UCID- 17267

Lawrence Livermore Laboratory

PROGRAMS NAES AND SS: USER-ORIENTED PROGRAMS FOR SOLVING NONLINEAR ALGEBRAIC EQUATIONS AND ORDINARY DIFFERENTIAL EQUATIONS

Howard K. McCue

August 11, 1976

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISCLAIMER

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

PROGRAMS NAES AND SS: USER-ORIENTED PROGRAMS FOR SOLVING NONLINEAR ALGEBRAIC EQUATIONS AND ORDINARY DIFFERENTIAL EQUATIONS

ABSTRACT

Program NAES (Nonlinear Algebraic Equation Solver) is a Fortran IV program used to solve the vector equation $\underline{f}(\hat{\underline{x}}) = \underline{0}$ for $\hat{\underline{x}}$. Two areas where Program NAES has proved to be useful are the solution for initial conditions and/or set points of complex systems of differential equations and the identification of system parameters from steady-state equations and steady-state data. Program SS (State Space) is a Fortran IV program used to solve a system of first-order, ordinary differential equations with a minimum of specialized coding. Program SS automatically provides a tabular listing and line-printer plots of the outputs. In addition, provisions are made to: perform one-time preintegration calculations, read specialized input data, establish specialized output labels, handle piecewise continuous $\underline{f}[\underline{x}(t),t]$, make x-y plots of output variables, and record the minimums/maximums of specified variables. Subroutines have been written to provide delay, level detection with hysteresis, and solutions to implicit equations.

INTRODUCTION

Programs NAES and SS were written to provide user-oriented, computer aids for solving nonlinear algebraic equations and ordinary differential equations of the initial-value type. For each program, only the Fortran coding describing the problem need be supplied by the user. This feature allows the user to concentrate his attention on that portion of the coding which describes his problem, and not on the details of the numerical method used to obtain the solution. This minimizes the time and effort required to obtain computer solutions. Program NAES (Nonlinear Algebraic Equation Solver) has proved useful in solving for the initial conditions and/or set points of complex systems of differential equations, and in solving for the model parameters from steady-state equations and data. Program SS (State Space) has been successfully used to provide numerical solutions for a wide variety of physical systems: helicopter flight control, gas-transfer systems with bang-bang control, synchronous generators and turbines with associated speed and voltage controls, process-control analysis for liquid-level control, temperature control of a laser optical room, etc. The main body of this report illustrates how Programs NAES and SS are used to solve a physical problem. Particular attention is directed to the thought process involved in the problem setup; this description should prove useful to people unfamiliar with the problem setup used in obtaining numerical solutions. This section should also allow a potential user to size up the effort required to obtain numerical solutions via NAES and SS. In Appendices A and B are the detailed write-ups for computer Programs NAES and SS; Appendix C describes how transfer functions are handled in Program SS.

EXAMPLES OF USAGE

To understand how one would use Programs NAES and SS, consider the system shown in Fig. 1. This system consists of two masses, two dashpots, and two nonlinear springs; the masses are acted on by a gravitational field. Prior to time $t = 0$, the system is in steady state with two forces, Force 1 and Force 2, acting on the two masses. The effect of these forces is to displace the masses from the normal position (where Force 1 = Force 2 = 0). At $t = 0$, Force 1 and Force 2 are released (i.e., they are set equal to zero for $t \geq 0$). Starting at $t = 0$, we wish to compute the displacements and velocities of the two masses plus the kinetic energy, the potential energy, and the total energy in the system. The displacements (d) are taken to be zero when Force 1 and Force 2 are zero and no gravitational force acts on the masses.

In this physical system are six mechanisms for storing energy: kinetic energy in the two masses, potential energy of the two masses in the gravitational field, and potential energy in the two nonlinear springs. The two dashpots provide the only means of dissipating energy in this system. At $t = 0$, the velocities of the masses are zero; thus, the initial value of the kinetic energy is zero. Since neither displacement is zero at $t = 0$, neither the potential energy stored in the springs nor the potential energy of the masses (both taken to be zero when $d_1 = d_2 = 0$) is zero.

Due to the initial displacement of the masses, the system will go through some coupled oscillations for $t > 0$. Since there are no energy

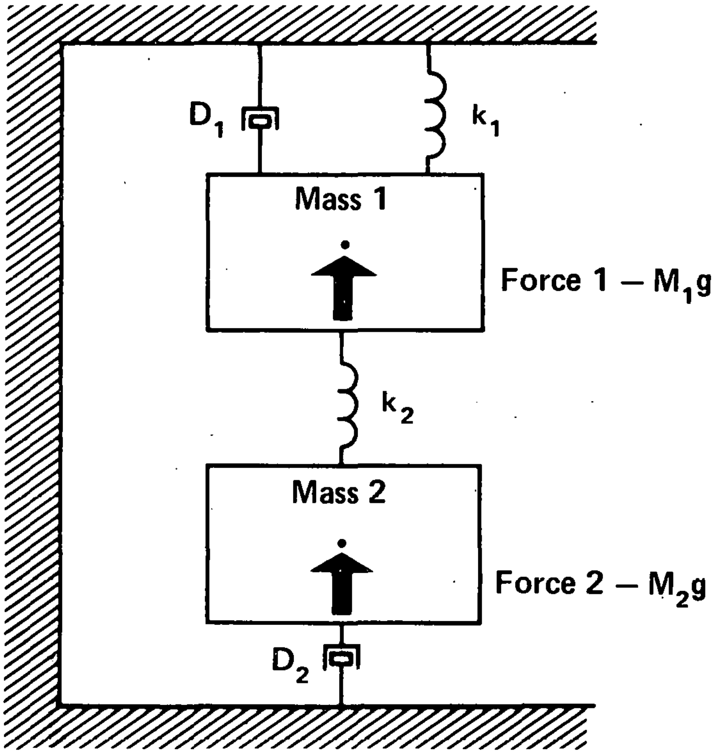

Fig. 1. Two-mass system with nonlinear Springs in a gravitational field. For $t < 0$, Force 1 = Force 2 = 1000 and $\dot{d}_1 = \dot{d}_2 = 0$. For $t \geq 0$, Force 1 = Force 2 = 0.

inputs to the system for $t > 0$ and since the dashpots will remove energy during these oscillations, we know the coupled oscillations will eventually decay to zero. During these oscillations, energy will be exchanged between the kinetic and potential modes. For $t > 0$, the total energy, the sum of the kinetic and potential energies, will slowly decay because of the energy dissipated by the dashpots. The dynamics of the physical system in Fig. 1 is governed by the following coupled, ordinary differential equations:

$$\text{Force 1} - M_1 g = M_1 \ddot{d}_1 + D_1 \dot{d}_1 + k_1(d_1) + k_2(d_1 - d_2) ,$$

$$\text{Force 2} - M_2 g = M_2 \ddot{d}_2 + D_2 \dot{d}_2 + k_2(d_2 - d_1) ,$$

where D_1 and D_2 are the dashpot coefficients and the nonlinear springs are defined by:

$$k_1(d_1) \triangleq c_1 d_1 + c_2 d_1^3 ,$$

$$k_2(d_1 - d_2) \triangleq c_3(d_1 - d_2) + c_4(d_1 - d_2)^3 .$$

The kinetic energy of this system at any instant of time is given by:

$$KE = \frac{1}{2} M_1 \dot{d}_1^2 + \frac{1}{2} M_2 \dot{d}_2^2 .$$

The potential energy of the masses in the gravitational field is taken to be zero for $d_1 = d_2 = 0$. Therefore, the potential energy of the two masses equals

$$PE_M = M_1 g d_1 + M_2 g d_2 .$$

Because of this selection of $d_1 = d_2 = 0$ as the point of zero potential energy, PE_M can take on positive and negative values.

The potential energy stored in a spring is obtained by integrating the force term over the displacement:

$$\text{Stored energy} = \int_0^x (an + bn^3)dn = a\frac{n^2}{2} + b\frac{n^4}{4} .$$

The potential energy stored in the two springs of our example is given by:

$$PE_S = c_1 \frac{d_1^2}{2} + c_2 \frac{d_1^4}{4} + c_3 \frac{(d_1 - d_2)^2}{2} + c_4 \frac{(d_1 - d_2)^4}{4} .$$

The total potential energy of the system is:

$$PE = PE_M + PE_S .$$

The total energy of the system then is given by:

$$\text{Total energy} \triangleq KE + PE_M + PE_S .$$

In this simulation, we wish to numerically solve for and plot: d_1, $\dot{d}_1$, d_2, $\dot{d}_2$, KE, PE, and KE + PE. Later, it will be shown that the displacements and velocities are state variables while the energy terms are nonlinear functions of the state variables. Program SS allows for the computation and plotting of both types of outputs. The differential equations that govern the system dynamics are given above. Before one can solve these differential equations, one must know the initial conditions. From the problem statement, $\dot{d}_1 = \dot{d}_2 = 0$ for all $t < 0$; this specifies two of the four initial conditions. Since $\dot{d}_1 = \dot{d}_2 = 0$ for all $t < 0$, $\ddot{d}_1 = \ddot{d}_2 = 0$ for all $t < 0$ too. Plugging these values into our differential equation yields the steady state equations for $t < 0$:

$$\text{Force 1} - M_1 g = c_1 d_1 + c_2 d_1^3 + c_3(d_1 - d_2) + c_4(d_1 - d_2)^3 ,$$

$$\text{Force 2} - M_2 g = c_3(d_2 - d_1) + c_4(d_2 - d_1)^3 .$$

Note that our differential equations have been reduced to nonlinear algebraic equations. In order to determine the last two initial conditions (d_1 and d_2) for our dynamics problem, we must first solve the above two coupled, nonlinear, algebraic equations. This is where Program NAES enters the problem. The nonlinear algebraic equations that determine the initial displacements are programmed in NAES; this coding appears in Fig. 2. The exact meaning of variables and where data should appear is covered in detail

```
C
C THE USER PLACES ALL OF HIS CODING BETWEEN THE TWO +-LINES.
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
100   CONTINUE
C DEFINE PROGRAM CONSTANTS N, GAIN, EPSC, EPSJ, MAX, IJAC, IAUTO,
C AND ISKIP HERE.
      N=2
      GAIN=1.0
      EPSC=1.0E-08
      EPSJ=1.0E-08
      MAX=200
      IJAC=0
      IAUTO=0
      ISKIP=0
C DEFINE THE INITIAL X-VECTOR HERE.
      X(1)=0.0
      X(2)=0.0
C DEFINE ANY ADDITIONAL PROBLEM CONSTANTS HERE.
      REAL MASS1, MASS2
      MASS1=1.0
      MASS2=1.0
      G=32.1740
      WRITE(NOUT,110)
110   FORMAT(20H--FORCE1--++FORCE2++)
      READ(NIN,120)FORCE1,FORCE2
120   FORMAT(2E10.3)
      FOR1=FORCE1-MASS1*G
      FOR2=FORCE2-MASS2*G
      C1=75.0
      C2=1.5
      C3=150.0
      C4=3.0
      GO TO 999
200   CONTINUE
C THE USER SPECIFIES THE N-DIMENSIONAL VECTOR-FUNCTION F.
      F(1)=C1*X(1)+C2*X(1)**3+C3*(X(1)-X(2))+C4*(X(1)-X(2))**3-FOR1
      F(2)=C3*(X(2)-X(1))+C4*(X(2)-X(1))**3-FOR2
      GO TO 999
300   CONTINUE
C IF IJAC.NE.0, THE USER SPECIFIES THE JACOBIAN HERE.
      GO TO 999
400   CONTINUE
C SPECIFY CONSTRAINTS ON THE ELEMENTS OF THE X-VECTOR HERE.
      IF(X(1).LT.-2.0)X(1)=-2.00
      IF(X(2).LT.-2.0)X(2)=-2.00
      GO TO 999
500   CONTINUE
C THIS SECTION PROFIDES A PLACE TO CALCULATE WITH THE SOLUTION VECTOR.
      PE=C1*X(1)**2/2.0+C2*X(1)**4/4.0+C3*(X(1)-X(2))**2/2.0+
     $   C4*(X(1)-X(2))**4/4.0+MASS1*G*X(1)+MASS2*G*X(2)
      WRITE(NOUT,510)PE
510   FORMAT(2/,19HPOTENTIAL ENERGY = ,E10.3)
      GO TO 999
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
```

Fig. 2. Fortran coding (in boxes) supplied by user for Program NAES.

in the NAES write-up of Appendix A. It is the purpose of this section to indicate the effort required to obtain a solution to the above equations.

In Fig. 2, the user supplied only the boxed-in information. In the 100 section, one defines the following data: convergence parameters, initial estimate of d_1 and d_2, and problem constraints. For a large class of problems, the convergence parameters are fixed, and one only changes n. For solution by NAES, the d_1 and d_2 variables are renamed x(1) and x(2). The initial estimates of x(1) and x(2) are zero. For this solution in NAES, it was decided to request the Force 1 and Force 2 data at execution time from the teletype*; this will allow one to rerun the problem for different values of forces without having to recompile NAES.

In the 200 section, we have written the nonlinear algebraic equations. The force terms have been moved to the other side of the equation. When F(1) = F(2) = 0, x(1) and x(2) are a solution to the nonlinear, algebraic equations. Program NAES will manipulate x(1) and x(2), driving F(1) and F(2) to essentially zero using the Newton-Raphson iterative technique. In section 400, interval constraints of $-2 \leq x(1)$ and $-2 \leq x(2)$ are specified. Finally, in section 500, we compute the potential energy based on values of d_1 and d_2 that satisfy our nonlinear, algebraic equations. Shown in Fig. 3 is the teletype dialogue for this problem. Again, only the boxed-in lines were typed by the user. For our problem, the answer is:

$$d_1 = 9.368,$$

$$d_2 = 13.93 .$$

Also shown in this dialogue are the values of F(1) and F(2):

$$F(1) = -4.729 \times 10^{-11} ,$$

$$F(2) = 7.276 \times 10^{-12} .$$

Therefore, the stated values for d_1 and d_2 are essentially solutions for the nonlinear equations. That is, the nonlinear equations are unbalanced

*Registered trademark of Teletype Corp.

```
NAES / .1 .1
--FORCE1--++FORCE2++
1000.0     1000.0
DO YOU WISH TO MODIFY CONVERGENCE VARIABLES--YES OR NO.
NO
DO YOU WISH TO MODIFY THE INITIAL X-VECTOR---YES OR NO.
NO
PROCESS CONVERGED IN    33 ITERATIONS.
THE CURRENT VALUE OF THE X-VECTOR IS...
  9.368E+00  1.393E+01
THE CURRENT VALUE OF THE VECTOR-FUNCTION F AT X IS...
 -4.729E-11  7.276E-12
THE PROGRAM CONSTANTS USED ARE...
THE CONVERGENCE EPSILON =  1.000E-08
THE MAXIMUM ITERATIONS ALLOWED =  200
GAIN ADJUSTED BY THE PROGRAM; FINAL GAIN =  1.000E+00
THE JACOBIAN WAS APPROXIMATED BY THE PROGRAM, WITH EPSJ = 1.000E-08

POTENTIAL ENERGY =  8.810E+03

 ALL DONE
```

Fig. 3. Teletype dialogue for NAES solution.

by the small amounts indicated by F(1) and F(2). Stated another way, the d_1 and d_2 terms are solutions to:

$$\text{Force 1} - M_1 g - 4.729 \times 10^{-11} = k_1(d_1) + k_2(d_1 - d_2) ,$$

$$\text{Force 2} - M_2 g + 7.276 \times 10^{-12} = k_2(d_2 - d_1) .$$

Since Force 1 = Force 2 = 1000, $M_1 = M_2 = 1$, and $g = 32.174$, one can ignore the slight perturbation that F(1) and F(2) will make to the solution of d_1 and d_2.

Now that the initial conditions for our differential equations (d_1, $\dot{d}_1$, d_2, and $\dot{d}_2$) are known, we can solve for the dynamic responses of interest. For most (if not all) numerical integration schemes, the differential equations must be placed in normal form (a set of first-order differential equations). To do this, first rearrange each equation such that the highest-order time derivative is isolated. For our example, solve for $\ddot{d}_1$ and $\ddot{d}_2$:

$$\ddot{d}_1 = \frac{1}{M_1} \left[\text{Force 1} - M_1 g - D_1 \dot{d}_1 - k_1(d_1) - k_2(d_1 - d_2) \right] ,$$

$$\ddot{d}_2 = \frac{1}{M_2} \left[\text{Force 2} - M_2 g - D_2 \dot{d}_2 - k_2(d_2 - d_1) \right] .$$

Note that lower-order time derivatives ($\dot{d}_1$, d_1, $\dot{d}_2$ and d_2) can be obtained by integrating $\ddot{d}_1$ and $\ddot{d}_2$:

$$\dot{d}_1(t) = \int_0^t \ddot{d}_1(n)dn ,$$

$$\dot{d}_2(t) = \int_0^t \ddot{d}_2(n)dn ,$$

$$d_1(t) = \int_0^t \dot{d}_1(n)dn ,$$

$$d_2(t) = \int_0^t \dot{d}_2(n)dn \; .$$

In block-diagram form, the above process appears as in Fig. 4(a). The differential equations are also shown in block-diagram form in Fig. 4(b). The IC signal entering each integrator indicates that each integrator has associated with it some initial condition. By assigning new variable names to the outputs of the integrators, we can transform our original differential equations into a set of first-order differential equations (normal form). Select the new variable as follows.

Let

$$x_1 = d_1 \; ,$$

$$x_2 = \dot{d}_1 \; ,$$

$$x_3 = d_2 \; ,$$

$$x_4 = \dot{d}_2 \; .$$

Then, the normal form equations are:

$$\dot{x}_1 = \dot{d}_i = x_2 \qquad \text{(i.e., } d_1 = \int \dot{d}_1 dt) \; ,$$

$$\dot{x}_2 = \ddot{d}_1 = \frac{1}{M_1}\left[\text{Force 1} - M_1 g - D_1 x_2 - k_1(x_1) - k_2(x_1 - x_3)\right] \; ,$$

$$\dot{x}_3 = \dot{d}_2 = x_4 \; ,$$

$$\dot{x}_4 = \ddot{d}_2 = \frac{1}{M_2}\left[\text{Force 2} - M_2 g - D_2 x_4 - k_2(x_3 - x_1)\right] \; .$$

This is the form required for numerical solution by Program SS. Using the same new variable names, one can likewise transform the initial conditions. For our example, these are:

$$x_1(0) = d_1(0) \; ,$$

$$x_2(0) = \dot{d}_1(0) \; ,$$

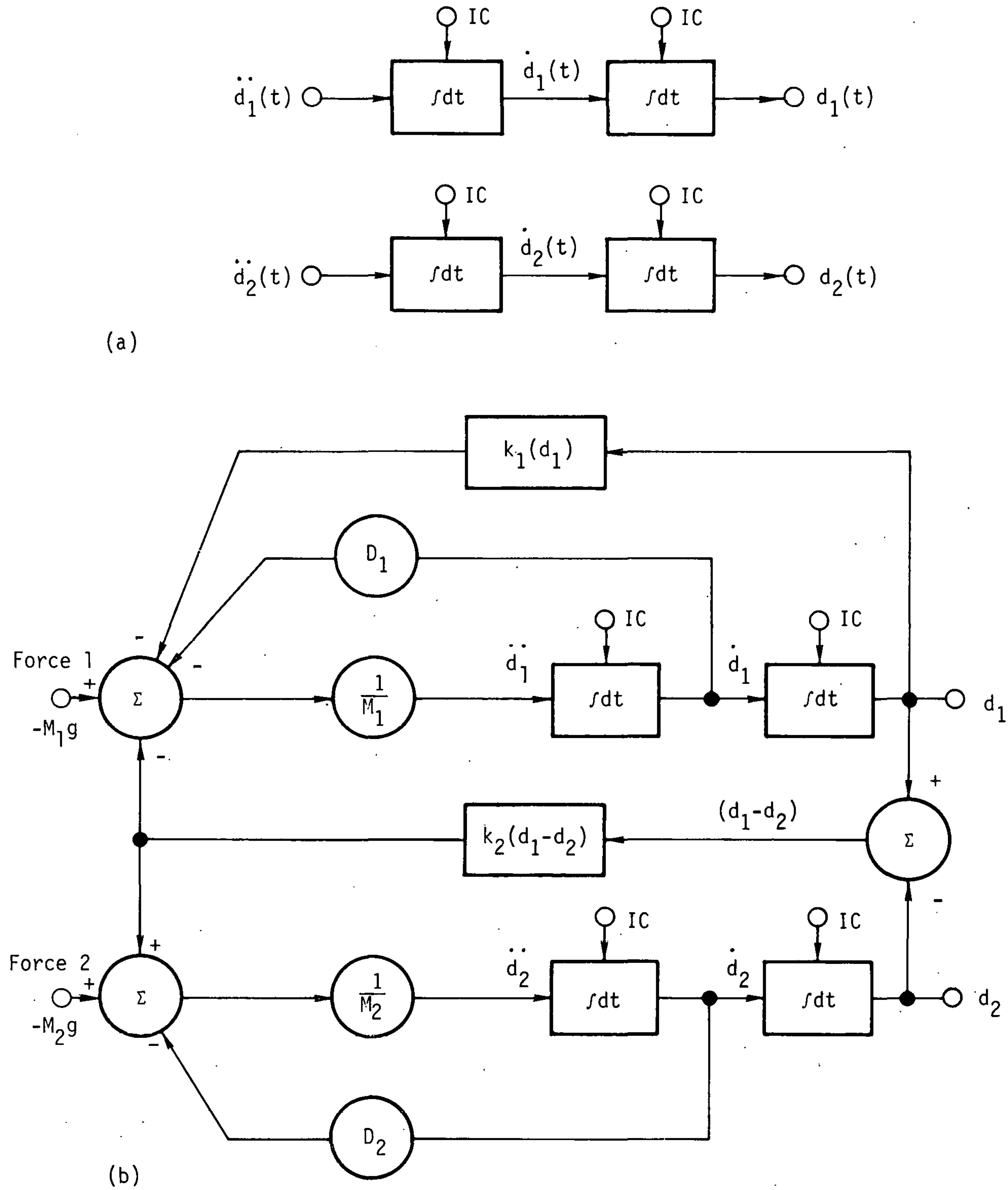

Fig. 4. Block diagrams of differential equations.

$$x_3(0) = d_2(0) \ ,$$

$$x_4(0) = \dot{d}_2(0) \ .$$

With the differential equations in normal form and with the initial conditions known, one is now prepared to numerically solve for the dynamic responses. The amount of Fortran IV coding required from the user is shown in Fig. 5, where the boxed-in lines were supplied by the user. In section 100, all specialized input-file data is read from SSIN (input file) into SS and echoed out to SSOUT (output file). SSIN contains information concerning t_{start}, t_{end}, stepsize, initial conditions, etc. The Fortran IV coding required to input and output this information is already part of Program SS.

