General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
AN AUTOMATED PROCEDURE FOR CALCULATING SYSTEM MATRICES FROM PERTURBATION DATA GENERATED BY AN EAI PACER 100 HYBRID COMPUTER SYSTEM

by Edward J. Milner and Susan M. Krosel

Lewis Research Center
Cleveland, Ohio 44135
December 1977
Techniques are presented for determining the elements of the A, B, C, and D state variable matrices for systems simulated on an EAI Pacer 100 hybrid computer. An automated procedure systematically generates disturbance data necessary to linearize the simulation model and stores these data on a floppy disk. A separate digital program verifies this data, calculates the elements of the system matrices, and prints these matrices appropriately labeled. The partial derivatives forming the elements of the state variable matrices are approximated by finite difference calculations.
AN AUTOMATED PROCEDURE FOR CALCULATING SYSTEM MATRICES FROM
PERTURBATION DATA GENERATED BY AN EAI PACER 100 HYBRID
COMPUTER SYSTEM

by Edward J. Milner and Susan M. Krosel

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

Most control system design techniques assume having available a linear representation of the system to be controlled. Although obtaining a linearized model of a complex, non-linear system is usually tedious and time consuming, this task can be made less so by using the automated procedure described in this report.

Techniques are presented for determining the elements of the A, B, C, and D state variable matrices for systems simulated on an EAI Pacer 100 hybrid computer system. The techniques are generalized to allow use with many different simulation models. An automated procedure systematically generates disturbance data necessary to linearize the simulation model and stores these data on an auxiliary TEKTRONIX model 4922 floppy disk.

A separate digital program verifies the floppy disk data, allows for data modification, if necessary, calculates approximations to the partial derivatives forming the elements of the system matrices, and prints the matrices appropriately labeled.

Use of the floppy disk as a storage medium permits the data generation process on the hybrid computer to be separate from the matrix element calculation process.

INTRODUCTION

The intricate nature of many dynamic systems necessarily requires rather complex simulation models to represent performance over their entire operating range. One is likely to find the model to include a combination of linear and nonlinear algebraic and differential equations and maps. Although such a model is useful for predicting system performance over a wide range of operating conditions, generally, its nonlinear characteristics hinder it from being well-suited for control analyses. A basic assumption of most control design techniques is having available a linear representation of the system to be controlled.
By considering operation in the vicinity of a particular operating point, state variable techniques can be employed to obtain a linear approximation of the nonlinear system. Doing so, the system equations can be written in vector notation as:

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

where \( x \) is a vector of the system time-varying states, \( \dot{x} \) is a vector of the time rate of change of the system states, \( u \) is a vector of system inputs, and \( y \) is a vector of system outputs. Matrices \( A, B, C, \) and \( D \) are characterized by:

\[ a_{ij} = \frac{\partial \dot{x}_i}{\partial x_j} \]  
\[ b_{ij} = \frac{\partial \dot{x}_i}{\partial u_j} \]  
\[ c_{ij} = \frac{\partial y_i}{\partial x_j} \]  
\[ d_{ij} = \frac{\partial y_i}{\partial u_j} \]  

One linearization technique is to obtain finite difference approximations for the partial derivatives which comprise the elements of the system matrices. To generate the partial derivatives, the steady-state level of each state variable and system input is independently stepped in both a positive and negative direction while holding the other state variables and system inputs fixed. Corresponding values of the state variable derivatives and system outputs are recorded for each step change. By calculating the deviation of each state variable derivative and output from its steady-state value and by forming appropriate quotients the matrix elements are formed.

This report describes an automated technique for determining the elements of the \( A, B, C, \) and \( D \) state variable matrices for systems that are simulated on an EAI Pacer 100 hybrid computer system. The technique is generalized to allow its use with many different simulation models. The data taking process is initiated by manually setting a sense switch at the Pacer 100 control console. The order in which the variables are to be perturbed along with other appropriate data are input from cards. Parameter values are transmitted from
the analog computer to the digital computer by means of analog-to-digital converters (ADC's). Perturbation data generated on the Pacer 100 system are recorded and stored on a TEKTRONIX model 4922 floppy disk. The data are later transferred from the floppy disk to an IBM TSS/360 computer where the matrix calculations are performed. The matrix calculation is contained in the program MATCALC which can generate matrices for systems having an arbitrary maximum of 20 states, 10 inputs, and 20 outputs.

The following sections provide details of the apparatus, the digital program, the perturbation process, the floppy disk data format, and the matrix calculation procedure. An example is provided to illustrate the procedure.

APPARATUS

The data taking technique, described in this section, applies to nonlinear system simulations implemented on an EAI Pacer 100 hybrid computer system. The technique could, of course, be modified for other hybrid computer systems. At the NASA Lewis Research Center, the Pacer 100 system includes the Pacer digital computer with 32K of core, a TEKTRONIX model 4010 terminal with auxiliary TEKTRONIX model 4922 floppy disk unit, two model 681 analog computers, and an interface to allow communication between the computers. That interface consists of 48 analog-to-digital converters (ADC's) and 24 digital-to-analog converters (DAC's).

Analog Setup Requirements

It is assumed that the integrations associated with the system dynamics are performed by the integrators on one or both of the analog computers. The integrator outputs then correspond to the system state variables. The perturbation program requires that each state variable, state derivative, input variable, and output variable be transmitted from the analog computer to the digital computer through the interface. Hence, each parameter of interest must be connected to an ADC. For the purposes of this report, parameters formulated on the "primary" analog console are transmitted to the digital computer via a block of ADC's beginning with ADC 1 and parameters formulated on the "secondary" analog console are transmitted via a block of ADC's beginning with ADC 33.

Digital Program Description

Appendix A contains FORTRAN listings of the major parts of a typical simulation program which incorporates the necessary perturbation logic.

The program MAIN is designed to control the execution of the major parts of the program. As the execution of each part is
completed, control is returned to MAIN. The analog computer setup
and checkout takes place during the execution of subroutine SET.
This includes setting and checking analog pots, digitally controlled
attenuators (DCA's), and digitally controlled function generators
(DCFG's). Subroutine STATIC provides for the reading of all ampli-
 fier outputs at any desired operating point. Subroutine LOOP is the
heart of the simulation insofar as this is the area where the dynam-
ics of the system being simulated are updated. By continually
cycling through component maps and table lookups, current values of
the system state derivatives are obtained. In general, LOOP will re-
ceive information from the analog computer and output information to
the analog computer through the interface.

The perturbation program consists of three primary digital sub-
routines - DATOUT, SGNFIX, and AVG. The function served by sub-
routine DATOUT is to set the integrator initial condition DCA's to
correspond to the steady-state operating point being considered and
to systematically perturb the state variables and the system inputs
one at a time in both a positive and negative sense. DCA's are used
because they can be set by the digital computer without changing
modes on the analog computer. Subroutine SGNFIX is used to guaran-
tee that the positive algebraic sign of each parameter is recorded
on floppy disk. For example, consider a variable, x, whose range
spans the set of all real numbers. Hence, x may have both positive
and negative real values. However, the analog computer representa-
tion can be either the variable (+x) or the variable (-x) depending
on the sign of the reference voltage used. If (-x) were the analog
computer variable, subroutine SGNFIX converts it to (+x). For each
perturbation, subroutine AVG scans the parameters of interest form-
ing 100 point averages for each of them. These data are then re-
corded on floppy disk.

Formatting the output data and recording it on floppy disk is
accomplished in subroutines ADDOUT and DISK, respectively. The sole
purpose of subroutine DISK is to control the recording of the data
on floppy disk. To record on floppy disk, the floppy disk unit is
plugged into the disk interface card on the TEKTRONIX 4010 CRT term-
inal. The write statements in the program are the same as if the
data were being written on the CRT terminal. Data is then automa-
tically written on the CRT terminal and recorded on the floppy disk.
Note that everything written on the CRT terminal is also recorded
on the floppy disk.

