This is to certify that the

thesis entitled

Sensitivity Calculations for Bond Graph
Models of Linear Resistive Systems

presented by

David Richard Reed

has been accepted towards fulfillment
of the requirements for

Master's degree in Mechanical Engineering

RC Rosenberg

Major professor

Date May 15, 1984

SENSITIVITY CALCULATIONS FOR BOND GRAPH MODELS OF LINEAR RESISTIVE SYSTEMS

By

David Richard Reed

A THESIS

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

MASTER OF SCIENCE

Department of Mechanical Engineering

1984

ABSTRACT

SENSITIVITY CALCULATIONS FOR BOND GRAPH MODELS OF LINEAR RESISTIVE SYSTEMS

By

David Richard Reed

Knowledge of the sensitivities of system response variables to a set of design parameters can be very useful in designing engineering systems. This thesis presents a method for calculating output sensitivities for linear resistive engineering systems modeled by bond graphs. A software module, written to interface with an existing bond graph processor, provides linear resistive system solutions and output sensitivities. The sensitivities are used to predict new solutions that result from parameter variations.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Ronald Rosenberg. His guidance, knowledge, and friendship have made this research possible.

I would also like to thank DuPont and Chevron for providing fellowship support during my graduate study at Michigan State University.

Finally, for her continuous love and support, I would like to thank my best friend and wife, Tamra.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Modern design of engineering systems often involves computer analysis of a mathematical model. Knowledge of the sensitivities of system response variables to a set of design parameters can greatly enhance the design process. The problem considered in this thesis is how to calculate the sensitivities of system outputs to resistive parameters for linear resistive engineering systems modeled by bond graphs [1]. Emphasis is placed upon efficient computation. Many practical system models fall into this class of systems.

In general, output sensitivities can be obtained by stepping the parameters over a range of values and calculating the associated changes in output. Although this may be acceptable for small systems with only a few parameters, the computational burden increases significantly for larger systems. Numerous complete system solutions would be necessary to find the sensitivity of a set of outputs to each parameter.

An alternative method for calculating output sensitivities for linear electrical resistive networks was given by Frank [2] and developed more fully by Calahan [3]. The method is based on a general network theorem by Tellegen and makes use of an adjoint network

(sensitivity model) approach. The adjoint network solution and the original network solution are used to calculate precisely output sensitivities without recourse to an analytic expression.

Adapting this adjoint system approach to bond graph models provided an efficient method for calculating output sensitivities. A computer software module called STATIC was written to interface with ENPORT-5 [4], an interactive bond graph processor for linear dynamic systems. The combined program will solve linear resistive systems with constant inputs, calculate output sensitivities, and use the sensitivities to predict new solutions due to one-port resistance parameter variations.

Chapter 2 describes the implementation and operation of STATIC and presents some examples. Chapter 3 illustrates a sample run through the STATIC module. Chapter 4 considers in detail the sensitivity calculation method and how it applies to bond graphs. Chapter 5 provides a summary of results and some suggestions for future work.

Chapter 2

THE STATIC MODULE

The STATIC module, when interfaced with ENPORT-5, allows solution of linear resistive systems modeled by bond graphs. Such systems contain no C or I elements. The ENPORT-5 program processes the bond graph through equation formulation. Program control is then transferred to the STATIC module for solution. The inputs are set and the outputs are computed. The effort and flow on each resistive port (R-port) is also available. Next the sensitivity of each system output to each one-port resistive parameter (R-parameter) may be calculated. Finally a prediction of new system outputs due to changes in one or more R-parameters may be made. The prediction makes use of the calculated output sensitivities to generate a linear approximation to the new solution.

2.1 Implementation and Operation

The calculation of sensitivities by the adjoint system method requires the computation of R-port flows for both the original system and the adjoint system (see Chapter 4). The structure of both systems is identical and is defined by the original model. ENPORT-5 will process the model to the following junction structure equations.

$$Di = S33 * Do + S34 * U \tag{2.1}$$

$$V = S43 * Do + S44 * U \tag{2.2}$$

$$Do = L * Di. \tag{2.3}$$

Di, Do, U, and V are vectors defined in Figure 1 and S33, S34, S43, S44, and L are matrices [5]. Equation (2.3) is substituted into (2.1) and the dissipation field (R-field) input Di is solved for as a function of U. The resulting equation is

$$Di = (I - S33*L)^{-1} * S34 * U. \tag{2.4}$$

The R-field output is computed using equation (2.3). The R-port efforts and flows are contained in Di and Do and are extracted using causality information. The junction structure output V is then computed using equation (2.2).

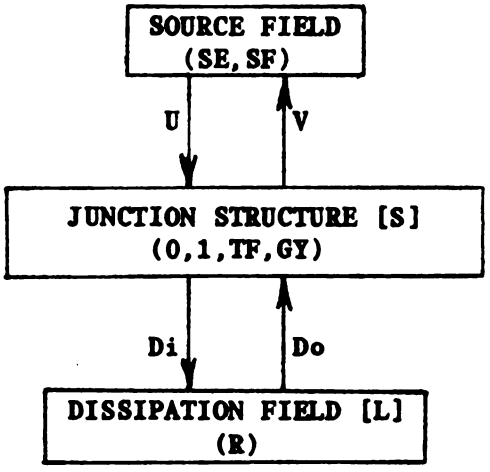


Figure 1. Structure of a Linear Resistive System

To calculate sensitivities, the adjoint system must now be solved. Since the structure is identical to the original system, equations (2.4) and (2.3) can be used with an adjoint input vector. All inputs are set to zero except the input corresponding to the desired output, which is

set to magnitude 1. The adjoint R-field vectors are computed and the adjoint R-port flows are extracted. The sensitivity of the desired system output to each one-port R-parameter is then calculated as the negative product of the R-port flows from the original and adjoint systems. Chapter 4 develops the the adjoint sensitivity method in some detail.

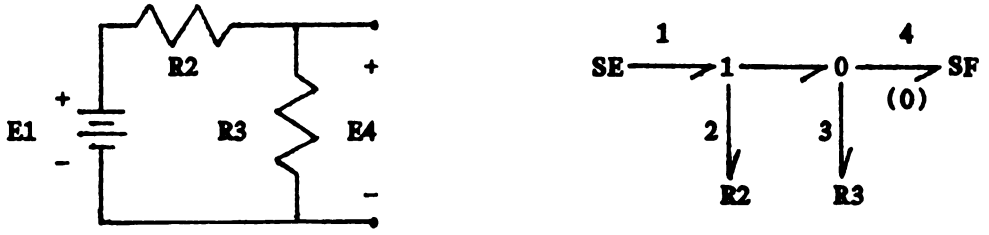
A prediction of a new solution can be made once the original solution, the output sensitivities, and a user-defined delta-R vector of changes are known. The following equation is used.

$$V(k,predicted) = V(k) + \sum ((dV(k)/dRi)*dRi)$$

where dV(k)/dRi is the sensitivity of output V(k) to parameter Ri and dRi is the change in resistance for parameter Ri. A dR value may be set for each one-port R-parameter and the above summation is taken over all i, where i = 1 to r, the number of one-port resistors.

2.2 Example 1 - Voltage Divider Circuit

Consider the linear resistive circuit of Figure 2 and its bond graph.



Circuit Diagram Bond Graph

Figure 2. Voltage Divider Circuit

Note that the zero flow source on port 4 of the bond graph represents the open circuit voltage output E4. The units are volts, amperes, and ohms. The inputs are E(1) = 10, F(4) = 0. The R-parameters are R2 = 25, R3 = 75. The bond graph was described to ENPORT-5 and processed. The STATIC module computed the system solution and calculated the sensitivities. The computer generated results are:

```
        SYSTEM INPUTS...              SYSTEM OUTPUTS...

     E(1) =  1.0000E+01          F(1) =  1.0000E-01
     F(4) =  0.0000E-01          E(4) =  7.5000E+00

                        R ELEMENTS...

     E(2) =  2.5000E+00          F(2) =  1.0000E-01
     E(3) =  7.5000E+00          F(3) =  1.0000E-01

           SENSITIVITIES...

     dF(1)/d(R2) = -1.0000E-03
     dF(1)/d(R3) = -1.0000E-03

     dE(4)/d(R2) = -7.5000E-02
     dE(4)/d(R3) =  2.5000E-02
```

For comparison, the analytical solution of the voltage divider circuit is:

OUTPUTS

$F(1) = E(1)/(R2 + R3) = 0.1$

$E(4) = E(1)*R3/(R2 + R3) = 7.5$

SENSITIVITIES

$dF(1)/dR2 = -E(1)/(R2 + R3)^2 = -0.001$

$dF(1)/dR3 = -E(1)/(R2 + R3)^2 = -0.001$

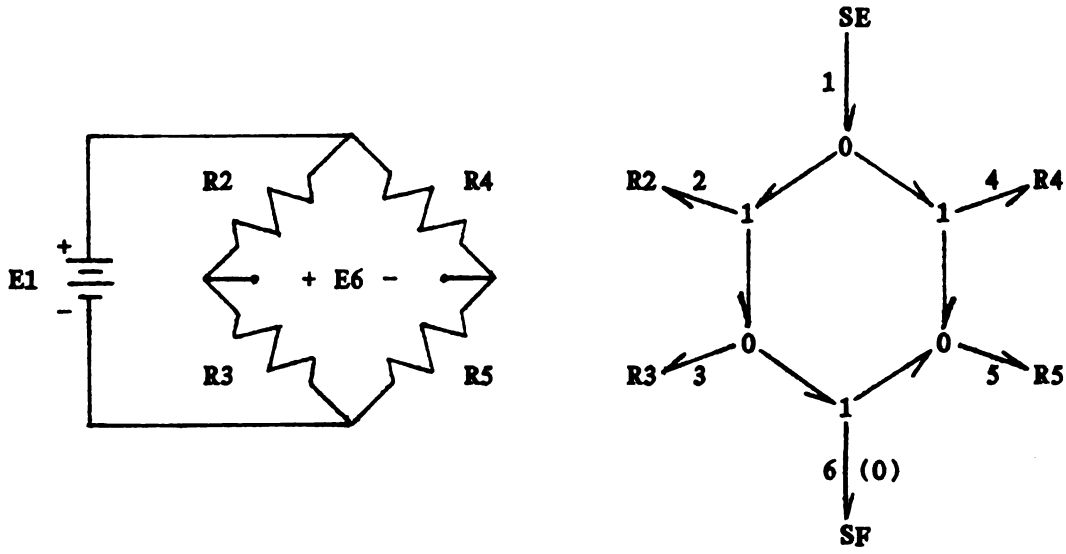
$dE(4)/dR2 = -E(1)*R3/(R2 + R3)^2 = -0.075$

$dE(4)/dR3 = E(1)*R2/(R2 + R3)^2 = 0.025$

The computer generated results match the analytical results.

