 Introduction

The PROSE system operates in conjunction with the SUPER*SCEPTRE package<sup>5</sup>. The system utilizes a standard unmodified SUPER*SCEPTRE system only as an equation generator, i.e., the execution phase (simulation) of SUPER*SCEPTRE is not required. The PROSE system is a "postprocessor" to the equation generator. Thus, the total PROSE system operates in six sequential job steps.

4.2 Steps of the PROSE system

The first two steps of the operation of PROSE are the standard SUPER and SCEPT1 steps of a SUPER*SCEPTRE job. The SUPER step performs any processing required on a standard SUPER*SCEPTRE input stream<sup>6</sup> (i.e., mechanical, digital, transfer function, etc. to produce actual SCEPTRE

<sup>6</sup>See previous reference AD-A011-348
coding. The SCEPTI step is also standard SCEPTRE input processing and equation generation. The state variable equations generated are placed in a FORTRAN subroutine (SIMUL8) and written on dataset PGMSAV. Also, data required for execution of the equations is placed on dataset DTASAV by SCEPT1.

With this setup stage performed, the PROSE system begins processing on the PGMSAV and DTASAV files. This third step reads the DTASAV file and verifies its legality according to the criteria setup in the PROSE manual.

After verification, the system searches the PGMSAV file for the state variable equations. The equations are pushed out in an equation stack (NPUSH) with the variable name (left hand side of the equation) and a pointer to its equation in NPUSH placed in array KLOCA. All expressions and state equations are thus saved in memory. During this process, the element names are also saved in array NSYM.

The system next begins writing its FORTRAN routine. The routine is one which can compute the A and B matrices of the linearized, state variable form:

\[ \dot{X} = AX + BU \]

where \( X \) is the vector of state variables and \( U \) is the vector of input sources.

Prior to the state equation generation, the system outputs the elements of any R, G, L, or S matrices which exist in the circuit.

The state equations are now generated by the following process. The symbol table (NSYM) is searched sequentially for state variables, and the equation for the derivative of that variable is initiated. The process involves location of the equation in the stack (NPUSH), popping the equation until a variable name is located, pushing the current equation pointer (onto KSHUV), and then recursively generating the equation for the current variable. At each location of a variable in an equation, either a set of parentheses are forced around the new equation (if the new equation has any sum terms in it), or not (if no sum terms exist); or, possibly, the variable is constant (element value) or a state variable itself, in which case the variable name is entered in the equation. This process may form awkwardly long state equations but decreases as much as practicable the number of parentheses.
and forces only constant valued variables to appear in the equations. Each state variable equation is thus generated by this same procedure.

The routine has now become a self-contained representation of the state equations of the input circuit.

The next two steps of PROSE perform the compilation of the SIMUL8 subroutine and linkage edit the SIMUL8 object with the object (load module in actuality) of the simulation portion of the PROSE system.

The simulation (GO step) reads the DTASAV file and begins the generation of the A and B matrices by alternately setting one and only one state variable to 1, all others are set to zero. Of course, all constant element values are set to their nominal values as specified in the input description. This A and B matrix generation is performed by subroutine EIGAN which also calls the IBM Scientific Subroutines HSBG and ATEIG to compute the eigenvalues of the real A matrix. Finally, the simulation performs a sensitivity analysis by increasing each constant non-input element value, respectively, by 1% and recomputing new A and B matrices and eigenvalues of the new A matrix.
5. Examples

Three examples are given which illustrate the use of the PROSE program in obtaining state equations for electrical networks. For each example the SCEPTRE listing is given together with the circuit diagram. The state equations are printed as the output. Notice that

$$\frac{dLX}{dt}$$

represents \( dI \)

and

$$\frac{dVCX}{dt}$$

represents \( dc \).

5.1 RLC Circuit

A simple RLC circuit is used for the first example. The circuit diagram is shown together with the SCEPTRE input in Figure 5.1a. The state equations produced as the output of the PROSE program is also listed in Figure 5.1b.

In Figure 5.2 the A and B matrices are given. Next the eigenvalues are calculated together with the sensitivities of the maximum eigenvalue of one per cent change in each parameter. If the sensitivity of the circuit element is within 3\% of unity, the circuit element can be removed from the network. How the circuit element can be removed (by setting it equal to zero or by setting it equal to infinity) is indicated in several of the examples that follow.
CIRCUIT DESCRIPTION
EXAMPLE:
E, G-1=1
R, 1-2=1
L, 2-3=1
C, 3-G=1
RL, 2-6=1

OUTPUTS
E
RUN: CONTAGL
STOP TIME=2
EXECUTE SETUP PHASE ONLY
END

END OF PRE-PROCESSOR

Figure 5.1a. RLC Circuit Example.
Figure 5.1b. State Equations for Figure 5.1a.
EXAMPLE:

STATE VARIABLE NAMES
IL
VC

MATRIX A

\[-1.000000000 \quad -1.000000000 \\
1.000000000 \quad -1.000000000\]

INDEPENDENT SOURCES
E

MATRIX B

\[0.000000000 \\
1.000000000 \\
0.000000000 \]

EIGENVALUES OF MATRICES A

<table>
<thead>
<tr>
<th>REAL PART</th>
<th>IMAG. PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.000000000</td>
<td>1.000000000</td>
</tr>
<tr>
<td>1.000000000</td>
<td>-1.000000000</td>
</tr>
</tbody>
</table>

SENSITIVITY ANALYSIS

MAXIMUM EIGENVALUE #2 MAGNITUDE = \[1.4142136D+00\]

ELEMENT SENSITIVITY

\[-C \quad C.496 \\
R \quad 0.250 \\
RL \quad -0.248 \\
L \quad -C.496\]

NO SENSITIVITIES WITHIN 3% OF 1.
C MAY BE REMOVED (I.E. SET TO ZERO IF SENSITIVITY < ZERO
SET TO INFINITY IF SENSITIVITY > ZERO.)

Figure 5.2. Matrices, Eigenvalues and Sensitivities for RLC Circuit.
5.2 Mutual Inductance

A simple transformer system is shown in Figure 5.3 together with the SUPER*SCEPTRE input. The state equations are given in Figure 5.4. Notice the generation of the inductor matrix equations, XML, and the references to the inverse elements of this matrix, XMLI. In Figure 5.5 the A matrix, B matrix, eigenvalues and sensitivities are listed.
CIRCUIT DESCRIPTION

MUTUAL INDUCTANCE TEST FOR PROSE

ELEMENTS
E, 1/2=1
F1,2=3=1
L1,3=1
L2,4=5=1
M, L1,L2=0.55
F2,4=5=1

CUTPLYS
E,V1,VL2,IL1,IL2,IR2

RUN CONTROLS
STOP TIME=3
EXECUTE SETUP PHASE CALY
END

END OF FPE-FRCESSOR

---

Figure 5.3. Mutual Inductance Example.
Figuring 5.4. State Equations for Mutual Inductance Example.
Figure 5.5. Matrices, Eigenvalues and Sensitivities for Mutual Inductance Example.
5.3 Integral Pulse Frequency Modulation Regulator

The circuit model for an integral pulse frequency modulation regulator* is given in Figure 5.5. The state equations follow in Figure 5.7. The A matrix is shown in Figure 5.8 with the eigenvalues for the A matrix. Also shown is the variation in eigenvalues that accompanies a one per cent change in each parameter.

* R. P. Iwens, Y. Yu, and J. Triner, "Time Domain Modeling and Stability Analysis of an Integral Pulse Frequency Modulated DC to DC Power Converter", PESC '75 Record
Figure 5.6. Integral Pulse Frequency Modulation Regulator Model.
Figure 5.7. State Equations for Figure 5.6.
Figure 5.8: Matrices, Eigenvalues and Sensitivities for Figure 5.6.
6. Summary and Conclusions

The work during the two years of the NASA Grant NSG-3096 has resulted in a computer program which will convert a SCEPTRE listing for an electrical or mechanical system into a set of state equations for the network. Additional features of the program to PROduce State Equations include subroutines to:

1. linearize the state equations about an operating point as specified by the initial conditions
2. obtain the eigenvalues for the linearized state equations
3. obtain the sensitivities of the eigenvalue with the maximum magnitude
4. use the sensitivity calculations to indicate which circuit element can be removed from the network without adversely affecting the systems dynamic operation

At the end of the first year the PROSE program had limitations on the type of networks it would accept. In the second year several of these limitations were removed at a great cost in programming manpower. The most significant improvement is the ability PROSE now has to accept networks with mutual inductance.

Additional research and development could profitably be spent in applying the PROSE program to the design of automatic control systems. Using SUPER*SCEPTRE an electromechanical description of the plant to be controlled would be reduced to the state equations for the systems. The state equations could be used in an algorithm to automatically compute the control matrix.
Appendix

This appendix gives the Fortran listing for the computer program that PROduces State Equations.
TRICENTENNIAL VERSION OF P-R-O-S-E

** IMPLICIT REAL*P(A-H,O-Z) **
VARIABLE DEFINITIONS - NASA

NSYM = ELEMENTS SYMBOL STORAGE ARRAY

KSYM = NUMBER OF SYMBOLS IN NSYM

NRAY = INPUT AND OUTPUT 80 BYTE ARRAY

KRAY = CURRENT LOCATION POINTER IN NRAY

NLEN = ARRAY OF LENGTHS OF EACH TYPE OF ELEMENT

NPT = POINTERS TO ELEMENT TYPES IN KSYM

NT = TYPE OF ELEMENT VALUE (=0 IS NOT EJ TYPE)

COMMON /NASA/ KSYM(6,300),KSYM,NRAY(80),KRAY,ALEN(8),NPT(8),NT(300)

VARIABLE DEFINITIONS - PUSHED

KLOCA = ARRAY OF LEFT HAND SIDE ENCODED NAMES AND

LOCATIONS OF EQUATION IN NPUSH AND

SUP, TFRMS FLAGS

KLOC = LENGTH OF KLOCA

NPUSH = EQUATION ARRAY

KPUSH = LENGTH OF EQUATION ARRAY NPUSH

KPMAX = MAXIMUM LENGTH OF NPUSH(1000)

COMMON /PLSHED/ KLOCA(3,400),KLOC,NPUSH(1000),KPUSH,KPMAX

VARIABLE DEFINITIONS - SHGR

NSHUV = PUSH DCLA STACK (FIFO TYPE)

KSHUV = CURRENT POINTER INTO NPUSH STACK

KST = START OF CURRENT EQUATION IN NPUSH

KEND = END OF CURRENT EQUATION IN NPUSH

KSUB = CURRENT VARIABLE'S SUBSCRIPT VALUE

KEXTD = CURRENT VARIABLE'S EXTENDED VALUE

KFAR = PARENTHESES FLAG

COMMON /SHGR/ NSHUV(500),KSHUV,KST,KEND,KSUB,KEXTD,KFAR

VARIABLE DEFINITIONS - CNTRL

LIST = CEBUG FLAG

L = RESERVEC WORD

CCTRL = CONTROLS FROM DTASAV

KXM = XM, XMRG, AND XMS ARRAY SIZES

COMMON /CNTRL/ LIST,L,CCTRL(160),KXM(3)
C VARIABLE DEFINITIONS - FILES
C NPRINT - PRINT FILE NUMBER
C NPGM - SCEPTRE PROGRAM FILE NUMBER
C NDATA - SCEPTRE DATA FILE NUMBER
C
C COMMON /FILES/ NPRINT, NPGM, NDATA, NEQU

C
C VARIABLE DEFINITIONS - CHAR
C NSP - SPACE
C NLP - LEFT PARENTHESIS
C NRP - RIGHT PARENTHESIS
C NCOM - COMMA
C NEQ - EQUAL SIGN
C ND - ALPHA LETTER "D"
C NPLUS - PLUS SIGN
C NMINUS - MINUS SIGN
C NSL - SLASH
C NZER - ZERO NUMERIC
C NXM(3) - ALPHABET "L", "R", "S"
C NUM(10) - NUMERICS
C
C COMMON /CHARS/ NSP, NLP, NRP, NCOM, NEQ, ND, NPLUS, NMINUS, NSL, NZER, NXM(3), NUM(10)

C
C VARIABLE DEFINITIONS - HOLDIT
C HEAD - HEADING CARD BUFFER
C KHEAD - NUMBER HEADING CARDS
C NCLM3 - RESERVED WORD 3
C ELTS - ELEMENT VALUES
C KELTS - NUMBER OF ELEMENTS
C KEL - NUMBER ELEMENTS- NUMBER EJ ELEMENTS
C SRC - SOURCE DERIVATIVE VALUES
C KSDR - NUMBER SOURCE DERIVATIVES
C NCUM3 - RESERVED WORD 3
C XIC - INITIAL STATE VARIABLE VALUES
C KSV - NUMBER STATE VARIABLES
C NCUM4 - RESERVED WORD 4
C PAK - DEFINED PARAMETER VALUES
C NPAR - DEFINED PARAMETER NAMES
C KDP - NUMBER DEFINED PARAMETERS
C KEUM3 - RESERVED WORD 3
C XTAB - TABULAR FUNCTION VALUES
C NTAB - TABLE NAMES
C KTAB - NUMBER ENTRIES IN XTAB
C RTAB2 - NUMBER OF TABLES
COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145

COMMON /MATERIAL/, HEAD(9,11), KHEAD, NDUM1, ELTS(300), KELTS, KEL, SDR(50)
1, KSDR(6,50), KSQR, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
PRI 0148

DATA KEJ - NUMBER OF INPUT (INDEPENDENT) SOURCES
PRI 0145
BEGIN MAIN ROUTINE

IMPLICIT REAL*8(A-H,Q-Z)
COMMON /NASA/ KSYM(6,300),KSYM,NRAY(80),KRAY,KLEN(8),NPT(8),NT(300)
COMMON /CNTRL/ LIST,LCTROL(160),KXM(3)
COMMON /FILES/ NPRINT,NPGN,NOTA,NEQU