Section 100 provides a place for the user to read specialized information from the input deck. For our example, Program SS will read the dashpot coefficients, D_1 and D_2. By placing D_1 and D_2 in the input deck, one can rerun the problem with different values of D_1 and D_2 without having to recompile Program SS. In section 200, one defines constants, performs initialization calculations, and defines output labels (used in plots and tabular listing). In this section, one can use the full power of Fortran IV coding to define the problem constants.

In our example, the forces FOR1 and FOR2 were computed. The terms will remain constant for the simulation unless modified in section 300 or 400. In section 300, the set of first-order differential equations are specified. Again, one can define Fortran variables to simplify the differential equations. In our example, Spring 1 and Spring 2 were defined to simplify the nonlinear force terms in the differential equations. Note that x_i becomes x(i) and that $\dot{x}_i$ becomes XDOT(i) in the Fortran coding.

In section 400, one defines the output variables. Any variable that one wishes to output must be equated to an element of the Y vector. For our example, the third output will be x(2) or $\dot{d}_1$; note that LABEL(3) (Velocity 1) corresponds to this output. Also note that the kinetic and potential energy terms were computed in the output section. These terms were not needed to solve the differential equations and can be computed directly from the X vector. This illustrates that Fortran can be used in the output section.

After Y(7), two calls to subroutine XYPLOT are made. These calls plot elements of the output vector against each other, with time as the parametric

```
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
100   CONTINUE
C
C THE USER INSERTS USER DEFINED INPUT READ/WRITE STATEMENTS HERE.
C THE INPUT TAPE UNIT NUMBER MUST BE NIN AND THE OUTPUT TAPE UNIT
C NUMBER MUST BE NOUT.
      READ(NIN,101)D1,D2                    Read special data from input file
101   FORMAT(2E10.3)
      WRITE(NOUT,101)D1,D2
      GO TO 999
200   CONTINUE
C
C ONE CAN DO ONE-TIME PRECALCULATIONS AND OUTPUT LABELLING IN
C THIS SECTION.
C
      REAL MASS1,MASS2                      Problem constants
      MASS1=1.0
      MASS2=1.0
      G=32.1740
      C1=75.0
      C2=1.5
      C3=150.0
      C4=3.0
      FOR1=-MASS1*G
      FOR2=-MASS2*G
C OVERWRITE THE STANDARD OUTPUT LABEL HERE.  AN EXAMPLE IS...
C     LABEL(1)=10HOUTPUT   1
      LABEL(1)=10HMASS1 DISP                Output labels
      LABEL(2)=10HMASS2 DISP
      LABEL(3)=10HVELOCITY 1
      LABEL(4)=10HVELOCITY 2
      LABEL(5)=10H KINETIC
      LABEL(6)=10HPOTENTIAL
      LABEL(7)=10HKE PLUS PE
      GO TO 999
300   CONTINUE
C
C THIS SECTION COMPUTES THE XDOT VECTOR GIVEN N, T, AND THE X-VECTOR.
C
C CALCULATE ANY INTERMEDIATE VARIABLES WHICH ARE FUNCTIONS OF THE STATES.
      SPRING1=C1*X(1)+C2*X(1)**3                          Differential equations
      SPRING2=C3*(X(1)-X(3))+C4*(X(1)-X(3))**3
C CALCULATE THE TIME DERIVATIVES OF THE STATE VARIABLES.
      XDOT(1)=X(2)
      XDOT(2)=(FOR1-D1*X(2)-SPRING1-SPRING2)/MASS1
      XDOT(3)=X(4)
      XDOT(4)=(FOR2-D2*X(4)+SPRING2)/MASS2
      GO TO 999
400   CONTINUE
C
C THE USER SPECIFIES THE VARIABLES THAT WILL BE OUTPUTTED IN THIS
C SECTION----THE OUTPUT VARIABLES ARE PLACED IN THE Y-VECTOR; THE
C Y VECTOR IS OF LENGTH M, WHERE M IS SPECIFIED IN THE INPUT
C DECK SSIN.
C
      KE=X(2)*X(2)*MASS1/2.0+X(4)*X(4)*MASS2/2.0          Output
      PE=C1*X(1)**2/2.0+C2*X(1)**4/4.0+C3*(X(1)-X(3))**2/2.0+
     $   C4*(X(1)-X(3))**4/4.0+MASS1*G*X(1)+MASS2*G*X(3)
      Y(1)=X(1)
      Y(2)=X(3)
      Y(3)=X(2)
      Y(4)=X(4)
      Y(5)=KE
      Y(6)=PE
      Y(7)=KE+PE
      CALL XYPLOT(1,1,3)
      CALL XYPLOT(2,2,4)
      GO TO 999
500   CONTINUE
C
C THIS SECTION IS PROVIDED FOR POST PROCESSING OF THE FINAL TIME DATA.
C
      GO TO 999
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig. 5. Fortran coding (in boxes) supplied by user for Program SS.

parameter. That is, CALL XYPLOT(1, 1, 3) plots Y(1) (X-AXIS) against Y(3) (Y-AXIS). Since Y(1) = d_1 and Y(3) = $\dot{d}_1$, this is a phase plane plot of MASS 1. In addition to XYPLOT, there are many other features already programmed in SS. Shown in Fig. 6 is the input deck SSIN for this problem. In this input deck are specified t_{start}, t_{end}, stepsize, initial conditions, specialized user-defined input, plot titles, etc. See Appendix B for details on inputting problems in Program SS.

Shown in Figs. 7 through 11 are selected portions of the output file, SSOUT. Figure 7 is the displacement of MASS 2 vs time. This output corresponds to setting Y(2) = x(3) in section 400. Figure 8 is the velocity of MASS 2 vs time. An X-Y plot of d_2 vs $\dot{d}_2$ was requested via the CALL XYPLOT(2, 2, 4); this plot is shown in Fig. 9. The total energy plot is shown in Fig. 10. Note that energy decreases as time progresses; this agrees with our reasoning in the problem description. Also note that the total energy at t = 0 from the SS run agrees with the energy calculation of the NAES run.

Finally, the first page of the tabular listing is shown. Note that the labels defined in SS are automatically incorporated in the plots and tabular listing. The last column is an estimate of the absolute value of the truncation error of the integration process. This error estimate is automatically outputted; see Appendix B for the details.

```
BOX R61 SS EXAMPLE  1 1 001 0 0 15000 000
THIS IS AN EXAMPLE OF TWO MASS SYSTEM WITH NONLINEAR SPRINGS IN A GRAVITY FIELD.
0.00         2.00         0.020        04 07
9.368        0.00         13.93        0.00
2.00         2.00
```

Fig. 6. SSIN input file.

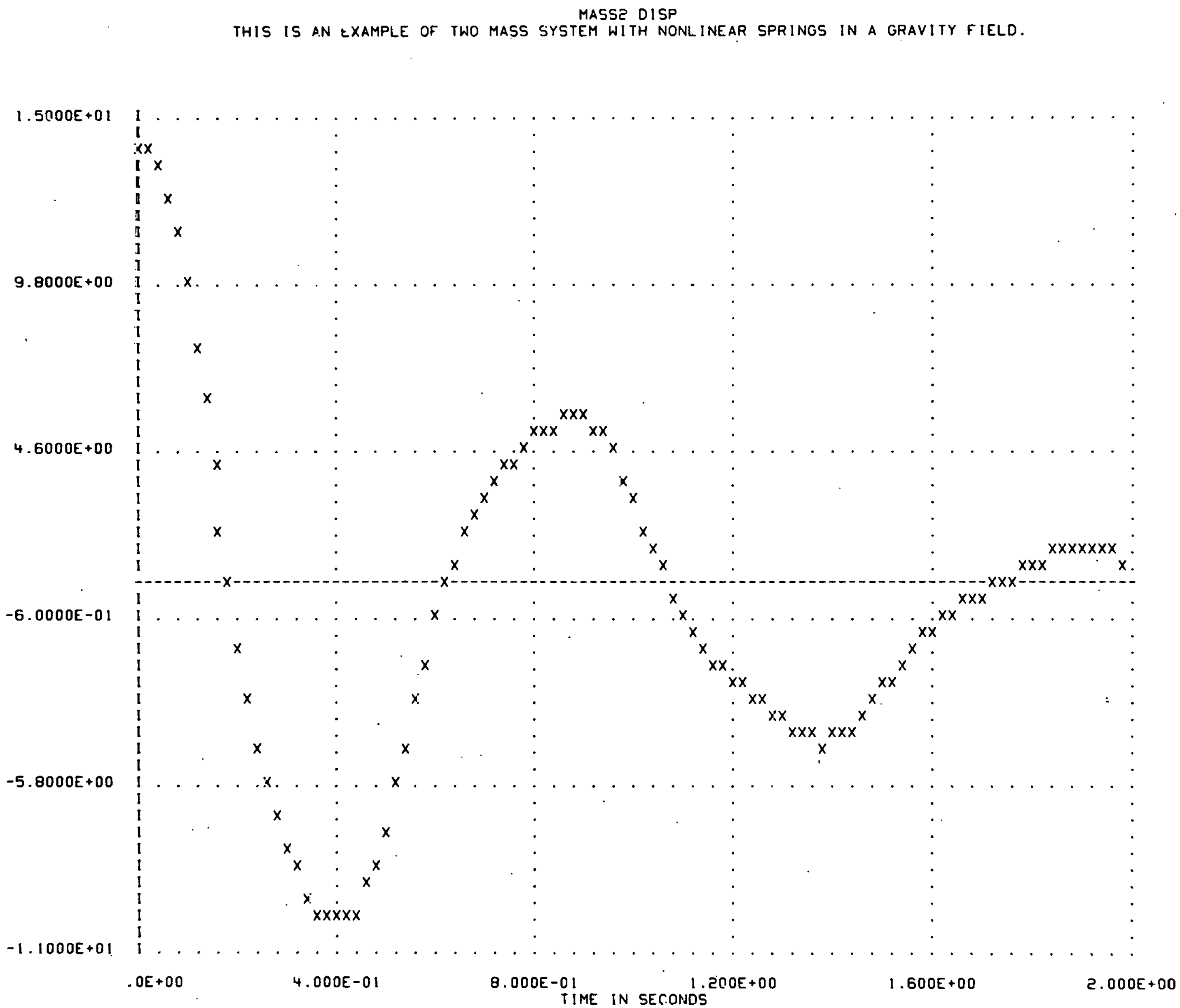

Fig. 7. Displacement of MASS 2 vs time.

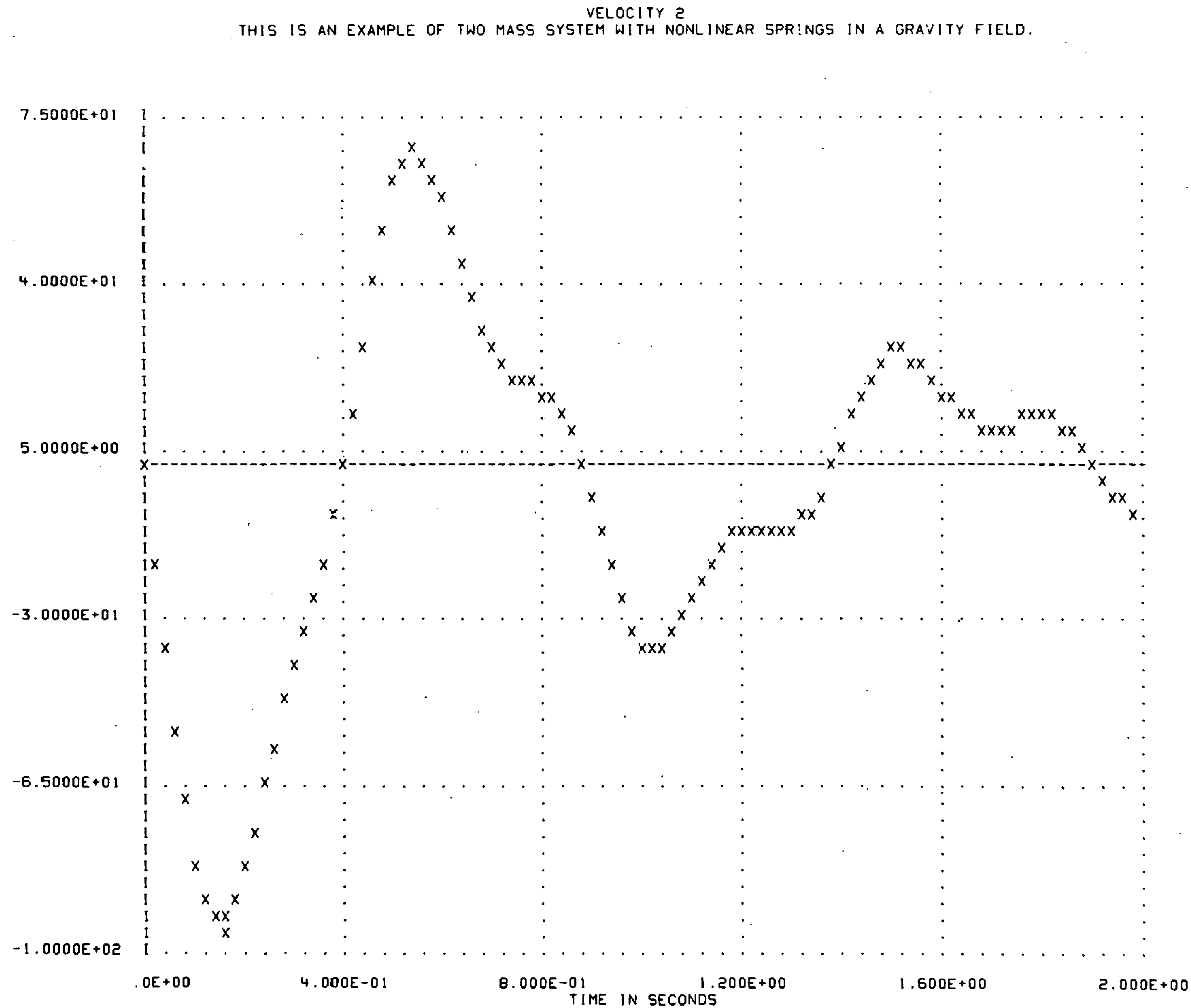

Fig. 8. Velocity of MASS 2 vs time.

VELOCITY 2 (Y-AXIS) VERSUS MASS2 DISP (X-AXIS)
THIS IS AN EXAMPLE OF TWO MASS SYSTEM WITH NONLINEAR SPRINGS IN A GRAVITY FIELD.

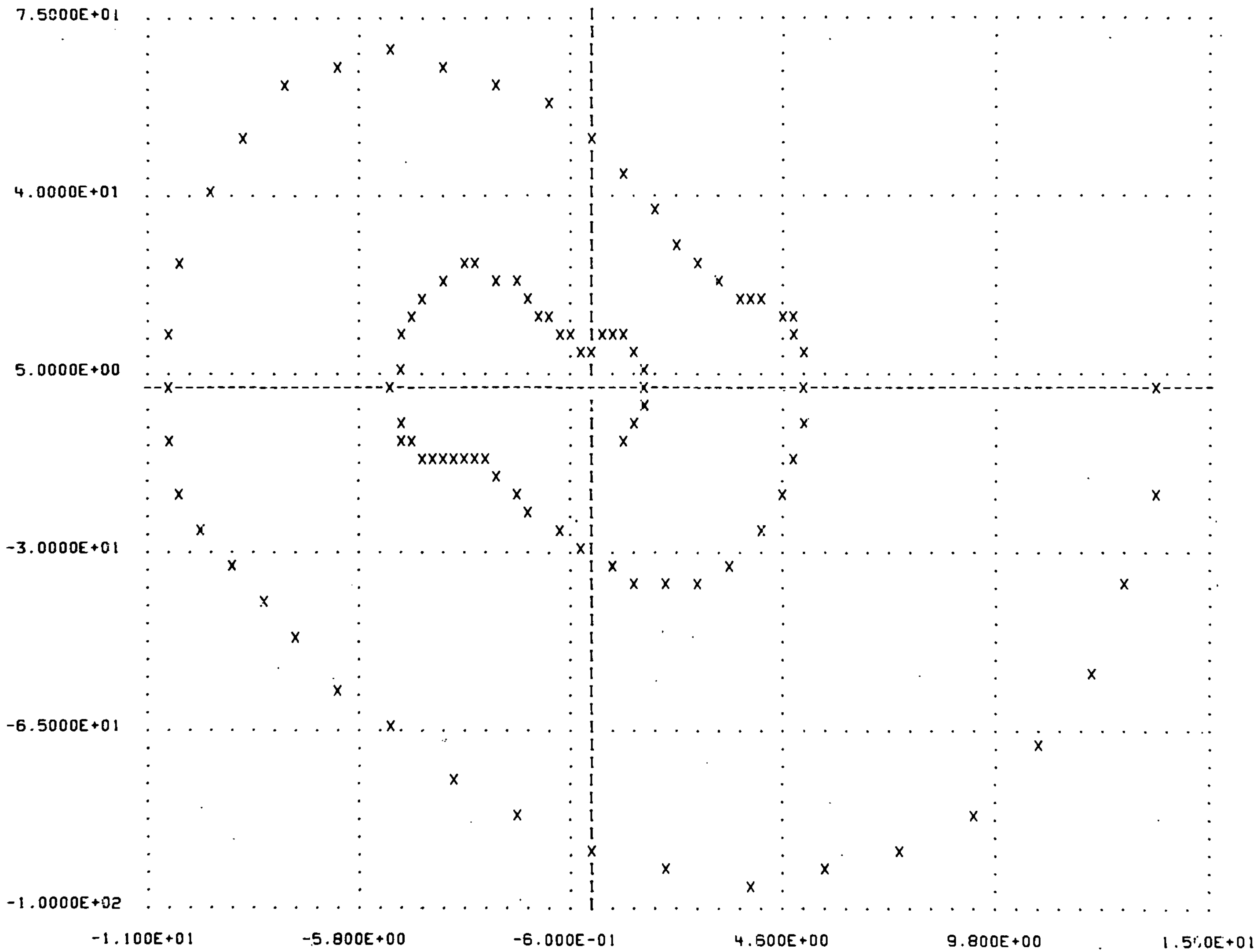

Fig. 9. Phase plane plot for MASS 2.

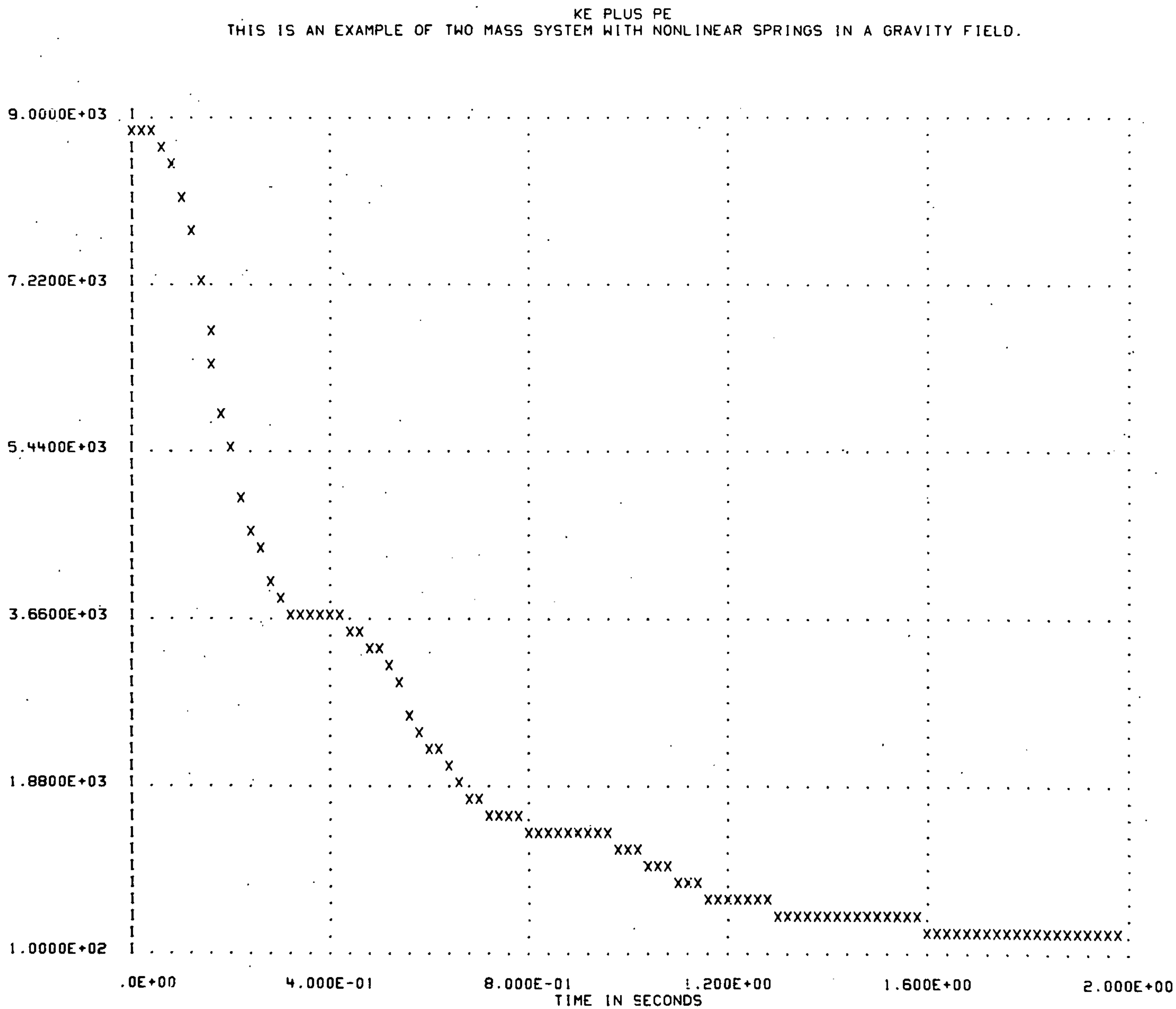

Fig. 10. Total energy of the system vs time.

TIME	MASS1 DISP	MASS2 DISP	VELOCITY 1	VELOCITY 2	KINETIC	POTENTIAL	KE PLUS PE	EST. ERROR
.0E+00	9.368E+00	1.393E+01	.0E+00	.0E+00	.0E+00	8.814E+03	8.814E+03	.0E+00
2.00E-02	9.174E+00	1.373E+01	-1.898E+01	-1.963E+01	3.720E+02	8.432E+03	8.804E+03	-1.000E+00
4.00E-02	8.635E+00	1.315E+01	-3.406E+01	-3.837E+01	1.316E+03	7.423E+03	8.739E+03	-1.000E+00
6.00E-02	7.848E+00	1.221E+01	-4.381E+01	-5.576E+01	2.514E+03	6.073E+03	8.587E+03	-1.000E+00
8.00E-02	6.915E+00	1.094E+01	-4.878E+01	-7.092E+01	3.704E+03	4.633E+03	8.337E+03	-2.000E+00
1.00E-01	5.916E+00	9.391E+00	-5.056E+01	-8.297E+01	4.720E+03	3.279E+03	7.999E+03	6.751E-04
1.20E-01	4.899E+00	7.642E+00	-5.086E+01	-9.133E+01	5.464E+03	2.126E+03	7.590E+03	2.101E-02
1.40E-01	3.881E+00	5.764E+00	-5.088E+01	-9.596E+01	5.898E+03	1.235E+03	7.133E+03	2.469E-04
1.60E-01	2.860E+00	3.826E+00	-5.124E+01	-9.721E+01	6.037E+03	6.177E+02	6.655E+03	5.828E-03
1.80E-01	1.828E+00	1.894E+00	-5.218E+01	-9.554E+01	5.925E+03	2.495E+02	6.175E+03	-2.000E+00
2.00E-01	7.694E-01	2.059E-02	-5.379E+01	-9.138E+01	5.622E+03	9.003E+01	5.712E+03	1.044E-04
2.20E-01	-3.276E-01	-1.747E+00	-5.596E+01	-8.511E+01	5.187E+03	9.150E+01	5.278E+03	2.343E-03
2.40E-01	-1.471E+00	-3.372E+00	-5.826E+01	-7.718E+01	4.675E+03	2.081E+02	4.883E+03	1.607E-05
2.60E-01	-2.654E+00	-4.827E+00	-5.985E+01	-6.823E+01	4.118E+03	4.131E+02	4.531E+03	1.053E-03
2.80E-01	-3.852E+00	-6.100E+00	-5.949E+01	-5.899E+01	3.509E+03	7.169E+02	4.226E+03	-2.000E+00
3.00E-01	-5.010E+00	-7.190E+00	-5.582E+01	-5.013E+01	2.814E+03	1.159E+03	3.973E+03	1.179E-04
3.20E-01	-6.053E+00	-8.111E+00	-4.783E+01	-4.200E+01	2.026E+03	1.753E+03	3.779E+03	1.293E-02
3.40E-01	-6.891E+00	-8.876E+00	-3.535E+01	-3.458E+01	1.222E+03	2.426E+03	3.648E+03	1.829E-04
3.60E-01	-7.443E+00	-9.496E+00	-1.944E+01	-2.748E+01	5.660E+02	3.013E+03	3.579E+03	9.980E-04
3.80E-01	-7.660E+00	-9.973E+00	-2.350E+00	-2.004E+01	2.030E+02	3.347E+03	3.550E+03	-2.000E+00
4.00E-01	-7.550E+00	-1.029E+01	1.304E+01	-1.153E+01	1.510E+02	3.387E+03	3.538E+03	4.386E-04
4.20E-01	-7.171E+00	-1.042E+01	2.416E+01	-1.311E+00	2.920E+02	3.230E+03	3.522E+03	2.493E-02
4.40E-01	-6.625E+00	-1.033E+01	2.969E+01	1.086E+01	4.990E+02	2.992E+03	3.491E+03	1.252E-04
4.60E-01	-6.022E+00	-9.976E+00	2.998E+01	2.447E+01	7.480E+02	2.694E+03	3.442E+03	9.908E-03
4.80E-01	-5.448E+00	-9.348E+00	2.691E+01	3.812E+01	1.088E+03	2.282E+03	3.370E+03	-2.000E+00
5.00E-01	-4.948E+00	-8.464E+00	2.295E+01	5.003E+01	1.514E+03	1.753E+03	3.267E+03	4.833E-04
5.20E-01	-4.518E+00	-7.371E+00	2.012E+01	5.880E+01	1.931E+03	1.199E+03	3.130E+03	2.378E-02
5.40E-01	-4.126E+00	-6.138E+00	1.937E+01	6.392E+01	2.230E+03	7.329E+02	2.963E+03	3.970E-04
5.60E-01	-3.730E+00	-4.837E+00	2.071E+01	6.562E+01	2.367E+03	4.117E+02	2.779E+03	8.169E-03
5.80E-01	-3.289E+00	-3.532E+00	2.371E+01	6.440E+01	2.355E+03	2.344E+02	2.589E+03	-2.000E+00
6.00E-01	-2.775E+00	-2.275E+00	2.787E+01	6.083E+01	2.238E+03	1.672E+02	2.405E+03	8.541E-05
6.20E-01	-2.170E+00	-1.110E+00	3.262E+01	5.543E+01	2.068E+03	1.647E+02	2.233E+03	3.042E-04
6.40E-01	-1.470E+00	-6.657E-02	3.738E+01	4.883E+01	1.890E+03	1.839E+02	2.074E+03	2.325E-05
6.60E-01	-6.797E-01	8.393E-01	4.146E+01	4.174E+01	1.730E+03	1.996E+02	1.930E+03	5.768E-04
6.80E-01	1.790E-01	1.605E+00	4.414E+01	3.490E+01	1.583E+03	2.142E+02	1.797E+03	-2.000E+00
7.00E-01	1.072E+00	2.242E+00	4.483E+01	2.892E+01	1.422E+03	2.542E+02	1.676E+03	2.635E-05
7.20E-01	1.955E+00	2.771E+00	4.309E+01	2.417E+01	1.220E+03	3.511E+02	1.571E+03	1.911E-03
7.40E-01	2.778E+00	3.217E+00	3.872E+01	2.073E+01	9.640E+02	5.191E+02	1.483E+03	1.308E-04
7.60E-01	3.487E+00	3.608E+00	3.182E+01	1.849E+01	6.770E+02	7.408E+02	1.418E+03	7.816E-04
7.80E-01	4.036E+00	3.962E+00	2.283E+01	1.710E+01	4.060E+02	9.682E+02	1.374E+03	-2.000E+00
8.00E-01	4.391E+00	4.294E+00	1.255E+01	1.609E+01	2.080E+02	1.143E+03	1.351E+03	1.857E-04
8.20E-01	4.538E+00	4.605E+00	2.105E+00	1.491E+01	1.130E+02	1.226E+03	1.339E+03	3.500E-03
8.40E-01	4.483E+00	4.886E+00	-7.331E+00	1.304E+01	1.110E+02	1.219E+03	1.330E+03	5.654E-06
8.60E-01	4.259E+00	5.118E+00	-1.474E+01	1.004E+01	1.590E+02	1.161E+03	1.320E+03	1.792E-03
8.80E-01	3.912E+00	5.278E+00	-1.948E+01	5.664E+00	2.050E+02	1.100E+03	1.305E+03	-2.000E+00
9.00E-01	3.500E+00	5.335E+00	-2.133E+01	-1.636E-01	2.270E+02	1.061E+03	1.288E+03	2.062E-04
9.20E-01	3.077E+00	5.263E+00	-2.056E+01	-7.228E+00	2.370E+02	1.032E+03	1.269E+03	7.907E-04
9.40E-01	2.690E+00	5.041E+00	-1.790E+01	-1.500E+01	2.720E+02	9.771E+02	1.249E+03	1.526E-04
9.60E-01	2.366E+00	4.663E+00	-1.437E+01	-2.268E+01	3.600E+02	8.645E+02	1.225E+03	3.042E-03
9.80E-01	2.113E+00	4.141E+00	-1.104E+01	-2.944E+01	4.940E+02	6.971E+02	1.191E+03	-2.000E+00
1.00E+00	1.917E+00	3.497E+00	-8.734E+00	-3.461E+01	6.370E+02	5.091E+02	1.146E+03	7.266E-05
1.02E+00	1.753E+00	2.770E+00	-7.925E+00	-3.786E+01	7.470E+02	3.426E+02	1.090E+03	1.799E-03
1.04E+00	1.589E+00	1.997E+00	-8.719E+00	-3.912E+01	8.030E+02	2.249E+02	1.028E+03	1.524E-04
1.06E+00	1.394E+00	1.217E+00	-1.098E+01	-3.854E+01	8.030E+02	1.607E+02	9.637E+02	2.133E-03
1.08E+00	1.142E+00	4.652E-01	-1.441E+01	-3.637E+01	7.650E 02	1.358E+02	9.008E+02	-2.000E+00
1.10E+00	8.135E-01	-2.298E-01	-1.856E+01	-3.296E+01	7.150E+02	1.263E+02	8.413E+02	6.818E-05
1.12E+00	3.990E-01	-8.482E-01	-2.287E+01	-2.880E+01	6.760E+02	1.100E+02	7.860E+02	4.953E-04

Fig. 11. Tabular output data.

APPENDIX A. PROGRAM NAES

Introduction

Program NAES (Nonlinear Algebraic Equation Solver) is a Fortran IV program used to solve the vector equation $\underline{f}(\hat{\underline{x}}) = \underline{0}$ for $\hat{\underline{x}}$. Two areas where Program NAES has proved to be useful are the solution for initial conditions and/or set points of complex systems of differential equations and system parameter identification based on steady-state equations and steady-state data. The method of solution is a modified Newton-Raphson iterative process. All information relating to a particular problem is placed in a standardized subroutine named USER. In this subroutine, one specifies program constants, vector function $\underline{f}(\underline{x})$, and an approximate value of $\hat{\underline{x}}$. Optional inputs for subroutine USER are interval constraints placed on the candidates for the solution vector $\hat{\underline{x}}$ and an analytical Jacobian. If an analytical Jacobian is not specified, the program will generate a numerical Jacobian. Program input/output is via TTY.

Examples of Usage

The following examples illustrate how one sets up a problem and converts it to Fortran coding. All user-supplied coding appears between the two + lines in the standardized subroutine USER.

Example One — Suppose one wishes to solve the following set of equations:

$$0 = x_1^{3} - 27 ,$$

$$0 = x_1 + x_2^{5} - 35 ,$$

subject to the limitations

$$0 \leq x_1 \leq 10 ,$$

$$-1 \leq x_2 \leq 10 ,$$

over the field of real numbers. The user-supplied Fortran coding to do this is shown in Fig. A-1. The boxed-in terms are user-supplied. The section