2.3 Example 2 - Wheatstone Bridge Circuit

Consider the linear resistive circuit of Figure 3 and its bond graph. The inputs are E(1) = 6, F(6) = 0. The R-parameters are R2 = 20, R3 = 100 , R4 = 200 , R5 = 40. The bond graph was described to ENPORT-5 and processed.



Circuit Diagram Bond Graph

Figure 3. Wheatstone Bridge Circuit

The STATIC module computed the system solution and calculated the sensitivities. The computer generated solution is:

SYSTEM INPUTS...		SYSTEM OUTPUTS...	
E(1) =	6.0000E+00	F(1) =	7.5000E-02
F(6) =	0.0000E-01	E(6) =	4.0000E+00

R ELEMENTS...

E(2) =	1.0000E+00	F(2) =	5.0000E-02
E(3) =	5.0000E+00	F(3) =	5.0000E-02
E(4) =	5.0000E+00	F(4) =	2.5000E-02
E(5) =	1.0000E+00	F(5) =	2.5000E-02

SENSITIVITIES...

dF(1)/d(R2) = -4.1667E-04
dF(1)/d(R3) = -4.1667E-04
dF(1)/d(R4) = -1.0417E-04
dF(1)/d(R5) = -1.0417E-04

dE(6)/d(R2) = -4.1667E-02
dE(6)/d(R3) = 8.3333E-03
dE(6)/d(R4) = 4.1667E-03
dE(6)/d(R5) = -2.0833E-02

Suppose that the nominal value given for R4 was uncertain. Let the original value be reduced by 10%. New outputs can be predicted by the STATIC module. First the dR vector is set by entering -20 (a 10% reduction) for dR4. Next the new solution is predicted. The computer results are:

THE dR VECTOR...

d(R2) = 0.0000E-01
d(R3) = 0.0000E-01
d(R4) = -2.0000E+01
d(R5) = 0.0000E-01

LINEAR PREDICTION OF SYSTEM SOLUTION...

SYSTEM INPUTS...		SYSTEM OUTPUTS...	
E(1) =	6.0000E+00	F(1) =	7.7083E-02
F(6) =	0.0000E-01	E(6) =	3.9167E+00

For comparison, the system was re-solved completely with R4 = 180. The results are:

SYSTEM INPUTS...		SYSTEM OUTPUTS...	
E(1) =	6.0000E+00	F(1) =	7.7273E-02
F(6) =	0.0000E-01	E(6) =	3.9091E+00

The predicted solution nearly matches the actual solution but required fewer calculations and much less time to get. The percent error between the actual solution and the predicted solution is 0.25% for F(1) and 0.20% for E(6).

Chapter 3

STATIC MODULE SAMPLE RUN

This chapter illustrates a sample run through the STATIC module. Consider the bond graph model of Figure 4.

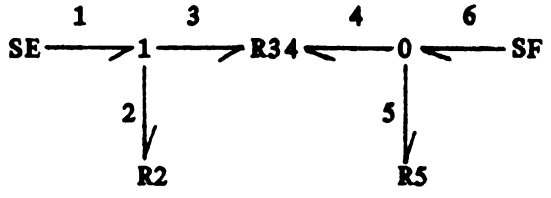


Figure 4. Sample Run System Model

The system parameters and defining constituitive equations are given below.

PARAMETERS	CONSTITUITIVE EQUATIONS
R2 = 250	E(2) = 250*F(2)
R34 = [350 200]	E(3) = 350*F(3) + 200*F(4)
[200 200]	E(4) = 200*F(3) + 200*F(4)
R5 = 200	E(5) = 200*F(5)

The following processing steps were performed in ENPORT-5.

(1) Input the bond graph
(2) Assign power directions
(3) Assign causality
(4) Set parameters
(5) Formulate the system equations

Program control was then transferred to the STATIC module for solution.

The next few pages show the transcript of the sample run through the STATIC module.

```
 STATIC SOLVER OPTIONS
-----------------------------------
 L: LIST INPUT VALUES
 M: MODIFY/SET INPUT VALUES
 G: GET THE SYSTEM SOLUTION
 D: DISPLAY SYSTEM SOLUTION
 C: CALCULATE SENSITIVITIES
 S: SHOW THE SENSITIVITIES
 P: PREDICT NEW SYSTEM SOLUTION
 H: HELP
 X: EXIT STATIC SOLVER (=DEFAULT)
-----------------------------------
 ENTER OPTION (X):M

    SET INPUTS...

ENTER E(1) ( 0.0000E-01):10
ENTER F(6) ( 0.0000E-01):

STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):G

COMPUTATION COMPLETED.

STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):D

   SYSTEM INPUTS...            SYSTEM OUTPUTS...

E(1) =  1.0000E+01          F(1) =  2.0000E-02
F(6) =  0.0000E-01          E(6) =  2.0000E+00

DISPLAY EFFORTS AND FLOWS ON R-PORTS? (Y):

                   R ELEMENTS...

E(2) =  5.0000E+00          F(2) =  2.0000E-02
E(3) =  5.0000E+00          F(3) =  2.0000E-02
E(4) =  2.0000E+00          F(4) = -1.0000E-02
E(5) =  2.0000E+00          F(5) =  1.0000E-02

STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):C

CALCULATION COMPLETED.
```

```
 STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):S

       SENSITIVITIES...

 dF(1)/d(R2) = -4.0000E-05
 dF(1)/d(R5) = -1.0000E-05

 dE(6)/d(R2) = -4.0000E-03
 dE(6)/d(R5) =  4.0000E-03

 SENSITIVITIES ASSOCIATED WITH MULTI-PORT R'S ARE NOT AVAILABLE.

 STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):P

  PREDICTOR OPTIONS
 ----------------------------------------
  L: LIST THE dR VECTOR
  M: MODIFY THE dR VECTOR
  S: SET ALL dR VALUES
  G: GET THE PREDICTED SYSTEM SOLUTION
  D: DISPLAY PREDICTED SYSTEM SOLUTION
  H: HELP
  X: EXIT PREDICTOR (=DEFAULT)
 ----------------------------------------
  ENTER OPTION (X):S

 ENTER d(R2) ( 0.0000E-01):20
 ENTER d(R5) ( 0.0000E-01):-20

 PREDICTOR OPTIONS (L,M,S,G,D,H,X,<FULL>):G

 COMPUTATION COMPLETED.

 PREDICTOR OPTIONS (L,M,S,G,D,H,X,<FULL>):D

       LINEAR PREDICTION OF SYSTEM SOLUTION...

    SYSTEM INPUTS...            SYSTEM OUTPUTS...

 E(1) =  1.0000E+01          F(1) =  1.9400E-02
 F(6) =  0.0000E-01          E(4) =  1.8400E+00

 PREDICTOR OPTIONS (L,M,S,G,D,H,X,<FULL>):X

 STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):X

 PROCEED? (Y):
```

At this point, program control was returned to ENPORT-5 and the sample run was ended.

Chapter 4

SENSITIVITY CALCULATION METHOD

The basis of the adjoint system method for calculating sensitivities is a general network theorem by Tellegen [6]. In bond graph terms, the theorem may be developed as follows.

Consider two bond graph models N and $\bar{N}$ which have the identical junction structure but possibly different types of field elements. The field elements are C, I, R, SE, and SF. Let (E(k),F(k)) and ($\bar{E}$(k),$\bar{F}$(k)) be the (effort,flow) on corresponding external ports (i.e., field ports of the junction structure) of N and $\bar{N}$. Further, let the corresponding power directions be the same. If n is the number of external ports on each junction structure, then Tellegen's theorem states that

$$\sum E(k)*\bar{F}(k) = 0 \quad \text{and} \quad \sum F(k)*\bar{E}(k) = 0, \text{ for } k = 1 \text{ to } n. \tag{4.1}$$

Now let the efforts and flows in N change by amounts dE(k) and dF(k). Tellegen's theorem must still be satisfied, so that

$$\sum (E(k)+dE(k))*\bar{F}(k) = 0 \quad \text{and} \quad \sum (F(k)+dF(k))*\bar{E}(k) = 0 \tag{4.2}$$

which requires

$$\sum dE(k)*\bar{F}(k) = 0 \quad \text{and} \quad \sum dF(k)*\bar{E}(k) = 0. \tag{4.3}$$

Equations (4.3) can be combined and written as

$$\sum (dE(k)*\bar{F}(k) - dF(k)*\bar{E}(k)) = 0. \tag{4.4}$$

Equation (4.4) provides the key to sensitivity calculations. If the

elements of $\bar{N}$ are chosen correctly, each term of (4.4) can be forced to zero except the terms from the output port and the varied parameter port. A sensitivity can be directly calculated from the non-zero terms.