Illustrative Example

Consider the following system of four uncoupled dynamic systems
linked through output relationships:
\[ ST81 = \int (\text{INPUT1} - \text{ST81}) \, dt \quad \text{where} \quad |\text{INPUT1}| \leq 2 \quad \text{and} \quad |\text{ST81}| \leq 2 \]  
(7)

\[ ST82 = \int (\text{INPUT2} - \text{ST82}) \, dt \quad \text{where} \quad |\text{INPUT2}| \leq 2.5 \quad \text{and} \quad |\text{ST82}| \leq 2.5 \]  
(8)

\[ ST83 = \int (\text{INPUT3} - \text{ST83}) \, dt \quad \text{where} \quad |\text{INPUT3}| \leq 5 \quad \text{and} \quad |\text{ST83}| \leq 5 \]  
(9)

\[ ST84 = \int (\text{INPUT4} - \text{ST84}) \, dt \quad \text{where} \quad |\text{INPUT4}| \leq 4 \quad \text{and} \quad |\text{ST84}| \leq 4 \]  
(10)

An analog diagram representing a typical system such as these is presented in Figure 1. Equations (7) and (8) are to be programmed on analog console 1. DCA 45 is to be used as the initial condition pot associated with ST81 and DCA 41 is to be used to set the perturbation amplitude of INPUT1. Likewise, DCA's 75 and 71 shall be used with ST82 and INPUT2, respectively.

Correspondingly, Eqs. (9) and (10) are to be programmed on analog console 2. However, on this console DCA's 75, 71, 45, and 41 shall be used with ST83, INPUT3, ST84, and INPUT4, respectively.

In all cases a 2 percent perturbation amplitude is deemed sufficient.

The solution of this example problem will be followed through in the discussion which follows.

**PROCEDURE**

The data taking process is activated by setting sense switch 4 and then exiting from the main system calculation subroutine (called LOOP) by setting sense switch 1. Control is then transferred to DATOUT. Hereafter, DATOUT controls reentry into LOOP. An internal check is made that the analog computer was in the operate mode at the time the data taking process was activated. Referring to Figure 2, a card is then read in (313) format giving the number of state variables, the number of system inputs, and the number of system outputs formulated on the primary analog console only. A series of cards punched in (F5.1, 13, S7) format are then read. Each card contains information characterizing one of the parameters of interest. For each state variable, the algebraic sign of the integrator output, the initial condition pot number, and the proportion of perturbation amplitude desired are required. For each system input, its algebraic sign, its perturbation pot number, and the proportion of perturbation amplitude desired are required. For each system output only its algebraic sign (as previously discussed in the explanation of subroutine SGNFIX) is required.
The next card read is a card in (3I3) format giving the number of state variables, the number of system inputs, and the number of system outputs formulated on the secondary analog console. Similarly, another series of cards in (F5.1, I3, S7) format are then read characterizing the parameters of interest on the secondary analog console as was just done for the primary console. Table I contains a listing of the cards used for the example and Figure 2 shows the data deck setup.

The order in which these data cards are read is very important. The parameter order determined by the cards must correspond with the parameter order determined by the analog ADC's used. State variable data is followed by system input data, which, in turn, is followed by system output data. Also, the order in which the parameters appear within each of the first two groups is important insofar as the order in which the cards are read determines the order in which the perturbations will occur. Note that the order of parameter perturbations is in itself arbitrary. Limitations on this order are imposed only by the analog console on which the integrator for a state variable or system input is located. As this procedure is defined, all state variables and system inputs on the primary analog console will be perturbed first. Then those on the second analog console will be perturbed. Table II gives the order in which the state variables and system inputs of the example are perturbed. Also, given is the analog console for each parameter and its associated ADC channel.

Following the reading of these data cards, the digital program proceeds to monitor the steady-state integrator outputs and sets DCA's corresponding to the integrator initial condition values. After the steady-state operating point is established, the analog computer is automatically put in IC mode where it remains during the perturbation and data taking process. The data taking process begins with a baseline reading of the steady-state values of the parameters of interest and the recording of them on floppy disk. The set of baseline data is designated as reading 1. Following the taking of this baseline data, each state variable and system input is stepped sequentially in both a positive and negative sense. Table III identifies the perturbations associated with each reading for the example.

For each perturbation, the data written on floppy disk are averages of the parameters of interest. The ADC's are read sequentially 100 times with the corresponding values summed and averaged. Since a parameter may be represented on the analog computer with either positive or negative algebraic sign, each average is multiplied by either (+1) or (-1), depending on the parameter sign. Thus, the data recorded on floppy disk corresponds to the positive algebraic sign of each parameter.

Data associated with each perturbation is recorded on floppy disk and each set of data on floppy disk is assigned its own identifying
reading number. Reading numbers are incremented as each set of data is recorded on floppy disk. Data is recorded on floppy disk according to the following format. The reading number is recorded twice in (215) format. The reason for this redundancy is so that it may be used as a check when reading the floppy disk. The parameter averages are recorded on floppy disk as integers (with 32767,10 representing full scale on the 32K Pacer) at the rate of 18 values per line together with a line checksum according to the format (1816,17). The checksum value is the sum of all the integers output on that particular line. If the number of averages exceeds 18 but is not evenly divisible by 18, the second last line of output will be the n remaining averages recorded in (n16) format and the final line will be the corresponding checksum in (17) format. If the number of averages is less than 18, the n averages will be recorded in (n16) format with a second line containing the checksum in (17) format. An example of the floppy disk data format is given in Table IV for the test case.

The final characters required on each line are a "line feed" and "X-off". This ending sequence is necessary to insure proper data transfer to the IBM TSS/360 computer. Because this is not the normal line termination, the contents of the digital computer core location controlling line termination must be changed. The address of this core location is obtained by taking the address of .FOR from the memory map (for the listing given in Appendix A, this address is 172348) and adding 2138. The address displacement value, 2138, depends on the version of the .FOR routine used. Version E01, December 1975 is in use on the Pacer 100 system at Lewis. After loading the program, this memory cell is opened and its contents set to zero by the user.

The entire procedure is repeated for each perturbation. When the data taking process is completed, an appropriate message is output on the line printer. At this point the system may be returned to normal operation by resetting sense switches 1 and 4.

MATRIX CALCULATION TECHNIQUE

This section describes a matrix calculation technique which uses the IBM TSS/360 computer with data input provided by the floppy disk. The matrix calculation is contained in the program MATCALC. This is a generalized version of a program developed at Lewis for matrix generation from data obtained from a hybrid simulation of the F100 engine.

The MATCALC program has three basic sections: data input, data verification, and matrix generation. The data input section uses a software checksum to insure that the floppy disk data was correctly transmitted to the IBM TSS/360 computer. After the data has been correctly transmitted, the data input section unscales it and stores this unscaled data in an internal program matrix for use in the other two program sections. The data verification section determines: (1) if only one state or input was perturbed for each reading and (2) if the
perturbation size of each state or input was at least 1% of the state's or input's steady-state value. The matrix generation section calculates the approximations to the partial derivatives forming the elements of the A, B, C, and D matrices and prints out the matrices in their final form. A listing of the MATCALC program is given in Appendix B.

Transfer of data from the floppy disk to the 360 computer is controlled by a procdef called READIN. A listing of this procdef is included in Appendix B and the procedure for use is given in Appendix D. The procdef, READIN creates the vs-dataset, 'DTRK' which contains the floppy disk data. The data on disk is a series of readings where each reading is a list of values of the states, state derivatives, inputs, and outputs. The required format of each reading was discussed in the preceding section. There should be 2n+1 readings on disk for each set of matrices, where n is equal to the number of states plus the number of inputs. The first reading is the 'base run' which is a reading of the unperturbed, steady-state values of the states, etc. (NOTE: The MATCALC program terminates if there is no 'base run'.) Corresponding to each reading is a disturbance number. The 'base run' must be assigned the disturbance number of 1. Since each state and input is perturbed in both a plus and minus direction, there are two readings associated with each state and input. These readings should be assigned disturbance numbers of m and m+1 where m is an even integer in the range 2 to 2n.