```
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
C
100   CONTINUE
C DEFINE PROGRAM CONSTANTS N, GAIN, EPSC, EPSJ, MAX, IJAC, IAUTO,
C AND ISKIP HERE.
      N=2
      GAIN=1.0
      EPSC=1.0E-08
      EPSJ=1.0E-08          A
      MAX=200
      IJAC=0
      IAUTO=0
      ISKIP=0
C DEFINE THE INITIAL X-VECTOR HERE.
      X(1)=10.0
      X(2)=-1.0             B
C DEFINE ANY ADDITIONAL PROBLEM CONSTANTS HERE.
      GO TO 999
200   CONTINUE
C THE USER SPECIFIES THE N-DIMENSIONAL VECTOR-FUNCTION F.
      F(1)=X(1)**3-27.0
      F(2)=X(1)+X(2)**5-35.0     C
      GO TO 999
300   CONTINUE
C IF IJAC.NE.0, THE USER SPECIFIES THE JACOBIAN HERE.
      GO TO 999                    D
400   CONTINUE
C SPECIFY CONSTRAINTS ON THE ELEMENTS OF THE X-VECTOR HERE.
      IF(X(1).LT.0.0)X(1)=0.0
      IF(X(2).LT.-1.0)X(2)=-1.0
      IF(X(1).GT.+10.0)X(1)=+10.0    E
      IF(X(2).GT.+10.0)X(2)=+10.0
      GO TO 999
500   CONTINUE
C THIS SECTION PROFIDES A PLACE TO CALCULATE WITH THE SOLUTION VECTOR.
      GO TO 999
C                                    F
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
```

Fig. A-1. Specialized Fortran coding for Example One.

marked A specifies convergence control variables (see section entitled "Brief Description of Method" in this appendix) and N (number of equations and number of elements in the x vector). Section B specifies the initial x vector; the user has specified $x_1 = 10$ and $x_2 = -1$ for this problem. Section C specifies the nonlinear algebraic equations. Section D provides a place for the Fortran coding of the Jacobian when an analytical Jacobian is used; for this example, a numerical (program-generated) Jacobian is used (IJAC = 0). Section E provides a place for the Fortran coding that constrains the x vector. Section F provides a place to calculate with the solution vector. The TTY dialogue for Example One is shown in Fig. A-2.

Example Two — Suppose one wishes to solve the problem of Example One using an analytical Jacobian. That is, solve

$$0 = x_1^3 - 27 ,$$

$$0 = x_1 + x_2^5 - 35 ,$$

subject to the limitations

$$0 \leq x_1 \leq 10 ,$$

$$-1 \leq x_2 \leq 10 ,$$

where the Jacobian J is

$$\begin{bmatrix} 3x_1^2 & 0 \\ 1 & 5x_2^4 \end{bmatrix} .$$

The user-supplied coding to solve this problem is shown in Fig. A-3; the TTY dialogue is shown in Fig. A-4.

Comments on Usage

- Since the algebraic equations that Program NAES solves are generally nonlinear, the vector equation $\underline{f}(\underline{x}) = \underline{0}$ may have many solutions (i.e.,

```
NAES / .2 .2
DO YOU WISH TO MODIFY CONVERGENCE VARIABLES--YES OR NO.
NO
DO YOU WISH TO MODIFY THE INITIAL X-VECTOR---YES OR NO.
NO
PROCESS CONVERGED IN   57 ITERATIONS.
THE CURRENT VALUE OF THE X-VECTOR IS...
  3.000E+00  2.000E+00
THE CURRENT VALUE OF THE VECTOR-FUNCTION F AT X IS...
 -5.684E-13 -1.364E-12
THE PROGRAM CONSTANTS USED ARE...
THE CONVERGENCE EPSILON =  1.000E-08
THE MAXIMUM ITERATIONS ALLOWED =  200
GAIN ADJUSTED BY THE PROGRAM; FINAL GAIN =  1.000E+00
THE JACOBIAN WAS APPROXIMATED BY THE PROGRAM, WITH EPSJ = 1.000E-08

 ALL DONE
```

Fig. A-2. TTY dialogue for Example One.

```
C
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
C
100    CONTINUE
C DEFINE PROGRAM CONSTANTS N, GAIN, EPSC, EPSJ, MAX, IJAC, IAUTO,
C AND ISKIP HERE.
       N=2
       GAIN=1.0
       EPSC=1.0E-08
       EPSJ=1.0E-08
       MAX-200
       IJAC=1
       IAUTO=0
       ISKIP=0
C DEFINE THE INITIAL X-VECTOR HERE.
       X(1)=10.0
       X(2)=-1.0
C DEFINE ANY ADDITIONAL PROBLEM CONSTANTS HERE.
       GO TO 999
200    CONTINUE
C THE USER SPECIFIES THE N-DIMENSIONAL VECTOR-FUNCTION F.
       F(1)=X(1)**3-27.0
       F(2)=X(1)+X(2)**5-35.0
       GO TO 999
300    CONTINUE
C IF IJAC.NE.0, THE USER SPECIFIES THE JACOBIAN HERE.
       JAC(1,1)=3.0*X(1)**2
       JAC(1,2)=0.00
       JAC(2,1)=1.00
       JAC(2,2)=5.0*X(2)**4
       GO TO 999
400    CONTINUE
C SPECIFY CONSTRAINTS ON THE ELEMENTS OF THE X-VECTOR HERE.
       IF(X(1).LT.0.0)X(1)=0.0
       IF(X(2).LT.-1.0)X(2)=-1.0
       IF(X(1).GT.+10.0)X(1)=+10.0
       IF(X(2).GT.+10.0)X(2)=+10.0
       GO TO 999
500    CONTINUE
C THIS SECTION PROFIDES A PLACE TO CALCULATE WITH THE SOLUTION VECTOR.
       GO TO 999
C
C
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
```

Fig. A-3. Specialized Fortran coding for Example Two.

```
NAES / .2 .2
DO YOU WISH TO MODIFY CONVERGENCE VARIABLES--YES OR NO.
NO
DO YOU WISH TO MODIFY THE INITIAL X-VECTOR---YES OR NO.
NO
PROCESS CONVERGED IN    61 ITERATIONS.
THE CURRENT VALUE OF THE X-VECTOR IS...
  3.000E+00  2.000E+00
THE CURRENT VALUE OF THE VECTOR-FUNCTION F AT X IS...
     .0E+00      .0E+00
THE PROGRAM CONSTANTS USED ARE...
THE CONVERGENCE EPSILON =  1.000E-08
THE MAXIMUM ITERATIONS ALLOWED =  200
GAIN ADJUSTED BY THE PROGRAM; FINAL GAIN =  1.000E+00
THE JACOBIAN WAS SUPPLIED BY THE USER.

 ALL DONE
```

Fig. A-4. TTY dialogue for Example Two.

$f(x) = x^2 - 1 = 0$). On the other hand, it may have no solution (over the field of real numbers) (i.e., $f(x) = x^2 + 1 = 0$). In addition, it may happen that a solution exists, but that Program NAES is not capable of solving for it.

- When Program NAES does not converge on a solution, try a different initial x vector, EPSJ, GAIN, etc. These are adjustable at the TTY.
- When the program estimates the Jacobian, there is an additional noise level added to the Jacobian. Since this Jacobian is used in an iterative process, this error typically does not affect the result, if convergence is obtained. If the error does affect the calculation, one must use an analytical Jacobian rather than a numerical Jacobian.
- The numerical Jacobian is only approximate but for large N saves the user from coding the Jacobian. This usually means a savings in debugging time.
- One can improve the accuracy of the numerical Jacobian by making EPSJ smaller up to the point of machine noise and number representation. For the CDC 6600 and CDC 7600 machines, one can represent 14.4 significant figures. If EPSJ is so small that changes in F occur in the fifteenth or higher significant figures, this information is lost. In addition, the calculations performed on the machines add noise. Based on the above and experimental running, EPSJ = 1.0×10^{-8} appears to be a reasonable starting value.
- As presently dimensioned, N must satisfy $1 \leq N \leq 20$.

<u>Brief Description of Method</u>

Suppose one has a vector function f such that

$$f : \mathbb{R}^N \rightarrow \mathbb{R}^N ,$$

that is differentiable, and can be written in Fortran. For this function one seeks an $\hat{x}$ vector in $\mathbb{R}^N$ (not necessarily unique) such that

$$\underline{f}(\hat{\underline{x}}) = \underline{0}.$$

One approach to solving for $\hat{\underline{x}}$ is the Newton-Raphson iteration. The basic idea is as follows. Assume Taylor's theorem applies, one can then expand $\underline{f}$ about a point $\underline{x}$, where $\underline{x} \neq \hat{\underline{x}}$. The result is:

$$0 = \underline{f}(\hat{\underline{x}}) = \underline{f}(\underline{x}) + \frac{\partial f}{\partial x} (\underline{x}) \; [\hat{\underline{x}} - \underline{x}] + \text{H.O.T.}$$

where H.O.T. denotes higher-order terms. Ignoring the H.O.T., one can solve for $\tilde{\underline{x}}$ as follows:

$$\tilde{\underline{x}} = \underline{x} - \left[\frac{\partial f}{\partial x} (\underline{x})\right]^{-1} f(\underline{x}) \; .$$

Since, in general, the H.O.T. contribute some information, $\tilde{\underline{x}}$ only approximates the true solution, $\hat{\underline{x}}$. Assuming the process converges, one can improve on this approximate $\tilde{\underline{x}}$ by using an iterative procedure based on the above approximate equation. Program NAES uses

$$\underline{x}_{N+1} = \underline{x}_N - (\text{GAIN}) \left[\frac{\partial f}{\partial x} (\underline{x})\right]^{-1} f(\underline{x}) \; .$$

GAIN is a user-specified convergence control term. For GAIN equal to one, this is the standard Newton-Raphson method. For efficiency, the computation of the inverse matrix is not performed, but rather the whole term

$$\left[\frac{\partial f}{\partial x} (\underline{x})\right]^{-1} f(\underline{x})$$

is computed via Gauss Elimination. Thus, given $\underline{x}_0$ (the initial x vector specified by the user), one can solve for $\underline{x}_1$; given $\underline{x}_1$, one can solve for $\underline{x}_2$, etc. There are two ways that this process can be halted. One, the maximum number of iterations specified by the user (MAX) is exceeded. Two, the process is judged to have converged. For this program, the process is said to have converged when each element of $\underline{f}$ and $\underline{x}$ satisfies:

$$\left| f_i(\underline{x}_N) \right| < \text{ESPC} \qquad i = 1,\ldots,N$$

and

$$\left| x_N(i) - x_{N-1}(i) \right| < \text{ESPC} \qquad i = 1,\ldots,N$$

where EPSC is a user-specified convergence variable. Whenever the process is halted, Program NAES prints out the current state of the iteration.

If IJAC equals zero, the program will approximate the Jacobian term via sequential perturbation of the x vector; EPSJ controls the amount of the perturbation. If IAUTO equals zero, the program will automatically adjust the value of GAIN; otherwise the value of GAIN is fixed. If ISKIP is less than or equal to zero, no intermediate printout occurs; otherwise, ISKIP is the ratio of iteration points to printout points.

Fortran IV Listing of Program NAES

A listing of the Fortran coding for Program NAES follows.