As an example, consider the linear resistive system of Figure 5. The inputs are E(1) = constant, F(4) = zero. Assume the desired output is E(4) and the R-parameter to be varied is R2. The Tellegen sum (equation (4.4)) for each external port will be evaluated and the elements of $\bar{N}$ will be determined.

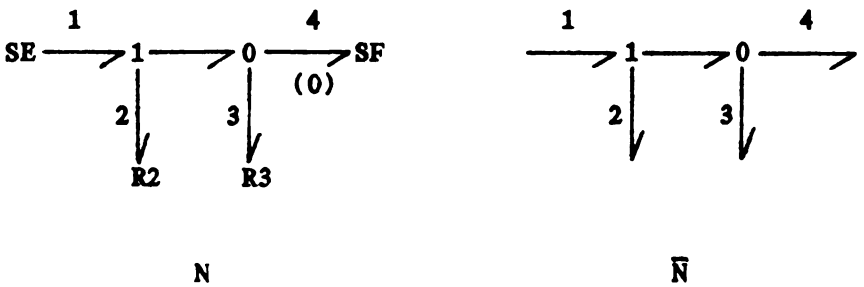


Figure 5. Linear Resistive System

For port 1 (k = 1), dE(1) = 0 because E(1) = constant. If $\bar{E}(1)$ is set to zero (implying a zero effort source on port 1 of $\bar{N}$), the Tellegen sum for port 1 becomes

$$(dE(1)*\bar{F}(1) - dF(1)*\bar{E}(1)) = 0.$$

For port 2 (k = 2), dR2 $\neq$ 0, so dE(2) = R2*dF(2) + F(2)*dR2. If R2 is duplicated in $\bar{N}$, cancellation results and the Tellegen sum for port 2 becomes

$$(dE(2)*\bar{F}(2) - dF(2)*\bar{E}(2)) = F(2)*\bar{F}(2)*dR2.$$

For port 3 (k = 3), dR3 = 0, so dE(3) = R3*dF(3). If R3 is duplicated in $\bar{N}$, cancellation results and the Tellegen sum for port 3 becomes

$$(dE(3)*\bar{F}(3) - dF(3)*\bar{E}(3)) = 0.$$

For port 4 (k = 4), dF(4) = 0 because F(4) = constant. If $\bar{F}(4)$ is set to 1 (implying a flow source on port 4 of $\bar{N}$), the Tellegen sum for port 4 becomes

$$(dE(4)*\bar{F}(4) - dF(4)*\bar{E}(4)) = dE(4)*(1).$$

Summing the terms of equation (4.4) for ports 1 through 4 results in

$$0 + F(2)*\bar{F}(2)*dR2 + 0 + dE(4) = 0.$$

Solving for the sensitivity,

$$dE(4)/dR2 = -F(2)*\bar{F}(2).$$

Similarly, if R2 is held constant and R3 is varied,

$$dE(4)/dR3 = -F(3)*\bar{F}(3).$$

Thus, for a linear resistive system with constant inputs, the sensitivity of an output to each one-port R-parameter can be calculated after solving for the R-port flows of N and $\bar{N}$. $\bar{N}$ is called the adjoint system.

In general, to construct the adjoint system for a linear resistive model, the following procedure is used. Note that the original bond graph model must contain a source corresponding to each desired output. (1) All R-elements in N are duplicated in $\bar{N}$ (including multi-port R-elements) and retain the identical resistance parameters. (2) All independent sources in N are duplicated in $\bar{N}$. The inputs are set to zero in $\bar{N}$. (3) If the output under consideration is an effort, then the flow for that port (which is an input) is set to 1 in $\bar{N}$. (4) If the

output under consideration is a flow, then the effort for that port (which is an input) is set to -1 in $\bar{N}$.

Thus, for a linear resistive model with one-port or multi-port resistors and independent effort or flow sources, the adjoint system $\bar{N}$ is nearly identical to the original system N. The only change is a different input vector U. Therefore, only a single reduction of the junction structure is necessary in order to solve both the original and adjoint systems.

For computer solution, the junction structure is reduced and Di and Do are computed given U using equations (2.4) and (2.3). The R-port flows are extracted from Di, Do. Next, the adjoint input vector $\bar{U}$ corresponding to the output under consideration is substituted and $\bar{D}i$ and $\bar{D}o$ are computed. The adjoint flows are extracted and the output sensitivities are calculated.

Chapter 5

SUMMARY

A sensitivity calculation method for network models based on an adjoint network was adapted and applied to bond graph models. This adjoint system method allows calculation of the sensitivities of system outputs to one-port resistive parameters for linear resistive systems with constant inputs. A software module, STATIC, was written to interface with ENPORT-5, a bond graph processor for linear dynamic systems. The STATIC module computes static system solutions, calculates output sensitivities and predicts new solutions due to variations in one or more resistive parameters.

5.1 Conclusion

For linear resistive bond graph models, the adjoint system approach provides an efficient method for computer calculation of sensitivities. Only one junction structure reduction is necessary to solve the original system and the one or more adjoint systems needed to calculate the sensitivities of system outputs to resistive parameters.

5.2 Future Work

Some suggestions for future work are:

(1) Develop a sensitivity calculation method for C-fields and I-fields. C-field problems have application in static deflection of compliant systems.

(2) Develop a sensitivity calculation method for linear dynamic systems modeled by bond graphs. Possible considerations are state variable sensitivity, eigenvalue sensitivity, and frequency response sensitivity.

(3) Make use of sensitivities in optimizing system parameters.

(4) Develop an efficient sensitivity calculation method for nonlinear systems.

LIST OF REFERENCES

LIST OF REFERENCES

1. Rosenberg, R. C. and Karnopp, D. C., INTRODUCTION TO PHYSICAL SYSTEM DYNAMICS, McGraw Hill, New York, 1983.

2. Frank, P. M., INTRODUCTION TO SYSTEM SENSITIVITY THEORY, Academic Press, New York, 1978.

3. Calahan, D. A., COMPUTER-AIDED NETWORK DESIGN, McGraw Hill, New York, 1972.

4. Rosenberg, R. C., 'ENPORT-5 User's Manual,' A. H. Case Center, College of Engineering, Michigan State University, 1981.

5. Karnopp, D. C. and Rosenberg, R. C., SYSTEM DYNAMICS: A UNIFIED APPROACH, Wiley, New York, 1975.

6. Penfield, P., Spence, R. and Duinker, S., TELLEGEN'S THEOREM AND ELECTRICAL NETWORKS, The M.I.T. Press, Cambridge, Massachusetts, 1970.

7. Karnopp, D. C., 'Power-conserving Transformations: Physical Interpretations and Applications using Bond Graphs,' Journal of The Franklin Institute, Volume 288, No. 3, September 1969, pp. 175-201.

8. Brayton, R. K. and Spence, R., SENSITIVITY AND OPTIMIZATION, Elsevier, New York, 1980.

APPENDICES

APPENDIX A

STATIC MODULE SUBROUTINES

The following is a list of subroutines in STATIC with a brief description of each.

STCALC calculates the sensitivity of each output to each one-port R-parameter.

STCOMP computes the effort and flow on each external bond.

STDRVR drives the STATIC module.

STINPT lists or modifies/sets the inputs.

STPRED predicts a new solution using the sensitivities and a user defined delta-R vector.

STRDUC reduces the junction structure matrices.

STSENS prints the sensitivities to the terminal screen.

STSGET gets the Sij sub-arrays from the S matrix.

STSOLN prints the system solution to the terminal screen.

The following is a calling tree for the subroutines in STATIC.

```
DRIVER (ENPORT)
      STDRVR
            GOON[1]
            STRDUC
                  CSMPY[2]
                  CSADD[2]
                  INPRD[2]
                  STSGET
            PROMPT[1]
            GETWD[1]
            STINPT
                  PROMPT[1]
                  GETRL[1]
                  NCHARS[3]
            STCOMP
                  CSMPY[2]
            STSOLN
                  NCHARS[3]
                  PROMPT[1]
                  YORN[1]
            STCALC
                  CSMPY[2]
            STSENS
                  NCHARS[3]
            STPRED
                  PROMPT[1]
                  GETWD[1]
                  NCHARS[3]
                  GETRL[1]
                  CSMPY[2]
                  CSADD[2]
            RHFILE[4]
            PROCED[1]
```

[1] Subroutine contained in IOUTIL (ENPORT)
[2] Subroutine contained in CSUTIL (ENPORT)
[3] Function contained in IOUTIL (ENPORT)
[4] Subroutine contained in UFILER (ENPORT)

APPENDIX B

STATIC MODULE DATA BASE

Many of the variables and arrays used in STATIC subroutines are shared with ENPORT-5 through common files. A list of the shared common files and the variables used from each is given below.

COMMON FILE	VARIABLES USED
SYBGBK	IELLST, NBIMX, IBMX, NPTR, IELNAM, IBNAM, NEL, NBD
CAUSBK	ICMX
CLASBK	NFL, NFS
RDUCBK	S, IPS, LS, T, IPT, LENT, WK, IWK, LENWK, FL, IPFL, LENFL, W2, IW2, LENW2
SOLNBK	MAXU, U
UTILBK	IERRF

In addition to the shared common files, the STATIC module also uses a local common file called STATBK. A listing of each STATIC common file is given in Appendix C.