GET SYMBOLS
CALL SETUP

GENERATE STATE EQUATIONS HEADER
CALL NTCUP

BUILD EQUATIONS
CALL BILD

FIN
CALL FINIE

STOP
END
SLBROLTINE BACK (NAME,NCHAR) 

CHANGE NAME TO NCHAR IN MECHANICAL FORMAT 

CHANGE 

GP=TP 

I=F EXCEPT ICW=TJ 

V=V EXCEPT %k = k.J 

ELEMENTS 

CU=K EU=U. JU=R LU=K RU=D 

CH=J EW=G. Jh=C 

DIMENSION NEL(5), NELF(4), NSTF(4), NAME(6) 

EQUIVALENCE (NG,NELF(7)), (NG,NELF(5)), (NG,NSTF(3)) 

EQUIVALENCE (NU,NELF(2)), (NG,NELF(8)), (NG,NSTF(4)) 

DATA NEL/IHC,IHE,IHJ,IHL,IHR/ 

DATA NST/IHF,IHT,IHW/ 

DATA NFL/IHC,IHE,IHJ,IHL,IHK,IHR/IHG,LHC/ 

DATA iST/IHI, I-V/ 

DATA NST/IHF,IHT,IHW/ 

KST=N+1 

KN=1 

IF(NCHAR(I).NE.NBL) KN=KN+1 

IF(NCHAR(I).NE.NG) RETURN 

IF(NCHAR(1).NE.NG.AND.NCHAR(2).NE.NP) GO TO 3 

NCHAR(1)=NT 

CONTINUE 

IF(NCHAR(1).NE.ND) GO TO 6 

NCHAR(1)=NT 

KST=N+1 

IF(NCHAR(KST).NE.NST(K1)) GO TO 7 

NCHAR(KST)=NSTF(K1) 

CONTINUE 

5 IF(NCHAR(1).NE.ND) GO TO 6 

NCHAR(1)=NT 

KST=N+1 

GO CONTINUE 

-40-
KEL=5
KELF=J
IF(NCHAR(KLCC).EQ.NU) GO TO 9
KEL=3
KELF=J
9
GO TO 1, KEL

IF(NCHAR(KST).NE.NEL(K1)) GO TO 10
NCAR(KST)=NELF(K1+KELF)
GO TO 11
CONTINUE
RETURN

IF(KLCC.EQ.KN) GO TO 13
KELF=KN7KLOC
GO 12 K1=1, KELF

NCHAR(KLOC+K1-1)=NCHAR(KLOC+K1)
12
NCHAR(KN)=NEL
GO TO 2
END
SUBROUTINE BCDOUT (KVAL)

PUT THE BCD CHARS FOR KVAL INTO OUTPUT STRING

VAL = KVAL
NLEN = INT(ALGIO(MAXO(MVAL,1))+1)
MOIV = I**(NLEN-1)
KFL = 0
DC 1 I = 1, NLEN
MPTR = MVAL/MDIV+1
IF (MPTR EQ 1 AND KFL EQ 0) GO TO 1
CALL PUTCHR (KNUMS(MPTR))
KFL = 1
NVAL = MVAL - (MPTR-1)*MDIV
MCIV = MCIV/10
RETURN
END
SUBROUTINE BILD

BUILD THE STATE VARIABLE EQUATIONS

IMPLICIT REAL*8(A-H,O-Z)

COMMON /NASA/ NSYM(6,3C),KSYM,NRAY(80),KRAY,ALEN(8),NRT(300),KMAX(10)

COMMON /PL-HED/ KLOCA(3,400),KLOC,NPUSH(1000),KPMAX

COMMON /CHAPS/ NSP,NLP,NRP,NPT,NEQ,D,NPLUS,NMINUS,NSLNZER,NXM(31),NUM(10)

COMMON /CNTRL/ LIST,LI,CTRL(160),KXM(3)

COMMON /FILES/ NPRINT,NPGM,NDTA,NEQU

DATA /RAY/0,1,2,3

IF(KFL.EQ.0) WRITE (NEQU,2)

CALL PUTNM (LXM)

CALL UCDOLT

CALL PUTCHR (NSPN)

DO 1 I=1,72

IF(KXM(I).EQ.0) GO TO 4

WRITE (NEQU,2) JXM(I),KXM(I),JXM(I)

CONTINUE

DO 6 I=1,3

IF(KXM(I).EQ.0) GO TO 6

JNX=KXM(I)

LXM(I)=JNX

DO 5 J=JNX

NB=I+(J-1)*JNX

NAX=(NB-1)*JNX

IF(NAX.GT.NAX) KVAL=1000000*(A-NAX)+NAX

CALL PUTNM (LXM)

CALL UCDOLT

CALL UCDOU (NA)

CALL PUTCN (NCX)

CALL PUTCHR (NRP)

CALL PUTCHR (NSC)
CALL POP (KVAL)
CALL PUTLIN
CONTINUE
WRITE (NEQU,7)
FORMAT (1HC,79X,/,1HC,79X)
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATION,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
DC 11 KSVPT=3,4
KSV=LEN(KSVPT)
IF(KSV.EQ.0) GO TO 11
DO 10 JSV=1,KSV
STOP
IF(KFL.EQ.1) CONTINUE
WRITE (NEQU,8)
FORMAT (3HC STATE VARIABLE EQUATIONS,50X,/,IHC,79X)
SUBROUTINE CECNAM (NAME, KSUB, KLOC, KVAL)

DECODE (NAME) INTO SUBSCRIPT LOCATION AND VALUE

IMPLICIT REAL*8(A-H,C-Z)

COMMON /NASA/ NSYM(NSYM), KSYM, NrAY(NRAT), KRAY, NLEN(NPT), NT(300),
       1)
COMMON /CHARS/ NSP, NLP, NRP, NCP, NSEQ, NQ, NPLUS, NMNUS, NSL, NZER, NX(13)

COMMON /CNTRL/ LIST, LI, CTROL(160), KXM(3)
COMMON /FILES/ NPRINT, NPGM, NDATA, NSEQ

DIMENSION NAME(6), NELTS(7), NTYPE(6)
DIMENSION NXTAB(5)

DATA NELTS/1HC, 1HR, 1HL, 1HE, 1HM, 1BP/
DATA NTYPE/IHD, IHV, 1HII, 1HF, 1HM/
DATA N/1HN/
DATA M/1MN/,
DATA N/1HT/, NXTAB(1), NELTS(7), NCHART, NXTAB(1)

EQUIVALENCE (NI, NTYPE(3)), (NP, NELTS(6))
EQUIVALENCE (NXTAB(1), NELTS(7)), (NCHART, NXTAB(1))

KLOC=0
KELTS=0
KTYPE=0
KEXTD=0

LOCATE SYMBOL TYPE
DC 1 I=1, 6
IF(NAME(I) .EQ. NTYPE(I)) GC TO 7
CONTINUE
KST=1

LOCATE SYMBOL ELEMENT
DO 3 I=1, 7
IF(NAME(KST) .EQ. NELTS(I)) GO TO 8
CONTINUE

MULTI TYPE?
IF(NAME(KST) .NE. NME) GO TO 4

CONTINUE
KELTS=9
GO TO 9

VIRTUAL/CURRENT TYPE
IF(KTYPE .EQ. 0) CR.KTYPE .EQ. 0.3) GO TO 12

FORM SYMBOL VALUE
KVAL=KLOC+1000*KELTS+10000*KTYPE+100000*KEXTD

IF(LIST .EQ. 1) WRITE (NPRINT, 6) NAME, KLOC, KELTS, KTYPE, KEXTD

FORMAT (6H DECNAM=, 6H KLCC=, 13.7H KELTS=, 13.7H KTYPE=, 13.7H KEXTD)
1XTD-1, 13)
RETURN
C GOTTA TYPE NAME
C EXPRESSION IS SPECIAL
7 IF(I.EQ.6) GO TO 17
C NOT AN "X"-TYPE
CC35 KST=2
CC36 KTYPE=I
CC37 GO TO 2
C GOTTA ELEMENT
8 KELTS=I
C PARAMETER IS SPECIAL
CC39 IF(KELTS.EQ.7) GO TO 5
C LOCATE NAME IN SYMBOL TABLE
CC40 C I=1,KSYM
CC41 DO 10 J=1,5
10 KSTJM=KST+J-1
CC42 IF(NAME(KSTJM).NE.NSYM(J,I)) GO TO 11
CC43 CONTINUE
C FOUND NAME IN STABLE
CC44 KLOC=I
CC45 GC TO 5
CC46 CONTINUE
C NOT IN STABLE OR V/I TYPE
CC47 KSTAR=2
C V#V/I TYPE
CC49 IF(LPUMP(NAME,KSTAR,I,KSTYL).EQ.11) GO TO 5
C V#V/I TYPE - STATE VARIABLE DERIV.
CC50 IF(NAME(KSTAR).EQ.ND) GO TO 15
CC51 IF(NAME(KSTAR).EQ.NP) GO TO 16
C V#P/I#P TYPE
CC52 IF(NAME(KSTAR).EQ.NP) GO TO 16
C FORM LOCATE CF SYMBOL
CC53 KLOC=NPT(KSTYL)+KSUB-1
CC54 IF(KSUB.EQ.100000) KLOC=KLOC+1
CC55 IF(KPL.GE.100000) KPL=KPL-100000
C X-ELEMENT TYPE
CC56 DC 14 KELTS=I,5
CC57 IF(NSYM(I,KPL).EQ.NELTS(KELTS)) GO TO 5
CC58 CONTINUE
CC59 KELTS=0
CC60 GC TO 5
C STATE VARIABLES
CC61 KTYPE=1
CC62 GO TO 13
C V2P, V3P, XIIP, OR XI5P
CC63 K=KTL=KSTYL+3
CC64 KTYPE=0
CC65 KLOC=KSUB
GO TO 5
C "X"-TYPE
C KSTAR=2
C "XM"-TYPE
C IF(NAME(KSTAR).EQ.NAME) GO TO 18
C "XI"-TYPE
C IF(NAME(KSTAR).NE.NI) GO TO 21
C KSTAR=3
C "XI"-TYPE
C IF(NAME(KSTAR).EQ.NAME) GO TO 5
C GO TO 16
C KLOC=KSUB
C "XM"-ELEMENT TYPE
C GO TO 5
C IF(NAME(3).EQ.NAME) KEXTD=1,3
C CONTINUE
C KEXTD=0
C CC TO 5
C "XM1"-TYPE
C IF(NAME(4).EQ.NAME) 'KTYPE=6
C GO TO 5
C "X" TYPE, KCN SPECIAL - "XTABLE"
C IF(JCATE(NAME,NXTAB,2,5).EQ.0) GO TO 22
C "XT" TYPE?
C IF(NAME(2).EQ.NAME) GO TO 23
C "XW#" TYPE
C KSTAR=3
C KPUMP=LPUMP(NAME,KSTAR,2,KSTYL)
C KSTYL=KSTYL/10
C IF(KPUMP.EQ.0.AND.KSTYL.NE.0) GO TO 13
C KEXTD=7
C KST=1
C GO TO 9
C "XTABLE"
C KELTS=0
C "X" TYPE=0
C GO TO 5
C ERRORS
C TABLE APPEARED IN XPRESSION ILLEGALLY
C CALL ERROR (30)
END

ERRORS
SUBROUTINE ERROR (NUM)

C PUT ERROR NUMBER

C

CMMCN /FILES/ NPRINT, NPGM, NOTA, NEGU

DIMENSION NBAD(34)

DATA NPACL/18/

DATA NBAD(1,4,5,6,7,8,17,18,19,20,23,24,27,28,30,31,32,34/)

WRITE (NPRINT,!) NUM

FORMAT (/!, 14H ERROR NUMBER f12.20H SEE ERROR LIST......./)

IF (NUM.EQ.0, NBAD(I)) GO TO 4

WRITE (NPRINT,3) NUM

3 FORMAT (38H STATE VARIABLE PRODUCTION SUPPRESSED.)

STOP 8

END
SUBROUTINE FINIE

FINISH UP THE STATE EQUATIONS PRODUCTION.