```
      PROGRAM NAES
C
C
C FOR A WRITEUP ON THE USAGE OF THIS PROGRAM, SEE UCRL-51657.
C
C
C PROGRAM NAES (NONLINEAR ALGEBRAIC EQUATION SOLVER) ATTEMPTS TO
C ITERATIVELY SOLVE (VIA NEWTON-RAPHSON METHOD) FOR THE N-DIMENSIONAL
C SOLUTION VECTOR X SUCH THAT THE N-DIMENSIONAL VECTOR-FUNCTION F
C EQUALS ZERO, THAT IS...
C     F:R**N---)R**N, DIFFERENTIABLE, CAN BE WRITTEN IN FORTRAN
C AND ONE SEEKS AN X-VECTOR IN R**N (NOT NECESSARILY UNIQUE) SUCH THAT
C     F(X) = 0
C
C PROVISIONS ARE MADE TO PLACE INTERVAL CONSTRAINTS ON THE ELEMENTS OF
C THE X-VECTOR.  THE JACOBIAN REQUIRED BY THE PROGRAM CAN BE SUPPLIED
C VIA FORTRAN STATEMENTS (ANALYTIC) OR ESTIMATED FROM FUNCTION F BY THE
C PROGRAM (NUMERICAL).  THE USER MUST SUPPLY AN INITIAL X-VECTOR.  AS
C PRESENTLY DIMENSIONED, PROGRAM NAES CAN SOLVE PROBLEMS WITH
C UNKNOWNS RANGING FROM 1 TO 20.  ALL USER CODING GOES IN SUB-
C ROUTINE USER.
C
C THIS PROGRAM WAS WRITTEN BY HOWARD MCCUE AS PART OF EECS 299 (THESIS)
C AT THE UNIVERSITY OF CALIFORNIA AT BERKELEY UNDER PROF. OTTO SMITH.
C
C IMPORTANT VARIABLES OF THIS PROGRAM ARE...
C   N        THE DIMENSIONAL OF THE PARTICULAR PROBLEM (I.E. 1 TO 20)
C   X        THE N-DIMENSIONAL ITERATION-VECTOR
C   F        THE N-DIMENSIONAL VECTOR-FUNCTION OF THE PARTICULAR
C            PROBLEM
C   JAC      THE N-BY-N JACOBIAN OF F
C   GAIN     THE NEWTON-RAPHSON ITERATION GAIN
C   EPSC     EPSILON USED TO JUDGE CONVERGENCE OF X-VECTOR
C   EPSJ     EPSILON USED TO APPROXIMATE THE JACOBIAN
C   MAX      MAXIMUM NUMBER OF NEWTON-RAPHSON ITERATIONS ALLOWED
C   IJAC     =0 MEANS JACOBIAN APPROXIMATED FROM F BY PROGRAM; OTHERWISE,
C            THE USER MUST PROVIDE FORTRAN CODING.
C   IAUTO    =0 MEANS THE GAIN TERM IS AUTOMATICALLY ADJUSTED; OTHER-
C            WISE, THE GAIN IS FIXED AT THE USER SPECIFIED VALUE.
C   ISKIP    .LE.0 MEANS NO INTERMEDIATE PRINTOUT.  OTHERWISE, THE
C            POSITIVE NUMBER IS THE RATIO OF THE CALCULATED ITERATIONS
C            TO PRINTOUT ITERATIONS.
C
C
C
      CALL CHANGE(5H+NAES)
      REAL X(20),F(20),JAC(20,20),Z(20),ERROR(20),XOLD(20),E(20)
      DIMENSION LABEL(20),FOLD(20)
      DATA (LABEL(I),I=1,20)/ 10H1111111111,10H2222222222,10H3333333333,
```

```
     210H4444444444,10H5555555555,10H6666666666,10H7777777777,
     310H8888888888,10H9999999999,10H1010101010,10H1111111111,
     410H1212121212,10H1313131313,10H1414141414,10H1515151515,
     510H1616161616,10H1717171717,10H1818181818,10H1919191919,
     610H2020202020 /
      NTTY=59
      NIN=NTTY
      NOUT=NTTY
      COMMON/IO/NIN,NOUT
C GET INITIAL X-VECTOR  AND OTHER PROGRAM CONSTANTS.
      CALL USER(1,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
C CHECK FOR N IN THE PROPER RANGE.
      NMAX=20
      IF((N.GT.0).AND.(N.LE.NMAX))GO TO 6
      WRITE(NOUT,7)
7     FORMAT(27HN IS .LE.0 OR GREATER THAN ,I3,2H .)
      GO TO 999
6     CONTINUE
C CHECK TO SEE IF MODIFICATIONS REQUESTED.
      WRITE(NOUT,1)
1     FORMAT(55HDO YOU WISH TO MODIFY CONVERGENCE VARIABLES--YES OR NO.)
      READ(NIN,2)ANS
2     FORMAT(A3)
      IF(ANS.NE.3HYES)GO TO 5
      WRITE(NOUT,3)
3     FORMAT(50H+++GAIN+++-CONV-EPS-+JAC-EPS++-MAX-ITS--SKIP RATIO,
     210H-GAIN MODE)
      READ(NIN,4)A,B,C,D,SKIP,AUTO
4     FORMAT(6E10.3)
      ID=D+0.1
      ISKIP=SKIP+0.1
      IF(AUTO.GT.0.0)IAUTO=1
      IF(AUTO.LT.0.0)IAUTO=0
      IF(ID.LE.0)ID=0
      IF(A.NE.0.0)GAIN=A
      IF(B.NE.0.0)EPSC=B
      IF(C.NE.0.0)EPSJ=C
      IF(ID.NE.0)MAX=ID
5     CONTINUE
C MODIFY GAIN, EPSC, EPSJ, AND MAX AS REQUIRED.
      EPSC=ABS(EPSC)
      EPSJ=ABS(EPSJ)
      IF(GAIN.EQ.0.0)GAIN=1.0
      IF(EPSC.EQ.0.0)EPSC=1.0E-06
      IF(EPSJ.EQ.0.0)EPSJ=1.0E-06
      IF(MAX.LE.0)MAX=1
      WRITE(NOUT,41)
41    FORMAT(55HDO YOU WISH TO MODIFY THE INITIAL X-VECTOR---YES OR NO.)
      READ(NIN,2)ANS
      IF(ANS.NE.3HYES)GO TO 45
      ID=N
      IF(N.GT.7)ID=7
      WRITE(NOUT,42)(LABEL(I),I=1,ID)
42    FORMAT(7A10)
      READ(NIN,43)(X(I),I=1,7)
43    FORMAT(7E10.3)
      IF(N.LE.7)GO TO 45
      ID=N
      IF(N.GT.14)ID=14
      WRITE(NOUT,42)(LABEL(I),I=8,ID)
      READ(NIN,43)(X(I),I=8,14)
      IF(N.LE.14)GO TO 45
      ID=N
      IF(N.GT.20)ID=20
      WRITE(NOUT,42)(LABEL(I),I=15,ID)
      READ(NIN,43)(X(I),I=15,20)
```

```
45    CONTINUE
      IPRINT=-ISKIP
C
C
C THIS LOOP DOES THE NEWTON-RAPHSON ITERATION.
C
C
C INITIALIZE F FOR THE FIRST PASS THROUGH DO LOOP 100.
      CALL USER(2,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
      DO 100 I=1,MAX
C STORE THE PREVIOUS VALUE OF X AND F.
      DO 10 J=1,N
      XOLD(J)=X(J)
      FOLD(J)=F(J)
10    CONTINUE
C GET THE VALUE OF F(X)
      CALL USER(2,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
      IF(IJAC.NE.0)GO TO 15
C GET A NUMERICAL APPROXIMATION TO THE JACOBIAN.
      DO 16 K=1,N
      X(K)=X(K)+EPSJ
      CALL USER(2,N,X,Z,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
      DO 17 J=1,N
      JAC(J,K)=(Z(J)-F(J))/EPSJ
17    CONTINUE
      X(K)=X(K)-EPSJ
16    CONTINUE
      GO TO 18
15    CONTINUE
C EVALUATE AN ANALYTIC EXPRESSION FOR THE JACOBIAN.
      CALL USER(3,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
18    CONTINUE
C SOLVE FOR THE CORRECTION TERM OF THE NEWTON-RAPHSON ITERATION.
      CALL GAUSS(N,JAC,F,ERROR,NOUT,IFLAG)
      IF(IFLAG.NE.0)GO TO 120
C VALID CORRECTION TERM CALCULATED; UPDATE THE ITERATION VECTOR.
      DO 20 J=1,N
      X(J)=X(J)-GAIN*ERROR(J)
20    CONTINUE
C IMPOSE CONSTRAINTS ON THE ELEMENTS OF THE ITERATION VECTOR.
      CALL USER(4,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
C UPDATE VALUE OF VECTOR FUNCTION F BASED ON CONSTRAINTED X VECTOR.
      CALL USER(2,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
C TEST FOR CONVERGENCE.
      DO 30 J=1,N
C CHECK THE RATE THAT X IS CHANGING.
      XX=ABS(X(J)-XOLD(J))
      IF(XX.GT.EPSC)GO TO 50
C CHECK THE CLOSENESS OF F(X) TO ZERO---THIS IS NEEDED WHEN
C X SATURATES ON THE CONSTRAINTS.
      XX=ABS(F(J))
      IF(XX.GT.EPSC)GO TO 50
30    CONTINUE
C ITERATIVE PROCESS IS JUDGED TO HAVE CONVERGED
      GO TO 130
      LOOP=0
50    CONTINUE
      IF(IAUTO.NE.0)GO TO 60
C CHECK FOR THE PROPER VALUE OF GAIN---ADJUST AS REQUIRED.
      DO 51 J=1,N
      XX=ABS(FOLD(J))-ABS(F(J))
      IF(XX.LT.0.0)GO TO 55
51    CONTINUE
C NO ELEMENT OF F INCREASED IN MAGNITUDE SINCE THE LAST ITERATION---
C GAIN VALUE JUDGED NOT TOO LARGE.  CHECK IF GAIN SHOULD BE INCREASED.
```

```
      DO 52 J=1,N
      IF(F(J).EQ.0.0)GO TO 52
      XX=ABS(FOLD(J)/F(J))
      IF(XX.GT.2.00)GO TO 60
52    CONTINUE
C GAIN IS JUDGED TO BE TOO SMALL.
      GAIN=2.0*GAIN
      IF(GAIN.GT.1000.0)GAIN=1000.0
      GO TO 60
55    CONTINUE
C THE GAIN IS JUDGED TO BE TOO LARGE.
      GAIN=GAIN/2.0
      IF(GAIN.LT.0.00001)GAIN=0.00001
56    CONTINUE
      LOOP=LOOP+1
      DO 57 J=1,N
      X(J)=XOLD(J)-GAIN*ERROR(J)
57    CONTINUE
C IMPOSE CONSTRAINTS ON THE ELEMENTS OF THE ITERATION VECTOR.
      CALL USER(4,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
C UPDATE VALUE OF VECTOR FUNCTION F BASED ON CONSTRAINTED X VECTOR.
      CALL USER(2,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
      IF(LOOP.GT.30)GO TO 60
      GO TO 50
60    CONTINUE
      IF(ISKIP.LE.0)GO TO 100
      IPRINT=IPRINT+1
      IF(IPRINT.LT.0)GO TO 100
C WRITE OUT THE CURRENT N, GAIN, X-VECTOR, AND F(X)-VECTOR.
      WRITE(NOUT,61)I,GAIN
61    FORMAT(2HI=,I4,3X,5HGAIN=,E10.3)
      WRITE(NOUT,141)(X(J),J=1,N)
      WRITE(NOUT,141)(F(J),J=1,N)
      IPRINT=-ISKIP
100   CONTINUE
C
C
      WRITE(NOUT,111)I
111   FORMAT(32HTHE PROCESS DID NOT CONVERGE IN ,I4,12H ITERATIONS.)
      GO TO 200
120   CONTINUE
      WRITE(NOUT,121)I
121   FORMAT(45HCAN NOT SOLVE FOR ERROR VIA SUBROUTINE GAUSS.,/,
     225HTHE NUMBER OF ITERATIONS=,I4)
      GO TO 200
130   CONTINUE
      WRITE(NOUT,131)I
131   FORMAT(21HPROCESS CONVERGED IN ,I4,12H ITERATIONS.)
      ICONV=1
      GO TO 200
200   CONTINUE
C WRITE OUT THE RESULTS.
      WRITE(NOUT,140)
140   FORMAT(39HTHE CURRENT VALUE OF THE X-VECTOR IS...)
      WRITE(NOUT,141)(X(I),I=1,N)
141   FORMAT(6(1X,E10.3))
      WRITE(NOUT,142)
142   FORMAT(53HTHE CURRENT VALUE OF THE VECTOR-FUNCTION F AT X IS...)
      WRITE(NOUT,141)(F(I),I=1,N)
      WRITE(NOUT,143)
```

```
143   FORMAT(33HTHE PROGRAM CONSTANTS USED ARE...)
      WRITE(NOUT,144)EPSC,MAX
144   FORMAT(26HTHE CONVERGENCE EPSILON = ,E10.3,/,
     233HTHE MAXIMUM ITERATIONS ALLOWED = ,I4)
      IF(IAUTO.EQ.0)WRITE(NOUT,147)GAIN
      IF(IAUTO.NE.0)WRITE(NOUT,148)GAIN
147   FORMAT(43HGAIN ADJUSTED BY THE PROGRAM; FINAL GAIN = ,E10.3)
148   FORMAT(14HGAIN FIXED AT ,E10.3)
      IF(IJAC.EQ.0)WRITE(NOUT,145)EPSJ
      IF(IJAC.NE.0)WRITE(NOUT,146)
145   FORMAT(45HTHE JACOBIAN WAS APPROXIMATED BY THE PROGRAM,
     212H WITH EPSJ = ,E10.3)
146   FORMAT(38HTHE JACOBIAN WAS SUPPLIED BY THE USER.)
      IF(ICONV.EQ.1)GO TO 300
      WRITE(NOUT,149)
149   FORMAT(3/,38HDO YOU WISH TO CONTINUE *---YES OR NO.)
      READ(NIN,2)ANS
      IF(ANS.EQ.3HYES)GO TO 6
300   CALL USER(5,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,ISKIP)
999   CONTINUE
      CALL EXIT
      END


      SUBROUTINE GAUSS(N,A,B,X,NOUT,IFLAG)
C
C SUBROUTINE GAUSS SOLVES THE VECTOR EQUATION A*X=B FOR THE X VECTOR
C GIVEN THAT THE A MATRIX AND B VECTOR ARE KNOWNS AND THAT THE
C A MATRIX HAS FULL RANK.  PROBLEMS MAY OCCUR FOR NEAR-SINGULAR A
C MATRICES; IF SO, ERROR MESSAGES ARE PRINTED AND IFLAG IS
C MADE NONZERO.  A,B, AND X ARE DEFINED OVER THE FIELD OF REAL
C NUMBERS.  INPUT/OUTPUT IS AS FOLLOWS...
C        N IS THE SYSTEM ORDER
C        A IS SYSTEM MATRIX
C        B IS INPUT VECTOR
C        X IS SOLUTION VECTOR
C        NOUT IS THE LOGICAL TAPE UNIT NUMBER
C        IFLAG=0    GAUSS ELIMINATION PERFORMED
C        IFLAG=1    GAUSS ELIMINATION CAN NOT BE PERFORMED
C
C
C THIS SUBROUTINE IS TAKEN FROM COMPUTER SOLUTION OF LINEAR ALGEBRAIC
C SYSTEMS BY G. FORSYTHE AND C. B. MOLER, PRENTICE-HALL  1967, PP 68-70.
C MODIFICATIONS WERE MADE TO THIS SUBROUTINE TO CHANGE THE MANNER
C IN WHICH ERROR MESSAGES ARE HANDLED.
C
C
C TO CHANGE THE MAXIMUM SIZE MATRIX THAT ONE CAN HANDLE, CHANGE
C THE VALUE OF NMAX IN THIS SUBROUTINE AND ALL DIMENSION STATEMENTS
C IN THIS SUBROUTINE PLUS SUBROUTINES DECOMP, SOLVE, AND IMPRUV.
      NMAX=20
      DIMENSION A(20,20),UL(20,20),B(20),X(20)
      IFLAG=0
C CHECK THE VALUE OF N
      IF((N.GT.0).AND.(N.LE.NMAX))GO TO 40
      IFLAG=1
      WRITE(NOUT,14)
14    FORMAT(38HIN A CALL TO GAUSS, N IS OUT OF RANGE.)
      GO TO 999
40    CONTINUE
```

```
      IF(N.NE.1)GO TO 41
      X=B(1)/A(1,1)
      GO TO 999
41    CONTINUE
C DECOMPOSE MATRIX A INTO UPPER AND LOWER TRIANGLE MATRICES, STORE IN UL
      CALL DECOMP(N,A,UL,IFLAG)
      IF(IFLAG.NE.0)GO TO 10
C SOLVE SYSTEM OF EQUATIONS USING U AND L MATRICES.
      CALL SOLVE(N,UL,B,X)
C USE IMPROVEMENT TO CONVERGE ON TRUE ANSWER.
      CALL IMPRUV(N,A,UL,B,X,DIGITS,IFLAG)
10    CONTINUE
      IFLAG=IFLAG+1
      GO TO(1,2,3,4),IFLAG
2     WRITE(NOUT,11)
   11 FORMAT(54H0MATRIX WITH ZERO ROW IN DECOMPOSE.                    )
      GO TO 1
3     WRITE(NOUT,12)
   12 FORMAT(54H0SINGULAR MATRIX IN DECOMPOSE.  ZERO DIVIDE IN SOLVE.  )
      GO TO 1
4     WRITE(NOUT,13)
   13 FORMAT(54H0NO CONVERGENCE IN IMPRUV. MATRIX IS NEARLY SINGULAR.  )
1     CONTINUE
      IFLAG=IFLAG-1
999   CONTINUE
      RETURN
      END


      SUBROUTINE DECOMP (NN, A, UL, IFLAG)
      DIMENSION A(20,20), UL(20,20), SCALES(20), IPS(20)
      COMMON / AA / IPS
      N = NN
C
C     INITIALIZE IPS, UL AND SCALES
      DO 5 I = 1,N
         IPS(I) = I
         ROWNRM = 0.0
         DO 2 J = 1,N
            UL(I,J) = A(I,J)
            IF(ROWNRM-ABS(UL(I,J))) 1,2,2
    1       ROWNRM = ABS(UL(I,J))
    2    CONTINUE
         IF (ROWNRM) 3,4,3
    3    SCALES(I) = 1.0/ROWNRM
         GO TO 5
    4    IFLAG=1
         GO TO 19
    5 CONTINUE
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
      NM1 = N-1
      DO 17 K = 1,NM1
         BIG = 0.0
         DO 11 I = K,N
            IP = IPS(I)
            SIZE = ABS(UL(IP,K))*SCALES(IP)
            IF (SIZE-BIG) 11,11,10
   10          BIG = SIZE
               IDXPIV = I
```

```
   11     CONTINUE
          IF (BIG) 13,12,13
   12        IFLAG=2
             GO TO 19
   13     IF (IDXPIV-K) 14,15,14
   14        J = IPS(K)
             IPS(K) = IPS(IDXPIV)
             IPS(IDXPIV) = J
   15     KP = IPS(K)
          PIVOT = UL(KP,K)
          KP1 = K+1
          DO 16 I = KP1,N
             IP = IPS(I)
             EM = -UL(IP,K)/PIVOT
             UL(IP,K) = -EM
             DO 16 J = KP1,N
                UL(IP,J) = UL(IP,J) + EM*UL(KP,J)
C            INNER LOOP.  USE MACHINE LANGUAGE CODING IF COMPILER
C         DOES NOT PRODUCE EFFICIENT CODE.
   16     CONTINUE
   17 CONTINUE
      KP = IPS(N)
      IF (UL(KP,N)) 19,18,19
   18 IFLAG=2
19    CONTINUE
      RETURN
      END


      SUBROUTINE SOLVE (NN, UL, B, X)
      DIMENSION UL(20,20), B(20), X(20), IPS(20)
      COMMON / AA / IPS
      N = NN
      NP1 = N+1
C
      IP = IPS(1)
      X(1) = B(IP)
      DO 2 I = 2,N
         IP = IPS(I)
         IM1 = I-1
         SUM = 0.0
         DO 1 J = 1,IM1
    1       SUM = SUM + UL(IP,J)*X(J)
    2 X(I) = B(IP) - SUM
C
      IP = IPS(N)
      X(N) = X(N)/UL(IP,N)
      DO 4 IBACK = 2,N
      I = NP1-IBACK
C        I GOES (N-1),....,1
         IP = IPS(I)
         IP1 = I+1
         SUM = 0.0
         DO 3 J = IP1,N
    3       SUM = SUM + UL(IP,J)*X(J)
    4 X(I) = (X(I)-SUM)/UL(IP,I)
      RETURN
      END
```