The readings contained in dataset, 'DTRK', are used by the data input section of MATCALC to form an internal program matrix, DATA. The rows of the DATA matrix are the readings in the same order as they appeared in the floppy disk data and in dataset 'DTRK'. Each row contains the unscaled values of the corresponding reading (i.e., the nth row contains the unscaled values in the nth reading). The order of the values is the same for the reading and the row. Table V lists the DATA matrix for the example.

The rows of the DATA matrix are used in the data verification and the matrix generation sections of MATCALC. The INDX array, also formed in the data input section, is used in referencing individual rows of the DATA matrix. Table V also includes the INDX array for the example. This referencing is accomplished by:

Row No. of DATA matrix = INDX (DISTURBANCE NO.)

If the Row No. is zero, the reading data associated with that disturbance number was not contained in the floppy disk data (i.e., the data for a plus or minus perturbation of a particular state or input was not recorded).

The NAME array associates each state or input with two disturbance numbers: one for the plus perturbation and the other for the minus
perturbation. This array contains the names of the state variables and the system inputs in the order in which they are perturbed (see Table II). The data for the NAME array is contained in the input dataset, 'QINFO'. 'QINFO' is a vs-dataset created by the user and contains data describing the contents of the floppy disk and specifying the structure of the A, B, C, and D matrices. The format of 'QINFO' is given in Appendix C and Table VI lists the 'QINFO' dataset for the example.

Appendix D contains a detailed procedure for processing of perturbation data which has been stored on a floppy disk. Instructions for operating the TEKTRONIX 4010 CRT terminal and TEKTRONIX 4922 floppy disk drive unit are included. IBM TSS/360 commands for reading the floppy disk data into the IBM TSS/360 computer and for performing the matrix calculations are provided. Also in Appendix D is the procedure to follow should checksum errors occur.

Table VII shows the resulting matrix listing for the example. As the results show, the simulated system consisted of four uncoupled dynamic systems linked only through the output relationships. The linearized system equations are:

\[
\begin{align*}
\dot{x}_1 &= -0.9938 \, x_1 + 0.9922 \, u_1 \\
\dot{x}_2 &= -0.9918 \, x_2 + 0.9888 \, u_2 \\
\dot{x}_3 &= -0.9939 \, x_3 + 0.9878 \, u_3 \\
\dot{x}_4 &= -0.9923 \, x_4 + 0.9861 \, u_4 \\
y_1 &= 1.003 \, x_1 - 0.001022 \, x_3 \\
y_2 &= -0.004666 \, x_1 + 1.002 \, x_2 + 0.001022 \, x_3 + 0.001019 \, u_3 \\
y_3 &= 0.999 \, x_3 - 0.001546 \, x_4 \\
y_4 &= 1.002 \, x_4
\end{align*}
\]
Program Considerations

Some modification of the perturbation program would be necessary if one of the state variables or inputs were at a limiting value while perturbation data were being taken. Because of that limit, that parameter could not be perturbed in both a positive and negative sense. Hence, proper logic would have to be included in subroutine DATOUT to restrict the perturbation of that parameter to the direction which would not violate the limiting value. Likewise, if the problem size were such that only one analog console were required, subroutine DATOUT could be easily modified to eliminate parallel operations addressed to the secondary analog console.

Use of a separate digital computer (the IBM TSS/360 computer) allows for the processing of the perturbation data and analysis of the resulting matrices to be done without interfering with use of the hybrid computer system. Calculation of the system matrices could have been done on the Pacer digital computer with the perturbation data stored on the Pacer's associated disk.

The format specified for data output to disk is arbitrary and can be changed if desired. The DISK subroutine of the perturbation program and the data input section of the matrix calculation program should be modified for the desired format. A new checksum error check should be developed to insure correct data transfer from disk.

CONCLUDING REMARKS

The A, B, C, and D state variable matrices used in controls design can be readily determined using the procedure described in this report. Data required to obtain a linear representation of the system to be controlled are systematically gathered using the techniques presented in the text. Disturbances are introduced in the nonlinear system one at a time. The user may determine the level of each disturbance and the sequential order in which each is to be introduced. For each disturbance, corresponding levels of essential system parameters are automatically recorded on floppy disk.

Use of the floppy disk as a storage medium permits the data generation process on the hybrid computer to be separate from the matrix calculation process. This floppy disk data is input to a separate digital program which contains the matrix calculation process. A software checksum is used to insure correct data transmission from the floppy disk. This program also checks the data for proper disturbance levels and correct number of disturbances. Appropriate messages are output if errors are detected, thus allowing data modification, if desired. The partial derivatives forming the elements of the system matrices are approximated by finite difference calculations. The system matrices, appropriately labeled, are printed out by this program.
Use of this procedure on a dynamic system simulated on a hybrid computer generates automatically the data required for linearization of the system and determines the matrices which represent the system.
APPENDIX A

SIMULATION AND PERTURBATION PROGRAMS

Main Program

C
C..LMN - ANALOG COMPUTER MODE INDICATOR
C
COMMON /LMN/ LMN
COMMON FLDSCL,E,1XOFF
LOGICAL HRED,
CALL GWAYIN (HRED,661,681)
C..TAKE ADDRESS OF FOR H65 AND 721X SET CONTENTS TO ZERO.
LF = 212
1XOFF = 223
CALL SET
CALL STATIC
10 CALL USE C0,1ERROR
CALL USE C1,1ERROR
CALL LOOP
IF C5(N64C4)) GO TO 36
CALL STATIC
20 IF CLMN EO 40 CALL LSENP CERROR)
GO TO 10
30 CALL EVENT (CLMN)
WRITE (16,61)
61 FORMAT (25D5,P35SP35,THEN P35P35CD2)
LA LF
OCT 3601
LA DF
OCT 3601
40 IF C5(N64C4)) GO TO 48
GO TO 20
END

SUBROUTINE SET
C
C ENTER YOUR PUTS, DC05, AND
C DC05 TO BE SET AND CHECKED
C
RETURN
END
SUBROUTINE STATIC
COMMON/HDISP/,LP,LDUFF
C
ENTER YOUR AMP DIODES TO BE READ AND CHECKED
C
WRITE (16,51)
61 FORMAT (26G24)YOU JUST EXECUTED STATIC/
LA LP
OCT 5000
LA 1DUFF
OCT 5001
RETURN
END

SUBROUTINE LOOP
C
C......LIM = ANALOG COMPUTER MODE INDICATOR
C
COMMON/CMODE/,LIM
LOGICAL LIM
10 CONTINUE
C
ENTER SYSTEM DYNAMIC EQUATIONS INCLUDING
BIVARIATE MAPS AND THREE LOOK-UPS
C
IF (LIM .NE. 1) GO TO 10
CALL DSC (2, LIM)
CALL DSC (1, LIM)
IF (LIM .EQ. 4) CALL DSC (KNN)\nRETURN
END

SUBROUTINE DATION (MODE)
C
C......NLIST = NUMBER OF STATE VARIABLES ON ANALOG CONSOLE 1
C......NILIN = NUMBER OF INPUTS ON ANALOG CONSOLE 1
C......NOUT = NUMBER OF OUTPUTS ON ANALOG CONSOLE 1
C......N2ST = NUMBER OF STATE VARIABLES ON ANALOG CONSOLE 2
C......N2LIN = NUMBER OF INPUTS ON ANALOG CONSOLE 2
C......N2OUT = NUMBER OF OUTPUTS ON ANALOG CONSOLE 2
COMMON/NIL10, N110, R10, R10K, R100, R1000, R10000, R100000, R1000000,

C DIMENSION N10R(20), N11R(20)

C SCALED FRACTION, SCALE(20), SCALE(20), SV1AMP(20), SV2AMP(20).