COMMON /FILES/ NPRINT,NPGM,NDTA,NEQU

WRITE (NEQU,1)

FORMAT (1HC,79X/,1HC,79X/,1HC,35(2HC),1HC,8X/,1HC,79X/,9H
  END,71X)

RETURN

END
SUBROUTINE GETEQS

C GET EQUATIONS FROM SIMTR INTO (NPUSH) POINTED TO BY
(KLOC1) AND KLOC

C
C implicit real(a-h,o-z)

COMMON /PLASHED/ KLCCA(3,400),KLOC,NPUSH(1000),KPMAX
COMMON /NASA/ NSYM(6,300),KSYM,NRAY(80),KRAY,NLET(8),NPT(8),NT(300)
1)
COMMON /CATRL/ LIST,1,CTRL(160),KXM(3)
COMMON /FILES/ NPRINT,NPGP,NDTA,NEQU
COMMON /CHARS/ NSP,NLP,NRP,NCOM,NEO,ND,NPLUS,NMINUS,NZER,NXM(3)
1),NUM(10)

DIMENSION NAME16),NAUGV(6),ICBXS(6)

C
KSLBSV=0
C
C GET NEXT CARD

1 NEOF=READ(NPGM,0)

2 IF(NRAY(1).EQ.NC) GO TO 1

C CHECK FOR "LO"

GO TO 3 KEO=6,50

3 IF(NRAY(KEO).EQ.NEO) GO TO 5

CC CONTINUE

4 IF(LOCATE(NRAY,NARGM,13,6).EQ.C) GO TO 14

5 IF(LOCATE(NRAY,NIONBX,13,6).EQ.0) 
C TEST "FADC"

C IF(NRAY(7).EQ.NP.OR.(NRAY(7).EQ.NX.AND.NRAY(8).EQ.NI))
GO TO 7

C IGNORE "DO" LOOPS

IF(NRAY(7).EQ.NC.AND.NRAY().EQ.NO)
GO TO 10

KRAY=7

C WRITE EQUATIONS

IF(NPRINT.5) WRITE (NPRINT,6) NRAY

C FORMAT (1X,63A1)

GO ON PARAMETER OR EXPRESSION

IF(NRAY(7).EQ.NPX.OR.(NRAY(7).EQ.NX.AND.NRAY(8).EQ.NI.AND.NRAY(8).EQ.NI))
GO TO 7

C IGNORE "DO" LOOPS

IF(NRAY(7).EQ.NC.AND.NRAY(8).EQ.NO) GO TO 11

C
FORTRAN IV GI RELEASE 2.0
GETEQS
DATE = 78178 14/44/45

0031 KSUB=0
0032 KEQI=KEO+1
0033 C IGNORE ZERO VALUES
0034 IF(NRAY(KEQI).*O.NZER) GO TO 1
0035 C GET SYMBOL
0036 IF(PLUS(ARAY, KRAY, NAME, O,KSUB).EQ.-1) GO TO 12
0037 C PUSH EQUATION
0038 KRAY=KEC+1
0039 IF(KSLBSV.EQ.O.AND.KSUB.EQ.1000000) KSUB=KSUBSV
0040 C CHECK FOR NOTHING PUSHED
0041 IF(KPSHSV.EQ.KPUSH) KLOC=KLOC-1
0042 C GIT NEXT STUFF
0043 GO TO 2
0044 C PARM. OR XPRESS. - SAVE NAME IN SYMBOL TABLE
0045 KSYM=KSYM+1
0046 IF(KSYM.GT.300) GO TO 13
0047 DO 8 I=1,6
0048 NSYM(I, KSYM)=NSP
0049 8 CONTINUE
0050 KRAY=KFO+1
0051 CALL PUSH (NSYM(1, KSYM), O)
0052 IF(KPSHSV.EQ.KPUSH) KLOC=KLOC-1
0053 C DETERMINE IF ONE PASS DO LOOP
0054 KSUBSV=NUMB(NRAY, 18, 20, I1)
0055 KSYM=NUMB(NRAY, 24, 26, I1)
0056 IF(KSYM.EQ.KSUBSV.OR.KSYM.EQ.1) KSUBSV=0
0057 GO TO 1
0058 C ERRORS
0059 C BAD SYMBOL
0060 CALL ERROR (25)
0061 C SYM. TABLE OVERFLD
0062 CALL ERROR (26)
0063 C PARSE OUT XML, XM, AND XMS LENGTHS
0064 IF(NRAY(31).EQ.NCGR) GO TO 4
0065 IF(NRAY(22).EQ.NXMI) GO TO 16

-51-
15 CONTINUE
GO TO 4
16 NEN=30
NST=28
0070 KXM(I)=NUMB(NRAY,NST,NEN,II)
IF(KXM(I).GT.7) CALL ERROR (35)
GO TO 4
0073 IF(NRAY(2E).NE.NC(W)
DO 19 I=1,3
19 CONTINUE
GO TO 4
CONTINUE
GO TO 4
20 NEN=27
NST=25
GO TO 17
END
SUBROUTINE MIDUP

WRITE DEBUG, MATRICES, AND HEADER

COMMON /HOLOIT/ HEAD(9,11), KHEAD, NUM1, ELTS(300), KELTS, KEL, SOR(5)
1, NSOR(6,10), KSEP, NDUM3, XIC(50), KSV, NDUM4, PAR(100), NPAR(6,100), KDP
2, KSYM, XTAE(500), NTAB(15,50), KTAB, KTAB2, KEJ

COMMON /CHARS/ NSP, NLP, NRP, NCNP, NQS, NDUM, KPLUS, KMINUS, NSL, NZER, NXN
1, NUM(10)

COMMON /FILES/ NPRINT, NPRINT, NPRINT

COMMON /CNTRL/ LIST, L1, CTRL(160), KX(13)

COMMON /NASA/ NISYM(6,300), KSYM, NRAY(80), KRAY, NLEN(8), NPT(8), NT(300)

EQUIVALENCE (TINE, CTRL(1))

DATA NE/10/ NEJ/IHJ/

PLT DBLG IF(LIST.NE.1) GO TO 19

WRITE (PRINT, 1) NPT, NLEN, KSYM

WRITE (NPRINT, 3) I, (NSYM(J,I), J=I, 6)

WRITE (NPRINT, 4) I, (NSYV(I), I=1, 6)

WRITE (NPRINT, 5) I, (NSYM(J,I), J=I, 6)

WRITE (NPRINT, 6) XMS

WRITE (NPRINT, 7) TIME, KELTS, KSV, KDP, KSR, KTA

WRITE (NPRINT, 8) TIME, KELTS, KSV, KDP, KSR, KTA

WRITE (NPRINT, 9) TIME, KELTS, KSV, KDP, KSR, KTA

WRITE (NPRINT, 10) TIME, KELTS, KSV, KDP, KSR, KTA

DO 10 T=1, KELTS

IF(NSYM(I), I=1, 6)

GO TO 10

CONTINUE
C PUT DEFINED PARAMETER VALUES
0030 IF(KDP.EQ.0) GO TO 13
0031 DC 11 I=1,KDP
0032 WRITE (NPRINT,12) I,PAR(I)
0033 IF(KSDR.EQ.0) GO TO 16
0034 DC 14 I=1,KSDR
0035 DC 11 I=1,KSOR
0036 WRITE (NPRINT,15) I,SDR(I)
0037 IF(KTAB.EQ.0) GO TO 19
0038 WRITE (APRIAT,18) I,XTAB(I)
0039 IF(KTAB.EQ.0) GO TO 19
0040 DC 17 I=1,KTAB
0041 WRITE (APRIAT,18) I,XTAB(I)
0042 IF(KTAB.EQ.0) GO TO 19
0043 WRITE (NEOU,20)
0044 IF(KTAB.EQ.0) GO TO 19
0045 RETURN
0046 END
SUBROUTINE POP (KPOPO)

POP THE EQUATION POINTED TO BY VALUE (KPOPO) FROM (NPUSH)

IMPLICIT REAL*8 (A-H,O-Z)

COMMON /NASA/ NSYM(6,300),KSYM,NRAY(80),KRAY,NLEN(8),NPT(8),NT(300)

COMMON /PUSHED/ KLOCA(3,4),KLCC,NPUSH(1000),KPUSH,KMAX

COMMON /SIGR/ NShUV(500),KShuv,KST,KEND,KSUB,KEXT,KPAR

COMMON /FILES/ NPRINT,NPGNCTA,NEQU

COMMON /CHARS/ NSP,NLP,NRP,NCCP,NEQ,NCPLUS,THINUS,NSL,NZERNXM(3),NUM(0)

COMMON /CNTRL/ LIST,LI,CONTROL(160),KXM(3),DIPENSIC,

DATA (911,NDV/HDH,1HV,lHI/C,NSHUV,KLCC(3,4),NPUSH,KPCPO)

GO TO 20

IF(LSTST(ILOC,KPCPO).NE.0) GO TO 20

IF(KPAR.EQ.0.LT.0.R.fNRAY(KRAY-1).LT.0.NSL) CALL PUTCHR (NLP)

IF(KPAR.NE.Q.OR.PUSH(KST).NE.NPLUS) CALL PUTCHR (NPUSH(KST))

IF(KST.LE.KEND) GO TO 3

CONTINUE DR!

ANYTHING LEFT ON STACK?

IF(LGRAP(KPOPO).EQ.0) GO TO 17

NC = LE CCNE?
IF(KST.LE.KEJD) GO TO 3

RETURN

SYMBOL VALUE GIVEN

KPOP=KPUSH(KST)

IF(NRAY(KRAY-1).NE.NSL) GC TO 8

KPAR=

CALL PUTCHR (NLP)

CHECK FOR INVERTED ARRAY

IF(KMX.EQ.16.OR.KMX.EQ.25 OR.KMX.EQ.36) GO TO 16

FCRM CORRECT VALUES FROM SUBSCRIPT

IF(KSUB.NE.0.AND.KPOP.GT.100000) KPOP=KPOP+KSUB-1000001

KPOP=KPOP+1

KST=KST+1

PUSH THIS VALUE AND POINT TO NEXT EQUATION

IF(KRAY+6.GT.7) CALL PUTLIN

PUT "V/I" IF NECESSARY

IF(KTYP.LE.0) CALL PLTCHR (MOVII(KTYP))

PUT NAME

CALL PUTNAM (NSYM(IKPOS))

GO TO 4

IF(KPCP.EQ.0) GO TO 15

EXTENDED SYMBOL AT BAD LOCATION

IF(MOD(KPOPO,ICCOO).NE.0) GO TO 15

PUT NAME - PUT ELEMENT

IF(KPOP.LT.400) GO TO 10

FORCE EQUATION IF NOT EXTENDED SYMBOL

IF(KEVC.EQ.0) GO TO 1

GO TO 21

LOCATE THIS EQUATION SUBSCRIPT IN STACK

DC 13 IS=1,KSHUV,5

CONTINUE

SUBSCRIPT NOT IN STACK

GO TO 21

GET SUBSCRIPT
KPOP = KPOP + KSUB + KSUBN
GO TO 1
C
PUT ZERO VALUE
CALL PUTCHE (NZER)
GO TO 4
C
START ARRAY MULTIPLY LOOP
C
INVALID SUBSCRIPT ON INVERTED ARRAY
IF (KSLBN1 > 1) GO TO 9
IF (KPCPC < 10) GO TO 21
C
SAVE OLD SUBSCRIPT AND GENERATE NEW ONE
KSUBN = KSUB
KSUBN1 = 1
C
PUT INVERTED ARRAY NAME
KX = KMX / 10
KSUB = KSLBN1
CALL PUTFKE (KX, KSUBN, KSUB)
GO TO 9
C
CONTINUE MATRIX MULTIPLY
KX = KPOP / 10
C
INCMMENT SUBSCRIPT 2
IF (KSUBN1 = 0 AND KX(KMX) > 8) GO TO 18
GO TO 19
KSUBN1 = KSUBN1 + 1
KSUB = KSLBN1
KST = KSHUV(KSUBN - 1)
NSHUV(KSUBN) = KSUB
KPCPC = KPCPC
CALL PUTCHE (KX, KSUBN, KSUB)
GO TO 4
C
END OF MATRIX MULTIPLY
GO TO 4
C
SAVE SUBSCRIPT
KSUB = KSUBN
NSHUV (KSHUV - 3) = NSHUV (KSHUV - 2) + 1
KSUBN1 = 0
GO TO 5
C
RETURN
C
INVALID SUBSCRIPT
CALL ERR (31)
C INVALID NAME TO POP
CALL ERROR (32)
END
SUBROUTINE PUSH (NAME, KSUB)

C

CALL CFPNAM (NAME, KSUB, KLOC1, KVAL)

IF(KVAL.EQ.0) GO TO 4

C

KLOC2 =KLOC+1

KLOC2(KLOC) = KVAL

KLOC1(KLOC) = 1

KLOC2(KLOC) = KSHV = KPUSH

IF(KVAL.EQ.0 OR KVAL.EQ.7000 OR KVAL.EQ.8000) GO TO 2

IF(KVAL.EQ.KLOC1(KLOC)) GC TO 1

NPLSH(KPUSH) = KVAL

IF(KPUSH.GT.KPMAX) GO TO 6

GC TO 1

C

DECODE EQUATION SYMBOL

CALL CFPNAM (NAME, KSUB1, KLOC1, KVAL)

IF(KVAL.EQ.0 AND KGO.EQ.1.OR. NPLSH(KPUSH).EQ.1) KLOC2(KLOC) = 1

KGO = 1

GO TO 1

C

DECODE EQUATION SYMBOL

CALL CFPNAM (NAME, KSUB, KLOC1, KVAL)

IF(KVAL.EQ.0) GO TO 5

C CHECK TO PUSH FULL NAME

IF(KVAL.EQ.6 OR. OR. KVAL.EQ.7000 OR. OR. KVAL.EQ.8000) GO TO 2

"X=X" FORMAT - IGNCRE

GO TO 2

IF(KVAL.EQ.KLOC1(KLOC)) GC TO 1

PUSH EQUATION SYMBOL VALUE ONLY

NPLSH(KPUSH) = KVAL

KPUSH=KPUSH+1

IF(KPUSH.GT.KPMAX) GO TO 6

GC TO 1

C

PUSH FULL NAME
CC27 2 KS=1
CC26 3 NPUSH(KPUSH)=NAM1(KS)
CC24 4 KPUSH=KPUSH+1
CC37 5 KS=KS+1
CC31 6 IF(KPUSH,GT,KMAX) GO TO 6
C END OF NAME?
CC32 7 IF(NAM1(KS),EQ,NSP) GO TO 1
CC33 8 IF(KS,GT,6) GO TO 1
GO34 9 GO TO 3
C ERRORS
C
CC35 4 CALL ERROR (27)
CC36 5 CALL ERROR (28)
CC37 6 CALL ERROR (29)
CC32 7 END
SUBROUTINE PUTCHR (NCHAR)

PLACE THE CHARACTER (NCHAR) INTO THE OUTPUT ARRAY (NRA)

IF (NRA) LONGER THAN 72, PUT (NRA) ON FORT20FQ01 AS FORTAN

COMMON /NASA/ NSYM(6,300), KSYM, NRAY(80), KRAY, ALEN(8), NPT(8), NT(300)