APPENDIX C

STATIC MODULE SOURCE CODE

The common files and subroutines in this appendix appear in the following sequence:

1. SYBGBK
2. CAUSBK
3. CLASBK
4. RDUCBK
5. SOLNBK
6. UTILBK
7. STATBK
8. STDRVR
9. STRDUC
10. STSGET
11. STINPT
12. STCOMP
13. STSOLN
14. STCALC
15. STSENS
16. STPRED

The next several pages contain the source code for the STATIC module.

```
CSYBGBK*************************************************************C
C                                                                    C
C--- COMMON FILE SYBGBK: SYSTEM BONDGRAPH BLOCK                      C
C                                                                    C
C--- SUPPORTS GRAPDR, BOND GRAPH DATA STRUCTURE                      C
C                                                                    C
C--- VARIABLES:
C     IELLST  LIST OF NODES BY TYPE NUMBER
C     NBIMX   BONDS INCIDENT ON A GIVEN NODE
C     IBMX    NODES ADJACENT TO A GIVEN BOND
C     NPTR    POINTER TO START OF BONDS IN NBIMX FOR NODE I
C     IELNAM  NAMES OF NODES
C     IBNAM   NAMES OF BONDS
C
C--- NOTE: THIS BLOCK SHOULD BE COORDINATED WITH GREDBK, SINCE
C          THE ARRAYS MATCH AND THE DIMENSIONS SET BY PARAMETER
C          DECLARATIONS SHOULD TOO.
C                                                                    C
      PARAMETER (MNEL=100)
      PARAMETER (MNBD=100)
C
      CHARACTER*32 IELNAM(MNEL),IBNAM(MNBD)
C                                                                    C
      COMMON /SYBGB1/ NEL,NBD,IELLST(MNEL),NBIMX(MNBD*2),
     +                IBMX(MNBD,2),NPTR(MNEL+1),NMCR
      COMMON /SYBGB2/ IELNAM,IBNAM
C                                                                    C
CENDSYBGBK**********************************************************C
C
CCAUSBK*************************************************************C
C                                                                    C
C--- CAUSBK  SUPPORTS THE CAUSAL PROCESSSING OF THE GRAPH            C
C                                                                    C
C---   ICMX    CONTAINS THE CAUSAL INFORMATION.                      C
C             OTHER VARIABLES ARE DEFINED IN ASCAU.                  C
C                                                                    C
C---  THIS COMMON REQUIRES THE PRIOR USE OF $INSERT SYBGBK.
C                                                                    C
      PARAMETER (MNSTKM=100)
      PARAMETER (MNSTKN=100)
      PARAMETER (MAXBD=13)
      PARAMETER (MAXND=13)
C
       COMMON /CAUSBK/ ICMX(MNBD*2),MELMNT(MNSTKM),MBND(MNSTKM),
     +             MIT(MNSTKM),NTEMP(MNSTKN),
     +             MTOP,NTOP,LEVEL,IBLIS(MAXBD),
     +             ICCNOD(MAXND),IBPNT(1+MAXND/4),NODLIS(1+MAXND/4),
     +             JPNT,KBOND,INODE
C                                                                    C
CENDCAUSBK**********************************************************C
C
```

```
CCLASBK*************************************************************C
C                                                                   C
C--- CLASBK  SUPPORTS CLASS, DATA FOR BOND TYPE CLASSIFICATION      C
C            BY FIELD.                                              C
C                                                                   C
C--- IBEQ    BOND EQUIVALENT LIST, MAPS INTEGER NAME INTO           C
C            WORKING POSITION IN FIELD VECTOR.                      C
C    IBT     BOND TYPE LIST, DEFINES BOTH FIELD TYPE AND CAUSALITY. C
C            SEE CLASS FOR DETAILS OF FIELD TYPES.                  C
C    NBEX    NUMBER EXTERNAL (FIELD) BONDS                          C
C    NBIN    NUMBER INTERNAL (JUNCTION) BONDS                       C
C    NFI     NUMBER INDEPENDENT STORAGE BONDS (STATE)               C
C    NFD     NUMBER DEPENDENT STORAGE BONDS                         C
C    NFL     NUMBER DISSIPATION (R) BONDS                           C
C    NFS     NUMBER SOURCE BONDS                                    C
C    NFT     NUMBER TWO-PORT (TF,GY) BONDS                          C
C    NFJ     NUMBER OF (0,1) BONDS (SAME AS NBIN NOW)
C                                                                   C
C---  REQUIRES THE PRIOR USE OF SYBGBK FOR PARAMETERS
C
      COMMON /CLASBK/ IBEQ(MNBD),IBT(MNBD),NBEX,NFI,NFD,NFL,NFS,
     +                NBIN,NFT,NFJ
C                                                                   C
CENDCLASBK**********************************************************C
C
CRDUCBK*************************************************************C
C                                                                   C
C--- RDUCBK : COMMON FOR REDUCE AND CSUTIL OPERATIONS
C
C--- VARIABLES:
C      S        IS THE JUNCTION STRUCTURE ARRAY
C      T        IS USED AS A WORKSPACE
C      WK       IS A WORKSPACE ARRAY
C      FL       IS THE DISSIPATION FIELD MATRIX
C      FS       IS THE STORAGE FIELD MATRIX
C      TP       IS THE TF,GY MODULUS FIELD MATRIX
C      W2       IS A WORKSPACE ARRAY
C      A        IS THE SYSTEM 'A' MATRIX
C      B        IS THE SYSTEM 'B' MATRIX
C      E        IS THE SYSTEM 'E' MATRIX (DEPENDENT C,I)
C      LS       IS THE JS MATRIX BREAK POINT ARRAY
C
C--- NOTE. SPARSE ARRAY STORAGE: S(I) CONTAINS THE SDATA,
C          IPS(I) GIVES THE K POSITION (RUNNING ROW INDEX),
C          AND LENS IS TOTAL NUMBER OF NONZERO ITEMS IN S.
C          OTHER SPARSE ARRAYS ARE SIMILAR.
C
```

```
C--- PARAMETER DECLARATIONS:
C
      PARAMETER (MAXS=400)
      PARAMETER (MAXT=MAXS)
      PARAMETER (MAXWK=2*MAXS/3)
      PARAMETER (MAXFL=100)
      PARAMETER (MAXFS=100)
      PARAMETER (MAXTP=100)
      PARAMETER (MAXW2=MAXWK)
      PARAMETER (MAXA=250)
      PARAMETER (MAXB=100)
      PARAMETER (MAXE=100)
      PARAMETER (MAXC=100)
      PARAMETER (MAXD=50)
C
C--- COMMON DECLARATIONS:
C
      COMMON /BK8/ S(MAXS),IPS(MAXS),LENS,LS(16),
     +             T(MAXT),IPT(MAXT),LENT,
     +             WK(MAXWK),IWK(MAXWK),LENWK
      COMMON /BK9/ FL(MAXFL),IPFL(MAXFL),LENFL,
     +             FS(MAXFS),IPFS(MAXFS),LENFS,
     +             TP(MAXTP),IPTP(MAXTP),LENTP,
     +             W2(MAXW2),IW2(MAXW2),LENW2
      COMMON /BK10/ A(MAXA),IPA(MAXA),LENA,
     +              B(MAXB),IPB(MAXB),LENB,
     +              E(MAXE),IPE(MAXE),LENE
C                                                                      C
CENDRDUCBK*************************************************************C
C
CSOLNBK****************************************************************C
C                                                                      C
C---  SOLNBK  SUPPORTS SOLUTION PHASE WITH CONTROL DATA
C
C---  VARIABLES:
C           AMX     IS THE A MATRIX IN FULL STORAGE MODE
C           BMX     IS THE B MATRIX IN FSM
C           NX      IS THE NUMBER OF STATE VBLS
C           NU      IS THE NUMBER OF INPUTS
C           XIN     IS THE INITIAL CONDITION VECTOR
C           X       IS THE STATE VECTOR AT TIME T
C           U       IS THE INPUT VECTOR AT TIME T
C           DX      IS THE X-DOT VECTOR AT TIME T
C           TIN     IS THE INITIAL TIME
C           TFIN    IS THE FINAL TIME
C           NSAV    IS THE NUMBER OF STAGES TO STORE RESULTS (TOTAL)
C           DTSTR   IS THE DT STORAGE INTERVAL
C           RES     IS THE RESULTS ARRAY
C           XSAVL   IS LOGICAL SAVE LIST FOR X VARIABLES
C           USAVL   IS LOGICAL SAVE LIST FOR U VBLS
C           DSAVL   IS LOGICAL SAVE LIST FOR DX/DT VBLS
C           NREQ    IS NUMBER OF REQUESTS TO BE SAVED
C
```

```
      PARAMETER (MAXX=15)
      PARAMETER (MAXU=10)
      PARAMETER (MAXXUD=15)
      PARAMETER (MAXRES=501)
C
      LOGICAL XSAVL(MAXX),USAVL(MAXU),DSAVL(MAXX)
      INTEGER NREQ
C
      COMMON /ABMXBK/ AMX(MAXX,MAXX),BMX(MAXX,MAXU)
      COMMON /XVBLBK/ NX,XIN(MAXX),X(MAXX),DX(MAXX)
      COMMON /UVBLBK/ NU,U(MAXU)
      COMMON /TCTLBK/ TIN,TFIN,NSAV,DTSTR
      COMMON /RSLTBK/ RES(MAXRES,MAXXUD)
      COMMON /SAVEBK/XSAVL,USAVL,DSAVL,NREQ
C                                                                       C