1

C.. CHECK THAT COMPUTER IS IN OPERATE

IF (IPRINT EQ 4) GO TO 10

WRITE (6, 61)

61 FORMH (200, 62) COMPUTERS ARE NOT IN OPERATE! THEN "NICE" WAS SET.

1 GO TO MAIN(7)

LA 10

LA 100

C.. CONSOLE 1 STATE VARIABLES, SYSTEM INPUTS, AND SYSTEM OUTPUTS

10 READ (6, 62) N1ST, N1IN, N1OUT

62 FORMH (211)

N1TOT = N1ST + N110R(N1ST)

N1TOT = N1TOT + N1IN(N1ST.

READ (6, 63) (CMPIFRED(CMP2NC(1), N12DC(1), SCALE(1)), 1-1, N1ST)

C.. READ STROBE STATE VALUES OF INTEGRATORS

CALL UBRADS (SV1AMP, 1, N1ST, 1, 1)

C.. CHANGE SIGN TO CORRESPOND TO DCR VALUES

DO 20 1 = 1, N1ST

SV1AMP(1) = -SV1AMP(1)

20 CONTINUE

CALL UBRADS (SV2AMP, 2, N1ST, 1, 1)

C.. CHANGE SIGN TO CORRESPOND TO DCR VALUES

DO 30 1 = 1, N1ST

SV2AMP(1) = -SV2AMP(1)

30 CONTINUE

C.. SET INTEGRATOR LCS (DCS) TO STROBE-STATE VALUES

CALL UDCS (1, 1, 1)

CALL UDCS (1, 1, 1)

CALL UNDCS (1, N1ST, N1ST, 1, 1)

CALL UDCS (1, 1, 1, 1)

CALL UDCS (1, 1, 1)

CALL UNDCS (N12DC, SV2AMP, N1ST, 1, 1)

C.. PUT COMPUTERS IN 1C

CALL UDCS (1, 1, 1)

CALL UDCS (1, 1, 1)

NIMPS = 100
IRDG = 1
CALL AVG (NHMP1S)
C........ INPUT VALUES
DO 40 J = 1,NN1J
J = J+1
K = N+JST1
SVMP(I,J) = NMPK(I,J)*NHMP1S(J)
40 CONTINUE
DO 50 J = 1,NN1J
J = J+1
K = N+JST1
SVMP(I,J) = NMPK(I,J)*NHMP1S(J)
50 CONTINUE
C........ PERTURB CONSOLE 1 VARIABLES
CALL USG (G, 1, ERR0)
CALL USG (G, 1, ERR0)
NMPK = NISTK
DO 60 J = 1,NMPK
60 CONTINUE
C........ PERTURB VARIABLE POSITIVE
SVMP = SVMP(I,J)*GMP1S(I,J)*SVMP(J)
C........ CHECK IF INPUT VARIABLE
IF GLOT RESTK, SVMP = GLOTRESTK*SVMP(J)
CALL USG (G, 1, ERR0)
CALL USG (G, 1, ERR0)
C........ PERTURB VARIABLE NEGATIVE
SVMP = SVMP(I,J)*GMP1S(I,J)*SVMP(J)
C........ CHECK IF INPUT VARIABLE
IF GLOT RESTK, SVMP = -GLOTRESTK*SVMP(J)
CALL USG (G, 1, ERR0)
CALL USG (G, 1, ERR0)
C........ RESTORE VARIABLE TO ORIGINAL VALUE
SVMP = SVMP(J)
C........ CHECK IF INPUT VARIABLE
IF GLOT RESTK, SVMP = .05
CALL USG (G, 1, ERR0)
CALL USG (G, 1, ERR0)
60 CONTINUE
C........ PERTURB CONSOLE 2 VARIABLES
CALL USG (G, 1, ERR0)
CALL USG (G, 1, ERR0)
NMPK = NISTK
DO 70 J = 1,NMPK
70 CONTINUE
C........ PERTURB VARIABLE POSITIVE
SVMP = SVMP(I,J)*GMP1S(I,J)*SVMP(J)
C. CHECK IF INPUT VARIABLE
   IF (LGT.NESTRO) VALUE = SIGN(1)*SVAMP(C)
   CALL UYDCS (N2DCSC), VALUE, I, TERROR
   CALL UYDC (C, I, TERROR)
   CALL LOOP
   CALL MVG (NUMPTS)
   CALL UYDC (C, I, TERROR)
   CALL UYDC (C, I, TERROR)
C. FERTI00K VARIABLE NEGETIVE
   VALUE = SVAMP(C)+SIGN(1)+SVAMP(C)
C. CHECK IF INPUT VARIABLE
   IF (LGT.NESTRO) VALUE = -SIGN(1)+SVAMP(C)
   CALL UYDCS (N2DCSC), VALUE, I, TERROR
   CALL UYDC (C, I, TERROR)
   CALL LOOP
   CALL MVG (NUMPTS)
   CALL UYDC (C, I, TERROR)
   CALL UYDC (C, I, TERROR)
C. RESET VARIABLE TO ORIGINAL VALUE
   VALUE = SVAMP(C)
C. CHECK IF INPUT VARIABLE
   IF (LGT.NESTRO) VALUE = 0
   RESULT (32064, 32064, PERTURBATION FOR COMPLETED)
   WRITE (15, 660)
64 FORMAT (32064, 32064, PERTURBATION FOR COMPLETED)
   CONTINUE
   WRITE (15, 660)
   CONTINUE
   WRITE (15, 660)
   CONTINUE
   RETURN

SUBROUTINE NORLX
C. NITOP = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C. ON ANALOG CONSOLE 1
C. N2TOT = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C. ON ANALOG CONSOLE 2
C. COMMON /MEAN/NETERS, N2TOT, N2TOT, UYDCS (C), AMP1SNF(C),
   I
   N2SNF, N2SNF, UYDCS (C), AMP1SNF(C), I
C. CONSOLE 1 VARIABLES
   DO 30 I = 1, NITOP
   IF (LGT.NESTRO) GO TO 30
   AVROI (C) = AMROI (C) + AMP1SNF (C)
30  GO TO 30

10 IF (I.GT.(2+M2ST8)) GO TO 20
   J = I-M2ST8
   AVRG1(I) = -AVRG(1)+M2P-1SN(J)
   GO TO 30
20 J = I-M2ST8
   AVRG(1) = -AVRG(1)+M2P+1SN(J)
   30 CONTINUE
C. CONSOLE 2 VARIABLES
   DO 60 J=1,N2TOPF
      IF (J.GT.(2+M2ST5)) GO TO 60
      AVRG3(J) = -AVRG2(J)+M2PP2SN(J)
      GO TO 60
60 IF (J.GT.(2+M2ST5)) GO TO 60
   J = I-M2ST8
   AVRG2(J) = -AVRG1(J)+M2PP2SN(J)
   GO TO 50
50 J = I-M2ST8
   AVRG2(J) = -AVRG2(J)+M2PP2SN(J)
   60 CONTINUE
   RETURN
   END