IF (KRAY .GT. 72) CALL PUTLT1A

NRAY(KRAY) = NCHAR
KRAY = KRAY + 1
RETURN
END
SUBROUTINE PUTIT (KGO, KSTCP, NCHAR, KLEN, KNT, KPT)

C PUT THE NAMES FROM ELEMENT LIST STARTING AT TYPE (KGO)
C TC TYPES (KSTCP) IN OUTPUT, PRECEDE BY (NCHAR) IF
C KLEN IF ZERO AND SET NUMBER ElTS IN (KNT)
C AND POINTER OF FIRST NAME IN (KPT)
C
COMMON /CHARS/ NSPMLPNRPNCChNEQND NPLUSNMINUSNSLNAZER, NM(3P1)
1, NUM(10)
COMMON /NSA/ KSYM(6, 300), KSYK, NRAY(80), KRAY, NLEN(8), NPT(8), NT(300P1)
1
COMMON /HOCIT/ HEAD(9, 11), KHEAD, NDUM(E, 6, 50), KELTS, KEL, SOR(50)
1, ASDR(6, 50), KCR, NCUM3, XIC(50), KSV, NOUN, PAR(100), NPAR(6, 100), KDP
2, NDUM5, XTAB(500), XTAB(7, 50), KTAB, KTAB2, KEJ
DIMENSION MRAY(6)
C
C ZERO POINTER AND COUNT
C KPT=0
C KNT=0
C LCCP THRU TYPES,
C DO 4 IPT=KGO, KSTCP
C FIND LENGTH OF THIS TYPE
C MPT=NCHAR(IPT)
C ZERO LENGTH IS NC PUTSKIS
C IF(MPT.EQ.0) GO TO 4
C FIND THIS GUYS POINTER LESS ONE
C LPT=MPT-1
C LOOP ON THAT COUNT
C DC 3 JPT=1, MPT
C PUT THIS EXTRA CHAR IF ONE
C NNPT=1
C IF(KLEN.EQ.1) GO TO 1
C MRAY(1)=NCHAR
C KNP=1
C FORM THIS GUYS ABSOLUTE POINTER
C LJT=LPT+JPT
C BUILD FIRST POINTER
C IF(KPT.EQ.0) KPT=LJT
C PUT THIS NAME
C HS=8-KNPT
C DC 2 M=V, PS
C MRAY(M+KNPT)=NSYM(M, LJPT)
C CALL PUTNAM (MRAY)
C UPDATE COUNT
KNT = KNT + 1

CALL PUTCHR (NCCM)

CONTINUE

RETURN

END
SUBROUTINE PUTLIN

WRITE THE CHARACTER STRING (NRAY) IN FORTRAN FORMAT
AFTER BLANKING END OF LINE: CONTINUATIONS ADDED

COMMON /NASA/ NSYM(6,300'), KSYM, NRAY(80), KRAY, NLEN(8), NT(30)
1)
COMMON /FILES/ NPRINT, NPGM, NDATA, NEQU
COMMON /CHARS/ NSP, NLP, NRP, NCOMM, NEQ, ND, NPLUS, NMINUS, NLS, NZER, NXM(3)
1), NUM(10)

DIMENSION NCOMM(6)
DATA NCOMM/6,1C,1H0,1HM,1H0,1HM/
DATA NAST/1H%/.

NOTHING ON LINE NOW

IF(KRAY.EQ.7) RETURN

C IF "COMMON " OR "=" APPEARS, THEN NO CONTINUATION

IF(LOCATE(NRAY, NCOMM, 7, 6).EQ.C) GO TO 2

KND=MINO(2, KRAY)
DO 1 I=8, KND
IF(NRAY(I).EQ.NEO) GO TO 2
CONTINUE

NRAY(6)=NAST
BLANK TO END OF LINE AND PUT LINE

CC 1 CONTINUE

2 CC 3 =KRAY, 73
CC 17 3 NRAY(I)=NSP

WRITE (NEQU, 4) (NRAY(I), I=1, 72)

CC 19 4 FORMAT (72A1, 8X)

CC 20 KRAY=7
CC 21 RETURN

END
SUBROUTINE PUTMX(KMX,KSUBN,KSUBN1)

COMMON /CHARS/ ASP,NLP,NRP,NCOM,VEQ,ND,NPLUS,NMINUS,NSL,NZER,NXM(3)
DIMENSION NAMX(6)
DATA NAMX/1HX,I-M,lH ,IIILH /

NAMX(3)=NMX(KMX)

IF(KSUBN.GT.1) CALL PUTCHR (NPLUS)
CALL PUTNAP (KAX)
CALL BCDOOLT (KSUBN)
CALL PUTCHR (NCOM)
CALL BCCUT (KSUBN1)
CALL PUTCHR (NRP)
RETURN
END.
SUBROUTINE PUTNAM (NAME)

C
C PUT THE (UP TO) SIX CHARACTER LIST (NAME) IN THE OUTPUT
C STRING (NRAY). WRITE (NRAY) IF OVER 72 CHAR. LONG
C
C
COMMON /NASA/ NSYM(6,300),KSYM,NRAY(80),KRAY,ALEN(8),NPT(8),NT(300),
1)

COMMON /CHARS/ ASP,NLP,NRP,NCOM,NEQ,NP,NPLUS,NMINUS,NSL,NZER,NXM(300),
11,NM(10)

DIMENSION NAME(6), NEW(6)

C
C RE-FORM MECHANICAL NAME
C
CALL BACK,(NAMEC, :w) LEN=1

1 IF(KRAY.GT.72) CALL PUTLIN

KRAY(KRAY)=NEW(LEN) KRAY=KRAY+1

ENC OF LINE?

LEN=LEN+1.

IF(NEW(LEN).EQ.NSP) RETURN

GO TO 1

END
SUBROUTINE SETUP

GET ELEMENT NAMES AND POINTERS INTO (NSYM), (NPT), (NLEN), AND
FROM SCIENTIFIC ROUTINE "SIMUL8"

INTEGRAL REAL (A-H,D-Z)

COMMON /FILES/ NPRINT, NPG, KACTA, NEQU

COMMON /NASA/ NSYM(6,300), KSYM, NRAY(80), KRAY, NLEN(B), NPT(8), NT(300)

COMMON /CNTRL/ LIST, LTIME, S1(21), XNDPQG, S2, XNHEAD, S3(6), XRUNNO, SPR1

COMMON /CHARS/ ASP, ALP, KRP, NCQMN, NCQ, NOS, NPLUS, NMINUS, NSL, NNER, NXM(3)

DIMENSION NCA0(13), NELS(8), NLOC(8)

DIMENSION DLH(300), NMUT(6,30)

DIMENSION NLHTR(5), NSRDOR(6), NTABLE(6), AMATE(6)

DIMENSION NAXUTL(5), NNAMEI(5), NPRTIV(6), NDERIV(6)

DIMENSION NGIAPPX(5), NDUPTR(6), NNDPD(3), NDQM(6)

DIMENSION CTROL(140)

EQUIVALENCES (CTROL(1), TIME)

DATA NAC/34,45,55,71,91,93,101,102,111,121,131,141,151/ 1127

DATA NELS/94,97,98,91,93,94,95,96,97,98,100/ 1128

DATA NLOC/7,5,6,1,2,3,8/ 1129

DATA ASIMTR/1HS,1HF,1HM,1HT,1HR/ 1130

DATA ANAPPS/1HA,1HY,1HE,1HS/ 1131

DATA NDRMTR/1HR,1HM,1HT,1HR,1HS/ 1132

DATA NDERIV/1HC,1HE,1HR,1HY,1HS/ 1133

DATA NDNAPXS/1HA,1HY,1HM,1HT,1HR,1HS/ 1134

DATA ACNX/1HI,1HO,1HN,1HL,1HX/ 1135

DATA NOLTP/1HP,1HU,1HT,1HP,1HT,1HS/ 1136

DATA NAST/1HP/ 1137

DATA ANDPDS/1IG,1ID,1HP/ 1138

DATA NSRDOR/1HS,1HR,1HC,1HO,1HR,1HY/ 1139

DATA NTABLE/1HT,1HA,1HB,1HL,1HS/ 1140

DATA NAXUTL/1HX,1HM,1HU,1HT,1HL/ 1141

GET DATA

READ TILL SUBROUTINE SIMTR FOUND

Kmut=0

-67-
READ (NPCM,2,ENC=39) NRAY
IF (LOCATE(NRAY,NSPTR,9,5).EQ.1) GO TO 1
CHECK WHICH SCFPTRE IS USED
READ (NPCM,3) NRAY
FORMAT (/////), 0A01
NAC=70
IF(NRAY(6).EQ.NAST) NAC=160
READ CONTROLS
READ (NDTA,4,END=38) CTRL(I), I=1,NAC
FORMAT (I5F14.7)
IF(NRAY.EQ.160) GC TC 6
DO 1 I=71,160
CTRL(I)=C.C00
C LCCATE CERTAIN STRINGS
IF(READ(NPGH,1).EQ.1) GO TO 40
IF(LOCATE(NRAY,NSRCD,15,6).EQ.0) GO TO 20
IF(LOCATE(NRAY,ARTAL,15,6).EQ.0) GO TO 22
IF(LOCATE(NRAY,NXMTL,15,5).EQ.0) GO TO 24
IF(NRAY(6).EQ.160) GC TO 8
IF(LOCATE(NRAY,NUMES,15,5).EQ.0) GO TO 26
IF(LOCATE(NRAY,APRMTR,15,6).EQ.0) GO TO 28
IF(LOCATE(NRAY,NERIV,15,5).EQ.0) GO TO 30
IF(LOCATE(NRAY,NONRX,15,5).EQ.0) GO TO 32
IF(LOCATE(NRAY,NCUTPT,15,6).EQ.0) GO TO 9
GO TO 6
C CHECK FOR INVALID FLAGS IN CONTROLS.
DC 10 TIER=1,13
KLOC=ABD(IER)
IF(CTRL(KLOC).NE.0.000) GC TC 36
CONTINUE
C FCRM CFUG FLAG
LIST=XWRD6G+XWRSTM
IF(LIST.GT.1) LIST=1
"EXECUTE SETUP PHASE ONLY MUST BE FLAGGED"
IF(EXTX,STP,NE.1,00) GC TO 37
C FORM ELEMENT TYPE POINTERS AND LENGTHS
NPT(7)=1
DO 11 I=1,8
MPT=VLOC(I)
KLOC=VELS(MPT)
NLEN(MPT)=CTRL(KLOC)
IF(MPT.EQ.7) GC TO 11
MPTM=VLOC(I-1)
NPT(MPT)=NPT(MPT)+NLEN(MPTM)
CONTINUE
DO 12 I=1,8
IF(NLEN(I).EQ.0) NPT(I)=0
CONTINUE
CONTINUE

KHEAD=KHEAD=HEAD

IF(KHEAD.NE.O) READ (NDTA,13) HEAD

FORMAT (9AB)

KELTS=KELTS

READ (NDTA,14) (ELTS(I),I=1,KELTS)

FORMAT (SC14.7)

IF(NAC.EQ.1S) READ (NDTA,15) ((NSYM(I,J),I=1,6),J=1,KELTS)

FORMAT (72A1)

KSYM=KSYM

READ (NDTA,14) (DUM(I),I=1,KSYM)

READ (NDTA,14) (DUM(I),I=1,KSYM)

IF(KSYM.EQ.0) GO TO 41

K2=NLEN(3)

IF(K2.EQ.K2) READ (NDTA,14) (XIC(I),I=1,K2)

M=K2+1

K2=NLEN(4)+K2

IF(K2.NE.NLEN(?)) READ (NDTA,14) (XIC(I),I=1,K2)

IF(K2.NE.K2) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

KDP=XNPD

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)

IF(KDP.NE.0) READ (NDTA,15) (PAR(I),I=1,KDP)
DO 19 I=1,KELTS
IF(NPAR1,NE.C.CO) GO TO 18
CALL GECNA (NSYM1,0,IXLOC,IVAL)
NT(I)=1-IXCTST(IXLOC,IVAL)
GO TO 19
CONTINUE
RETURN
C SOURCE DERIVATIVES EXIST - GIT THEM
KRAY=22
KSDR=3
IF(LPULL(NRAY,KRAY,NSDR1,0,KSBR,EQ.-1) GO TO 7
IF(KSDR=KSBR) GO TO 21
RETURN
C TABLES EXIST - GIT THEM TOO
KRAY=22
KTA=J
KTA20
IF(LPULL(NRAY,KRAY,NTAB1,KTAB1,0,NTAB7,KTAB2).EQ.-1) GO TO 7
KTA=NTABJ,KTAB2)
KTA2=NTAB7,KTAB2+1
GO TO 23
C GET MUTUALS
KRAY=21
IF(LPULL(NRAY,KRAY,NSYM1,KMUT1,0,KSBR).EQ.-1) GO TO 7
KMUT=KMUT+1
GO TO 25
C GET ELEMENT NAMES AND NUMBERS
KRAY=21
IF(LPULL(NRAY,KRAY,NSYM1,KELTS1,0,KSBR).EQ.-1) GO TO 7
KELTS=KELTS+1
XNLETS=XNLETS
GO TO 27
C GET NUMBER AND NAMES OF DPS
KRAY=22
IF(LPULL(NRAY,KRAY,NSYM1,KPAR1,KDP1,0,KSBR).EQ.-1) GO TO 7
KDP=KDP1
IF(KDP=KDP) GO TO 25
GO TO 27
C GET NUMBER OF DEFINED PARAMETER DERIVS
DC 31 I=22,70
IF(INCONT(I) EQ.0) XNPD=XNPD+1
GO TO 6
C GET NUMBER OF ELEMENT TYPES
IF(LPULL(NRAY,KRAY,ADUM,0,KSBR).EQ.-1) GO TO 7
CONTINUE
C47    DC 34 I=2,10
C48    IF (NDLM(3).EQ.NUM(I)) GO TO 35
C49    CONTINUE
C50    I=2
C51    TRY(NELS(I-1))=K50
C52    GO TO 33
C      ERRORS
C      INVALID FLAG EQRCRS
C53    CALL ERROR (16)
C      EXECUTE SETUP PHASE ONLY NOT SPECIFIED
C54    CALL ERROR (14)
C      NO CONTROLS(NO DTASAV)
C55    CALL ERROR (17)
C      NO PROGRAM SIMTR(NO PGMSAV)
C56    CALL ERROR (18)
C      NO TRANSIENT EQUATIONS IN SIMTR
C57    CALL ERROR (19)
C      NO STATE VARS
C58    CALL ERROR (20)
C59    END
FUNCTION LGRAB (KPOP)

C PULL THE PAST LOCATORS FROM (KSHUV)

CALL FILE5/ LPRINT,NPGM,NDTA,NEQU

COMMON /SHGR/ NSHUVC, KSHUV, KEND, KSUB, KEXTD, KPAR

COMMON /CNTRL/ LIST, L1, CTRL(160), KXM(3)

C

IF(LIST.EQ.11) WRITE (NPRINT,1)

FORMAT (9H LGRABBED)

IF(KSHUV.EQ.0) RETURN

KPAR=NSHUV(KSHUV)

KEND=NSHUV(KSHUV)

KSUB=NSHUV(KSHUV)

RETURN

END
FUNCTION LOCATE (ARAY, NSTR, KSTR, KLEN)

LOCATE STRING (NSTR) IN ARRAY (ARAY) WITH LENGTH (KLEN) STARTING AT (KSTR). LOCATE=0 IF STRING NOT FOUND. LOCATE=1 IF STRING NOT FOUND.