CENDSOLNBK*************************************************************C
C
CUTILBK****************************************************************C
C                                                                       C
C--- UTILBK  GENERAL SUPPORT FUNCTIONS                                  C
C                                                                       C
C--- VARIABLES:
C              IERRF   ERROR RETURN VALUE
C              ABNEW   .TRUE. IF A,B ARE NEW TO SOLVIT
C              DUMPFG  SYSTEM LEVEL FLAG FOR DUMPING COMMON
C              ITERM   SPECIFIES KIND OF TERMINAL THE USER HAS
C              DYNFLG  IF TRUE WE HAVE A DYNAMIC SYSTEM
C              STTFLG  IF TRUE WE HAVE A STATIC SYSTEM
C
      LOGICAL ABNEW,DUMPFG,DYNFLG,STTFLG
C
      COMMON /UTILBK/IERRF,ABNEW,DUMPFG,ITERM,DYNFLG,STTFL
C                                                                       C
CENDUTILBK*************************************************************C
C
```

```
CSTATBK*****************************************************************C
C                                                                       C
C--- STATBK - LOCAL COMMON FOR THE ENPORT STATIC MODULE
C                                                                       C
C--- THIS COMMON REQUIRES THE PRIOR USE OF $INSERT SOLNBK.
C                                                                       C
C--- VARIABLES
C                                                                       C
C    MLTPRT - TRUE IF MULTI-PORT R IS PRESENT
C    POWRIN - TRUE FOR POWER INTO SYSTEM (MAXU*1)
C                                                                       C
C    MAXR   - MAXIMUM R-PORTS ALLOWED
C    MAXU   - MAXIMUM SOURCES ALLOWED (FROM SOLNBK)
C    INLST  - LIST OF NODES OF ONE-PORT R'S
C    IRLST  - LIST OF BONDS ON R'S W/CAUSALITY
C    ISLST  - LIST OF BONDS ON SOURCES W/CAUSALITY
C    IPDU   - POINTER FOR DI,DO,U,V VECTORS
C                                                                       C
C    DI     - R-FIELD INPUT VECTOR (Di)
C    DO     - R-FIELD OUTPUT VECTOR (Do)
C    EFRT   - EFFORT VECTOR (SORTED FROM Di AND Do)
C    FLOW   - FLOW VECTOR (SORTED FROM Di AND Do)
C    S4     - STORAGE FOR S44 MATRIX
C    SN     - SENSITIVITY ARRAY d(Vi)/d(Rj)
C    DR     - DELTA-R VECTOR
C    U      - SYSTEM INPUT VECTOR (FROM SOLNBK)
C    V      - SYSTEM OUTPUT VECTOR
C                                                                       C
C    NOTE:  MAXR MUST BE GREATER THAN MAXU.
C                                                                       C
      PARAMETER (MAXR=25)
      PARAMETER (MAXRU=MAXR*MAXU)
      PARAMETER (MAXUU=MAXU*MAXU)
C                                                                       C
      LOGICAL MLTPRT,POWRIN(MAXU)
C                                                                       C
      COMMON /STATB1/ MLTPRT,POWRIN
      COMMON /STATB2/ INLST(MAXR),IRLST(MAXR),ISLST(MAXU)
      COMMON /STATB3/ DI(MAXR),DO(MAXR),EFRT(MAXR),FLOW(MAXR),
     +                DR(MAXR),IPDU(MAXR),V(MAXU)
      COMMON /STATB4/ S4(MAXUU),IPS4(MAXUU),LENS4,
     +                SN(MAXRU),IPSN(MAXRU),LENSN
C                                                                       C
CENDSTATBK**************************************************************C
```

```
C    ENPORT STATIC MODULE
C                                                                      C
CSTDRVR >>>>>>>>>>>>>>>>>>>>>> STDRVR <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STDRVR(PROCFG)
C                                                                      C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C--- STDRVR DRIVES ENPORT'S STATIC MODULE.
C                                                                      C
C--- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT SOLNBK
$INSERT UTILBK
$INSERT STATBK
C                                                                      C
      LOGICAL PROCFG,FULL,NEWLIN,ENDLIN,SENSIT,
     +        SETFLG,SOLFLG,SENFLG
      CHARACTER*1 ANS
      CHARACTER*32 FNAME
      CHARACTER*70 STRING
C                                                                      C
C***STDRVR*************************************************************C
C                                                                      C
C--- CHECK NUMBER OF R'S AND U'S
      IF (NFL.GT.MAXR) THEN
        ANS='R'
        WRITE(*,9000)ANS,MAXR,ANS
        CALL GOON
        PROCFG=.FALSE.
        RETURN
      ENDIF
9000  FORMAT(/' *** TOO MANY',A1,'''S, MAXIMUM IS ',I3,
     +       ' NO FURTHER ACTION POSSIBLE.',
     +      /' *** WE SUGGEST INCREASING MAX',A1,'.')
      IF (NFS.GT.MAXU) THEN
        ANS='U'
        WRITE(*,9000)ANS,MAXU,ANS
        CALL GOON
        PROCFG=.FALSE.
        RETURN
      ENDIF
C
C--- INITIALIZE
      DO 10 I=1,MAXU
10    U(I)=0.0
      DO 20 I=1,MAXR
20    DR(I)=0.0
      SOLFLG=.FALSE.
      SENFLG=.FALSE.
      IERRF=0
      CALL STRDUC(SENSIT)
```

```
      IF (IERRF.NE.0) THEN
        WRITE(*,9001) IERRF
9001    FORMAT(/' *** ERROR NUMBER ',I6,' FOUND.',
     +         /5X,'NO FURTHER PROCESSING POSSIBLE.')
        CALL GOON
        PROCFG=.FALSE.
        RETURN
      ENDIF
C
C--- PRINT STATIC SOLVER OPTIONS
      FULL=.TRUE.
100   CONTINUE
      IF (FULL) THEN
        WRITE(*,1000)
1000  FORMAT(/'  STATIC SOLVER OPTIONS'
     +       /' ----------------------------------'
     +       /'  L: LIST INPUT VALUES'
     +       /'  M: MODIFY/SET INPUT VALUES'
     +       /'  G: GET THE SYSTEM SOLUTION'
     +       /'  D: DISPLAY SYSTEM SOLUTION'
     +       /'  C: CALCULATE SENSITIVITIES'
     +       /'  S: SHOW THE SENSITIVITIES'
     +       /'  P: PREDICT NEW SYSTEM SOLUTION'
     +       /'  H: HELP'
     +       /'  X: EXIT STATIC SOLVER (=DEFAULT)'
     +       /' ----------------------------------')
        STRING='  ENTER OPTION (X):'
      ELSE
        WRITE(*,'(1X)')
        STRING=' STATIC SOLVER OPTIONS (L,M,G,D,C,S,P,H,X,<FULL>):'
      ENDIF
C
C--- REQUEST AND GET SELECTED OPTION
      CALL PROMPT(STRING)
      NEWLIN=.TRUE.
      ANS='#'
      CALL GETWD(ANS,NEWLIN,ENDLIN)
C
C--- PROCESS SELECTED OPTION
      IF (ANS.EQ.'#') THEN
        IF (FULL) THEN
          ANS='X'
        ELSE
          FULL=.TRUE.
          GO TO 100
        ENDIF
      ENDIF
C
      IF (ANS.EQ.'L') THEN
        SETFLG=.FALSE.
        CALL STINPT(SETFLG)
      ELSEIF (ANS.EQ.'M') THEN
        SETFLG=.TRUE.
        CALL STINPT(SETFLG)
```

```
      ELSEIF (ANS.EQ.'G') THEN
        IERRF=0
        CALL STCOMP(SOLFLG)
        IF (IERRF.NE.0) THEN
          WRITE(*,9001) IERRF
        ENDIF
      ELSEIF (ANS.EQ.'D') THEN
        IF (SOLFLG) THEN
          CALL STSOLN
        ELSE
          WRITE(*,9010)
9010      FORMAT(/' YOU MUST GET THE SYSTEM SOLUTION',
     +           /' BEFORE DISPLAYING IT.')
        ENDIF
      ELSEIF (ANS.EQ.'C') THEN
        IF (SENSIT) THEN
          IF (SOLFLG) THEN
            IERRF=0
            CALL STCALC(SENFLG)
            IF (IERRF.NE.0) THEN
              WRITE(*,9001)IERRF
            ENDIF
          ELSE
            WRITE(*,9020)
9020        FORMAT(/' YOU MUST GET THE SYSTEM SOLUTION',
     +             /' BEFORE CALULATING SENSITIVIES.')
          ENDIF
        ELSE
          WRITE(*,9015)
9015      FORMAT(/' SENSITIVITIES ARE NOT AVAILABLE',
     +           /' DUE TO BOND ACTIVATION.')
        ENDIF
      ELSEIF (ANS.EQ.'S') THEN
        IF (SENFLG) THEN
          CALL STSENS
        ELSE
          WRITE(*,9030)
9030      FORMAT(/' YOU MUST CALCULATE THE SENSITIVITIES',
     +           /' BEFORE SHOWING THEM.')
        ENDIF
      ELSEIF (ANS.EQ.'P') THEN
        SENFLG) THEN
          CALL STPRED
        ELSE
          WRITE(*,9040)
9040      FORMAT(/' YOU MUST CALCULATE THE SENSITIVITIES',
     +           /' BEFORE PREDICTING A NEW SOLUTION.')
        ENDIF
      ELSEIF (ANS.EQ.'H') THEN
        FNAME='PUBLIC>E5.2>HELP.STATIC'
        CALL RHFILE(FNAME)