SUBROUTINE AVG (NMPK1)
C. NMPK1 = NUMBER OF POINTS TO USE FOR AVPK1
C. NTOPF = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C. ON ANALOG CONSOLE 1
C. N2TOPF = TOTAL NUMBER OF STATE VARIABLES, INPUTS, AND OUTPUTS
C. ON ANALOG CONSOLE 2
C.
C. COMMON/HERN/M2ST8, N1TOPF, M2PP2SN(J), M2PP2SN(J). 1
   N2ST8, N2TOPF, M2PP2SN(J), M2PP2SN(J). 1
   DIMENSION SUM1,(2), SUM2,(2), SUM3,(2), SUM4,(2), SUM5,(2)
   DO 10 J=1,N1TOPF
      SUM1(J) = 0
10 CONTINUE
   DO 20 J=1,N2TOPF
      SUM2(J) = 0
20 CONTINUE
F3 = NMPK1
C. GENERATE AVPK1s
   DO 50 J=1,F3
      CONTINUE
C. CONSOLE 2
   CALL (HMPK1, TEMP(33), N2TOPF, TEMP(3))
   DO 30 J=1,N2TOPF
      SUM2(J) = SUM1(J)+TEMP(1)
30 CONTINUE
50 CONTINUE
C... CONSOLE I
CALL USCK(C,1,10904.1,10904)
DO 40 I=1,N1010F
SUM(I) = SUM(I)+N1010F
40 CONTINUE
CALL USCK(C,1,10904)
CALL USCK(C,1,10904)
CALL LOOP
CALL USCK(C,1,10904)
50 CONTINUE
C... DETERMINE AVERAGES
DO 60 J=1,N1010F
AVRG(J) = SUM(J)/PL Industries
SUM(J) = 0.
60 CONTINUE
DO 70 J=1,N1010F
AVRG(J) = SUM(J)/PL Industries
SUM(J) = 0.
70 CONTINUE
CALL SUMDFK
CALL FILE
RETURN
END

SUBROUTINE DISK
C
C... THE ONLY WAY THAT THE FLIPPER DISK CAN BE WRITTEN ON IS THROUGH
C... THIS SUBROUTINE.
C
COMMON/HEH/R158, N1010, N1010F, NV602(60), NV62SN(20).
COMMON/FILEL/DP, DISK
DIMENSION IR1(60), IR2(100), IR3(100)
WRITE (1,63) IR1, IR2, IR3
61 FORMAT (215)
63 FORMAT (215)
LA LP
OCT 3000
LA 11000
OCT 3000
IRGB = IR1+1
DO 10 I=1,N1010F
IRGB = NWK(I)*12767
10 CONTINUE
DO 20 J=1,N1010F
J = I*N1010F
SUBROUTINE BILOP (110110, 110101, 110110, 110110, 110110)

C. ITO110 = (RUNNING TOTAL) / 100
C. ITO11 = MASC(RUNNING TOTAL - 100 + ITO110)
I. ITSIGN = SIGN OF RUNNING TOTAL
C. IN = NTH NUMBER TO BE ADDED TO RUNNING TOTAL
C. IM1 = IN - 100
C. IM2 = MASC(IN - 1000 + 100)

I. ITSIGN = ITOSIGN(IN)
I. IM1 = IM1 / 100
I. IM2 = IM2 / 100 + IM1 / 100
IF (IM2 = IM1) GO TO 20
IF (ITSIGN * IM1 < 0) GO TO 30

C. ITO110 AND IM HAVE THE SAME SIGN
C. ITO110 = IM1 + 100
C. ITO11 = IM2 + 100
IF (ITO11.LT.100) RETURN
C. ITO110 = IM1 - 100
I. ITO110 + ITO110 RETURN

C. ITO110 AND IM HAVE THE DIFFERENT SIGN
IF (IM2 < IM1) GO TO 20
IF (IM2 > IM1) GO TO 30

C. ITO110 AND IM CAN BE NEGATIVE
C. ITO110 = IM2 + ITO110 + IM1
C. ITO110 = IM2 + ITO110
I. ITSIGN = 1
IF (ITO110.LT.100) ITSIGN = -1
C. ITO110 = ITO110 / 100
I. ITO110 + ITO110 RETURN

C. MAGNITUDE OF TOTAL IS GREATER THAN MAGNITUDE OF A
C. ITO110 = ITO110 + IM10
APPENDIX B

'MATCALC' PROGRAM AND 'READIN' PROCDEF

****MATRIX GENERATION ROUTINE

C C

C DIMENSION NAME(30), INDEX(61)
C DIMENSION ID(18), DATA(61,70)
C DIMENSION NS(20), NS(20), NS(20), NS(20)
C DIMENSION SPH(70), CTEST(30)
C DIMENSION A(20,20), A(20,20), C(20,20), D(20,20)
C DIMENSION H(60), ADG(30), NAME(20), NAME(20)

C DATA DATA(470)-0.0/
C DATA LAST/0/, (70)/610/
C DATA CTEST/0/, (40)/0/
C INTEGER VA, VD, ADG
C DATA HDG/60*W5/, ADG/30*W5

C READ(9,8001) NS, NS, NS

8001 FORMAT (313)
C NWORDS = 2*NS+NC+NY
C NS = 2*(NS+NC)+1
C NOUT = NS+NC
C NINPUT = NS

C READ(9,8002) (NAME(I),I=1,NTOT)

8002 FORMAT (30A)
C READ(9,8003) (SPH(I),I=1,NWORD)

8003 FORMAT (4I10.6/)
C READ(9,8002) (NAME(I),I=1,NS)
C READ(9,8002) (NAME(I),I=1,NT)

*********

*********

FLOPPY DISK READ IN AND UNSCALING OF DATA

****READ IN START

I=C

1 I=I+1
C READ(9,5001,END=2,EP=2) NA, NS

5001 FORMAT (215)

C

C **DISTURBANCE NUMBER CHECK
C IF (NA .EQ. NS) GO TO 3
C WRITE (6,6100) NAME

6100 FORMAT (5X,'NA VALUE ('I3,') IS NOT EQUAL TO NB VALUE ('I3,')',/,-
C 'ENTER CORRECT DISTURBANCE VALUE (15) FORMAT')
C READ(9,7101) NA

7101 FORMAT (15)
GO TO 3
2 WRITE (*,7002)
7002 FORMAT (5X,'NO. OF SBS NOT EQUAL TO EXPECTED NO. OF DISTURBANCES',/,-
ASK,'ENTER 1 TO STOP, DEFAULT TO CONTINUE'/)
READ (7,7003) TEST
7003 FORMAT (I1)
IF (TEST .EQ. 1) STOP
GO TO 500
3 IF (INDX(NA) .EQ. 0) GO TO 4
WRITE (*,1001) NA,INDX(NA)
1001 FORMAT (5X,'INDX(*,12,) PREVIOUSLY SET TO SEQ, IN,/--
* 5X,'DEFAULT TO REPLACE, 1 TO STOP'/)
READ (7,7003) TEST
IF (TEST .EQ. 1) STOP
C
**INDX - CONTENTS ARE SEQUENCE NUMBERS
* INDX(NA) = I
LCT = 1
J = 1
SEND = 10
5 JA = J+17
IF (LCT .EQ. 1 .AND. SEND .GT. NWORDS) GO TO 10
IF (J .GT. NWORDS) GO TO 10
SEND (J,5001) (IVAL(K),K=1,SEND),ICK
5002 FORMAT (1616,17)
LAST = LCT+N
GO TO 15
10 JA = NWORDS
SEND = JA-J+1
READ (5,5002) (IVAL(K),K=1,SEND)
READ (9,5003) ICK
5003 FORMAT (17)
LCT = LCT+2
C
**CHECKSUM CHECK
15 ISUM = 0
DO 16 K=1,SEND
16 ISUM = ISUM+IVAL(K)
IF (ISUM .NE. ICK) GO TO 50
100 ILast = ICK
WRITE (*,5001) NA,NA
IF (SEND .NE. 16) GO TO 17
WRITE (9,5002) (IVAL(K),K=1,SEND),ICK
GO TO 20
17 WRITE (9,5002) (IVAL(K),K=1,SEND)
WRITE (9,5003) ICK
C
**UNSCALING OF SAMPLED VARIABLES
20 K = 0
DO 25 N=J,JA
K = K+1
Y1 = IVAL(K)
TO: Y/32768.
25 DATA(I,J) = Y*SF(N)
J = JA+1
IF (J.GT. NWORDS) GO TO 5
IF (J.GT. NWORDS) 70 TO 1
WRITE (6,6001)
6001 FORMAT (5X,'MATRIX DATA READ IN AND UNSCALING COMPLET.'/
,'A DATA BASE ON UNIT 9 CONTAINS (CORRECTED IF WEC) FLOPPY DISK DATA')
GO TO 500