DIMENSION NRAY(80), NSTR(80)

LOCATE=1
CONTINUE
LOCATE=0
RETURN

END
FUNCTION LOCTST (ILOC,IPOP)

C DETERMINE IF THE LOCATOR (IPOP) IS A VALID SYMBOL
C AND FIND ITS POINTER (ILOC) IN (KLCCA) AND SET LOCTST=0
C IF INVALID SET LOCTST=1.

C

C DETERMINE IF THE LOCATOR (IPOP) IS A VALID SYMBOL
C AND FIND ITS POINTER (ILOC) IN (KLCCA) AND SET LOCTST=0
C IF INVALID SET LOCTST=1.

C

C DETERMINE IF THE LOCATOR (IPOP) IS A VALID SYMBOL
C AND FIND ITS POINTER (ILOC) IN (KLCCA) AND SET LOCTST=0
C IF INVALID SET LOCTST=1.

C
0033 IF(1POP.LT.1000000.AND.KLOCA(I,ILOC).LT.1000000) GO TO 4 - PRI 1408
CC39 IF(MLOC.LT.JLOC) GO TO 4 PRI 1409
C FIND SUBSCRIPT POSSIBLE RANGE
C130 EQ. 2 J=1,8
C131 JPPT=VPT(J) PRI 1411
C132 IF(JLOC.LT.NPT(JMPT)) GO TO 3 PRI 1413
C133 CONTINUE
C134 J=0
C135 I=J-1
C CHECK SUBSCRIPT WITHIN RANGE
C136 IF(J.LT.0) GO TO 4 PRI 1416
C137 JPPT=VPT(J) PRI 1417
C138 IF(NLENC(JMPT).EQ.0) GO TO 3 PRI 1419
C139 IF((KLOC-JLOC).LE.NLENC(JMPT)-1) GO TO 6 PRI 1420
C140 CC9 4 CCNITALE
C NO POSSIBLE MATCH
C141 I=LOCST=1 PRI 1421
C142 GO TO 7 PRI 1422
C FORM SUBSCRIPT FROM GIVEN RANGE
C143 IF(IPJP.GT.ICCC=0.OR.KLOCA(I,ILOC).GT.1000000.OR.KEXTD.NE.0) KSUBPRI 1423
C144 1=KLOC-JLOC+1 PRI 1425
C145 IF(1POP.LT.4000000.AND.LOCTST.EQ.0) GO TO 10 PRI 1426
C146 IF(1LIST.EQ.1) WRITE (NPRINT,9) LOCTST PRI 1427
C147 RETURN PRI 1430
C47 INVLAD VARIABLE ELEMENT
C148 CALL ERROR (34) PRI 1431
C149 END
FUNCTION LPULL (NRAY, KSTR, NSYM, KDO, KSUB)  

PULL NEXT SYMBOL FROM (NRAY) BEGINNING AT (KSTR)  
UP TO COLUMN 72 INTO (NSYM) LEFT JUSTIFIED. READ  
NEXT CARD IF END OF LINE FOUND AND CONTINUE  
NEXT CARD MUST BE CONTINUATION CARD, ELSE LPULL=-0  
UPDATE (KSTR) TO NEXT LOCATION.  
LPULL=0 GOT SYMBOL  
LPULL=1 GOT SYMBOL AND GOT NEXT CARD  
LPULL=-1 DID NOT GET SYMBOL BUT GOT NEXT CARD  

 LPULL=7 FIND BEGINNING (ALPHA) CHAR OF SYMBOL  
 IF(NRAY(KSTR).EQ.NSP) GO TO 3  
 ELSE  I=1,26  
 IF(NRAY(KSTR).EQ.I) GC TO 8  
 CONTINUE  
 NOT A SYMBOL  
 IF(KDC.EQ.0) GC TO 4  
 PLUSH KCA-SYMBOL CHARACTER  
 IF(NRAY(KSTR).EQ.NSP) GO TO 4  
 N=1 KPUSH KPUSH+1  
 SET KDO TO 2 IF SUM OF PRODUCTS TERM  
 IF(NRAY(KSTR).EQ.NPLUS.OR.NRAY(KSTR).EQ.NMINUS) KDO=2  
 IF(KPUSH.GT.KPMA) GO TO 27  
 KSTR=KSTR+1  
 TRY TO FIND THE SYMBOL AGAIN  
 IF(KSTR.LE.72) GO TO 1  
 END OF RECORD-GET ANOTHER
NEOF = LREAD(NPGN+1)
FORMAT (1X,BCAl)
CHECK CONTINUATION CHARACTER
IF(NRAY(6).EQ.ASP) GO TO 6
IF(LIST.EQ.1) WRITE(NPRINT,5) NRAY
FIND SYMBOL IN NEW RECORD
GO TO 1
SYMBOL NOT IN THIS RECORD-NC CONTINUATION FCUNA
LPULL = -1
KSTR=7
RETURN
IF SYMBOL IS "CCC" Ignore it
GO TO 10
CLEAR SYMBOL
DO 9 I=1,6
NSYM(I)=NSP
GET SYMBOL
DO 14 I=1,6
IGNORE SPACES IN SYMBOL
IF(NRAY(KSTR).NE.NSP) GO TO 11
KSTR=KSTR+1
IF(KSTR.LE.72) GO TO 10
END OF RECORD-FAKE END OF SYMBOL
KSTR=KSTR-1
GO TJ 15
CHECK ALPHANUMERIC CHAR. HERE
DO 12 J=1,36
IF(NRAY(KSTR).EQ.NALPHA(J)) GC TO 13
CONTINUE
END OF SYMBOL
GO TO 14
GOT AN ALPHANUM CHAR. - SAVE IT
NSYM(I)=NRAY(KSTR)
KSTR=KSTR+1
CHECK FOR END OF RECORD
IF(KSTR.LE.72) GO TO 14
END OF RECORD-GET NEW RECORD
NEOF = LREAD(NPGN+1)
CHECK FOR CONTINUATION FLAG
IF(NRAY(6).EQ.ASP) GO TO 25
OK CONTINUE RECORD-FIND SYMBOL
GO TO 7
CONTINUE
CHECK FOR SUBSCRIPT
IF(NRAY(KSTR).NE.NLP) RETURN
KSTR=KSTR+1
C FAKE AN "N" SUBSCRIPT
     KSUB=1CCCCCO
     IF(NRAY(KSTR).EQ.NN) GO TO 19
     IF(NRAY(KSTR).EQ.NCM) GO TO 16
     KSTR=KSTR+1
     GC TO 19
     IF(LPUPP2(NRAY,KSTR,KV1).EQ.1) GO TO 22
     IF(NSYM(3).EQ.MX(II)) GO TO 18
     CONTINUE
     GC TO 22
     IF(KSYM(II).EQ.C) GO TO 22
     KSYM=KSYM+1
     KCSTR=KSTR+1
     IF(LPUPP2(NRAY,KSTR,KV2).EQ.1) GO TO 22
     KSYM=KSYM+1
     KSTR=KSTR+1
     GC TO 22
     IF(KSUB.NE.100*N) GO TO 21
     IF(NRAY(KSTR).EQ.NRP) GO TO 21
     KSTR=KSTR+1
     GO TO 20
     RETURN
     "XTABLE" FOUND
     KSTR=KSTR-1
     RETURN
     "VALID SUBSCRIPT NUMBER-CHECK EXPRESSION"
     IF(NSYM(N).NE.NX) GO TO 28
     KSYM=KSYM+1
     IF(Locate(NSYM,NXTAB,2,5).EQ.C) GO TO 26
     DC 23 I=KSTP,50
     CONTINUE
     IF(NSYM(N).EQ.NRP) GO TO 24
     CONTINUE
     "NE END CF ARG. FOUND.
     KSTR=KSTR+1
     GO TO 20
     "ERRORS"
     "STACK OVERFLOW"
     CALL EQROR (22)
     "BAD SUBSCRIPT"
     CALL ERROR (23)
     "BAD EXPRESSION ARGUMENT STRING"
     CALL ERROR (24)
FORTRAN IV G1 RELEASE 2.0
CCG3
END

LPULL

DATE = 78178
14/44/45

PR1 1580

ORIGINAL PAGE IS OF POOR QUALITY

-79-
FUNCTION LPUMP (NRAY,KRAY,KLEN,KVAL)  

C PUMP OUT THE INTEGER VALUE (KVAL) FROM BCD ARRAY (NRAY)  
C STARTING AT (KRAY) FOR LENGTH (KLEN). UPDATE (KRAY) AND  
C IGNORE BLANKS. LPUMP=0 IF CKAY CONVERSION  
C LPUMP=1 IF ERROR IN CONVERSION  
C
C
FUNCTION LPUMP (NRAY,KRAY,KLEN,KVAL)  

COMMON /CHARS/ NSP,NLP,NRP,NCON,NEQ,ND,NPLUS,NMINUS,NSL,NLERNXM  

DIMENSION NRAY(80)  

KVAL=0  
LPUMP=0  
DO 3 1=1,KLEN  
IF(NRAY(KRAY).GT.NSP) GO TO 3  
CC 3   
IF(NRAY(KRAY).EQ.NNUM(J)) GO TO 2  
CC 2   
CONTINUE  
LPUMP=1  
RETURN  
CC 3   
KRAY=KRAY+1  
RETURN  
END
FUNCTION LPUMP2 (NRAY,KRAY,KVAL)

PUMP THE NEXT SUBSCRIPT SLRRGUDED NUMBER FROM NRAY
RETURN VALUE AS LPUMP

COMMON /CIARS/ NSP,NLP,NRP,NCOM,NEQ,NO,NPLUS,NMINUS,NSL,NZERO,NXM(3)
DIMENSION NRAY(80)

LPUMP2=1
LRAY=KRAY
LRAY=LRAY+1
IF(LRAY.GT.771 RETURN
IF(NRAY(LRAY).NE.NRP.AND.NRAY(LRAY).NE.NCOM) GO TO 1

KXRAY=LRAY-KRAY

LPUMP2=LPUMP(NRAY,KRAY,KXRAY,KVAL)
RETURN

END
FUNCTION LREAD (NFIL, LIST)

C READ A RECORD FROM FILE (NFIL) INTO (NRAY)
C LIST IF (LIST) IS = 1, RETURN CODES:
C LREAD=0 NO END OF FILE AND "END" NOT IN COLS 7-9
C =1 END OF FILE OR "END" IN COLS 7-9
C
C COMMON /NASA/ NSYM(6,300),KSYM,NRAY(80),KRAY,KLEN(8),NPT(8),NT(300)

COMMON /FILES/ NPRINT,NPGM,NOTA,NEQU
C DIMENSION NEND(3)
C DATA NEND/LHE,IHN,1HD/
C
C SET POINTER AND FLAG
C KRAY=7
C LREAD=0
C
C READ NEW LINE
C READ (NFIL,1,ENC=3,ERR=3) NRAY
C 1 FORMAT (8CAI)
C IF(LIST.EQ,1) WRITE (NPRINT,2) NRAY
C 2 FORMAT (1X,8OA1)
C "END" APPEARS?
C 3 IFLOCATE(NRAY,NEND,7,3).EQ.0) LREAD=1
C
C RETURN
C
C RETURN
C END
FUNCTION LSHUW(KPOP, IPOS, KPCPC)

C PUSH THE CURRENT SYMBOL LOCATORS INTO (KSHUV) IF THE
C NEXT SYMBOL IS LOCATED IN THE LIST (KLOCA) OF SYMBOLS
C
C
C IMPLICIT REAL*8 (J,H,O,I)
C COMMON /PLSHELD/ KLOCA(3,400), KLOC, NPSH(1000), KPSH, KMAX
C COMMON /SHGR/ NSHUV(500), KSHUV, KST, KEND, KSUB, KEKTD, KPAR
C COMMON /CTRL/ LIST, L1, CTRL(160), KXM(3)
C COMMON /FILES/ NPRINT, KPOP, NACT, NEXIT