      ELSEIF (ANS.EQ.'X') THEN
        CALL PROCED(PROCFG)
        RETURN
```

```
      ELSE
        WRITE(*,*)' *** THIS IS NOT A VALID OPTION.'
      ENDIF
      FULL=.FALSE.
      GO TO 100
      END
C                                                                      C
CENDSTDRVR                                                             C
C                                                                      C
CSTRDUC >>>>>>>>>>>>>>>>>>>>>> STRDUC <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STRDUC(SENSIT)
C                                                                      C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C--- STRDUC REDUCES THE JUNCTION STRUCTURE EQUATIONS AND
C    SETS UP AN R NODE LIST, AN R BOND LIST, AND A
C    SOURCE BOND LIST FOR ENPORT'S STATIC MODULE.
C                                                                      C
C--- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CAUSBK
$INSERT CLASBK
$INSERT RDUCBK
$INSERT SOLNBK
$INSERT UTILBK
$INSERT STATBK
C
      LOGICAL SENSIT
      CHARACTER*40 STRING
C                                                                      C
C***STRDUC*************************************************************C
C                                                                      C
C--- SET UP INLST, IRLST, AND ISLST
C    + = EFFORT INTO ELEMENT (BOND LISTS)
C    - = FLOW   INTO ELEMENT (BOND LISTS)
      NR=1
      NS=1
      MLTPRT=.FALSE.
      DO 30 I=1,NEL
        IF (IELLST(I).EQ.3) THEN
C--- R ELEMENT
          NP=NPTR(I+1)-NPTR(I)
          IF (NP.EQ.1) THEN
            INLST(NR)=I
          ELSE
            DO 10 J=NR,NR+NP-1
10          INLST(J)=0
          ENDIF
          IF (NP.GT.1) MLTPRT=.TRUE.
          DO 20 J=1,NP
            J1=NPTR(I)+(J-1)
            IRLST(NR)=NBIMX(J1)
```

```
            IF ((ICMX(J1).EQ.6).OR.(ICMX(J1).EQ.5) THEN
              IRLST(NR)= -IRLST(NR)
            ENDIF
            NR=NR+1
20        CONTINUE
        ELSEIF ((IELLST(I).EQ.4).OR.(IELLST(I).EQ.5)) THEN
C--- SOURCE ELEMENT
          J=NPTR(I)
          K=NBIMX(J)
          IF (IBMX(K,1).EQ.I) THEN
            POWRIN(NS)=.TRUE.
          ELSE
            POWRIN(NS)=.FALSE.
          ENDIF
          IF (IELLST(I).EQ.4) K=-K
          ISLST(NS)= K
          NS=NS+1
        ENDIF
30    CONTINUE
C
C--- CHECK FOR ACTIVATED BONDS
      SENSIT=.TRUE.
      DO 40 I=1,2*NBD
        IF ((ICMX(I).EQ.2).OR.(ICMX(I).EQ.5)) THEN
          SENSIT=.FALSE.
        ENDIF
40    CONTINUE
C
C--- REDUCE THE JUNCTION STRUCTURE EQUATIONS
C
C--- SET UP AND COMPUTE {(INV(I-S33*L))*S34}
C
C--- PUT IDENTITY MATRIX (NFLxNFL) INTO T/IPT/LENT
      DO 50 NR=1,NFL
        T(NR)=1.0
50    IPT(NR)=(NR-1)*NFL+NR
      LENT=NFL
C
C--- MULTIPLY S33*L AND PUT -(S33*L) INTO W2/IW2/LENW2
      CALL CSMPY(NFL,NFL,LS(11),IPS,S,
     +                NFL,LENFL,IPFL,FL)
      IF (IERRF.NE.0) RETURN
      DO 60 I=1,LENWK
        W2(I)=-WK(I)
60    IW2(I)=IWK(I)
      LENW2=LENWK
C
C--- COMPUTE (I-S33*L) AND PUT INTO W2/IW2/LENW2
      CALL CSADD(LENT,IPT,T,LENW2,IW2,W2)
      IF (IERRF.NE.0) RETURN
      DO 70 I=1,LENWK
        W2(I)=WK(I)
70    IW2(I)=IWK(I)
      LENW2=LENWK
```

```
C
C--- EXTRACT S34 FROM S AND PUT INTO T/IPT/LENT
      IF (LS(12).GT.MAXT) THEN
        STRING=' *** T ARRAY TOO SMALL FOR S34'
        WRITE(*,9000)STRING(1:NCHARS(STRING))
9000    FORMAT(/,A,/' NO FURTHER ACTION POSSIBLE.')
        RETURN
      ENDIF
      L1=LS(11)+1
      L2=LS(11)+LS(12)
      CALL STSGET(T,IPT,S,IPS,L1,L2)
      LENT=LS(12)
C
C--- COMPUTE INV(I-S33*L)*S34 AND LEAVE IN T/IPT/LENT
      DET=0.0
      CALL INPRD(NFL,W2,IW2,LENW2,MAXW2,
     +           NFS,T,IPT,LENT,MAXT,DET)
      IF (IERRF.NE.0) RETURN
      IF (ABS(DET).LT.1.0E-10) THEN
        STRING=' *** DET(I-S33*L) = 0.0'
        WRITE(*,9000)STRING(1:NCHARS(STRING))
        RETURN
      ENDIF
C
C--- EXTRACT S43 FROM S AND PUT INTO W2/IW2/LENW2
      IF (LS(15).GT.MAXW2) THEN
        STRING=' *** W2 ARRAY TOO SMALL FOR S43'
        WRITE(*,9000)STRING(1:NCHARS(STRING))
        RETURN
      ENDIF
      L1=L2+1
      L2=L2+LS(15)
      CALL STSGET(W2,IW2,S,IPS,L1,L2)
      LENW2=LS(15)
C
C--- EXTRACT S44 FROM S AND PUT INTO S4/IPS4/LENS4
      IF (LS(16).GT.MAXUU) THEN
        STRING=' *** S4 TOO SMALL FOR S44'
        WRITE(*,9000)STRING(1:NCHARS(STRING))
        RETURN
      ENDIF
      L1=L2+1
      L2=L2+LS(16)
      CALL STSGET(S4,IPS4,S,IPS,L1,L2)
      LENS4=LS(16)
C
      RETURN
      END
C                                                                    C
CENDSTRDUC                                                           C
C                                                                    C
CSTSGET >>>>>>>>>>>>>>>>>>>>>> STSGET <<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                    C
      SUBROUTINE STSGET(SIJ,IPSIJ,S,IPS,L1,L2)
```

```
C                                                                       C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                       C
C   -STSGET EXTRACTS AN SIJ SUB-MATRIX FROM THE JUNCTION
C    STRUCTURE MATRIX S (ALL IN SPARSE FORM).
C                                                                       C
C--- DECLARATIONS
C                                                                       C
      DIMENSION SIJ(*),IPSIJ(*),S(*),IPS(*)
C                                                                       C
C***STSGET*************************************************************C
C                                                                       C
      DO 10 I=L1,L2
        SIJ(I-L1+1)=S(I)
        IPSIJ(I-L1+1)=IPS(I)
10    CONTINUE
      RETURN
      END
C                                                                       C
CENDSTSGET                                                              C
C                                                                       C
CSTINPT >>>>>>>>>>>>>>>>>>>>> STINPT <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                       C
      SUBROUTINE STINPT(SETFLG)
C                                                                       C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                       C
C--- STINPT MODIFIES/SETS OR LISTS THE CONSTANT INPUTS
C    FOR ENPORT'S STATIC MODULE.
C                                                                       C
C--- DECLARATIONS
C                                                                       C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT SOLNBK
$INSERT STATBK
C                                                                   C
      LOGICAL SETFLG,NEWLIN,ENDLIN
      CHARACTER*1 OTYPE
      CHARACTER*(8) IBOND
      CHARACTER*70 STRING
C                                                                       C
C***STINPT*************************************************************C
C                                                                       C
C--- SET/MODIFY OR LIST U (INPUTS)
      IF (SETFLG) THEN
        WRITE(*,1000)
1000    FORMAT(/'     SET INPUTS...'/)
      ELSE
        WRITE(*,1010)
1010    FORMAT(/'     INPUTS...'/)
      ENDIF
      RLO=-1.0E+10
      RHI= 1.0E+10
```

```
      NEWLIN=.TRUE.
      DO 100 NS=1,NFS
        IF (ISLST(NS).LT.0) THEN
          OTYPE='E'
        ELSE
          OTYPE='F'
        ENDIF
        IBOND=IBNAM(ABS(ISLST(NS)))
        IF (SETFLG) THEN
          WRITE(STRING,1020)OTYPE,IBOND(1:NCHARS(IBOND)),U(NS)
1020      FORMAT(' ENTER ',A1,'(',A,') (',1PE11.4,'):')
          CALL PROMPT(STRING)
          RVAL=U(NS)
          CALL GETRL(RVAL,RLO,RHI,NEWLIN,ENDLIN)
          U(NS)=RVAL
        ELSE
          WRITE(*,1030)OTYPE,IBOND(1:NCHARS(IBOND)),U(NS)
1030      FORMAT(1X,A1,'(',A,') = ',1PE11.4)
        ENDIF
100   CONTINUE
      RETURN
      END
C                                                                      C
CENDSTINPT                                                             C
C                                                                      C