C **CHECKSUM ERROR ROUTINE
50 WRITE (6,100) NA, 1, LINF
6101 FORMAT (' CHECKSUM ERROR',3X,' NUM ',I3,' SEQ ',I3, ' LINE ',I4)
WRITE (6,6102) ILAST, ICK, ISUM
6102 FORMAT (' CHECKSUMS : PREVIOUS = ',I7, ' READ = ',I7, ' CALC = ',I7)
WRITE (6,6103) (IVAL(K),K=1,NWEND)
6103 FORMAT (' THE VALUES ARE ',/I,I,I)
200 WRITE (6,6106)
6104 FORMAT (' ENTER CORRECTED LINE')
IF (NWEND .LT. 18) GO TO 6010
READ (7,5002) (IVAL(K),K=1,NWEND)
READ (7,5003) ICK
GO TO 6520
6010 READ (7,5002) (IVAL(K),K=1,NWEND),ICK
6020 ISUM = 0
DO 55 K=1,NWEND
55 ISUM = ISUM+IVAL(K)
IF (ISUM .NE. ILM) GO TO 100
WRITE (6,6105) ISUM, ICK
6105 FORMAT (' CHECKSUMS DIFFER, CALC = ',I7, ' READ = ',I7)
WRITE (6,6106)
6106 FORMAT (' ENTER 1 TO RE-ENTER, 2 TO STOP, DEFAULT TO CONTINUE')
READ (7,7003) NTEST
NTEST = NTEST+1
GO TO (100,200,300),NTEST
300 STOP

********
500 CONTINUE

IF DESIRE A LISTING OF DATA MATRIX WHICH NOW CONTAINS UNSCALED DISTURBANCE DATA,
INSERT WRITE STATEMENTS HERE

********
** ****
********
INITIALIZATION FOR DISURANCE VERIFICATION

NBRUN = INDEX(1)
IF (NBRUN .NE. 0) GO TO 544
WRITE (6,6002)
6002 FORMAT (5X,'NO BASE DATA,TERMINATE')
STOP
504 READ(5,8005) (NDELT(K),K=1,NTOT)
READ(5,8005) (NDESY(K),K=1,NT)
READ(5,8005) (NORSV(K),K=1,NT)
READ (5,8005) (NCOL(K),K=1,NTOT)
8005 FORMAT (3013)
C
510 ATEST(N) = SP(ND)/1000.
C
********
C
DISTURBANCE CHECK AND VERIFICATION
C
DO 700 N=1,NTOT
NB=NSIZE
NAME(N) = NAME(N)
DO 400 K=1,2
NC=0
NI = 2*N+K-1
IF (NI .EQ. 0) N1 = NBRUN
DO 400 I=1,NTOT
SD = NDELT(I)
IF (ABS(DATA(NBRUN,ND)) .LT. ATEST(I)) GO TO 399
TEST = (DATA(NI,ND)-DATA(NBRUN,ND))/DATA(NBRUN,ND)
IF (TEST .LT. ATEST) GO TO 400
312 IF (NAME(I) .NE. NAME(N)) GO TO 333
NC = I
GO TO 400
333 WRITE (6,201) N1,N1,N1,N1,DATA(NI,ND),DATA(NBRUN,ND)
201 FORMAT (5X,'**SEQUENCE ',I2,' INDEX ',I2,' 2ND, ',A4,' IS PERTURBED IN ADDITION TO ',A4,','
A4,'  VALUE = ',F15.5,' BASE VALUE = ',F15.5)
WRITE (6,203) Ni
203 FORMAT (5X,'**BASE IS BEING USED FOR SEQ ',I2,'/')
WRITE (6,204) Ni
204 FORMAT (5X,'**BASE IS BEING USED FOR SEQ ',I2,'/')
GO TO 400
399 WRITE (6,205) DATA(NI,ND),DATA(NBRUN,ND)
400 CONTINUE
IF (NAME(I) .EQ. NAME(N)) GO TO 590
WRITE (6,202) NAME(I),DATA(NI,ND),DATA(NBRUN,ND)
202 FORMAT (5X,'**BASE IS BEING USED FOR SEQ ',I2,'/')
GO TO 400
**ORIGINAL PAGE IS OF POOR QUALITY**

202 FORMAT (1X, 'SEQUENCE', I2, ', INDEX ', I2, ', ', AA, ', ' DISTURBANCE LOW OR ZERO', /, -
          AA1, ' VALUE = ', G13.5, ', BASE VALUE = ', G13.5)
          GO TO 600
590 WRITE (6,209) NRI, NS3
209 FORMAT (' INDEX ', I2, ', NO DISTURBANCE FOR ', AA, ',', /
          A*, USING BASE RUN')
600 CONTINUE
700 CONTINUE

```
******
******
**MATRIX CALCULATION
******
******
J = 1
DO 600 M = 1, N \\
   NCL = NCOL(N) \\
   NN = N = NSIZE \\
   NW = NAME(NCL) \\
   NROW = NDELT(NCL) \\
   NTWO = 2 * NCL \\
   NROW2 = INDEX(NTWO) \\
   IF (NROW1 .EQ. 0) NROW1 = NBUN \\
   IF (NROW2 .EQ. 0) NROW2 = NBUN \\
607 DNOMN = DATA(NROW1, NROW) - DATA(NROW2, NWORD) \\
   IF (ABS(DNOMN) .LT. 1.E-06) GO TO 609 \\
   IF (NROW1 .EQ. 0) GO TO 608 \\
   WRITE (6,205) NS3, NI \\
205 FORMAT (1X, 'DISTURBANCE IDENTICAL, SETTING SEQUENCE ', I2, ' TO BASE') \\
   NBUN = NBUN \\
608 GO TO 607 \\
609 WRITE (6,206) NS3 \\
206 FORMAT (1X, 'DISTURBANCES, MATRIX COLUMN ZERODED') \\
610 DUM = 1.0E50 \\
C **CALCULATION OF A & B MATRIX COLUMNS
609 DO 620 K = 1, NSIZE \\
   KROW = NROW(K) \\
   IF (K .LT. NSIZE) GO TO 610 \\
   A(K,N) = (DATA(KROW1, NDUM) - DATA(KROW2, NDUM)) / DNOMN \\
   GO TO 620 \\
610 C(K,K) = DATA(KROW1, NDUM) - DATA(KROW2, NDUM) / DNOMN \\
620 CONTINUE \\
C **CALCULATION OF C & D MATRIX COLUMNS
609 DO 630 K = 1, NSIZE \\
   KROW = NROW(K) \\
   IF (K .LT. NSIZE) GO TO 625 \\
   C(K,N) = (DATA(KROW1, NDUM) - DATA(KROW2, NDUM)) / DNOMN \\
   GO TO 630 \\
630 C(K,K) = (DATA(KROW1, NDUM) - DATA(KROW2, NDUM)) / DNOMN
```
630 CONTINUE
J = J + 1
VAR = NCOL(N)
1*(N .GT. NS + 1) GO TO 710
1*(N .GT. VS) GO TO 705
HDG(J) = NAME(VAR)
GO TO 800
705 J = 2
710 HDG(J) = NAME(VAR)
800 CONTINUE