C KSHUV = 1
C GO IF ELATION FOUND
C IF (LOCST(IPOS, KPOP), EQ. 0) GO TO 1
C USE LAST VALUE

C KPS = KMOD
C KPOP = KPOP
C GO TO 2
C 1
C PUSH SUBSCRIPT, POINTERS, VALUE AND SUM-PRODUCTS FLAG

C IF (KSHUV .GT. 500) GO TO 5
C KSHUV = KSHUV + 1
C NSHUV(KSHUV) = KSLB
C KSHUV = KSHUV + 1
C NSHUV(KSHUV) = KST
C KSHUV = KSHUV + 1
C NSHUV(KSHUV) = KEND
C KSHUV = KSHUV + 1
C NSHUV(KSHUV) = KPCPC
C KSHUV = KSHUV + 1
C NSHUV(KSHUV) = KPAR
C KPCPC = KPOP
C 2
C IF (LIST .EQ. 0) RETURN
C K5 = KSHUV - 4
C WRITE (NPRINT, 3) IPOS, KPOP, KSUB, KEKTD, LSHUV, KSHUV
C IF (K5 .GT. 0) WRITE (NPRINT, 4) (NSHUV(I), I = K5, KSHUV)
C 3
C FORMAT (7X, 511C)
C 4
C RETURN
C ERRORS
C
C SHUV STACK OVERFLOW
CALL ERROR (33) END
FUNCTION NUMB (NRAY, KST, KEN, I)

DECODER NUMBER AT KST IN NRAY UNIITL KEN

COMMON /FILES/ NPRINT, NPGM, NCTA, NEQU
COMMON /CTRL/ LIST, LI, CTROL(160), KXM(3)
COMMON /CHAR/ ASP, NLP, NRP, NCOM, VEQ, ND, NPLUS, NMINUS, NSLP, NZER, NXM(3)
DIMENSION /ARS/ NRAY, NCOM, NLP, NRP, NPLUS, NMINUS, NSLP, NZER, NXM(3)

I = 7
CONTINUE
IF (NPAY(I) .EQ. ASP) GO TO 4
DO 3 J = 1, 10
IF (NRAY(I) .NE. NUMS(J)) GO TO 3
NUMB = NUMB + 10 + J - 1
CONTINUE
RETURN
END
IMPLICIT REAL*8(A-H,O-Z)
STANDARD SCIENTIFIC COMMON AREAS
COMMON /CUTPTS/ (51)
COMMON /CURRTS/ C(I)
COMMON /DERIVS/ D(I)
COMMON /DNAMS/ DPS(I)
COMMON /ELNAMS/ LELS(6,300)
COMMON /HCRDS/ HEAD(9,11)
COMMON /HEADS/ XN(1)
COMMON /OUTR/ OUTR(5,1)
COMMON /CLTPUT/ NCUTP(6,50)
COMMON /PRMTRS/ P(I)
COMMON /SAVICT/ CICT(1)
COMMON /SCRTMS/ S(I)
COMMON /TBLUTS/ T(I)
COMMON /TAPUTS/ NPRT, NOTA, APMW
COMMON /VGLTGS/ V(I)
COMMON /XPUTL/ XM(I)
COMMON /PROSE/ A(50,50), B(50,50), L(32), (46), (50), (60), (70), (80), (90)
COMMON /XPUTL/ X(50)
DIMENSION CTROLS(160), ER(50), E(50), XER(50), XEI(50)
DIMENSION NL(80), NUT2(6), NL(20), NL(10), NL(5), NL(3)
DATA NSRCDR/IHS,IHR,IHC,IHD,IHRJHV/
DATA XNCLTPT/IHTIHA,IHB,IFL, IPE,IHS/
DATA NCLTPT/IHRJlU,IHT,IhPLT,LHS/
DATA NCDN/IH,/,NBL/IH /
DATA NUMS/IHI.,I-1,1H2,IH3,1H4,1H5,1H6,1H7,1H8,1H9/
DATA NT/IPI/,NV/1-V/
EQUIVALENCE /PTPE,CTROLS(1))
C
C SET I/C DEVICE CODES
C
CC29 NPRT=6
CC30 NDATA=1
CC31 NPGM=2
CC32 CALL SETUP2 (NAC,LEL,NSRC,NTA3)
C
C033 IF(XWRITSM.EQ.1.00) WRITE (NPRT,1) CONTROLS
C034 1 FORMAT (21H <<<<CONTROLS<<<<>>,32(IX,1P5D14.7,/) )
C035 2 FORMAT (1P5D14.7)
C
CC36 READ I/FACING CARDS
CC37 NHEAD=IXNHEAD
CC38 IF(NHEAD.NE.0) READ (NDTA,4) HEAD
CC39 4 FORMAT (26H <<<<<<HEAD CARDS<<<<>>,/ ,I(1X,9A8,/) )
CC40 READ ELEMENT VALUES
CC41 NEL=XNELTS
CC42 READ (NCTA,2) (XN(I),I=,NEL)
CC43 IF(XWRITSM.EQ.1.00.AND.NHEAD.NE.0) WRITE (NPRT,5) (HEAD(I,J),I=1,NH)
CC44 5 FORMAT (26H <<<<<<ELEMENT VALUES<<<<>>,/ ,60(I,1X,1P5D14.7,/) )
C
CC45 READ ELEMENT NAMES
CC46 IF(NAC.EQ.160) READ (NDTA,7) (LELS(I,J),I=1,6/J=1,NEL)
CC47 7 FORMAT (26H <<<<<<ELEMENT NAMES<<<<>>,/ ,30(I,1X,72A1,/) )
C
CC48 READ ELEMENT VALUES AND CURRENTS (INITIAL CONDITIONS)
CC49 READ (NCTA,2) (V(I),I=1,NEL)
CC50 IF(XWRITSM.EQ.1.00) WRITE (NPRT,8) (V(I),I=1,NEL)
CC51 8 FORMAT (26H <<<<<<VOLTAGES<<<<>>,/ ,60(I,1X,1P5D14.7,/) )
CC52 READ (NCTA,2) (G(I),I=1,NEL)
CC53 IF(XWRITSM.EQ.1.00) WRITE (NPRT,9) (G(I),I=1,NEL)
CC54 9 FORMAT (26H <<<<<<CURRENTS<<<<>>,/ ,60(I,1X,1P5D14.7,/) )
C
CC55 READ MUTUAL VALUES
CC56 NMUT=XNP33+XNML3+YNEC+XNBL
CC57 IF(NMUT.NE.0) READ (NDTA,2) (XM(I),I=1,NMUT)
CC58 10 FORMAT (26H <<<<<<MUTUALS<<<<>>,/ ,10(I,1X,1P5D14.7,/) )
C
CC59 READ SOURCE DERIVATIVES
CC60 IF(NAC.NE.0) READ (NDTA,2) (S(I),I=1,NSRC)
CC61 IF(XWRITSM.EQ.1.00.AND.NSRC.NE.0) WRITE (NPRT,11) (S(I),I=1,NSRC)
CC62 11 FORMAT (26H <<<<<<SOURCE DERIVS<<<<>>,/ ,10(I,1X,1P5D14.7,/) )
C
CC63 READ STATE VARIABLE SAVES I/C'S
CC64 NIC=XNL3+XNLL+YNEC+XNBL
CC65 IF(NIC.NE.0) READ (NDTA,2) (CICT(I),I=1,NIC)
CC66 IF(XWRITSM.EQ.1.00.AND.NIC.NE.0) WRITE (NPRT,12) (CICT(I),I=1,NIC)
CC67 12 FORMAT (26H <<<<<<STATE I/C'S<<<<>>,/ ,10(I,1X,1P5D14.7,/) )
C   COMPUTE NUMBER OF STATE VARIABLES
NSV=NSV+XNLL
C   READ DEFINED PARAMETER DERIVATIVES
NDPD=NDPD+1
IF(XWTSN.EQ.0) READ (NDTA,2) (P(I),I=1,NDPD)
C   WRITE (NPRT,16) (P(I),I=1,NDPD)
C   TABLES
IF(IPWTSN.EQ.0) READ (NDTA,17) ((NOUTP(I,J),I=1,6),J=1,NOPRQ)
C   OUTPUT REQUESTS
C   ZERO OUT DERIVATIVES
DC 22 I=1,NSV
DC 23 I=1,NEL
DC 24 I=1,NINDC
C FORM INDEPENDENT VARIABLE TABLE
N(J)=0
DO 26 I=2,NOPRQ
   DO 25 J=1,NEL
      DC 24 K=1,6
      IF(NOLTP(XI),AE.LELS(K,J)) GO TO 25
      CONTINUE
      KIND(J)=1
      N(J)=J-1-1
      XN(J)=C(CD0)
   GO TO 26
25 CONTINUE
26 CONTINUE
C COMPUTE OFFSETS TO STATE VARIABLE NAMES
L4S=XN7CT+1
L4P=XNAC+L4S-1
L3S=XN7CT+XNBL+XNLC+XNLR+1
L3P=XNLL+L3S-1
C CLEAR STATE VARIABLES
IF(L3S.GT.L3P) GO TO 28
DO 27 l=35,L3P
27 C(I)=C.O
IF(L3S.GT.L4P) GO TO 30
DO 29 I=L4S,L4P
29 V(I)=J.O
C PERFORM EIGAN ANALYSIS
CALL EIGAN (ER,E1)
C PRINT HEADER
WRITE (APRT,31)
31 FORMAT (27HNASA-LEWIS RESEARCH CENTER,1,27H----- ---------- -----)
1----,1/27H PROSE PROGRAM,1/27H ----- ---------- -----)
2----,1/27H UNIVERSITY OF SOUTH FLORIDA,1/27H ----- ---------- -----)
3----,1/27H----- ---------- -----)
C WRITE STATE VARIABLE NAMES
IF(L3S.GT.L3P) GO TO 37
DO 35 I=I,9A6,
35 NLT2(I+I)=LELS(I,I)
CALL BACK (NUT2,NUT1)
WRITE (APRT,33)
33 FORMAT (21H----- ---------- -----),/21H STATE VARIABLE NAMES,1,21H----- ---------- -----)
C WRITE STATE VARIABLE NAMES
IF(L3S.GT.L3P) GO TO 37
DO 35 I=I,9A6,
35 NLT2(I+I)=LELS(I,I)
NLT2(I)=NI
CALL BACK (NUT2,NUT1)
WRITE (APRT,36)
36 FORMAT (1X,6A1)
C WRITE STATE VARIABLE NAMES
C4C 37 IF(L4S.GT.L4P) GO TO 41
014 38 DO 39 I=L4S,L4P
0142 39 DO 38 II=1,5
0143 38 NUT2(II+1)=NELS(II,I)
0144 39 NUT2(I)=NV
0145 39 CALL BACK (NUT2,NUT)
0146 39 WRITE (NPR,40) NUT
C147 40 FORMAT (1X,6A1)
0148 41 WRITE (NPR,42)
0149 42 FORMAT (1/20H -------- -------,1/20H INDEPENDENT SOURCES,1/20H)
014A 43 WRITE A MATRIX HEADER
014B 44 WRITE (NPR,44) (A(I,J),J=1,NSV)
014C 45 WRITE A MATRIX
014D 46 FORMAT (1X,6A1)
014E 47 CONTINUE
C162 48 WRITE (NPR,48)
0163 49 WRITE (NPR,49) (B(I,J),J=1,NSV)
0164 48 WRITE A MATRIX
0165 49 WRITE (NPR,44) (B(I,J),J=1,NSV)
C166 49 WRITE EIGENVALUES
0167 50 WRITE (NPR,51)
0168 51 FORMAT (1/24H ------- -------,1/24H EIGENVALUES OF MATRIX)
0169 51 X=0.0 DO 52 ER(I),EI(I)
0170 52 ER(I)=ABS(ER(I))
0171 52 XI=ABS(X)
0172 52 FORMAT (1X,1PE4.7,1PE4.7)
0173 52 IF(DABS(X).GT.DABS(XI)) GO TO 53
0174 53 CONTINUE
C176 53 DC SENSITIVITY ANALYSIS
-90-
0177 WRITE (NPRT,54) IL,XL
0178 54 FORMAT (/,'21H SENSITIVITY ANALYSIS,'/,'21H MAXIMUM EIGENVALUE #,'/,'13H MAGNITUDE'/,31)
0179 
0180 
0181 DO 61 I=1,NEL
0182 C IGNORE IF INDEP. SOURCE
0183 IF(NIND(I).EQ.1) GO TO 61
0184 C IGNORE IF ZERO VALUED
0185 IF(XN(I).EQ.0.0D0) GO TO 61
0186 C SAVE OLD VALUE
0187 SV=XN(I)
0188 C BUMP VALUE BY 10 PER CENT
0189 XN(I)=XN(I)*1.1D0
0190 C COMPUTE SENSITIVITIES
0191 EG=DSCRT(XER(IT)*XER(IL)*XEI(IL)*XEI(IL))-XL
0192 EG=0.0D0/100
0193 C COMPUTE SENSITIVITIES
0194 EG=DSCRT(XER(IT)*XER(IL)*XEI(IL)*XEI(IL))-XL
0195 EG=0.0D0/100
0196 C BUMP VALUE BY 10 PER CENT
0197 XN(I)=XN(I)*1.1D0
0198 C ADD SENS APPROX. -1
0199 CALL BACK (LELS(I,1),NUT)
0200 WRITE (NPRT,27) NUT,EG
0201 27 FORMAT (2X,6A1,5X,F6.3)
0202 C ADD SENS APPROX. 1
0203 CALL BACK (LELS(I,1),NUT)
0204 WRITE (NPRT,27) NUT,EG
0205 27 FORMAT (2X,6A1,5X,F6.3)
0206 C RESTORE VALUE
0207 XN(I)=SV
0208 61 CONTINUE
0209 60 XN(I)=SV
0210 61 CONTINUE
0211 IF(I.EQ.1) STOP
CALL BACK (LCLS(1,AL),NUT)
WRITE (NPAT,62) NUT
62 FORMAT (/,'NO SENSITIVITIES WITHIN 3% OF 1 . ,/' ,8X,6AI,' MAY PR2 0291
       1BE REMOVED (I.E. SET TO ZERO IF,19H SENSITIVITY < ZERO,/',29X,3PR2 0292
       29H SET TO INFINITY IF SENSITIVITY > ZERO.)')
STOP
END
SUBROUTINE ATETG (M, A, RRRIIANA, IA)