CSTCOMP >>>>>>>>>>>>>>>>>>>>>> STCOMP <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STCOMP(SOLFLG)
C                                                                      C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C--- STCOMP COMPUTES ALL EXTERNAL BOND EFFORTS AND FLOWS FOR
C    A STATIC BOND GRAPH WITH CONSTANT INPUTS.
C                                                                      C
C--- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT RDUCBK
$INSERT SOLNBK
$INSERT UTILBK
$INSERT STATBK
C                                                                      C
      LOGICAL SOLFLG
C                                                                      C
C***STCOMP*************************************************************C
C                                                                      C
C--- INITIALIZE
      SOLFLG=.FALSE.
C
C--- COMPUTE EFFORTS AND FLOWS
C
C--- INITIALIZE DI,DO,EFRT,FLOW,IPDU
      DO 10 NR=1,MAXR
```

```
        DI(NR)=0.0
        DO(NR)=0.0
        EFRT(NR)=0.0
        FLOW(NR)=0.0
10      IPDU(NR)=NR
C
C--- COMPUTE Di = INV(I-S33*L)*S34*U
       J=1
       CALL CSMPY(NFL,NFS,LENT,IPT,T,
     +                  J,NFS,IPDU,U)
       IF (IERRF.NE.0) RETURN
       DO 20 I=1,LENWK
20     DI(IWK(I))=WK(I)
C
C--- COMPUTE Do=L*Di
       CALL CSMPY(NFL,NFL,LENFL,IPFL,FL,
     +                  J,NFL,IPDU,DI)
       IF (IERRF.NE.0) RETURN
       DO 30 I=1,LENWK
30     DO(IWK(I))=WK(I)
C
C--- COMPUTE S43*Do AND PUT INTO EFRT (TEMPORARILY)
       CALL CSMPY(NFS,NFL,LENW2,IW2,W2,
     +                  J,NFL,IPDU,DO)
       IF (IERRF.NE.0) RETURN
       DO 40 I=1,LENWK
40     EFRT(IWK(I))=WK(I)
C
C--- COMPUTE S44*U AND PUT INTO FLOW (TEMPORARILY)
       CALL CSMPY(NFS,NFS,LENS4,IPS4,S4,
     +                  J,NFS,IPDU,U)
       IF (IERRF.NE.0) RETURN
       DO 50 I=1,LENWK
50     FLOW(IWK(I))=WK(I)
C
C--- COMPUTE V=S43*Do+S44*U
       DO 60 NS=1,NFS
60     V(NS)=EFRT(NS)+FLOW(NS)
C
C--- SORT Di AND Do VECTORS INTO EFRT AND FLOW VECTORS
C
       DO 70 NR=1,NFL
         IF (IRLST(NR).GT.0) THEN
           EFRT(NR)=DI(NR)
           FLOW(NR)=DO(NR)
         ELSE
           EFRT(NR)=DO(NR)
           FLOW(NR)=DI(NR)
         ENDIF
70     CONTINUE
C
C--- NORMAL RETURN
       SOLFLG=.TRUE.
       WRITE(*,1000)
```

```
1000  FORMAT(/' COMPUTATION COMPLETED.')
      RETURN
      END
C                                                                      C
CENDSTCOMP                                                             C
C                                                                      C
CSTSOLN >>>>>>>>>>>>>>>>>>>>>> STSOLN <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STSOLN
C                                                                      C
C---- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C---- STSOLN DISPLAYS THE SYSTEM SOLUTION FOR
C     ENPORT'S STATIC MODULE.
C                                                                      C
C---- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT SOLNBK
$INSERT STATBK
C                                                                      C
      LOGICAL ISYES
      CHARACTER*1 ITYPE,OTYPE
      CHARACTER*8 IBOND
      CHARACTER*70 STRING
C                                                                      C
C***STSOLN*************************************************************C
C                                                                      C
C---- PRINT SYSTEM SOLUTION (RESULTS)
      WRITE(*,1000)
1000  FORMAT(/,4X,'SYSTEM INPUTS...',T28,4X,'SYSTEM OUTPUTS...'/)
      DO 200 NS=1,NFS
        IBOND=IBNAM(ABS(ISLST(NS)))
        IF (ISLST(NS).GT.0) THEN
          ITYPE='F'
          OTYPE='E'
        ELSE
          ITYPE='E'
          OTYPE='F'
        ENDIF
        WRITE(*,2000)ITYPE,IBOND(1:NCHARS(IBOND)),U(NS),
     +               OTYPE,IBOND(1:NCHARS(IBOND)),V(NS)
2000    FORMAT(1X,A1,'(',A,') = ',1PE11.4,
     +     T28,1X,A1,'(',A,') = ',1PE11.4)
200   CONTINUE
C
      STRING=' DISPLAY EFFORTS AND FLOWS ON R PORTS? (Y):'
      WRITE(*,'(1X)')
      CALL PROMPT(STRING)
      ISYES=.TRUE.
      CALL YORN(ISYES)
      IF (ISYES) THEN
        CONTINUE
```

```
      ELSE
        RETURN
      ENDIF
C---- PRINT EFFORTS AND FLOWS ON R ELEMENTS
      WRITE(*,3000)
3000  FORMAT(/,20X,'R ELEMENTS...'/)
      ITYPE='E'
      OTYPE='F'
      DO 300 NR=1,NFL
        IBOND=IBNAM(ABS(IRLST(NR)))
        WRITE(*,2000)ITYPE,IBOND(1:NCHARS(IBOND)),EFRT(NR),
     +               OTYPE,IBOND(1:NCHARS(IBOND)),FLOW(NR)
300   CONTINUE
      RETURN
      END
C                                                                      C
CENDSTSOLN                                                             C
C                                                                      C
CSTCALC >>>>>>>>>>>>>>>>>>>>>> STCALC <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STCALC(SENFLG)
C                                                                      C
C---- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C---- STCALC CALCULATES THE SENSITIVITY OF EACH OUTPUT TO EACH
C     ONE-PORT R FOR A STATIC BOND GRAPH WITH CONSTANT INPUTS.
C                                                                    C
C---- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT RDUCBK
$INSERT SOLNBK
$INSERT UTILBK
$INSERT STATBK
C                                                                      C
      LOGICAL SENFLG
      REAL UADJ(MAXU)
C                                                                      C
C***STCALC*************************************************************C
C                                                                      C
C---- INITIALIZE
      SENFLG=.FALSE.
      DO 10 NS=1,MAXU
10    UADJ(NS)=0.0
C
C---- CALCULATE SENSITIVITIES
C
      DO 100 NS=1,NFS
C---- SET INPUT VECTOR (UADJ)
        IF (ISLST(NS).GT.0) THEN
C---- FLOW SOURCE
          IF (POWRIN(NS)) THEN
            UADJ(NS)=-1.0
```

```
          ELSE
            UADJ(NS)= 1.0
          ENDIF
        ELSE
C--- EFFORT SOURCE
          IF (POWRIN(NS)) THEN
            UADJ(NS)= 1.0
          ELSE
            UADJ(NS)=-1.0
          ENDIF
        ENDIF
C
C--- ZERO Di AND Do
        DO 20 NR=1,MAXR
          DI(NR)=0.0
20      DO(NR)=0.0
C
C--- COMPUTE Di = INV(I-S33*L)*S34*UADJ
        J=1
        CALL CSMPY(NFL,NFS,LENT,IPT,T,
     +                     J,NFS,IPDU,UADJ)
        IF (IERRF.NE.0) RETURN
        DO 30 I=1,LENWK
30      DI(IWK(I))=WK(I)
C
C--- COMPUTE Do=L*Di
        CALL CSMPY(NFL,NFL,LENFL,IPFL,FL,
     +                     J,NFL,IPDU,DI)
        IF (IERRF.NE.0) RETURN
        DO 40 I=1,LENWK
40      DO(IWK(I))=WK(I)
C
C--- SORT DI AND DO, RETAIN ADJOINT FLOW ONLY
        DO 50 NR=1,NFL
          IF (IRLST(NR).GT.0) THEN
            FADJ=DO(NR)
          ELSE
            FADJ=DI(NR)
          ENDIF
C
C--- STORE SENSITIVITES
          SN((NS-1)*NFL+NR)=-1.0*FLOW(NR)*FADJ
          IPSN((NS-1)*NFL+NR)=(NS-1)*NFL+NR
50      CONTINUE
C
C--- RESET ADJOINT INPUT VECTOR
        UADJ(NS)=0.0
100   CONTINUE
      LENSN=NFS*NFL
C
C--- NORMAL RETURN
      SENFLG=.TRUE.
      WRITE(*,1000)
1000  FORMAT(/' CALCULATION COMPLETED.')
```

```
      RETURN
      END
C                                                                      C
CENDSTCALC                                                             C
C                                                                      C
CSTSENS >>>>>>>>>>>>>>>>>>>>>> STSENS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                      C
      SUBROUTINE STSENS
C                                                                      C
C---- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                      C
C---- STSENS SHOWS THE SENSITIVITIES FOR
C     ENPORT'S STATIC MODULE.
C                                                                      C
C---- DECLARATIONS
C                                                                      C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT SOLNBK