**********
PRINT OUT OF MATRICES
10 COLS MAX PER LINE
NC = INPUT
WRITE (6, 6900)
6900 FORMAT (5X, 'THE A MATRIX/')
IF (NS .LE. 10) GO TO 6950
NCOL1 = NS/2
NCOL2 = NCOL1 + 1
WRITE (6, 6960) (HDG(L), L = 1, 30)
6960 FORMAT (' ', 5X, 30A4)
DO 6906 I = 1, NS
6906 WRITE (6, 6901) NAME(I), (A(I, J), J = 1, NCOL1)
WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6907 I = 1, NS
6907 WRITE (6, 6901) NAME(I), (A(I, J), J = NCOL2, NS)
GO TO 6955
6950 WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6908 I = 1, NS
6908 WRITE (6, 6901) NAME(I), (A(I, J), J = 1, NS)
6901 FORMAT (' ', 4X, 5X, 10D12.4)
6955 WRITE (6, 6902)
6902 FORMAT (5X, 'THE B MATRIX/')
WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6911 I = 1, NS
6911 WRITE (6, 6901) NAME(I), (B(I, J), J = 1, NC)
WRITE (6, 6903)
6903 FORMAT (5X, 'THE C MATRIX/')
IF (NS .LE. 10) GO TO 6970
WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6921 I = 1, NY
6921 WRITE (6, 6901) NAME(I), (C(I, J), J = 1, NCOL1)
WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6922 I = 1, NY
6922 WRITE (6, 6901) NAME(I), (C(I, J), J = NCOL2, NS)
GO TO 6975
6970 WRITE (6, 6960) (HDG(L), L = 1, 30)
DO 6923 I = 1, NY
6923 WRITE (6, 6901) NAME(I), (C(I, J), J = 1, NS)
6975 WRITE (6, 6904)
6904 FORMAT(5X,'THE D MATRIX')
   WRITE (6,6905) (ADD(I),I=1,30)
   DO 6931 I=1,30
   6931 WRITE(6,6901) NAME(I),(D(I,J),J=1,NC)
   C IF DESIRE A PERMANENT DATASET WITH A,B,C,D MATRICES
   C INSERT WRITE STATEMENTS HERE WITH DESIRED FORMAT
   C
   C **ERROR MESSAGE NUMBER PRINTOUT
   C IF (ERROR = NE. 0) WRITE(6,711) ERROR
   711 FORMAT (///,** AT LEAST **,1,** ERRORS ***)
   STOP
   END

PROCDEF READIN

PROCDEF READIN
PARAM DTK
IF ('EXIT'='') GO TO 'END'
DISPLAY 'ENTER DATA - FINISH WITH _END'
ERASE DTK
END DTK
DEFAULT SYSNY=5
END DTK
VSEFLY DTK:
DEFAULT SYSNY=5
DISPLAY 'V5-DATASET COMPLETE - MORE IF NEC'
'END'
APPENDIX C

'QINFO' dataset

Format:

<table>
<thead>
<tr>
<th>line 1</th>
<th>NS, NC, NY</th>
<th>313</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME array</td>
<td>30A4</td>
<td></td>
</tr>
<tr>
<td>Sf array</td>
<td>8(10F8.2)</td>
<td></td>
</tr>
<tr>
<td>NAM2 array</td>
<td>30A4</td>
<td></td>
</tr>
<tr>
<td>NAM3 array</td>
<td>30A4</td>
<td></td>
</tr>
<tr>
<td>NDELT array</td>
<td>30I3</td>
<td></td>
</tr>
<tr>
<td>NDRIV array</td>
<td>30I3</td>
<td></td>
</tr>
<tr>
<td>NOBSV array</td>
<td>30I3</td>
<td></td>
</tr>
<tr>
<td>last line</td>
<td>NCOL array</td>
<td>30I3</td>
</tr>
</tbody>
</table>

Description:

NS: number of states
NC: number of inputs
NY: number of outputs
NAME: names of the states and inputs in the order perturbed for generation of hybrid data.

description of each name - 4 alphanumeric characters

Each position of the NAME array contains the name of a state or an input. Since each state and each input is perturbed in both a plus and minus direction, each position of the NAME array corresponds to two readings and hence, two disturbance numbers. The correspondence between a position in the NAME array and two disturbance numbers is as follows:

N: position in NAME array
(e.g., NAME(N) = ST1 name of state 1)

| disturbance number of plus perturbation | 2*N |
| disturbance number of minus perturbation | 2*N+1 |

NCOL: determines columnar structure of the matrices
The integer values represent the positions of the NAME array in the order desired for the columns of matrices.
(e.g., NAME array ST1, ST2, IN1, IN2, ST3, ST4, IN3, IN4

position 1 2 3 4 5 6 7 8
Want columns 1, 2, 3, and 4 of A and C matrices to be with respect to ST1, ST2, ST3, ST4 and want columns 1, 2, 3, and 4 of B and D matrices to be with respect to IN1, IN2, IN3, IN4.

Then NCOL array is 1, 2, 5, 6, 3, 4, 7, 8)
The data for the arrays SF, NDELT, NDRIV, and NOBSV depend on the order of the values in the rows of the DATA matrix. Let 'ROW' mean the general form of a row in the DATA matrix.

**SF**
- scale factors corresponding to values in 'ROW'.
- the real x values in 1/x scale factor representation.

**NDELT:** positions in 'ROW' occupied by the values of the states and inputs.

**NDRIV:** positions in 'ROW' occupied by the values of the state derivatives ordered as the rows of A and B matrices are desired.

**NOBSV:** positions in 'ROW' occupied by the values of the outputs ordered as rows of C and D matrices are desired.

**NAM2:** names of the state derivatives in order specified in NDRIV.
- length per name - 4 alphanumeric characters.

**NAM3:** names of the outputs in order specified in NOBSV.
- length per name - 4 alphanumeric characters.
APPENDIX D
DATA PROCESSING PROCEDURE

Note: This procedure is specialized for use with the NASA-Lewis Research Center EAI Pacer 100 computer system.

TEKTRONIX 4010 TERMINAL INITIALIZATION

1. turn terminal ON (power light will come on).
2. turn 360 switch to the ON position.
3. turn SEL switch to the OFF position.
4. set BAUD switch for 1200.
5. set NORM/TP-WR switch for NORM.
6. set LOCAL/LINE switch for LINE.
7. on the phone modem, momentarily depress the CONNECT button. (the CONNECT light will come on when line is connected).
8. hit 'BREAK' key on terminal.
9. 'logon' to 360 as normally done.
   (check TERMINAL ID; it should be posted).

TEKTRONIX 4922 FLOPPY DISK INITIALIZATION

1. turn POWER switch ON.
2. depress RESET button.
3. turn PROMPT MODE switch OFF.
4. select proper disk.
   a. check disk notebook.
   b. make sure disk moves freely in envelope.
5. set DISK switch for disk being used.
   a. A if disk 1.
   b. B if disk 2.
6. set TERMINAL switch (switch #1) for terminal being used.
   a. LEFT if left terminal.
   b. RIGHT if right terminal.
7. set TRANSMISSION switch (switch #2) to the LEFT.
   (LEFT setting - data transmission stops at 'XOFF' character).
8. set R-W switch to MANUAL READ.
DATA TRANSFER FROM FLOPPY DISK

1. depress RESET button on disk.
2. set beginning track address in TRACK ADDRESS.
3. depress LA button on disk.
4. type at terminal
   READIN 'DTRK' (CR).
   when the procdef starts, the message:
   'ENTER DATA - FINISH WITH - END'
   will print at the terminal. Data transfer will be automatic;
   the user can determine this by the periodic "noise" the disk
   will make during this transfer.
5. type at terminal:
   END (CR)
   when the end of the data is reached.
   This will end the EDIT dataset created by procdef READIN.
   It may be necessary to type _END (CR) more than once before
   the 360 accepts it. These unaccepted lines are entered into
   the dataset 'DTRK' and must be deleted through REDIT
   before using 'DTRK' as input to the MATCALC program.