```
      SUBROUTINE IMPRUV (NN, A, UL, B, X, DIGITS, IFLAG)
      DIMENSION A(20,20), UL(20,20), B(20), X(20), R(20), DX(20)
C     USES ABS(), AMAX1(), ALOG10()
      DOUBLE PRECISION SUM
      N = NN
C
      EPS = 2.**(-47)
      ITMAX = 29
C     +++  EPS AND ITMAX ARE MACHINE DEPENDENT. +++
C
      XNORM = 0.0
      DO 1 I = 1,N
    1    XNORM = AMAX1(XNORM,ABS(X(I)))
      IF (XNORM) 3,2,3
    2    DIGITS = -ALOG10(EPS)
         GO TO 10
C
    3 DO 9 ITER = 1,ITMAX
         DO 5 I = 1,N
            SUM = 0.0
            DO 4 J = 1,N
    4          SUM = SUM + A(I,J)*X(J)
            SUM = B(I) - SUM
    5    R(I) = SUM
C        +++ IT IS ESSENTIAL THAT A(I,J)*X(J) YIELD A DOUBLE PRECISION
C             RESULT AND THAT THE ABOVE + AND - BE DOUBLE PRECISION.  +++
         CALL SOLVE (N,UL,R,DX)
         DXNORM = 0.0
         DO 6 I = 1,N
            T = X(I)
            X(I) = X(I) + DX(I)
            DXNORM = AMAX1(DXNORM,ABS(X(I)-T))
    6    CONTINUE
         IF (ITER-1) 8,7,8
    7       DIGITS = -ALOG10(AMAX1(DXNORM/XNORM,EPS))
    8    IF (DXNORM-EPS*XNORM) 10,10,9
    9 CONTINUE
C     ITERATION DID NOT CONVERGE
      IFLAG=3
10    CONTINUE
      RETURN
      END


      SUBROUTINE USER(MODE,N,X,F,JAC,GAIN,EPSC,EPSJ,MAX,IJAC,IAUTO,
     2ISKIP)
C
C IN THIS SUBROUTINE, THE USER SPECIFIES THE PARTICULAR PROBLEM.  THE
C INPUT/OUTPUT IS AS FOLLOWS...
C   MODE    THIS BRANCHES PROGRAM TO VARIOUS PARTS OF THE SUBROUTINE.
C   N       THE DIMENSIONAL OF THE PARTICULAR PROBLEM (I.E. 1 TO 20)
C   X       THE N-DIMENSIONAL ITERATION-VECTOR
C   F       THE N-DIMENSIONAL VECTOR-FUNCTION OF THE PARTICULAR
C           PROBLEM
C   JAC     THE N-BY-N JACOBIAN OF F
C   GAIN    THE NEWTON-RAPHSON ITERATION GAIN
C   EPSC    EPSILON USED TO JUDGE CONVERGENCE OF X-VECTOR
C   EPSJ    EPSILON USED TO APPROXIMATE THE JACOBIAN
C   MAX     MAXIMUM NUMBER OF NEWTON-RAPHSON ITERATIONS ALLOWED
C   IJAC    =0 MEANS JACOBIAN APPROXIMATED FROM F BY PROGRAM; OTHERWISE,
```

```
C             THE USER MUST PROVIDE FORTRAN CODING.
C    IAUTO    =0 MEANS THE GAIN TERM IS AUTOMATICALLY ADJUSTED; OTHER-
C             WISE, THE GAIN IS FIXED AT THE USER SPECIFIED VALUE.
C    ISKIP    .LE.0 MEANS NO INTERMEDIATE PRINTOUT.  OTHERWISE, THE
C             POSITIVE NUMBER IS THE RATIO OF THE CALCULATED ITERATIONS
C             TO PRINTOUT ITERATIONS.
C
C SECTIONS 100 AND 200 ARE REQUIRED WHILE SECTIONS 300, 400, AND
C ARE OPTIONAL.
C
C
      COMMON/IO/NIN,NOUT
      REAL X(20),F(20),JAC(20,20)
      GO TO(100,200,300,400,500),MODE
C
C THE USER PLACES ALL OF HIS CODING BETWEEN THE TWO +-LINES.
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
C
100   CONTINUE
C DEFINE PROGRAM CONSTANTS N, GAIN, EPSC, EPSJ, MAX, IJAC, IAUTO,
C AND ISKIP HERE.
      N=2
      GAIN=1.0
      EPSC=1.0E-08
      EPSJ=1.0E-08
      MAX=200
      IJAC=0
      IAUTO=0
      ISKIP=0
C DEFINE THE INITIAL X-VECTOR HERE.
      X(1)=10.0
      X(2)=-1.0
C DEFINE ANY ADDITIONAL PROBLEM CONSTANTS HERE.
      GO TO 999
200   CONTINUE
C THE USER SPECIFIES THE N-DIMENSIONAL VECTOR-FUNCTION F.
      F(1)=X(1)**3-27.0
      F(2)=X(1)+X(2)**5-35.0
      GO TO 999
300   CONTINUE
C IF IJAC.NE.0, THE USER SPECIFIES THE JACOBIAN HERE.
      GO TO 999
400   CONTINUE
C SPECIFY CONSTRAINTS ON THE ELEMENTS OF THE X-VECTOR HERE.
      IF(X(1).LT.0.0)X(1)=0.0
      IF(X(2).LT.-1.0)X(2)=-1.0
      IF(X(1).GT.+10.0)X(1)=+10.0
      IF(X(2).GT.+10.0)X(2)=+10.0
      GO TO 999
500   CONTINUE
C THIS SECTION PROFIDES A PLACE TO CALCULATE WITH THE SOLUTION VECTOR.
      GO TO 999
C
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C


999   CONTINUE
      RETURN
      END
```

APPENDIX B. PROGRAM SS (STATE SPACE)

Introduction

In controls and systems engineering, the process under study is often described by a system of first-order, ordinary differential equations of the initial-value type. Problems of this type can be characterized by vector differential equations of the form:

$$\dot{\underline{x}} = \underline{f}\left[\underline{x}(t),t;\ \underline{x}(t_0),t_0\right] \qquad t \geq t_0 \ . \tag{B-1}$$

Program SS was written with the intent of providing a simple method of obtaining numerical solutions for problems of this type with a minimum of specialized programming. For restrictions on the form of $\underline{f}[\underline{x}(t),t;\ \underline{x}(t_0),t_0]$, see the section in this appendix entitled "Discussion of the Integration Method." Used in its simplest form, Program SS only requires the user to provide Fortran coding for the vector function $\underline{f}[\underline{x}(t),t;\ \underline{x}(t_0,t_0]$, specify the outputs, and supply a standardized input deck. Program SS will then generate a tabular listing of the outputs and make line-printer plots of the outputs vs time. In addition, provisions are also made to: perform one-time preintegration calculations, perform one-time postintegration calculations, read specialized input data, establish specialized output labels, handle piecewise continuous $\underline{f}[\underline{x}(t),t]$, make x-y plots of output variables, and record the minimums and maximums of specified variables. Subroutines have been written to provide delay, level detection with hysteresis, and solutions to implicit equations. Program SS is written totally in Fortran IV; the output is in line-printer format.

Example of Usage

The following example illustrates how one can use Program SS to obtain the numerical solution to a set of nonlinear differential equations. Suppose the system under study can be described by the following three differential equations:

$$\dot{x}_1 = -0.5x_1 \ , \tag{B-2}$$

$$\dot{x}_2 = -ax_2 \ , \tag{B-3}$$

$$\dot{x}_3 = -cx_3 + x_1^2 - x_2^2 - d \ , \qquad \text{(B-4)}$$

where

$$t_0 = 0 \ ,$$

$$x_1(0) = x_2(0) = x_3(0) = 1 \ ,$$

$$d = \sqrt{a + b} \ .$$

Observe that for the first differential equation the constant -0.5 is fixed, while for the second and third equations the coefficients are written as variables. Because the coefficient in Eq. (B-2) is fixed, it can be explicitly written in the Fortran coding. Assume that because of the nature of the problem, one wishes to observe the solution of a set of differential equations for different values of constants a, b, and c. The approach used is to compile Program SS with the differential equations but have the constants a, b, and c specified by the input deck. This technique allows one to use the same binary file (results of compilation) with different input decks to generate solutions for the different sets of constants. For this example, specify the constants as follows:

$a = 1,$

$b = 0.5,$

$c = 0.25.$

The solution starts at $t_0 = 0$ s; specify the final time as $t_{final} = 20$ s and the stepsize (constant over the run) as 0.1 s. If the stepsize is too large, the numerical solution will go unstable. This will cause Program SS to halt the solution and output all data up to that time.

The user-written input deck SSIN for this program is shown in Fig. B-1. A detailed description of the input-deck format is given in the following section of this appendix. The specialized Fortran coding for this problem is shown in Fig. B-2, where the boxed-in portions are written by the user. Later in this appendix, a detailed description of the subroutine USER is given in which all specialized Fortran coding relating to this problem appears.

```
)2 4 6 8(1)2 4 6 8(2)2 4 6 8(3)2 4 6 8(4)2 4 6 8(5)2 4 6 8(6)2 4 6 8(7)
BOX R61 SS EXAMPLE  2 1 001 0 0 15000 004
THIS IS AN EXAMPLE OF A SYSTEM OF NONLINEAR DIFFERENTIAL EQUATIONS.
THE FIRST TWO D.E.S ARE LINEAR WHILE THE THIRD IS NONLINEAR
0.00      20.0      0.10      03 04
1.00      1.00      1.00
1.000     0.500     0.250
THE VALUES OF A, B, AND C FOR THIS RUN ARE...
         A=1.000
         B=0.500
         C=0.250
```

Fig. B-1. Input deck SSIN for example given in text.

```
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
100   CONTINUE
C
C THE USER INSERTS USER DEFINED INPUT READ/WRITE STATEMENTS HERE.
C THE INPUT TAPE UNIT NUMBER MUST BE NIN AND THE OUTPUT TAPE UNIT
C NUMBER MUST BE NOUT.
      READ(NIN,101)A,B,C
101   FORMAT(3E10.3)
      WRITE(NOUT,101)A,B,C
      GO TO 999
200   CONTINUE
C
C ONE CAN DO ONE-TIME PRECALCULATIONS AND OUTPUT LABELLING IN
C THIS SECTION.
      D=SQRT(A+B)
C OVERWRITE THE STANDARD OUTPUT LABEL HERE.  AN EXAMPLE IS...
C     LABEL(1)=10HOUTPUT   1
      LABEL(1)=10HSTATE NO 1
      LABEL(2)=10HSTATE NO 2
      LABEL(3)=10HSTATE NO 3
      LABEL(4)=10H XDOT(3)
      GO TO 999
300   CONTINUE
C
C THIS SECTION COMPUTES THE XDOT VECTOR GIVEN N, T, AND THE X-VECTOR.
C
C CALCULATE AN INTERMEDIATE VARIABLE WHICH IS A FUNCTION OF THE STATES.
      Z=-C*X(3)+X(1)**2-X(2)**2-D
      CALL MINMAX(1,10H XDOT(3)   ,Z)
      IF(T.GT.15.0)CALL STOP
C CALCULATE THE TIME DERIVATIVES OF THE STATE VARIABLES.
      XDOT(1)=-0.5*X(1)
      XDOT(2)=-A*X(2)
      XDOT(3)=Z
      GO TO 999
400   CONTINUE
C
C THE USER SPECIFIES THE VARIABLES THAT WILL BE OUTPUTTED IN THIS
C SECTION----THE OUTPUT VARIABLES ARE PLACED IN THE Y-VECTOR; THE
C Y VECTOR IS OF LENGTH M, WHERE M IS SPECIFIED IN THE INPUT
C DECK SSIN.
      Y(1)=X(1)
      Y(2)=X(2)
      Y(3)=X(3)
      Y(4)=Z
      CALL XYPLOT(1,4,2)
      GO TO 999
500   CONTINUE
C
C THIS SECTION IS PROVIDED FOR POST PROCESSING OF THE FINAL TIME DATA.
C
C CALCULATE THE SUM OF THE THREE STATES AT THE FINAL TIME
      SUM=X(1)+X(2)+X(3)
      WRITE(NOUT,501)SUM
501   FORMAT(1H1,5/,6HSUM = ,E10.3)
      GO TO 999
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig. B-2. Specialized Fortran coding required for example given in text.

Figures B-3 through B-5 show portions of the output for this problem. Figure B-3 shows the echoing of the input-deck data. Figure B-4 illustrates a typical line-printer plot, and Fig. B-5 shows the initial portion of the tabular listing.

Standardized Input Deck

The standardized input is that portion of the input deck for which the Fortran coding has already been written. The standardized input includes the following type of information: starting time, final time, stepsize, initial conditions, plot title cards, etc. The user must write additional Fortran coding (in subroutine USER) for any specialized data he wishes to read-in via the input deck. In the above example, the specialized input is the values of constants a, b, and c. The input deck is named SSIN and cards in it have the following formats:

Control Card

Columns

1-20 Not used for this version of Program SS.

21 The number of plot title cards that appear on each line-printer plot. The minimum is zero and the maximum is four.

23 Not used for this version of Program SS.

25-27 This is the ratio of output stepsize to integration stepsize. Data is written in I3 format. If left blank, the default value of one is assigned by the program. The minimum value is one and the maximum value is 999.

29 This switch controls the selection of output modes:

= 0 means plots and tabular listing,
= 1 means tabular listing only,
= 2 means plots only,
= 3 means no plots or tabular listing.

31 This switch controls the line-printer plot size:

= 0 means full-size plots; otherwise, reduced size plots.

33-37 These locations specify the output-file size (I5 format).

```
THE DATA IN THE INPUT FILE IS...


BOX R61 SS EXAMPLE  2 1    1 0 0 15000    4
THIS IS AN EXAMPLE OF A SYSTEM OF NONLINEAR DIFFERENTIAL EQUATIONS.
THE FIRST TWO D.E.S ARE LINEAR WHILE THE THIRD IS NONLINEAR
    .0E+00 2.000E+01 1.000E-01 3  4
 1.000E+00 1.000E+00 1.000E+00
 1.000E+00 5.000E-01 2.500E-01



     THE VALUES OF A, B, AND C FOR THIS RUN ARE...
              A=1.000
              B=0.500
              C=0.250
```

Fig. B-3. The echoing of the input deck into the output deck.

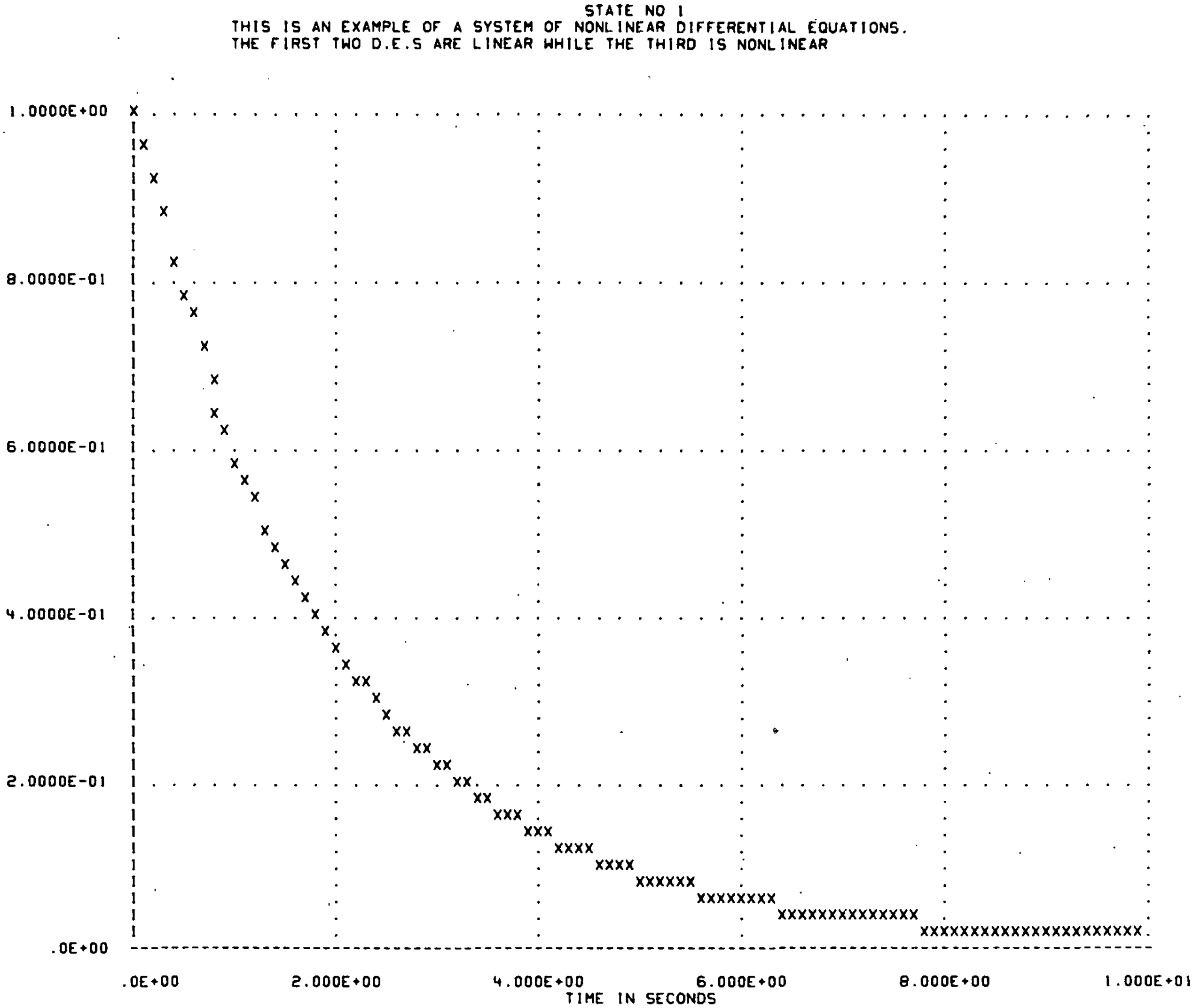

Fig. B-4. Typical line-printer plot.

TIME	STATE NO 1	STATE NO 2	STATE NO 3	XDOT(3)	EST. ERROR
.0E+00	1.000E+00	1.000E+00	1.000E+00	-1.475E+00	.0E+00
1.00E-01	9.512E-01	9.048E-01	8.588E-01	-1.353E+00	-1.000E+00
2.00E-01	9.048E-01	8.187E-01	7.284E-01	-1.258E+00	-1.000E+00
3.00E-01	8.607E-01	7.408E-01	6.065E-01	-1.184E+00	-1.000E+00
4.00E-01	8.187E-01	6.703E-01	4.910E-01	-1.127E+00	-2.000E+00
5.00E-01	7.788E-01	6.065E-01	3.807E-01	-1.081E+00	6.728E-09
6.00E-01	7.408E-01	5.488E-01	2.745E-01	-1.046E+00	1.702E-07
7.00E-01	7.047E-01	4.966E-01	1.713E-01	-1.018E+00	1.482E-06
8.00E-01	6.703E-01	4.493E-01	7.076E-02	-9.950E-01	-2.000E+00
9.00E-01	6.376E-01	4.066E-01	-2.779E-02	-9.765E-01	5.500E-09
1.00E+00	6.065E-01	3.679E-01	-1.246E-01	-9.610E-01	1.144E-07
1.10E+00	5.769E-01	3.329E-01	-2.201E-01	-9.477E-01	6.222E-07
1.20E+00	5.488E-01	3.012E-01	-3.142E-01	-9.357E-01	-2.000E+00
1.30E+00	5.220E-01	2.725E-01	-4.072E-01	-9.247E-01	4.503E-09
1.40E+00	4.966E-01	2.466E-01	-4.992E-01	-9.142E-01	7.669E-08
1.50E+00	4.724E-01	2.231E-01	-5.901E-01	-9.039E-01	2.504E-07
1.60E+00	4.493E-01	2.019E-01	-6.800E-01	-8.936E-01	-2.000E+00
1.70E+00	4.274E-01	1.827E-01	-7.688E-01	-8.832E-01	3.687E-09
1.80E+00	4.066E-01	1.653E-01	-8.566E-01	-8.726E-01	5.140E-08
1.90E+00	3.867E-01	1.496E-01	-9.433E-01	-8.617E-01	9.310E-08
2.00E+00	3.679E-01	1.353E-01	-1.029E+00	-8.505E-01	-2.000E+00
2.10E+00	3.499E-01	1.225E-01	-1.113E+00	-8.389E-01	3.019E-09
2.20E+00	3.329E-01	1.108E-01	-1.197E+00	-8.270E-01	3.446E-08
2.30E+00	3.166E-01	1.003E-01	-1.279E+00	-8.148E-01	2.895E-08
2.40E+00	3.012E-01	9.072E-02	-1.360E+00	-8.023E-01	-2.000E+00
2.50E+00	2.865E-01	8.208E-02	-1.439E+00	-7.896E-01	2.471E-09
2.60E+00	2.725E-01	7.427E-02	-1.518E+00	-7.766E-01	2.310E-08
2.70E+00	2.592E-01	6.721E-02	-1.595E+00	-7.634E-01	4.484E-09
2.80E+00	2.466E-01	6.081E-02	-1.670E+00	-7.501E-01	-2.000E+00
2.90E+00	2.346E-01	5.502E-02	-1.745E+00	-7.366E-01	2.023E-09
3.00E+00	2.231E-01	4.979E-02	-1.818E+00	-7.230E-01	1.548E-08
3.10E+00	2.122E-01	4.505E-02	-1.889E+00	-7.094E-01	3.597E-09
3.20E+00	2.019E-01	4.076E-02	-1.959E+00	-6.958E-01	-2.000E+00
3.30E+00	1.920E-01	3.688E-02	-2.028E+00	-6.821E-01	1.657E-09
3.40E+00	1.827E-01	3.337E-02	-2.096E+00	-6.685E-01	1.038E-08
3.50E+00	1.738E-01	3.020E-02	-2.162E+00	-6.549E-01	5.286E-09
3.60E+00	1.653E-01	2.732E-02	-2.227E+00	-6.414E-01	-2.000E+00
3.70E+00	1.572E-01	2.472E-02	-2.290E+00	-6.280E-01	1.356E-09
3.80E+00	1.496E-01	2.237E-02	-2.352E+00	-6.148E-01	6.957E-09
3.90E+00	1.423E-01	2.024E-02	-2.413E+00	-6.016E-01	4.752E-09
4.00E+00	1.353E-01	1.832E-02	-2.473E+00	-5.886E-01	-2.000E+00
4.10E+00	1.287E-01	1.657E-02	-2.531E+00	-5.757E-01	1.111E-09
4.20E+00	1.225E-01	1.500E-02	-2.588E+00	-5.630E-01	4.663E-09
4.30E+00	1.165E-01	1.357E-02	-2.644E+00	-5.505E-01	3.653E-09
4.40E+00	1.108E-01	1.228E-02	-2.698E+00	-5.381E-01	-2.000E+00
4.50E+00	1.054E-01	1.111E-02	-2.751E+00	-5.259E-01	9.092E-10
4.60E+00	1.003E-01	1.005E-02	-2.803E+00	-5.140E-01	3.126E-09
4.70E+00	9.537E-02	9.095E-03	-2.854E+00	-5.022E-01	2.591E-09
4.80E+00	9.072E-02	8.230E-03	-2.904E+00	-4.907E-01	-2.000E+00
4.90E+00	8.629E-02	7.447E-03	-2.952E+00	-4.793E-01	7.444E-10
5.00E+00	8.208E-02	6.738E-03	-3.000E+00	-4.682E-01	2.095E-09
5.10E+00	7.808E-02	6.097E-03	-3.046E+00	-4.572E-01	1.739E-09
5.20E+00	7.427E-02	5.517E-03	-3.091E+00	-4.465E-01	-2.000E+00
5.30E+00	7.065E-02	4.992E-03	-3.135E+00	-4.360E-01	6.095E-10
5.40E+00	6.721E-02	4.517E-03	-3.178E+00	-4.257E-01	1.405E-09
5.50E+00	6.393E-02	4.087E-03	-3.220E+00	-4.156E-01	1.110E-09
5.60E+00	6.081E-02	3.698E-03	-3.261E+00	-4.057E-01	-2.000E+00

Fig. B-5. Initial portion of the tabular output.