$INSERT STATBK
C                                                                      C
      CHARACTER*1 OTYPE
      CHARACTER*6 INODE
      CHARACTER*8 IBOND
C                                                                      C
C***STSENS*************************************************************C
C                                                                      C
C---- PRINT SENSITIVITIES
      WRITE(*,1000)
1000  FORMAT(/,7X,'SENSITIVITIES...')
      DO 200 NS=1,NFS
        IF (ISLST(NS).GT.0) THEN
          OTYPE='E'
        ELSE
          OTYPE='F'
        ENDIF
        IBOND=IBNAM(ABS(ISLST(NS)))
        WRITE(*,'(1X)')
        DO 100 NR=1,NFL
          IF (INLST(NR).EQ.0) GO TO 100
          INODE=IELNAM(INLST(NR))
          WRITE(*,2000)OTYPE,IBOND(1:NCHARS(IBOND)),
     +          INODE(1:NCHARS(INODE)),SN((NS-1)*NFL+NR)
2000      FORMAT(' d',A1,'(',A,')/d(',A,') = ',1PE11.4)
100     CONTINUE
200   CONTINUE
      IF (MLTPRT) WRITE(*,3000)
3000  FORMAT(/' SENSITIVITIES ASSOCIATED WITH MULTIPORT',
     +        ' R''S ARE NOT AVAILABLE.')
      RETURN
      END
C                                                                      C
CENDSTSENS                                                             C
C                                                                      C
```

```
CSTPRED >>>>>>>>>>>>>>>>>>>>>> STPRED <<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                  C
      SUBROUTINE STPRED
C                                                                  C
C--- PROGRAMMER:  D.R.REED (SPRING 1984)
C                                                                  C
C--- STPRED PREDICTS STATIC SYSTEM SOLUTIONS USING THE CALCULATED
C    SENSITIVITIES AND A USER SUPPLIED DELTA-R VECTOR.
C                                                                  C
C--- DECLARATIONS
C                                                                  C
$INSERT SYBGBK
$INSERT CLASBK
$INSERT RDUCBK
$INSERT SOLNBK
$INSERT UTILBK
$INSERT STATBK
C                                                                  C
      LOGICAL FULL,NEWLIN,ENDLIN,PREFLG
      CHARACTER*1 ANS,ITYPE,OTYPE
      CHARACTER*6 INODE
      CHARACTER*8 IBOND
      CHARACTER*70 STRING
      REAL DV(MAXU)
C                                                                  C
C***STPRED*********************************************************C
C                                                                  C
C--- INITIALIZE
      FULL=.TRUE.
      RLO=-1.0E+10
      RHI= 1.0E+10
      NEWLIN=.TRUE.
      DO 1 NS=1,MAXU
1     DV(NS)=0.0
      PREFLG=.FALSE.
C--- BE SURE THERE ARE ONE-PORT R ELEMENTS TO VARY
      DO 10 NR=1,NFL
10    IF (INLST(NR).NE.0) GO TO 100
C--- OOPS, NO ONE-PORT R'S TO VARY, RETURN TO STDRVR
      WRITE(*,9000)
9000  FORMAT(/' NO ONE-PORT R PARAMETERS TO VARY.')
      IF (MLTPRT) WRITE(*,9010)
9010  FORMAT(' VARYING OF MULTIPORT R''S IS NOT AVAILABLE.')
      RETURN
C
C--- PRINT PREDICTOR OPTIONS
100   CONTINUE
      IF (FULL) THEN
        WRITE(*,1000)
1000    FORMAT(/'  PREDICTOR OPTIONS'
     +         /' ---------------------------------------'
     +         /'  L: LIST THE dR VECTOR'
     +         /'  M: MODIFY THE dR VECTOR'
     +         /'  S: SET ALL dR VALUES'
```

```
     +          /'  G: GET THE PREDICTED SYSTEM SOLUTION'
     +          /'  D: DISPLAY PREDICTED SYSTEM SOLUTION'
     +          /'  H: HELP'
     +          /'  X: EXIT PREDICTOR (=DEFAULT)'
     +          /' ----------------------------------------')
        STRING='  ENTER OPTION (X):'
       ELSE
        WRITE(*,'(1X)')
        STRING=' PREDICTOR OPTIONS (L,M,S,G,D,H,X,<FULL>):'
       ENDIF
C
C--- REQUEST AND GET SELECTED OPTION
       CALL PROMPT(STRING)
       NEWLIN=.TRUE.
       ANS=' '
       CALL GETWD(ANS,NEWLIN,ENDLIN)
C
C--- PROCESS SELECTED OPTION
       IF (ANS.EQ.' ') THEN
         IF (FULL) THEN
           ANS='X'
         ELSE
           FULL=.TRUE.
           GO TO 100
         ENDIF
       ENDIF
C
       IF (ANS.EQ.'L') THEN
         WRITE(*,1010)
1010     FORMAT(/'    THE dR VECTOR...'/)
      O 110 NR=,NFL
           IF (INLST(NR).EQ.0) GO TO 110
           INODE=IELNAM(INLST(NR))
           WRITE(*,1020)INODE(1:NCHARS(INODE)),DR(NR)
1020       FORMAT(' d(',A,') = ',1PE11.4)
110      CONTINUE
C
       ELSEIF (ANS.EQ.'M') THEN
C--- GET R NODE NAME
120      INODE='QUIT'
         WRITE(*,'(1X)')
         STRING=' ENTER R NODE NAME (<RETURN> TO QUIT):'
         CALL PROMPT(STRING)
         CALL GETWD(INODE,NEWLIN,ENDLIN)
C--- PROCESS NODE NAME
         IF (INODE.NE.'QUIT') THEN
           DO 130 NR=1,NFL
             IF (INLST(NR).EQ.0) GO TO 130
             IF (INODE.EQ.IELNAM(INLST(NR))) GO TO 140
130        CONTINUE
           WRITE(*,9020)
9020       FORMAT(/' *** BAD NODE NAME - TRY AGAIN')
           GO TO 120
C--- GOOD NODE NAME, NOW GET dR
```

```
140         RVAL=DR(NR)
            WRITE(STRING,1030)INODE(1:NCHARS(INODE)),RVAL
1030        FORMAT(' ENTER d(',A,') (',1PE11.4,'):')
            CALL PROMPT(STRING)
            CALL GETRL(RVAL,RLO,RHI,NEWLIN,ENDLIN)
            DR(NR)=RVAL
            GO TO 120
          ENDIF
C
        ELSEIF (ANS.EQ.'S') THEN
          WRITE(*,'(1X)')
          DO 150 NR=1,NFL
            IF (INLST(NR).NE.0) THEN
              INODE=IELNAM(INLST(NR))
              RVAL=DR(NR)
              WRITE(STRING,1030)INODE(1:NCHARS(INODE)),RVAL
              CALL PROMPT(STRING)
              CALL GETRL(RVAL,RLO,RHI,NEWLIN,ENDLIN)
              DR(NR)=RVAL
            ENDIF
150       CONTINUE
C
        ELSEIF (ANS.EQ.'G') THEN
C--- GET PREDICTION
          DO 160 NS=1,MAXU
160       DV(NS)=0.0
          J=1
          IERRF=0
          CALL CSMPY(NFS,NFL,LENSN,IPSN,SN,
     +                       J,NFL,IPDU,DR)
          IF (IERRF.NE.0) THEN
            WRITE(*,9030) IERRF
9030        FORMAT(/' *** ERROR NUMBER ',I6,' FOUND.',
     +             '  NO FURTHER ACTION POSSIBLE.')
            RETURN
          ENDIF
          DO 170 I=1,LENWK
170       DV(IWK(I))=WK(I)
          CALL CSADD(NFS,IPDU,DV,NFS,IPDU,V)
          IF (IERRF.NE.0) THEN
            WRITE(*,9030) IERRF
            RETURN
          ENDIF
          DO 180 I=1,LENWK
180       DV(IWK(I))=WK(I)
          PREFLG=.TRUE.
          WRITE(*,1050)
1050      FORMAT(/' COMPUTATION COMPLETED.')
C
        ELSEIF (ANS.EQ.'D') THEN
C--- DISPLAY PREDICTION
          IF (PREFLG) THEN
            WRITE(*,1060)
1060        FORMAT(/6X,' LINEAR PREDICTION OF SYSTEM SOLUTION...'/
```

```
     +      /,4X,'SYSTEM INPUTS...',T28,4X,'SYSTEM OUTPUTS...'/)
            DO 190 NS=1,NFS
              IBOND=IBNAM(ABS(ISLST(NS)))
              IF (ISLST(NS).GT.0) THEN
                ITYPE='F'
                OTYPE='E'
              ELSE
                ITYPE='E'
                OTYPE='F'
              ENDIF
              WRITE(*,1070)ITYPE,IBOND(1:NCHARS(IBOND)),U(NS),
     +                     OTYPE,IBOND(1:NCHARS(IBOND)),DV(NS)
1070          FORMAT(1X,A1,'(',A,') = ',1PE11.4,
     +           T28,1X,A1,'(',A,') = ',1PE11.4)
190         CONTINUE
          ELSE
            WRITE(*,9040)
9040        FORMAT(/' YOU MUST GET THE PREDICTION',
     +             /' BEFORE DISPLAYING IT.')
          ENDIF
C
        ELSEIF (ANS.EQ.'H') THEN
          WRITE(*,1080)
1080      FORMAT(/' THE dR VECTOR CONTAINS dR FOR EACH ONE-PORT R.',
     +           /' EACH ENTRY IN THE dR VECTOR EQUALS THE CHANGE',
     +           /' IN R FOR THE ASSOCIATED ONE-PORT R ELEMENT.')
C
        ELSEIF (ANS.EQ.'X') THEN
          RETURN
        ELSE
          WRITE(*,*)' *** THIS IS NOT A VALID OPTION.'
        ENDIF
        FULL=.FALSE.
        GO TO 100
        END
C                                                                    C
CENDSTPRED                                                           C
C                                                                    C
CENDSTATIC                                                           C
```