'MATCALC' PROGRAM EXECUTION

After creation of datasets 'DTRK' and 'QINFO', the program
can be executed by typing at the terminal:

DDEF FT05F001, VS, 'DTRK'
DDEF FT08F001, VS, 'QINFO'
DDEF FT09F001, VS, 'QOUTPUT'
MATCALC

NOTES:

1. When the message 'MATRIX DATA READ IN' etc. is printed
   at the terminal, depress RESET button on disk.
2. Final printout at terminal will be the A, B, C, and D
   matrices
3. If errors are detected, appropriate messages will be
   printed out, allowing the user opportunity to make
   corrections. The procedure for checksum error cor-
   rection follows.
CHECKSUM ERROR PROCEDURE

If the read and calculated values of the checksum for a line of floppy disk data are not equal, a checksum error has occurred. An error message will print at the terminal, allowing the data line to be re-entered as follows:

1. set LOCAL/LINE switch to LOCAL on terminal.
2. depress RESET button on disk.
3. set beginning track address in TRACK ADDRESS.
4. depress LA button.
5. at the terminal,
   the 'CTRLQ' key will input one line of floppy disk data
to the terminal CRT screen.
   a. Lines of data should be inputted until the proper
disturbance reading data is reached. This is done
by comparing the RUN number outputted in the error
message to the values in the line containing the
disturbance number for each reading.
   b. Lines of data from that reading should then be
inputted, noting the checksum for each line. When
the checksum equal to the PREVIOUS checksum
printed in the error message) is reached, the
LOCAL/LINE switch should be set to LINE.
   c. The next data line should be inputted by the 'CTRLQ'
key. This line will be accepted by the 360 as the
re-entered data line. If the checksum of this data
line is on the next line, that line will automatically
input to the 360.
TABLE I. - PERTURBATION CONTROL

CARDS FOR 4-STATE, 4-INPUT,

4-OUTPUT EXAMPLE

\[
\begin{array}{ccc}
2 & 2 & 2 \\
1 & 45 & .02 \\
1 & 75 & .02 \\
-1 & 41 & .02 \\
-1 & 71 & .02 \\
-1 & & \\
-1 & & \\
2 & 2 & 2 \\
1 & 75 & .02 \\
1 & 45 & .02 \\
-1 & 71 & .02 \\
-1 & 41 & .02 \\
-1 & & \\
-1 & & \\
\end{array}
\]

TABLE II. - PERTURBATION ORDER OF STATE VARIABLES AND SYSTEM INPUTS FOR 4-STATE,

4-INPUT, 4-OUTPUT EXAMPLE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Console</th>
<th>ADC Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST1</td>
<td>Primary</td>
<td>1</td>
</tr>
<tr>
<td>ST2</td>
<td>Primary</td>
<td>2</td>
</tr>
<tr>
<td>IN1</td>
<td>Primary</td>
<td>5</td>
</tr>
<tr>
<td>IN2</td>
<td>Primary</td>
<td>6</td>
</tr>
<tr>
<td>ST3</td>
<td>Secondary</td>
<td>33</td>
</tr>
<tr>
<td>ST4</td>
<td>Secondary</td>
<td>34</td>
</tr>
<tr>
<td>IN3</td>
<td>Secondary</td>
<td>37</td>
</tr>
<tr>
<td>IN4</td>
<td>Secondary</td>
<td>38</td>
</tr>
</tbody>
</table>

Note: The perturbation order of state variables and system inputs per console is arbitrary.
TABLE III. - READING NUMBERS FOR 4-STATE, 4-INPUT, 4-OUTPUT EXAMPLE

Rdg. # (= disturbance #)
1  BASE RUN
2  ST1 +
3  ST1 -
4  ST2 +
5  ST2 -
6  IN1 +
7  IN1 -
8  IN2 +
9  IN2 -
10  ST3 -
11  ST3 -
12  ST4 +
13  ST4 -
14  IN3 +
15  IN3 -
16  IN4 +
17  IN4 -

Order of values per reading:
ST1, ST2, DRV1, DRV2, IN1, IN2, OUT1, OUT2, ST3, ST4, DRV3, DRV4,
IN3, IN4, OUT3, OUT4

Note - STJ represents state variable \( x_j \) where \( J = 1 \) to \( 4 \)
INJ represents input variable \( u_j \) where \( J = 1 \) to \( 4 \)
OUTJ represents output variable \( y_j \) where \( J = 1 \) to \( 4 \)
DRVJ represents state derivative \( \dot{x}_j \) where \( J = 1 \) to \( 4 \)
# Table IV - Floppy Disk Data for 4-State, 4-Input, 4-Output Example

<table>
<thead>
<tr>
<th>State</th>
<th>Input 1</th>
<th>Input 2</th>
<th>Input 3</th>
<th>Input 4</th>
<th>Output 1</th>
<th>Output 2</th>
<th>Output 3</th>
<th>Output 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- State 1: Inputs are 1, 2, 3, 4. Outputs are 0, 0, 0, 0.
- State 2: Inputs are 4, 5, 6, 7. Outputs are 0, 0, 0, 0.
- State 3: Inputs are 8, 9, 10, 11. Outputs are 0, 0, 0, 0.
- State 4: Inputs are 12, 13, 14, 15. Outputs are 0, 0, 0, 0.
TABLE V. "INDX" ASSAY AND "DATA" IMTAIK FOR 4-STATE, 4-INPUT, 4-OUTPUT EXAMPLE

<table>
<thead>
<tr>
<th>INDX ARRAY</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA MATRIX

<table>
<thead>
<tr>
<th>Row</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00049</td>
<td>1.9726</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>2</td>
<td>1.20045</td>
<td>1.3798</td>
<td>-0.204688E-01</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>3</td>
<td>0.91201</td>
<td>1.3792</td>
<td>0.85547E-01</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>4</td>
<td>1.30264</td>
<td>1.1792</td>
<td>0.00000</td>
<td>-1.37573E-01</td>
<td>0.999939</td>
</tr>
<tr>
<td>5</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>6</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>7</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>8</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>9</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>10</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>11</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>12</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>13</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>14</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>15</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>16</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
<tr>
<td>17</td>
<td>1.00049</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9726</td>
<td>3.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>2</td>
<td>1.3792</td>
<td>-0.204688E-01</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>3</td>
<td>1.3792</td>
<td>0.85547E-01</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>4</td>
<td>1.1792</td>
<td>0.00000</td>
<td>-1.37573E-01</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>5</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>6</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>7</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>8</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>9</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>10</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>11</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>12</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>13</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>14</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>15</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>16</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
<tr>
<td>17</td>
<td>1.8792</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.999939</td>
<td>1.8792</td>
</tr>
</tbody>
</table>
**TALLI VI. - 'QINFO' DATASET FOR 4-INPUT, 4-OUTPUT EXAMPLE**

<table>
<thead>
<tr>
<th>ST1</th>
<th>ST2</th>
<th>IN1</th>
<th>IN2</th>
<th>ST3</th>
<th>ST4</th>
<th>IN3</th>
<th>IN4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>2.</td>
<td>2.</td>
<td>2.5</td>
<td>2.</td>
<td>2.</td>
<td>2.5</td>
<td>2.</td>
</tr>
<tr>
<td>2.</td>
<td>2.</td>
<td>5.</td>
<td>4.</td>
<td>5.</td>
<td>4.</td>
<td>5.</td>
<td>4.</td>
</tr>
</tbody>
</table>

Note: Appendix C gives the required format specifications for 'QINFO' dataset and defines the arrays in the MATCALC program which are initialized by this data.
### Table VII: A, B, C, D Matrix Results for 4-State

#### 4-Input, 4-Output Example

**A Matrix**

<table>
<thead>
<tr>
<th></th>
<th>ST1</th>
<th>ST2</th>
<th>ST3</th>
<th>ST4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>0.0</td>
<td>-0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN2</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>IN3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
</tr>
<tr>
<td>IN4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**B Matrix**

<table>
<thead>
<tr>
<th></th>
<th>ST1</th>
<th>ST2</th>
<th>ST3</th>
<th>ST4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>IN2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**C Matrix**

<table>
<thead>
<tr>
<th></th>
<th>ST1</th>
<th>ST2</th>
<th>ST3</th>
<th>ST4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**D Matrix**

<table>
<thead>
<tr>
<th></th>
<th>ST1</th>
<th>ST2</th>
<th>ST3</th>
<th>ST4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>IN4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 1. - General analog circuit based on ST8 + \int (INPUT-ST8)dt.

Figure 2. - Data deck setup for example.