COMPLETE EIGENVALUES OF UPPER HESSENBERG MATRIX A
M=ORDER OF MATRIX A
RR=REAL EIGENVALUE RESULTS ARRAY
RI=IMAGINARY EIGENVALUE ARRAY
IANA=COMPUTE METHOD ARRAY (0 OR 2=DIRECT)
IA=FIRST DIMENSION OF A MATRIX

IBM SSP ROUTINE

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION A(I), RR, RI, PRR(2), PRI(2), IANA(I)
INTEGER*4 P, PI, C

E7=1.0D-8
E6=1.0D-6
E10=1.0D-10
DELTA=0.5
MAXIT=50

INITIALIZATION
N=M
N1=N-1
IN=IN+1
NA=IN+N
IF(N1) 2,69,2
NP=NP+1

ITERATION COUNTER
IT=0
RCCTS OF THE 2ND ORDER MAIN SUBMATRIX AT THE PREVIOUS ITERATION

DC 3 I=1,2
DC 3 I=1,2

LAST TWO SUPERDIAGONAL ELEMENTS AT THE PREVIOUS ITERATION
PAN=0.0
PANI=C.0
ORIGIN SHIFT
R=C.0
S=0.0

ROOTS OF THE LOWER MAIN 2 BY 2 SUBMATRIX
N2=NI-1
IN1=NI-1

N1=NI+N1
N1N=IN+N1
N1N1=NI+NI

T=A(NIN1)-A(NN)
U=T*T

V=4.0*A(LN)+A(KN)
IF(JABS(V)-A'E7) 11,11,5
T=U+V
IF(DABS(T)-CMAX1(UJABS(V))*E6) 6,6,7
T=C.0
U=(A(N)+A(NN))/2.0
V=DSORT(UJABS(U))/2.0
IF(T) 15,8,8
IF(U) 10,9,9
RR(NI)=U+V
RR(N)=U-V
GO TO 14
RR(NI)=L-V
RR(N)=U*V
GO TO 14
IF(T) 13,12,12
RR(N1)=A(N1)
RR(N)=A(N1)
GO TO 14
RR(NI)=A(NN)
RR(N)=A(NN1)
GO TO 14
RR(NI)=C.0
RR(N)=C.C
GO TO 16
RR(N1)=U
RR(N)=U
RR(N)=U
IF(N2)=68,68,17
TESTS OF CONVERGENCE
-94-
CC6C 17 NIN2=NIN1+IA
CC6D 18 RVD=RR(N1)+RR(N1)+RI(N1)+RI(N1)
CC6E 19 EPS=10**DSQRT(RVD)
CC6F 20 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6G 21 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6H 22 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6I 23 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6J 24 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6K 25 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6L 26 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6M 27 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6N 28 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6O 29 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6P 30 IF(DABS(A(NIN2))=EPS) 68,68,18
CC6Q 31 IF(DABS(A(NIN2))=EPS) 68,68,18

COMPUTE THE SHIFT

J=1

K=N-1

IF(DABS(RR(K)-PRR(I))+DABS(RI(K)-PRI(I))-DELTA*CDABSRR(K))+DABS(RPRI(K)) 23,24,24

J=J+1

CONTINUE

R=0.0
S=0.0

R=RR(J)+RR(J)
S=RR(J)+RR(J)

SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF THE
SUBMATRIX BEFORE ITERATION

PAN=A(NIN1)
PAN1=A(NIN2)

DO 29 I=1,2
K=IP-I
PRR(I)=RR(K)
PRI(I)=RI(K)

SEARCH FOR A PARTITION OF THE MATRIX, DEFINED BY P AND Q

P=N2
IF(N-3) 35,35,30

DO 34 J=2,N2

IPI=IPI+IA
GO TO 49
ALPHA = 2.0
PSII = 0
PSII = 0
IF (I - J) .GE. 50, 53, 56
IF (I - J) .GE. 50, 52, 51
A(IJ + 1) = - ETA
Go TO 53
A(IJ) = - ETA
IF (I - J) .GE. 55, 57, 57
A(IP2J) = A(IP2J) - ETA
IF (I - K) .GE. 58, 58, 58
K = N
Go TO 60
IP = IP + 1
GO TO 64
JIP = J + P
J = J + 1
K = I + 2
GO TO 66
JIP = J + 1
I = I + 1
JIP = J + 1
IF (I - J) .GE. 61, 62, 62
JIP = J + 1
A(JIP) = A(JIP) - ETA
PSII
IF (I - J) .GE. 63, 64, 64
A(JIP2) = A(JIP2) - ETA
CONTINUE
IF (I - J) .GE. 65, 66, 66
JIP = J + 1
CONTINUE
FORTRAN IV G1 RELEASE 2.0  ATEIG  DATE = 78178  14/44/45  PAGE 0006

0180  JIP2=JIP+IA
0181  ETA=ALPHA*PSI2*A(JIP2)
0182  A(JI)=-ETA
0.03  A(JIP)=-ETA*PSI1
0184  A(JIP2)=A(JIP2)-ETA*PSI2
0185  II=IJP+1
0186  IT=IT+1
0187  GC TO 4

C  END OF ITERATION
C

0188  IF(OABS(.%(NI)-OA8S(A(N1N2J1))>69,68,68
C  TWO EIGENVALUES HAVE BEEN FOUND
C
0189  IANA(N)=0
018C  IANA(N1)=2
0191  N=N2
0192  IF(N2) 71,71,1

C  ONE EIGENVALUE HAS BEEN FOUND
C

0193  RR(N)=A(ΚΑ1)
0194  Ρ1(Κ)Α=Ν.Ο
0195  IANA(Ν)=1
0196  IF(Ν1) 71,71,70

0197  NN=1
0198  G0 TO 1
0199  RETURN
0200  END
SUBROUTINE E2CK (NAMENCHAR)

CHANGE NAME TO NCHAR IN MECHANICAL FORMAT.

GP=TP DXXX=ZXXX AND CHANGE XXX

I=F EXCEPT ICW=TJ

EQUALS V=V EXCEPT VCW=HJ

ELEMTS

CU=M EU=U JU=R LU=K RU=D

C

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
0033  KEL=5
0036  KELF=0
0037  IF(NCHAR(KLOC).EQ.NU) GO TO 9
0038  KEL=3
0039  KELF=5
0040  GO TO 10
0041  KELF=(N-KLOC)
0042  DO 12 K=1,KELF
0043  NCHAR(KLOC+K-1)=NCHAR(KLOC+K)
0044  CONTINUE
0045  RETURN
0046  IF(KLOC.EQ.KN) GO TO 13
0047  KELF=KLOC-KLOC
0048  DO 12 K=1,KELF
0049  NCHAR(KLOC+K-1)=NCHAR(KLOC+K)
0050  12 CONTINUE
0051  NCHAR(KN)=NBL
0052  GO TO 2
0053  END
SUBROUTINE EIGAN (ER, EI)

SUPERVISE COMPUTATION OF EIGENVALUES FROM SPECIFIED ROUTINE SIMTR

DIMENSION ER(5), EI(50), XA(50), SSV(50)

BEGIN CODE

CALL SIMTR (KDUM, C)
DO 1 I=1, NSV
1 SSV(I)=C(I)
LOOP THROUGH STATE VARIABLES THROUGH TYPE 3'S FIRST

IF (L3S.GT.L3P) GO TO 4
DC 3 I=L3S, L3P
C(I)=L.COC

COMPUTE DERIVATIVES CALL SIMTR (KDUM, 0)
FORM AN A MATRIX ROW
DO 2 J=1, NSV
2 A(J, I-L3S+1)=0(J)-SSV(J)

THROUGH TYPE 4 STATE VARIABLES

4 IF (L4S.GT.L4P) GO TO 7
DC 6 I=L4S, L4P
V(I)=1.COC

CALL SIMTR (KDUM, 0)
DO 5 J=1, NSV
5 A(J, I+L3P-L3S-L4S+2)=0(J)-SSV(J)
CC28  6  Y(I)=C(I,DO)
CC29  7  IF (NEJ.EQ.0) GO TO 10
CC30  8  II=1
CC31  9  DO 9 I=1,NEL
CC32  10  IF (NIND(I),NE.1) GO TO 9
CC33  11  XN(I)=1.0DO
CC34  12  CALL SIMTR (KOUY,J)
CC35  13  DO 8 J=1,NSV
CC36  14  B(J,II)=C(J)-SSV(J)
CC37  15  II=II+1
CC38  16  CONTINUE
CC39  17  CFORM EIGEN VALUES
CC40  18  USE ANOTHER ARRAY (SINCE A IS DESTROYED BY HSBG)
CC41  19  DO 11 I=1,NSV
CC42  20  DO 11 J=1,NSV
CC43  21  XA(I,J)=A(I,J)
CC44  22  CALL HSBG (NSV,XA,50)
CC45  23  CALL ATEIG (NSV,XA,ER,IE,IPERF,50)
CC46  24  RETURN
CC47  25  END
SUBROUTINE HSBG (A,IA,IA)

REDUCE A MATRIX TO UPPER HESSENBERG FORM

N=ORDER OF A, IA=FIRST DIMENSION OF A

IMPLICIT REAL*8(A-H,O-Z)

DIMENSION A(1)

L=N

LIA=IA

LIA=NIA-IA

L IS THE ROW INDEX OF THE ELIMINATION

SEARCH FOR THE PIVOTAL ELEMENT IN THE LTH ROW

ISLB=LIA+L

IPIV=ISLB-IA

PIV=DABS(A(IPIV))

IF(L-3) 19,2,2

L1=L-1

L2=L1-1

INTERCHANGE THE COLUMNS

M=IPIV-L

DO 5 I=IL,IA

T=A(J)

A(J)=A(K)

A(K)=T

IF(IPIV-18) 11,11,8

IF(IPIV) 7,18,7

IF(IPIV-CABS(A(ISUB))) 11,11,8

CONTINUE

DO 5 I=IL,IA

T=A(J)

A(J)=A(K)

A(K)=T

5 CONTINUE
INTERCHANGE THE ROWS

\[ P = 12 - \frac{\mu}{\nu} \]

DO 10 \( i = L1, N \)

\[ T = A(i) \]

J = I - P

\[ A(i) = A(j) \]

10 \( A(j) = T \)

TERMS OF THE ELEMENTARY TRANSFORMATION

DO 11 \( i = L, L1, A \)

\[ A(i) = A(i) / A(ISUE) \]

RIGHT TRANSFORMATION

J = -IA

DO 12 \( i = 1, L2 \)

\[ J = J + 1A \]

LJ = L + J

DO 13 \( k = 1, L1 \)

\[ KJ = K + J \]

KL = K + LIA

\[ A(KJ) = A(KJ) - A(LJ) * A(KL) \]

CONTINUE

LEFT TRANSFORMATION

K = -IA

DO 16 \( i = 1, N \)

\[ K = K + 1A \]

LJ = K + L1

S = A(LK)

LJ = L + IA

DO 15 \( j = 1, L2 \)

\[ JK = K + J \]

LJ = LJ + IA

S = S + A(LJ) * A(JK) * 1.0

\[ A(LK) = S \]

SET THE LOWER PART OF THE MATRIX TO ZERO

DO 17 \( i = L, L1, A \)

\[ A(i) = 0.0 \]

L = L1

GO TO 1

RETURN
CCC1
SLBROLIST INTEGRB
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<PR2 0759
CCC2
RETURN
CCC3
END
SUBROUTINE INVERS (NAMAT, AMAT, NR, NC, IERROR)
IMPLICIT REAL*4(A-H,O-Z), INTEGER*4(I-N)

INNAMMAT = NAME OF MATRIX TO BE OPERATED ON
AMAT = THE MATRIX TO BE SOLVED OR INVERTED
NR = THE NUMBER OF ROWS IN AMAT
NC = THE NUMBER OF COLUMNS IN AMAT
I = NC = NR + 1 FOR EQUATION SOLUTION
I = AC = 2 * AR FOR MATRIX INVERSION
IERROR = ONE IF THE MATRIX IS SINGULAR, OTHERWISE ZERO

OC01 REAL*4 NAMAT
OC02 COMMON /TAPES/ NOUTTP, INTAPE, LIDTP, INOUTP, MEDTP, NSAVTP
OC03 COMMON /CNTRLS/ TIMEX, STOPX, XERTX, XISSX, XISSY, XISSZ, XISST
OC04 1, XIXATX, XIEXR, XIXMKPA, XIXMNI, XIXMCKX, XIXICP, XIXICER, XIXICER
OC05 2R, XIXMCTP, XIXMITY, XIXMICS, XIXICPA, XIXMTRP, XIXMTRT, XIXMTNT
OC06 3ERN, XIXCITL, XIXTPAC, XIXPSNO, XIXRAC, XIXTPS2
OC07 DIMENSION AMAT(NR,NC)
OC08 EQUIVALENCE (NCUTP,OUTAPE)
OC09 INTCEP OUTAPE
OC10 IF(XICPA.NE.X'XICP) GO TO 2
OC11 WRITE (CNOUTP,I) X'MICP
OC12 FORMAT (34HTHE NEWTON-RAPHSON PASS LIMIT OF ,F5.0,48H HAS BEEN EXP2 082)
OC13 1CEEDED WITHOUT ATTAINING CONVERGENCE)
OC14 IERROR=10
OC15 GC TO 20
GC17 2 IERROR=2
OC18 NRX2=2*NR
OC19 INC=NR
OC20 IF(NRX2.NE.NC) INC=NC
OC21 C CHECK FOR 1X1 MATRIX AND SOLVE IF FOUND
OC22 IF(INRC.NE.1) GO TO 3
OC23 AMAT(1,2)=AMAT(1,2)/AMAT(1,1)
OC24 GO TO 2C
OC25 NR2=2*NR
OC26 3 IF MATRIX INVERSION DESIRED, GENERATE IDENTITY MATRIX
OC27 IF(NC/LT,NR) GC TO 6
OC28 JINC=NR+1
OC29 GO TO 5 INC
GC30 5 INC=INC+1, NR
OC31 INCRD=INCR+NR
OC32 DO 5 INCOLM=JINC,NC
OC33 IF(INCOLM.EQ.INCROW) GO TO 4
OC34 AMAT(INCR,INCOLM)=9.
OC35 GO TO 5