39-41 The number of problem-comment cards (I3 format). These comment cards will appear only once at the beginning of the output. The minimum number is 0 and the maximum is 999.

Plot Title Cards

The plot title cards are reproduced at the top of each plot. One may have from zero to a maximum of four plot title cards, and each card can have up to 80 characters.

Problem Information Card

Columns

1-10 Initial time in seconds (E10.3 format).

11-20 Final time in seconds (E10.3 format).

21-30 Integration stepsize in seconds (E10.3 format). The stepsize is fixed for the numerical solution of the differential equations.

31-32 The number of integrator state variables N (I2 format).

34-35 The number of outputs M (I2 format). If M = 0, then the program assigns a default value of M = N.

As presently dimensioned, N and M must satisfy:

$$0 < N \leq 20 \ ,$$
$$0 \leq M \leq 30 \ .$$

Initial Condition Cards

The N initial values of the N integrators are read in 8E10.3 format. The first position corresponds to $x_0(1)$, the next to $x_0(2)$, etc.

User Defined Input

The formats used here are user specified in subroutine USER, section 100.

Problem Comment Cards

The problem comment cards appear only once at the beginning of the output. One may have 0 to 999 cards; each card may have up to 80 characters.

Standardized Subroutine USER

All user-written Fortran IV coding, which specifies the particular problem, appears in a standardized subroutine USER. Subroutine USER is called five different ways by Program SS. The manner in which subroutine USER is used is determined by the value of mode (set by Program SS). For mode = 1, subroutine USER branches to section 100, for mode = 2, to section 200, etc. The basic form of the standardized subroutine USER is shown in Fig. B-6.

In section 100, Program SS reads user-defined input data. The input-tape unit number is NIN and the output tape unit number is NOUT. All read statements accept data contained in the input file SSIN; all write statements place data in the output file SSOUT. In section 200, one can do precalculations based on data read in section 100 and constants defined in section 200. Typically, the results of the precalculations will be constants used in the integration portion. One can also overwrite the standardized output labels in this section. In section 300, one specifies the first-order, ordinary differential equation. Given N, T, and X (where N is the number of first-order differential equations, T is the current time, and X is the current value of the state vector at time T), the user must provide Fortran IV coding that determines XDOT, the current value of the time-derivative of X at time T.

In section 400, the output vector Y is specified. One can place any desired variable in any order in the Y vector. The first element of Y is plotted first, the second element is plotted second, etc. In section 500, one can do postintegration calculation. The value of X will be that of the last calculated time. Any input or output must observe the tape unit numbers discussed in section 100. Observe that no user-written common statements are required to exchange information between Program SS and subroutine USER or between sections in subroutine USER.

Additional Features

This section discusses features of Program SS not illustrated in the four previous sections of this appendix.

- For efficiency in core utilization, the output vector Y is dimensioned to hold 101 output points (not integration points) per element. To provide adjustment between integration stepsize and output stepsize, a countdown ratio

```
      SUBROUTINE USER(MODE,N,T,X,XDOT)
C
C
C THE VARIABLES USED BY PROGRAM SS ARE AS FOLLOWS...
C          MODE     SWITCH USED BY PROGRAM SS TO SELECT VARIOUS PARTS
C                   OF SUBROUTINE USER.
C          NIN      TAPE UNIT NUMBER FOR READING USER DEFINED INPUT
C          NOUT     TAPE UNIT NUMBER FOR ECHOING USER DEFINED INPUT
C          N        DIMENSION OF THE STATE VECTOR X
C          M        NUMBER OF VARIABLES TO BE OUTPUTTED
C          T        CURRENT VALUE OF TIME
C          X        STATE VECTOR---THESE VARIABLES ARE THE RESULT OF THE
C                   DIGITAL INTEGRATION.
C          XDOT     CURRENT VALUE OF THE TIME DERIVATIVE OF X EVALUATED
C                   AT THE CURRENT TIME T.
C          Y        OUTPUT VECTOR---THESE VARIABLES WILL BE OUTPUTTED.
C
C NOTE:    EVERYTHING IN SECTIONS 300 AND 400 IS REQUIRED.  EVERY-
C          THING IN SECTIONS 100, 200, AND 500 IS OPTIONAL.
C
      DIMENSION X(20),XDOT(20),Y(31),LABEL(80)
      COMMON/TIE1/NIN,NOUT,M,ALINE
      COMMON/TIE3/Y
      COMMON/TIE4/LABEL
      GO TO(100,200,300,400,500)MODE
C
C
C THE USER PLACES ALL OF HIS CODING BETWEEN THE TWO + LINES.
C
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
100   CONTINUE
C
C THE USER INSERTS USER DEFINED INPUT READ/WRITE STATEMENTS HERE.
C THE INPUT TAPE UNIT NUMBER MUST BE NIN AND THE OUTPUT TAPE UNIT
C NUMBER MUST BE NOUT.
      GO TO 999
200   CONTINUE
C
C ONE CAN DO ONE-TIME PRECALCULATIONS AND OUTPUT LABELLING IN
C THIS SECTION.
C
C OVERWRITE THE STANDARD OUTPUT LABEL HERE.  AN EXAMPLE IS...
C     LABEL(1)=10HOUTPUT   1
      GO TO 999
300   CONTINUE
C
C THIS SECTION COMPUTES THE XDOT VECTOR GIVEN N, T, AND THE X-VECTOR.
C
C CALCULATE INTERMEDIATE VARIABLES WHICH ARE FUNCTIONS OF THE STATE.
C
C CALCULATE THE TIME DERIVATIVES OF THE STATE VARIABLES.
      GO TO 999
400   CONTINUE
C
C THE USER SPECIFIES THE VARIABLES THAT WILL BE OUTPUTTED IN THIS
C SECTION----THE OUTPUT VARIABLES ARE PLACED IN THE Y-VECTOR; THE
C Y VECTOR IS OF LENGTH M, WHERE M IS SPECIFIED IN THE INPUT
C DECK SSIN.
C
      GO TO 999
500   CONTINUE
C
C THIS SECTION IS PROVIDED FOR POST PROCESSING OF THE FINAL TIME DATA.
C
      GO TO 999
C
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
999   CONTINUE
      RETURN
      END
```

Fig. B-6. Basic form of subroutine USER.

is provided in the first card of the input deck. If one sets this ratio to two, every other calculated value of the Y vector will be outputted. If one specifies the initial time, final time, and countdown ratio such that more than 101 time points are outputted, Program SS will first compute the outputs for the initial 101 time points, then output this data in the normal manner. Next, the outputs associated with the 101st time point of Y are copied into the storage locations of the first time point of Y, and Program SS continues the numerical solution by refilling the Y vector. If the Y vector is filled again, Program SS will output the data and then proceed on again. This technique allows the storage area in Program SS to remain small; a small object file is a useful goal when running in a time-sharing computer environment.

- It has been observed that for realistic simulation problems (that is, problems where the number of integrations and outputs are approximately equal, the ratio of output stepsize to integration stepsize is not over four, and plots are requested), the IO time (time to output data) is larger than the CPU time (time used for digital integration). The IO-to-CPU charge times will be dependent on the computer center used.

Table B-1. Useful subroutines in Program SS.

Subroutine	Function
STOP	Terminates Program SS and outputs data up to that point.
RESTART	Initiates the Runge-Kutta integration method. This subroutine is used when discontinuities occur in $\underline{f}[\underline{x}(t),t]$.
MINMAX	Records the minimum and maximum of specified variables and the times at which these occur. This subroutine is useful for checking equilibrium points.
XYPLOT	Performs X-Y plots for variables that appear in the output vector Y.
LD	Simulates a level detector with hystersis.
IMPEQS	Solves implicit equations. The nonlinear algebraic equations appear in subroutine NAE.
DELAY	Delays any variable by an integral number of integration steps.

• Additional useful subroutines that are currently in Program SS are given in Table B-1. These subroutines can be called from subroutine USER. For more information on these subroutines, read the instructions that appear in each subroutine listing. (The complete program listing appears at the end of this appendix.)

Discussion of the Integration Method

Program SS uses an Adams-Bashforth-Moulton predictor-corrector (fourth-order) to carry out the integration. The predictor equation is:

$$x^*_{n+1} = x_n + \frac{h}{24}\left[55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right] ,$$

and the corrector equation is:

$$\tilde{x}_{n+1} = x_n + \frac{h}{24}\left[9f^*_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right] ,$$

where h is the stepsize and f_n denotes the time-derivative of x at time t_n. Program SS uses one correction per integration step. One can estimate the local truncation error (TR), assuming a constant fifth time-derivative of x, to be:

$$TR = (-19/270)\left(\tilde{x}_{n+1} - x^*_{n+1}\right) .$$

Program SS uses this value of TR to update $\tilde{x}_{n+1}$ as follows:

$$x_{n+1} = \tilde{x}_{n+1} + TR.$$

The above three equations are the basis of subroutine ESODEQ. To start the fourth-order predictor/corrector, subroutine ESODEQ uses the standard fourth-order Runge-Kutta integration. The equations for the Runge-Kutta section are:

$$x_{n+1} = x_n + \frac{h}{6}\left[k_0 + 2k_1 + 2k_2 + k_3\right] ,$$

where

$$k_0 = f\left[x(t_n), t_n\right],$$

$$k_1 = f\left[x(t_n) + \frac{hk_0}{2}, t_n + \frac{h}{2}\right],$$

$$k_2 = f\left[x(t_n) + \frac{hk_1}{2}, t_n + \frac{h}{2}\right],$$

$$k_3 = f\left[x(t_n) + hk_2, t_n + h\right].$$

The Runge-Kutta method is accurate over a few steps, but note that four function evaluations of $f[x(t),t]$ per step are required. The fourth-order predictor/corrector for one correction requires only two evaluations of $f[x(t),t]$ per step. In addition, the predictor/corrector provides a simple estimate of the TR, which is used to determine a stepsize consistent with a maximum value of TR. Both the Runge-Kutta and predictor/corrector methods used in Program SS require that up to and including the fifth time-derivative of $x(t)$ exist. For the case where $f[x(t),t]$ is piecewise continuous (e.g., in digital switching), one can solve each continuous section by these methods and restart at the discontinuity. That is, suppose $f[x(t),t]$ has a discontinuity at t_n. One can solve for $x(t_n^-)$ by the predictor/corrector and use $x(t_n^-)$ as the initial conditions for a solution starting at t_n^+ (i.e., restart the solution at t_n^+ with the Runge-Kutta method). One can accomplish this in subroutine USER by call to RESTART.

The basic concepts of how the predictor/corrector performs the integration follows. Suppose one wishes to solve an ordinary differential equation of the initial-value type; that is, one wishes to solve:

$$x(t) = \int_{t_0}^{t} f[x(n),n]\, dn,$$

subject to $x(t_0) = x_0$. The numerical solution to this problem consists of solving a sequence of single-step problems such as:

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f[x(n),n]\, dn,$$

with $x(t_n) = x_n$. If one can solve the single-step problem, one can then sequentially solve for x_1, x_2, etc. The basic idea of the predictor/corrector method is to approximate $f[x(t),t]$ by a polynomial and then integrate the polynomial over the single step. This is performed in two stages: predicting and correcting. For Program SS the polynomial used (Newton Backward Interpolating Polynomial) can be shown to be equivalent to:

$$p(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$

(i.e., a third-order polynomial curve-fit of $f[x(t),t]$).

Assume one knows some past values of $f[x(t),t]$; one can use a third-order, polynomial curve-fit to predict the value of $f[x(t),t]$ at t_{n+1}. This is shown in Fig. B-7.

The predicted value of $x(t_{n+1})$ is:

$$x^*(t_{n+1}) = x(t_n) + A_p ,$$

where A_p is the area under the polynomial in Fig. B-7 from t_n to t_{n+1}. Thus, $x^*(t_{n+1})$ is a reasonable estimate of the true solution at t_{n+1}. The corrector takes this initial guess and improves upon it. The corrector for Program SS curve-fits the $f[x(t),t]$ function using the predicted value of $x^*(t_{n+1})$. This is shown in Fig. B-8. The corrected value of $x(t_{n+1})$ is:

$$x^c(t_{n+1}) = x(t_n) + A_c ,$$

where A_c is the area under the polynomial in Fig. B-8 from t_n to t_{n+1}.

Notice that the predictor extrapolates the f function while the corrector interpolates the f function from t_n to t_{n+1}. One can reapply the corrector formula as many times as desired, but Program SS uses only one correction. The TR is defined as the difference between the true solution at t_{n+1} and the corrector output at t_{n+1} for infinitely precise calculations, an exact differential equation, and an exact value of x_n. Thus, TR ignores

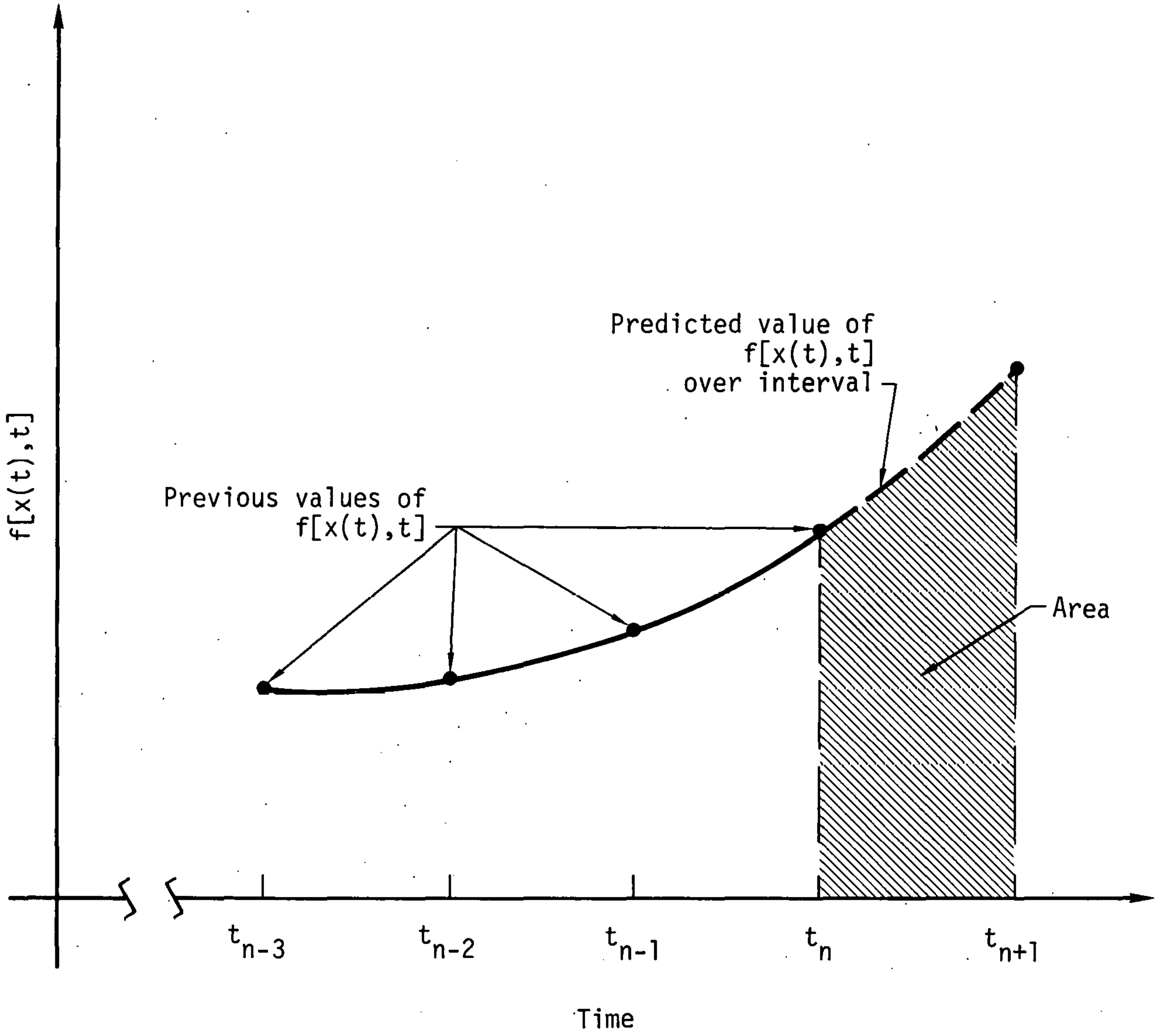

Fig. B-7. Predicting process.

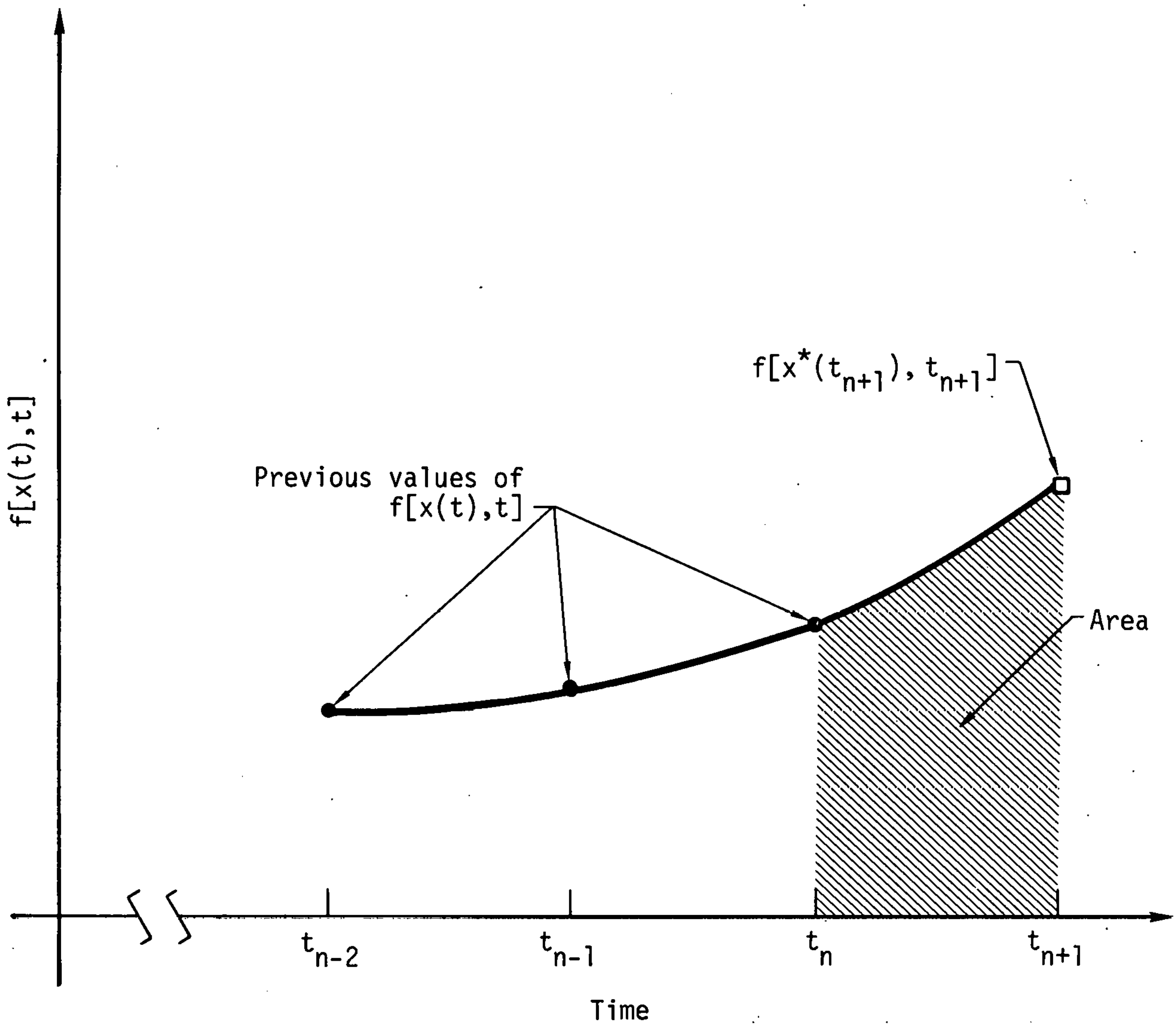

Fig. B-8. Correcting process.

propagation errors, machine roundoff errors, and modeling errors; it is simply the error introduced by the polynomial-curve fit for a single step. An estimate of the TR is automatically printed out in the tabular listing. This estimate assumes a constant fifth time-derivative of x over the interval h. To decode the estimated TR printout, read the comments in subroutine ODE.