-107-
CC30  4   AMAT(INCR,INCOLP)=1.
CC31  5   CONTINUE
CC32  6   DO 19 I=1,NR
CC33     M=M+1
CC34     J=1
CC35     C   CHECK FOR LAST DIAGONAL ELEMENT
CC36     IF(I.EQ.NR) GO TO 9
CC37     C   FIND THE LARGEST VALUE BELOW DIAGONAL ELEMENT IN THIS COLUMN
CC38  7   J=N
CC39     C   CONTINUE
CC40     C   SEE IF THE LARGEST ELEMENT IN ROW IS ZERO
CC41     9   IF(AMAT(J,I)) 12,10,12
CC42     C   INDICATES THAT MATRIX IS SINGULAR
CC43  10   IERR=11
CC44     WRITE (NOUTTP,11) NAMMAT
CC45     C   CONTINUE
CC46     C   SEE IF ROW INTERCHANGE IS NEEDED TO GET LARGEST ELEMENT ON DIAG.
CC47  12   IF(J-I) 15,15,13
CC48     C   INTERCHANGE ROWS OF DIAGONAL AND THOSE TO THE RIGHT OF IT
CC49  13   DO 14 N=M,NC
CC50     SAVE=AMAT(I,N)
CC51     AMAT(I,N)=AMAT(J,N)
CC52     AMAT(J,N)=SAVE
CC53     C   DIVIDE ALL ELEMENTS TO RIGHT OF DIAG. IN DIAG. ROW BY DIAG. ELEM.
CC54  15   DO 16 J=M,NC
CC55     AMAT(I,J)=AMAT(I,J)/DIAG
CC56     AMAT(I,I)=0.
CC57  16   DO 18 K=1,NR
CC58     IF(AMAT(K,I).EQ.0.) GO TO 18
CC59     IF(I.EQ.J) GO TO 17
CC60  17   AMAT(K,J)=AMAT(K,J)-AMAT(K,I)*AMAT(I,J)
CC61  18   CONTINUE
CC62  19   CONTINUE
CC63  20   RETURN
END
SUBROUTINE OUTP

DUMMY OUTP ROUTINE

RETURN

END
SUBROUTINE SAVEPD

DUMMY SAVEPD ROUTINE

RETURN

END
SUBROUTINE SETUP2 (NAC,LELS,NSRC,NTAB)

SETUP ROUTINE

IMPLICIT REAL*8(A-H,O-Z)

DIMENSION NSIMTR(5), NSRCDR(6), NTABLE(6), NNDPD(3)
DIMENSION NClrPT(6), NNMULT(5), NNAME(5), NPRMTR(6)
DIMENSION NICNX(5), INDERIV(6),内马尔(800)
DIMENSION NLMS(IO), LOCS(E)
DATA ITABLE /IHT,IHA,IHB,IHL/IHE,IHS/
DATA MICP/1H,1-D,1HP/
DATA KCP/1HC,1HT,1FP,1FH,1HS/
DATA NXNUTL/!NX,IHP,'HU,HT,IHL/
DATA NNAMES/IHN,HA,IHI,HHE,HS/
DATA NPRYTR/IP,IR,NIH-,IFTl-R,1HS/
DATA NUCNX/1HT,1HC,1HM,1HT,1HR/
DATA NCUTPT/IHC,1HM,1HT,1FP,1FH,1HS/
DATA NKP/1HM,1HT,1FP,1FH,1HS/
DATA NSRCER/IHS,i-R;IHC,If-C,IfRIHV/
DATA ITABLE /IHT,IHA,IHB,IHL/IHE,IHS/
DATA ECUIVLENCE (TIME,CTROLS(1))
DATA ECUIVLENCE (NCOP,NNDPD(1))
DATA ECUIVLENCE (NRLK,NONB(4))
NTAB=0
NSRC=2
READ (APGP,2,END=33) NRAY
FORMAT (ECAL)
IF (LOCATE(NRAY,NSIMTR,9,5).EQ.1) GO TO 1
READ (APGP,3) NRAY
FORMAT (/*/*/*,80AL)
NAC=70
IF(NRAY(6).EQ.MST) NAC=160
READ (NDTA,4,END=35) (CTRLS(I),I=1,NAC)
FORMAT (1P5D14.7)
IF(NAC.EQ.160) GO TO 6
00 5 I=71,160
CTRLS(I)=C.CDC
0040 6 READ (NPGF,2) NRAY
0041 7 IFLOCATE(NRAY,NSRCDR,15,6).EQ.0) GO TO 9
0042 IFLOCATE(NRAY,NTABLE,15,6).EQ.0) GO TO 11
0043 IF(NAC.EQ.160) CC TC 8
0044 IFLOCATE(NRAY,NXMUTL,15,5).EQ.0) GO TO 12
0045 IFLOCATE(NRAY,NAMES,15,5).EQ.0) GO TO 14
0046 IFLOCATE(NRAY,NPRMTR,15,6).EQ.0) GO TO 21
0047 IFLOCATE(NRAY,NDERIV,15,6).EQ.0) GO TO 23
0048 IFLOCATE(NRAY,NIONBX,13,5).EQ.0) GO TO 25
0049 8 IFLOCATE(NRAY,NOUTPT,15,6).EQ.0) GO TO 32
0050 GO TO 6
0051 9 K=22
0052 10 NHOL=NCNT(NRAY,K)
0053 NSRC=NSRC+NHOL
0054 IF(NRAY(72).EQ.NBLK) GO TO 6
0055 READ (NPGM,2) NRAY
0056 IF(NRAY(6).NE.NAST) GO TO 7
0057 GO TO 10
0058 11 NTAB=NUMB(NRAY,25,27,K)
0059 GC TO 10
0060 12 K=21
0061 13 N=OL=NCNT(NRAY,K)
0062 XRAY=XXM33+NHCL
0063 IF(NRAY(72).EQ.NBLK) GO TO 6
0064 READ (NPGM,2) NRAY
0065 IF(NRAY(6).NE.NAST) GO TO 7
0066 K=7
0067 GO TO 12
0068 14 NELTS=0
0069 I=21
0070 N=1
0071 15 DO 16 J=1,6
0072 LELS(J,NELTS+1)=NBLK
0073 16 IF(NRAY(I)).EQ.NKCM) GO TO 18
0074 LELS(N,NELTS+1)=NRAY(I)
0075 17 IF(NRAY(I)).EQ.NBLK) GO TO 19
0076 N=N+1
0077 I=I+1
0078 18 IF(I.LE.72) GO TO 17
0079 READ (NPGM,2) NRAY
0080 IF(NRAY(I)).NE.NAST) GO TO 20
0081 I=7
0082 19 NELTS=NELTS+1
0083 XNELTS=NELTS
0084 I=I+1
0085 CC TO 18
OC88  IF(I.LE.72) GO TO '15  
C183  READ (NPGR,2) NRAY  
C190  IF(NRAY(6).NE.NAST) GO TO 7  
C191  I=7  
C192  GO TO 15  
C193  19  NELTS=NELTS+1  
C194  XNELTS=XNELTS  
C195  GO TO 6  
C196  20  NELTS=NELTS+1  
C197  XNELTS=XNELTS  
C198  GO TO 7  
C199  21  K=22  
C200  NHOL=NHOL+1  
C201  XNPD=XNPD+NHOL  
C202  22  IF(NRAY(NRAY(2),EQL.BHRK) GO TO '6  
C203  READ (NPGR,2) NRAY  
C204  IF(NRAY(6).NE.NAST) GO TO 7  
C205  K=7  
C206  GO TO 22  
C207  23  DO 24 I=22,70  
C208  IF(LCATE(NRAY,EQL.NPD,I,3).EQ.O) XNPD=XNPD+1  
C209  CONTINUE  
C210  GO TO 6  
C211  25  IF(NRAY(16).EQL.NRAY(9)) GO TO 6  
C212  K=19  
C213  26  IF(NRAY(K).EQL.NRAY(9)) GO TO 6  
C214  DO 27 I=2,10  
C215  IF(NRAY(K).EQL.NRAY(9)) GO TO 29  
C216  CONTINUE  
C217  27  WRITE (K,28) NRAY,K  
C218  IF(NRAY(6).EQL.NRAY(9)) GO TO 29  
C219  CONTINUE  
C220  28  FORMAT (42H *** ERROR *** - - INVALID NUMBER IN SIMTR.1/80A1.6H  
C221  10L=,13)  
C222  STCP 8  
C223  29  J=I-1  
C224  K=K+2  
C225  30  IF(K.GT.72) READ (NPGR,2) NRAY  
C226  IF(K.GT.72) K=K-72+6  
C227  31  IF(NRAY(K).EQL.NRAY(9)) GO TO 31  
C228  K=K+2  
C229  32  GC TO 30  
C230  KST=K  
C231  33  KEN=K+2  
C232  CRTOLS (LCJ(J))=NUME(NRAY,KST,KEJ,K)  
C233  34  K=KEN+5  
C234  IF(K.LE.72) GO TO 26  
C235  READ (NPGR,2) NRAY  
C236  IF(NRAY(6).NE.NAST) GO TO 7  
C237  K=K-72+6  

-113-
0135  GO TO 26
0136  RETURN
0137  WRITE (NPRT,34)
0138  FORMAT (39H *** ERROR *** - END OF FILE IN SIMTR)
0139  STOP 8
0140  WRITE (NPRT,36)
0141  FORMAT (40H *** ERROR *** - END OF FILE ON DTASAV)
0142  STOP 8
0143  END
SUBROUTINE TPHEAD
DUMMY TPHEAD ROLTINE
RETURN
END
SUBROUTINE XLOAD

C

C DUMMY XLOAD ROUTINE

C

RETURN

END
FUNCTION LOCATE (NRAY, NSTR, KSTR, KLEN)

LOCATE STRING (NSTR) IN ARRAY (NRAY) WITH
LENGTH (KLEN) STARTING AT (KSTR).
LOCATE = 1 IF STRING FOUND
LOCATE = 0 IF STRING NOT FOUND

DIMENSION NRAY(80), NSTR(80)

LOCATE = 1
DO 1, KLEN
KSTRIN = KSTR + I - 1
IF (NRAY(KSTRIN) NE NSTR(I)) RETURN
CONTINUE

LOCATE = 0
RETURN
END
FUNCTION NCNT (NRAY,NST)  

COUNT NUMBER OF COMMAS+1 IN ARRAY NRAY  
STARTING AT NST  

DO NOT COUNT COMMAS IN BLK!  

DIMENSION NRAY(80)  
DATA BLK/1H/,NCWF/1H/,/  
NCNT=1  
DO 1 I=NST,72  
1 IF(NRAY(I).EQ.NCWF) NCNT=NCNT+1  
IF(NRAY(I).EQ.NBLK) RETURN  
CONTINUE  
RETURN  
END
FUNCTION NUPB (NRAY,KST,KEN,I)

DECODE NUMBER AT KST IN NRAY UNITL KEN

DIMENSION NRAY(80), NUMS(10)

DATA NUMS(10)=1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/

COMMON /TAPES/ NPRT,NRTA,APGM

DATA NBLK/IH/

NUMB=0

I=KST,KEN

IF(I.LE.72) GO TO 2

READ (NPGM,1) NRAY

FORMAT (8GA1)

I=7

KEY=KEN-72+6

CONTINUE

IF(NRAY(I).EQ.NBLK) GO TO 4

DO 3 J=1,10

IF(NRAY(I).NE.NUMS(J)) GO TO 3

NUMB=NUMB+J-1

CONTINUE

RETURN

END
FUNCTION XTABLE (TABLE, X)

IMPLICIT REAL*8 (A-H,O-Z), INTEGER*4 (I-N)

FUNCTION TO PERFORM STRAIGHT LINE EXTRAPOLATION OF TABULAR DATA

DIMENSION TABLE(2)

LCP=TABLE(1)-2.

DO 4 INDX=3,LOOP,2

TEST LOWER COORDINATE

IF(TABLE(INCX)-X) 1,2,5

TEST UPPER COORDINATE

IF(TABLE(INDX)-X) 4,5,6

EXACT MATCH TO LOWER COORDINATE - EQUAT AND RETURN

Y=TABLE(INDX+1)

GO TO 6

EXACT MATCH TO UPPER COORDINATE - EQUAT AND RETURN

Y=TABLE(INDX+3)

GO TO 6

CONTINUE

INDX=LCP-1

PERFORM STRAIGHT LINE EXTRAPOLATION

SLOPE=(TABLE(INCX+3)-TABLE(INCX+1))/TABLE(INDX+2)-TABLE(INDX))

CALCULATE Y INTERCEPT

Y=TABLE(INDX+1)-SLOPE*TABLE(INDX)

CALCULATE DEPENDENT VARIABLE

Y=SLOPE*X+YINT

OUTPUT PERTINANT VARIABLES

XTABLE=Y

RETURN

END