Fortran IV Listing of Program SS

A Fortran IV listing of Program SS (which compiles under the Control Data Corporation's PUTT compiler as implemented at the Lawrence Livermore Laboratory) follows:

```
      PROGRAM SS (SSIN,TAPE1=SSIN,SSOUT,TAPE2=SSOUT)
C
C PROGRAM SS (STATE SPACE) IS A GENERAL PURPOSE ORDINARY, FIRST-ORDER
C DIFFERENTIAL-EQUATION (OF THE INITIAL VALUE TYPE) SOLVER.
C
C PROGRAM SS WAS WRITTEN BY HOWARD MCCUE AT LAWRENCE LIVERMORE LABS.,
C LIVERMORE, CALIFORNIA AS PART OF EECS 299 (THESIS) AT THE
C UNIVERSITY OF CALIFORNIA, BERKELEY UNDER PROF. OTTO SMITH.  FOR A
C WRITEUP OF PROGRAM USAGE, SEE UCRL-51657, STABILIZATION OF DISTANT
C AND LOCAL POWER SYSTEM DISTURBANCES BY OPTIMIZED FIELD CONTROL,
C APPENDIX M.  THIS PROGRAM WAS LAST MODIFIED ON NOVEMBER 9TH, 1973.
C
C AS PRESENTLY DIMENSIONED, PROGRAM SS CAN HANDLE 20 (NMAX)
C INTEGRATIONS AND 30 (MMAX) OUTPUTS.  ONE CAN CHANGE THESE
C NUMBERS AS FOLLOWS:  TO CHANGE THE MAXIMUM NUMBER OF INTEGRATIONS
C FROM 20 TO 50, CHANGE ALL VARIABLES PRESENTLY DIMENSIONED 20
C TO 50; SET NMAX IN THE MAIN PROGRAM TO 50.  TO CHANGE THE MAXIMUM
C NUMBER OF OUTPUTS TO 60, CHANGE ALL VARIABLES PRESENTLY
C DIMENSIONED 31 TO 61 (I.E. 60+1, THE EXTRA ONE IS FOR THE
C TRUNCATION ERROR OUTPUT); SET MMAX IN THE MAIN PROGRAM TO 60.
C TO INCREASE MMAX BEYOND 79, ONE MUST INCREASE THE DIMENSION OF LABEL.
C THE FOLLOWING NUMBER OF DIMENSION STATEMENTS MUST BE MODIFIED FOR
C THE ABOVE MENTIONED CHANGES: INTEGRATORS--5, OUTPUTS--4, LABELS--2.
      NMAX=20
      MMAX=30
C
C CREATE THE * BINARY FILE.
      CALL CHANGE(3H*SS)
C
C SPECIFY THE INPUT/OUTPUT TAPE UNITS.
      NIN=1
      NOUT=2
      COMMON/INPUT1/IFLAG,IKEEP,ISKIP,ISIZE,NMAX,MMAX
      COMMON/TIE1/NIN,NOUT,M,ALINE
      DIMENSION TITLES(4,8),X0(20)
C
C INPUT ANY USER SPECIFIED PARAMETERS.
      CALL INPUT(T0,TFINAL,H,X0,N,M,TITLES)
      IF(IFLAG.EQ.1)GO TO 999
      TFINAL=0.99999999*TFINAL
C SOLVE THE ORDINARY DIFFERENTIAL EQUATIONS.
      CALL ODE(T0,TFINAL,H,X0,N,M,NOUT,TITLES,ISIZE,ISKIP)
C
C
999   CONTINUE
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE INPUT(TO,TFINAL,H,XO,N,M,TITLES)
C
C THIS SUBROUTINE READS THE STANDARIZED INPUT DATA AND USER
C DEFINED INPUT DATA FROM INPUT FILE ΔSSINΔ.  IN ADDITION, THIS
C SUBROUTINE CALLS UP SECTION 200 IN SUBROUTINE USER.
C
C
      DIMENSION NAME(2),TITLES(4,8),XO(20),COMMENT(8),XDOT(20)
      COMMON/INPUT1/IFLAG,IKEEP,ISKIP,ISIZE,NMAX,MMAX
      COMMON/TIE1/NIN,NOUT,M,ALINE
      COMMON/INPUT2/IDEL
C
C
C READ IN REQUIRED DATA.
      IFLAG=0
      READ(NIN,1)NAME(1),NAME(2),ITITLES,IKEEP,ISKIP,IDEL,ISIZE,LLL,ICOM
1     FORMAT(2A10,I1,1X,I1,1X,I3,1X,I1,1X,I1,1X,I5,1X,I3)
      IF(LLL.LE.0)LLL=10000
C CREATE A DISK FILE FOR THE OUTPUT.
      CALL CREATE(5HSSOUT,LLL,IERROR)
      IF(IERROR.LT.0)IFLAG=1
      IF((ITITLES.LT.0).OR.(ITITLES.GT.4))IFLAG=1
      IF(ISKIP.LE.0)ISKIP=1
      IF((IDEL.LT.0).OR.(IDEL.GT.3))IFLAG=1
      IF(ICOM.LE.0)ICOM=0
      IF(IFLAG.EQ.1)GO TO 999
      WRITE(NOUT,7)
7     FORMAT(5/,32HTHE DATA IN THE INPUT FILE IS...,3/)
      WRITE(NOUT,1)NAME(1),NAME(2),ITITLES,IKEEP,ISKIP,IDEL,ISIZE,LLL,
     2ICOM
      IF(ITITLES.EQ.0)GO TO 5
      DO 3 I=1,ITITLES
      READ(NIN,4)(TITLES(I,J),J=1,8)
4     FORMAT(8A10)
      WRITE(NOUT,4)(TITLES(I,J),J=1,8)
3     CONTINUE
5     CONTINUE
      READ(NIN,2)TO,TFINAL,H,N,M
2     FORMAT(3E10.3,2(I2,1X))
      WRITE(NOUT,2)TO,TFINAL,H,N,M
      IF(TO.GT.TFINAL)IFLAG=1
      IF(H.EQ.0.0)IFLAG=1
      IF((N.LE.0).OR.(N.GT.NMAX))IFLAG=1
      IF((M.LT.0).OR.(M.GT.MMAX))IFLAG=1
      IF(M.EQ.0)M=N
      READ(NIN,6)(XO(I),I=1,N)
6     FORMAT(8E10.3)
      WRITE(NOUT,6)(XO(I),I=1,N)
C GET ANY USER WRITTEN INPUT DATA.
      CALL USER(1,N,TO,XO,XDOT)
      WRITE(NOUT,9)
9     FORMAT(5/)
      IF(ICOM.EQ.0)GO TO 10
      DO 11 I=1,ICOM
      READ(NIN,4)(COMMENT(I),I=1,8)
      WRITE(NOUT,12)(COMMENT(I),I=1,8)
12    FORMAT(5X,8A10)
11    CONTINUE
10    CONTINUE
C DO ANY PRECALCULATIONS THAT ARE REQUIRED.
      CALL USER(2,N,TO,XO,XDOT)
999   CONTINUE
      RETURN
      END


      SUBROUTINE ODE(TO,TFINAL,H,XO,N,M,NOUT,TITLES,ISIZE,ISKIP)
```

```
C
C
C SUBROUTINE ODE (ORDINARY DIFFERENTIAL EQUATIONS) IS A DRIVER FOR
C SUBROUTINE ESODEQ.  SUBROUTINE ESODEQ COMES FROM THE UNIVERSITY OF
C CALIFORNIA AT DAVIS COMPUTING CENTER.  ESODEQ USES A FOUR POINT
C ADAMS-BASHFORTH-MOULTON PREDICTOR-CORRECTOR METHOD TO CARRY OUT ITS
C INTEGRATION.  THIS IS A CONSTANT STEP-SIZE INTEGRATION SUBROUTINE.
C
C THE INPUTS TO ODE ARE AS FOLLOWS....
C         TO       INITIAL TIME
C         TFINAL   FINAL TIME OF SOLUTION
C         H        STEP SIZE
C         XO       INITIAL VALUE OF THE STATE VECTOR
C         N        NUMBER OF FIRST-ORDER ODE
C         M        TOTAL NUMBER OF VARIABLES TO BE OUTPUTTED.
C         NOUT     TAPE UNIT NUMBER FOR OUTPUT
C         TITLES   AN ARRAY OF TITLE CARDS USED FOR PLOTTING
C         ISIZE    CONTROLS THE SIZE OF THE LINE PRINTER PLOTS;
C                  =0 MEANS FULLSIZED PLOTS, OTHERWISE, ONE GETS
C                  REDUCED-SIZED PLOTS.
C         ISKIP    RATIO OF CALCULATED TO OUTPUTTED POINTS
C
C
C
C TR IS AN ESTIMATE OF THE LOCAL TRUNCATION ERROR IN THE COMPUTED
C SOLUTION AT TIME T.  SUPPOSE TR IS THE ASSOCIATED ERROR FOR THE ITH
C ELEMENT OF THE STATE VECTOR X AT TIME T; THEN THE COMPUTED STATE
C ELEMENT X(I) DIFFERS FROM THE TRUE SOLUTION AT TIME T BY AN ESTIMATED
C ERROR BOUND OF + OR - TR.  SEE ESODEQ WRITEUP FOR MORE DETAILS.
C THE VALUE OF TR IS INTERPRETED IN THE FOLLOWING WAY...
C
C   TR=-1.0      STARTING PREDICTOR-CORRECTOR VIA RUNGA-KUTTA---NO
C                ESTIMATE OF THE LOCAL TRUNCATION ERROR IS AVAILABLE.
C   TR=-2.0      MARKER FOR LOCAL TRUNCATION ERROR PRINTOUT---THE NEXT
C                VALUE OF TR IS FOR X(1)
C   TR=-3.0      A DISCONTINUITY IN F(X(T),T) HAS OCCURRED WHILE
C                IN THE PREDICTOR MODE---THE PROGRAM HAS INITIATED A
C                CHANGE TO THE RUNGA-KUTTA INTEGRATION METHOD FOR
C                THREE INTEGRATION STEPS (SUBROUTINE RESTART).
C   TR.GE.0.0    ABSOLUTE VALUE OF ESTIMATE OF LOCAL TRUNCATION ERROR
C                ----SEE DECODING BELOW.
C
C ONE DECODES THE PRINTOUT AS FOLLOWS:  SUPPOSE ONE HAS N STATES AND
C TR=-2.0 AT TIME T=TT.  THEN THE LOCAL TRUNCATION ERROR ASSOCIATED WITH
C X(I) OCCURS AT TIME T=TT+I*H FOR I.LE.N.  FOR T=TT+H*(N+1), TR=-2.0
C AND THE PROCESS REPEATS.  THUS, ONE GETS AN UPDATE OF THE STATE VECTOR
C ERROR EVERY N+1 PRINTOUTS.  FOR THE PURPOSE OF HALTING THE NUMERICAL
C SOLUTION, THE PROGRAM EXAMINES THE ESTIMATES OF THE LOCAL TRUNCATION
C ERRORS OF ALL INTEGRATIONS AT EACH CORRECTION.
C
C
C
C
      DIMENSION TITLES(4,8),XO(20),X(20),XDOT(20),Y(31)
      COMMON/TIE3/Y
      COMMON/TIE2/T
      COMMON /ISTOPS/ ISTOP
C
      ISTOP=0
      T=TO
C INITIALIZE THE OUTPUT VECTOR Y FOR T=TO.
      CALL USER(3,N,TO,XO,XDOT)
      CALL USER(4,N,TO,XO,XDOT)
```

```
      MP1=M+1
      Y(MP1)=0.0
      IF((T0.GE.TFINAL).OR.(ISTOP.NE.0))GO TO 2
      CALL OUTPUT(MP1,T0,Y,NOUT,TITLES,ISIZE,0,ISKIP)
C START-UP THE INTEGRATION PROCESS.
      CALL ESODEQ(1,N,T0,X0,T,X,TR,H,ISTOP,ISKIP)
      CALL USER(4,N,T,X,XDOT)
      Y(MP1)=TR
      IF((T.GE.TFINAL).OR.(ISTOP.NE.0))GO TO 2
      CALL OUTPUT(MP1,T,Y,NOUT,TITLES,ISIZE,0,ISKIP)
1     CONTINUE
C PERFORM A SINGLE INTEGRATION STEP.
      CALL ESODEQ(3,N,T0,X0,T,X,TR,H,ISTOP,ISKIP)
      CALL USER(4,N,T,X,XDOT)
      Y(MP1)=TR
      IF((T.GE.TFINAL).OR.(ISTOP.NE.0))GO TO 2
      CALL OUTPUT(MP1,T,Y,NOUT,TITLES,ISIZE,0,ISKIP)
      GO TO 1
2     CONTINUE
      CALL OUTPUT(MP1,T,Y,NOUT,TITLES,ISIZE,1,ISKIP)
      IF(ISTOP.EQ.0)GO TO 5
      WRITE(NOUT,9)
9     FORMAT(1H1,25/)
      IF(ISTOP.LT.0)GO TO 6
C
C PROGRAM TERMINATED BECAUSE INTEGRATION ERROR WAS JUDGED TOO
C LARGE.  PRINT-OUT WHICH ELEMENT OF THE STATE VECTOR BLEW-UP.
C
      WRITE(NOUT,10)ISTOP
10    FORMAT(28HPROGRAM TERMINATED----STATE ,I2,09H BLEW UP.)
      GO TO 5
6     CONTINUE
C
C PROGRAM TERMINATED BECAUSE OF A CALL TO SUBROUTINE STOP.
C
      WRITE(NOUT,7)
7     FORMAT(47HPROGRAM TERMINATED BY A CALL TO SUBROUTINE STOP)
5     CONTINUE
999   CONTINUE
C POST PROCESS THE FINAL TIME DATA.
      CALL MINMAX(-1,10H DUMP NOW ,0.0)
      IF(T.EQ.T0)CALL USER(5,N,T0,X0,XDOT)
      IF(T.NE.T0)CALL USER(5,N,T,X,XDOT)
      RETURN
      END


      SUBROUTINE ESODEQ(KODE,N,XI,YI,X,Y,TR,H,ISTOP,ISKIP)
C ESODEQ IS FROM THE UNIVERSITY OF CALIFORNIA AT DAVIS COMPUTING CENTER.
C
C FOR DETAILS ON THE METHOD SEE...
C   ISAACSON AND KELLER,  ANALYSIS OF NUMERICAL METHODS,PP384--388
C   MCCRACKEN AND DORN,  NUMERICAL METHODS AND FORTRAN PROGRAMMING P334
C
C KSTP=1 MEANS ESODEQ WILL USE THE CORRECTOR AFTER EACH PREDICTION-----
C SEE ESODEQ WRITEUP FOR MORE DETAILS.
      KSTP=1
C THIS COMMON GOES BETWEEN SUBROUTINE ESODEQ AND DELAY.
      COMMON/FINAL/IFINAL
      COMMON/NODER/INODER
```

```
      IFINAL=0
      DIMENSION YI(20),Y(20),DY(20),YC(20),DYP(4,20),S(20)
      DATA HO/0/,DYP/80*0./
      DATA MSTP, NSTP, IN1, IN2, IN3, IN4/ 0, 0, 1, 2, 3, 4/
      IS1=IS1+1
      GO TO (1000, 2000, 3000), KODE
C                            INITIALIZE PROCESS TO START WITH RUNGE-KUTTA
C                            INTEGRATION ON INITIAL VALUES
 1000 DO 1001 I=1,N
1001  Y(I) = YI(I)
      CALL USER(3,N,XI,YI,DY)
      X = XI
      IN1 = 1
      IN2 = 2
      IN3 = 3
      IN4 = 4
C INITIALIZE THE TR PRINTOUT SELECTOR.
      NP1=N+1
      ITR=NP1
      IS1=-ISKIP+1
 1050 MSTP = KSTP
      NSTP = 0
      HO = H
      GO TO 4000
C                            START R-K INTEGRATION WITH CURRENT X,Y VALUES
 2000 GO TO 1050
 3000 IF (HO.NE.H) GO TO 1050
C CHECK FOR A PROBLEM SPECIFIED RESTART BASED ON THE CORRECTER OUTPUT.
      IF(INODER.NE.0)GO TO 1050
      GO TO 4000
C                            INTEGRATE 1 STEP
C                            SAVE CURRENT DERIVATIVE VALUES
 4000 DO 4001 I= 1,N
 4001 DYP(IN1,I) = DY(I)
C                            CHECK FOR R-K CONTINUATION
      IF (NSTP.LE.2) GO TO 4500
C                            USE ABM FORMULAE
C                            PREDICTOR
      DO 4002 I=1,N
 4002 YC(I) = Y(I) + H*(55.0*DYP(IN1,I)-59.0*DYP(IN2,I)+37.0*DYP(IN3,I)
     1                       -9.0*DYP(IN4,I))/24.0
C                            CHECK IF CORRECTOR STEP IS DESIRED
      MSTP = MSTP-1
      IF (MSTP.LE.0) GO TO 4100
      DO 4003 I=1,N
 4003 Y(I) = YC(I)
      X = X + H
      GO TO 4800
C                            CORRECTOR
 4100 X = X + H
      CALL USER(3,N,X,YC,DY)
C CHECK FOR A PROBLEM SPECIFIED RESTART BASED ON THE PREDICTOR OUTPUT.
C IF A RESTART IS REQUESTED, USE ONLY THE PREDICTOR FOR THIS
C STEP (THIS WILL YIELD THE T-MINUS VALUE OF THE STATE VECTOR, ONE CAN
C THEN USE THIS VALUE AS THE INITIAL CONDITION FOR THE RESTART.).
      IF(INODER.EQ.0)GO TO 120
      DO 121 I=1,N
      Y(I)=YC(I)
121   CONTINUE
      IF(IS1.LT.0)GO TO 125
```

```
      ISI=-ISKIP
      TR=-3.0
125   CONTINUE
      GO TO 4800
120   CONTINUE
      DO 4102 I=1,N
 4102 Y(I) = Y(I) + H*(9.0*DY(I)+19.0*DYP(IN1,I)-5.0*DYP(IN2,I)
     1                  +DYP(IN3,I))/24.0
      MSTP = KSTP
C
C CALCULATE AN ESTIMATE OF THE LOCAL TRUNCATION ERROR FOR THE PURPOSE
C OF TERMINATING THE PROGRAM IF TR IS TOO LARGE----THIS SECTION
C CHECKS EACH STATE ELEMENT AT EVERY CORRECTION TIME.
C
      DO 101 I=1,N
      TR=-0.07037037*(Y(I)-YC(I))
C UPDATE THE CORRECTOR OUTPUT---SEE PAGE 341 OF MCCRACKEN AND DORN.
      Y(I)=Y(I)+TR
      IF(Y(I).NE.0.0)FRAC=ABS(TR/Y(I))
      ABSYY=ABS(Y(I))
      IF((FRAC.GE.0.25).AND.(ABSYY.GT.1.00))ISTOP=I
101   CONTINUE
C
C CALCULATE AN ESTIMATE OF THE LOCAL TRUNCATION ERROR WHICH IS
C SYNCHRONIZED WITH THE OUTPUT.
C
      IF(ISI.LT.0)GO TO 110
C CALCULATE TR FOR THIS TIME.
      ISI=-ISKIP
      IF(ITR.NE.NP1)GO TO 105
      TR=-2.0
      ITR=1
      GO TO 110
105   CONTINUE
      TR=ABS(-0.07037037*(Y(ITR)-YC(ITR)))
      ITR=ITR+1
110   CONTINUE
      GO TO 4800
C                        USE R-K STEP
C               NOTATION -  Y(J+1) = Y(J) + (K0+2K1+2K2+K3)/6
C                        COMPUTE SUM = K0
 4500 DO 4501 I=1,N
      S(I) = H*DY(I)
 4501 YC(I) = Y(I) + S(I)/2.0
      XC = X + H/2.0
C                        ADD 2*K1 TO SUM
      CALL USER(3,N,XC,YC,DY)
      DO 4502 I=1,N
      S(I) = S(I) + 2.0*H*DY(I)
 4502 YC(I) = Y(I) + (H*DY(I))/2.0
C                        ADD 2*K2 TO SUM
      CALL USER(3,N,XC,YC,DY)
      DO 4503 I=1,N
      S(I) = S(I) + 2.0*H*DY(I)
 4503 YC(I) = Y(I) + H*DY(I)
      XC = X + H
C                        ADD K3 TO SUM AND GET NEW Y VALUE
      CALL USER(3,N,XC,YC,DY)
      DO 4504 I=1,N
      S(I) = S(I) + H*DY(I)
```

```
 4504 Y(I) = Y(I) + S(I)/6.0
      X = XC
      NSTP = NSTP + 1
C UPDATE THE RUNGA-KUTTA FLAG IN TR AS REQUIRED.
      IF(IS1.LT.0)GO TO 130
      IS1=-ISKIP
      TR=-1.0
130   CONTINUE
C RESET THE NO DERIVATIVE SWITCH
      INODER=0
C                         COMPUTE CURRENT DY VALUES
4800  CONTINUE
      IFINAL=1
      CALL USER(3,N,X,Y,DY)
C                         ROTATE INDICES OF DYP ARRAY
      I = IN4
      IN4 = IN3
      IN3 = IN2
      IN2 = IN1
      IN1 = I
      GO TO 3001
 3001 RETURN
      END


      SUBROUTINE RESTART
C
C SUBROUTINE RESTART IS CALLED IN SUBROUTINE USER (OR ANY OTHER
C LOCATION REQUIRED) WHEN A DISCONTINUITY IN A PIECEWISE CONTINUOUS
C F(X(T),T) OCCURS.  SUBROUTINE RESTART SETS A SWITCH IN SUBROUTINE
C ESODEQ WHICH CHANGES THE INTEGRATION METHOD FROM PREDICTOR/CORRECTOR
C TO RUNGA-KUTTA; THIS CHANGE REMAINS FOR THREE INTEGRATION STEPS.
C
C
      COMMON/NODER/INODER
      INODER=1
      RETURN
      END


      SUBROUTINE STOP
C
C THE PURPOSE OF SUBROUTINE STOP IS TO PROVIDE A MEANS OF
C TERMINATING THE PROBLEM SOLUTION (AND GETTING THE REQUESTED PLOTS
C AND/OR TABULAR DATA) VIA FORTRAN CODING IN SUBROUTINE USER.
C
      COMMON /ISTOPS/ ISTOP
C SET A SWITCH WHICH WILL TERMINATE THE SOLUTION IN SUBROUTINE ODE.
      ISTOP=-1
      RETURN
      END


      SUBROUTINE OUTPUT(M,T,Y,NOUT,TITLES,ISIZE,IDUMP,ISKIP)
C
C THIS SUBROUTINE STORES THE OUTPUTS AND CALLS UP THE OUTPUTTING OF
C RESULTS WHEN THE STORAGE ARRAYS ARE FULL.
C
C THE INPUT VARIABLES ARE...
C      M          NUMBER OF ELEMENTS IN THE OUTPUT VECTOR Y.
```

```
C       T           CURRENT TIME
C       Y           VECTOR TO BE OUTPUTTED
C       NOUT        OUTPUT TAPE UNIT NUMBER
C       TITLES      ARRAY OF TITLE CARDS FOR PLOTS
C       ISIZE       SWITCH WHICH DETERMINES PLOT SIZE
C       IDUMP       =1 FORCES ALL STORED DATA TO BE OUTPUTTED
C       ISKIP       RATIO OF CALCULATED POINTS TO OUTPUTTED POINTS
C       IDEL        SWITCH THAN DETERMINES OUTPUT MODES
C
C
      COMMON/INPUT2/IDEL
      COMMON/XY1/IX(10),IY(10)
      COMMON/XY2/ISPEC,LABELX,LABELY
      DIMENSION TT(101),YY(101,31)
      DIMENSION Y(31),TITLES(4,8)
      DIMENSION TTT(101),YYY(101)
C THESE LABELS APPEAR ON BOTH THE TABULAR AND PLOTTED DATA.
      COMMON/TIE4/LABEL
      DIMENSION LABEL(80)
      DATA (LABEL(I),I=1,32 ) /10H OUTPUT 1 ,10H OUTPUT 2 ,
     210H OUTPUT 3 ,10H OUTPUT 4 ,10H OUTPUT 5 ,10H OUTPUT 6 ,
     310H OUTPUT 7 ,10H OUTPUT 8 ,10H OUTPUT 9 ,10H OUTPUT 10,
     410H OUTPUT 11,10H OUTPUT 12,10H OUTPUT 13,10H OUTPUT 14,
     510H OUTPUT 15,10H OUTPUT 16,10H OUTPUT 17,10H OUTPUT 18,
     610H OUTPUT 19,10H OUTPUT 20,10H OUTPUT 21,10H OUTPUT 22,
     710H OUTPUT 23,10H OUTPUT 24,10H OUTPUT 25,10H OUTPUT 26,
     810H OUTPUT 27,10H OUTPUT 28,10H OUTPUT 29,10H OUTPUT 30,
     910H OUTPUT 31,10H OUTPUT 32/
      DATA (LABEL(I),I=33,64) /10H OUTPUT 33,10H OUTPUT 34,
     210H OUTPUT 35,10H OUTPUT 36,10H OUTPUT 37,10H OUTPUT 38,
     310H OUTPUT 39,10H OUTPUT 40,10H OUTPUT 41,10H OUTPUT 42,
     410H OUTPUT 43,10H OUTPUT 44,10H OUTPUT 45,10H OUTPUT 46,
     510H OUTPUT 47,10H OUTPUT 48,10H OUTPUT 49,10H OUTPUT 50,
     610H OUTPUT 51,10H OUTPUT 52,10H OUTPUT 53,10H OUTPUT 54,
     710H OUTPUT 55,10H OUTPUT 56,10H OUTPUT 57,10H OUTPUT 58,
     810H OUTPUT 59,10H OUTPUT 60,10H OUTPUT 61,10H OUTPUT 62,
     910H OUTPUT 63,10H OUTPUT 64/
      DATA (LABEL(I),I=65,80) /10H OUTPUT 65,10H OUTPUT 66,
     210H OUTPUT 67,10H OUTPUT 68,10H OUTPUT 69,10H OUTPUT 70,
     310H OUTPUT 71,10H OUTPUT 72,10H OUTPUT 73,10H OUTPUT 74,
     410H OUTPUT 75,10H OUTPUT 76,10H OUTPUT 77,10H OUTPUT 78,
     510H OUTPUT 79,10HEST. ERROR/
      DATA IS1/-1/
      IF(M.LE.1)GO TO 999
      IS1=IS1+1
C CHECK IF THIS CALCULATED POINT SHOULD BE STORED.
      IF(IS1.GE.0)GO TO 11
C CHECK FOR PROBLEM TERMINATION WHEN USING ISKIP GREATER THAN ONE.
      IF(IDUMP.EQ.1)GO TO 11
      GO TO 999
11    CONTINUE
      IS1=-ISKIP
C STORE THE OUTPUT FOR TIME=T
      ICOUNT=ICOUNT+1
      TT(ICOUNT)=T
      DO 1 I=1,M
1     YY(ICOUNT,I)=Y(I)
C CHECK IF THE STORAGE ARRAYS ARE FULL.
      IF(ICOUNT.GT.100)GO TO 100
C CHECK IF THE LAST POINT HAS BEEN CALCULATED---IF SO TERMINATE PROGRAM.
```

```
      IF(IDUMP.EQ.1)GO TO 100
      GO TO 999
100   CONTINUE
C
C THE PLOTTING ARRAYS ARE FULL---OUTPUT THE DATA IN ARRAYS.
C
C FIRST, DO THE PLOTTING OF THE DATA POINTS.
      MM1=M-1
      IF((IDEL.EQ.1).OR.(IDEL.EQ.3))GO TO 150
      DO 110 I=1,MM1
      DO 111 J=1,ICOUNT
      TTT(J)=TT(J)
111   YYY(J)=YY(J,I)
C CAUTION!  ARRAYS TTT AND YYY WILL BE MODIFIED BY SUBROUTINE PLOTS.
110   CALL PLOTS(TTT,YYY,TITLES,ICOUNT,LABEL(I),NOUT,ISIZE)
C PERFORM THE X-Y PLOTS AS REQUESTED VIA SUBROUTINE XYPLOT.
      DO 200 I=1,10
      IF(IX(I).EQ.0)GO TO 200
      KX=IX(I)
      KY=IY(I)
      LABELX=LABEL(KX)
      LABELY=LABEL(KY)
      ISPEC=1
      DO 201 J=1,ICOUNT
      TTT(J)=YY(J,KX)
      YYY(J)=YY(J,KY)
201   CONTINUE
      CALL PLOTS(TTT,YYY,TITLES,ICOUNT,LABEL(I),NOUT,ISIZE)
      ISPEC=0
200   CONTINUE
150   CONTINUE
C NEXT, DO THE TABULAR LISTING OF DATA POINTS.
      IF((IDEL.EQ.2).OR.(IDEL.EQ.3))GO TO 998
      WRITE(NOUT,112)
112   FORMAT(1H1)
C WRITE OUT THE LABELS.
      WRITE(NOUT,119)(LABEL(I),I=1,MM1),LABEL(80)
119   FORMAT(1X,9H  TIME    ,10(1X,A10),/,10X,10(1X,A10),/,
     210X,10(1X,A10),/,10X,10(1X,A10),/,10X,10(1X,A10),/,
     310X,10(1X,A10),/,10X,10(1X,A10),/,10X,10(1X,A10))
      WRITE(NOUT,118)
118   FORMAT(1H )
      DO 120 I=1,ICOUNT
C WRITE THE DATA.
      WRITE(NOUT,121)TT(I),(YY(I,J),J=1,M)
121   FORMAT(1X,E9.2,10(1X,E10.3),/,10X,10(1X,E10.3),/,
     210X,10(1X,E10.3),/,10X,10(1X,E10.3),/,10X,10(1X,E10.3),/,
     310X,10(1X,E10.3),/,10X,10(1X,E10.3),/,10X,10(1X,E10.3))
120   CONTINUE
998   CONTINUE
C COPY THE LAST POINT INTO THE FIRST POSITION FOR THE CONTINUATION PLOT.
      TT(1)=TT(ICOUNT)
      DO 130 I=1,M
130   YY(1,I)=YY(ICOUNT,I)
C RESET THE ICOUNT FLAG.
      ICOUNT=1
999   CONTINUE
      RETURN
      END
```

```
      DIMENSION X(101),Y(101),TITLES(4,8)
      DIMENSION POINTS(101),POINT(41),XLABEL(6)
      COMMON/XY2/ISPEC,LABELX,LABELY
      MAXPTS=101
C CHECK TO SEE IF NUMPTS IS OUT OF RANGE.
      IF(NUMPTS.GT.MAXPTS)GO TO 999
      IF(NUMPTS.LT.2)GO TO 999
C WRITE THE HEADING FOR THE PLOT.
      IF((ISPEC.NE.1).AND.(ISIZE.EQ.0))WRITE(NOUT,6)NAME
6     FORMAT(1H1,59X,A10)
      IF((ISPEC.NE.1).AND.(ISIZE.NE.0))WRITE(NOUT,61)NAME
61    FORMAT(1H1,30X,A10)
      IF((ISPEC.EQ.1).AND.(ISIZE.EQ.0))WRITE(NOUT,62)LABELY,LABELX
62    FORMAT(1H1,37X,A10,17H (Y-AXIS) VERSUS ,A10,9H (X-AXIS))
      IF((ISPEC.EQ.1).AND.(ISIZE.NE.0))WRITE(NOUT,63)LABELY,LABELX
63    FORMAT(1H1,12X,A10,17H (Y-AXIS) VERSUS ,A10,9H (X-AXIS))
C DETERMINE THE PLOT SIZE
      IF(ISIZE.NE.0)GO TO 301
C THESE CONSTANTS ARE USED FOR THE FULL-SIZE PLOT.
      LX=100
      LY=50
      LXL=6
      GO TO 302
C THESE CONSTANTS ARE USED FOR THE REDUCED-SIZE PLOT.
301   CONTINUE
      LX=40
      LY=30
      LXL=3
302   CONTINUE
      XLX=LX
      YLY=LY
      LXP1=LX+1
      LYP1=LY+1
C WRITE OUT THE TITLE CARDS
      DO 3 I=1,4
      IF(ISIZE.EQ.0)WRITE(NOUT,4)(TITLES(I,J),J=1,8)
4     FORMAT(25X,8A10)
      IF(ISIZE.NE.0)WRITE(NOUT,41)(TITLES(I,J),J=1,8)
41    FORMAT(1X,8A10)
3     CONTINUE
      WRITE(NOUT,5)
5     FORMAT(1H )
C
C ORDER THE (X,Y) PAIRS BY DECREASING VALUES OF Y
C
C SOLVE FOR MAX
      I=1
20    CONTINUE
      JJ=I
      YMAX=Y(I)
      DO 10 J=I,NUMPTS
      IF(Y(J).LE.YMAX)GO TO 10
      YMAX=Y(J)
      JJ=J
10    CONTINUE
C INTERCHANGE
      YUPPER=Y(I)
      XX=X(I)
      Y(I)=Y(JJ)
      X(I)=X(JJ)
```

```
      Y(JJ)=YUPPER
      X(JJ)=XX
      I=I+1
      IF(I.EQ.NUMPTS)GO TO 30
      GO TO 20
30    CONTINUE
C SOLVE FOR MIN/MAX OF X AND Y.
      XMIN=X(1)
      XMAX=X(1)
      YMIN=Y(1)
      YMAX=Y(1)
      DO 2 I=1,NUMPTS
      IF(X(I).LT.XMIN)XMIN=X(I)
      IF(X(I).GT.XMAX)XMAX=X(I)
      IF(Y(I).LT.YMIN)YMIN=Y(I)
      IF(Y(I).GT.YMAX)YMAX=Y(I)
2     CONTINUE
C IF THE PLOT DATA IS CONSTANT, DO NOT PLOT---THIS WILL SAVE ON
C WRITING FORMATTED IO.
      IF((YMIN.NE.YMAX).OR.(ISPEC.EQ.1))GO TO 320
      WRITE(NOUT,322)
322   FORMAT(10/)
      WRITE(NOUT,321)NAME,YMIN
321   FORMAT(10X,A10,24H IS A CONSTANT OF VALUE ,E20.13,/)
      GO TO 999
320   CONTINUE
      IF((ISPEC.NE.1).OR.(XMIN.NE.XMAX).OR.(YMIN.NE.YMAX))GO TO 330
      WRITE(NOUT,322)
      WRITE(NOUT,321)LABELX,XMIN
      WRITE(NOUT,321)LABELY,YMIN
      GO TO 999
330   CONTINUE
C RESET THE END POINTS.
      CALL ENDPTS(XMIN,XMAX)
      CALL ENDPTS(YMIN,YMAX)
C CALCULATE DELX AND DELY.
      DELX=(XMAX-XMIN)/XLX
      DELY=(YMAX-YMIN)/YLY
C XTHRES AND YTHRES ARE USED AS NOISE THRESHOLDS IN LABELLING THE AXES.
      XTHRES=ABS(XMAX)
      IF(ABS(XMIN).GT.ABS(XMAX))XTHRES=ABS(XMIN)
      XTHRES=0.001*XTHRES
      YTHRES=ABS(YMAX)
      IF(ABS(YMIN).GT.ABS(YMAX))YTHRES=ABS(YMIN)
      YTHRES=0.001*YTHRES
C
C GENERATE THE PLOT
C
C CALCULATE THE POSITION (IF ANY) OF THE X-AXIS
      KX=ABS(XMIN/DELX)+1.0
      IF(XMIN.EQ.0.0)KX=1
      IF(XMAX.EQ.0.0)KX=LXP1
      IF(KX.GT.LXP1)KX=LXP1
      IZERO=0
      IF((XMIN.LE.0.0).AND.(XMAX.GE.0.0))IZERO=1
C CALCULATE THE LINE (IF ANY) OF THE Y-AXIS
      KY=ABS(YMAX/DELY)+1.0
      IF((YMAX.LT.0.0).OR.(YMIN.GT.0.0))KY=0
      IF(YMAX.EQ.0.0)KY=1
      IF(YMIN.EQ.0.0)KY=LYP1
```

```
      ICOUNT=10
      LIST=1
      YLOWER=YMAX
      DO 100 I=1,LYP1
      YUPPER=YLOWER
      YLOWER=YMAX-I*DELY
C ZERO THE POINTS ARRAY (START A NEW LINE OF THE PLOT)
      DO 101 J=1,LXP1
101   POINTS(J)=1H
      IF(ICOUNT.NE.10)GO TO 105
      DO 106 J=1,LXP1,2
106   POINTS(J)=1H.
105   CONTINUE
C WRITE OUT COORDINATE MARKERS
      POINTS(  1)=1H.
      POINTS( 21)=1H.
      POINTS( 41)=1H.
      POINTS( 61)=1H.
      POINTS( 81)=1H.
      POINTS(101)=1H.
C WRITE OUT THE ZERO-MARKER FOR X=0
      IF(IZERO.EQ.1)POINTS(KX)=1HI
C WRITE OUT THE ZERO-MARKER FOR Y=0
      IF(I.NE.KY)GO TO 137
      DO 136 J=1,LXP1
136   POINTS(J)=1H-
137   CONTINUE
C LOOPING AROUND LOOP 102 PLACES THE SYMBOL X ON THE X-AXIS FOR EACH
C (X,Y) PAIR THAT SATISFIES....
C    (Y.GT.YLOWER).AND.(Y.LE.YUPPER)
102   CONTINUE
      IF(LIST.GT.NUMPTS)GO TO 110
      IF(Y(LIST).LE.YLOWER)GO TO 110
      K=(X(LIST)-XMIN)/DELX+1.0
      IF(K.GT.LXP1)K=LXP1
      POINTS(K)=1HX
      LIST=LIST+1
      GO TO 102
110   CONTINUE
C WRITE OUT A SINGLE LINE OF THE PLOT.  DETERMINE WHICH OF FOUR
C POSSIBLE WRITE STATEMENTS TO USE.
      IF(ICOUNT.EQ.10)GO TO 112
      ICOUNT=ICOUNT+1
C FOR PROGRAM EFFICIENCY, OUTPUT ARRAYS POINTS AND POINT AS FOLLOWS.
      IF(ISIZE.NE.0)GO TO 210
C WRITE STATEMENT FOR LARGE PLOTS.
      WRITE(NOUT,111)POINTS
111   FORMAT(15X,101A1)
      GO TO 220
210   CONTINUE
C WRITE STATEMENT FOR SMALL PLOTS.
      DO 211 J=1,LXP1
      POINT(J)=POINTS(J)
211   CONTINUE
      WRITE(NOUT,215)POINT
215   FORMAT(15X,41A1)
220   CONTINUE
      GO TO 100
112   CONTINUE
      ICOUNT=1
```

```
      IF((YUPPER.GT.-YTHRES).AND.(YUPPER.LT.YTHRES))YUPPER=0.0
C FOR PROGRAM EFFICIENCY, OUTPUT ARRAYS POINTS AND POINT AS FOLLOWS.
      IF(ISIZE.NE.0)GO TO 230
C WRITE STATEMENT FOR LARGE PLOTS.
      WRITE(NOUT,113)YUPPER,POINTS
113   FORMAT(2X,E11.4,2X,101A1)
      GO TO 240
230   CONTINUE
C WRITE STATEMENT FOR SMALL PLOTS.
      DO 231 J=1,LXP1
      POINT(J)=POINTS(J)
231   CONTINUE
      WRITE(NOUT,235)YUPPER,POINT
235   FORMAT(2X,E11.4,2X,41A1)
240   CONTINUE
100   CONTINUE
      DO 121 I=1,6
      XI=I-1
      XLABEL(I)=XMIN+20.0*DELX*XI
      IF((XLABEL(I).LT.XTHRES).AND.(XLABEL(I).GT.-XTHRES))XLABEL(I)=0.0
121   CONTINUE
      WRITE(NOUT,122)(XLABEL(J),J=1,LXL)
122   FORMAT(/,10X,6(E10.3,10X))
      IF(ISPEC.EQ.1)GO TO 999
      IF(ISIZE.EQ.0)WRITE(NOUT,202)
202   FORMAT(58X,15HTIME IN SECONDS)
      IF(ISIZE.NE.0)WRITE(NOUT,203)
203   FORMAT(28X,15HTIME IN SECONDS)
999   CONTINUE
      RETURN
      END


      SUBROUTINE ENDPTS(XMIN,XMAX)
C THIS SUBROUTINE RESETS THE END POINTS FOR SUBROUTINE XYPLOT.  THIS
C INSURES EVEN NUMBERS ON THE PLOTS.
      DIMENSION A(38)
      DATA (A(I),I=1,38)/0.0,0.1,0.25,0.50,0.75,1.0,1.1,1.25,1.50,1.75,
     22.00,2.50,3.00,3.50,4.00,4.50,5.0,6.0,7.0,8.0,9.0,10.0,11.,12.5,
     315.,17.5,20.,25.,30.,35.,40.,45.,50.,60.,70.,80.,90.,100./
C CHECK FOR EQUAL ENDPOINTS (I.E. A CONSTANT)
      IF(XMIN.NE.XMAX)GO TO 1
      IF(XMIN.NE.0.0)GO TO 3
      XMIN=XMIN-5.0E-99
      XMAX=XMAX+5.0E-99
      GO TO 999
3     CONTINUE
      XMIN=XMIN*(0.999999)
      XMAX=XMAX*(1.000001)
      GO TO 999
1     CONTINUE
C CHECK FOR CORRECT ALGEBRAIC ORDERING
      DEL=XMAX-XMIN
      IF(DEL.GT.0.0)GO TO 2
      XX=XMAX
      XMAX=XMIN
      XMIN=XX
      DEL=-DEL
2     CONTINUE
C DEL IS POSITIVE AT THIS POINT.
```

```
      VALUE=1.0
      IF(DEL.LE.1.0)GO TO 10
5     CONTINUE
      IF(DEL.LT.VALUE)GO TO 20
      VALUE=VALUE*10.0
      GO TO 5
10    CONTINUE
      IF(DEL.GE.VALUE)GO TO 11
      VALUE=VALUE*0.1
      GO TO 10
11    VALUE=VALUE*10.0
20    CONTINUE
C AT THIS POINT, ONE HAS SELECTED VALUE SUCH THAT...
C         0.1*VALUE.LE.DEL    AND    DEL.LT.VALUE
      XX=XMIN/VALUE
      IXX=XX
      XX=IXX
      XX=XX*10.0
C XX REPRESENTS THOSE DIGITS COMMON TO BOTH XMIN AND XMAX
      XXMIN=XMIN*10.0/VALUE-XX
      XXMAX=XMAX*10.0/VALUE-XX
      IF(XXMIN.EQ.0.0)GO TO 30
      IF(XXMIN.LT.0.0)GO TO 35
C XXMIN IS POSITIVE.
      DO 32 I=2,38
      AAA=A(I)
      IF(XXMIN.LT.AAA)GO TO 33
32    CONTINUE
33    I=I-1
      XXMIN=A(I)
      GO TO 30
35    CONTINUE
C XXMIN IS NEGATIVE.
      XXMIN=-XXMIN
      DO 36 I=2,38
      AAA=A(I)
      IF(XXMIN.LT.AAA)GO TO 37
36    CONTINUE
37    XXMIN=-A(I)
30    CONTINUE
      IF(XXMAX.EQ.0.0)GO TO 40
      IF(XXMAX.LT.0.0)GO TO 45
C XXMAX IS POSITIVE
      DO 42 I=2,38
      AAA=A(I)
      IF(XXMAX.LE.AAA)GO TO 43
42    CONTINUE
43    XXMAX=A(I)
      GO TO 40
45    CONTINUE
C XXMAX IS NEGATIVE.
      XXMAX=-XXMAX
      DO 46 I=2,38
      AAA=A(I)
      IF(XXMAX.LE.AAA)GO TO 47
46    CONTINUE
47    I=I-1
      XXMAX=-A(I)
40    CONTINUE
C SOLVE FOR NEW END POINTS.
```

```
      XMIN=(XX+XXMIN)*VALUE/10.0
      XMAX=(XX+XXMAX)*VALUE/10.0
999   CONTINUE
      RETURN
      END


      SUBROUTINE MINMAX(ID,NAME,VALUE)
C
C SUBROUTINE MINMAX IS USED IN PROGRAM SS TO FIND THE MINIMUM AND
C MAXIMUM OF THE SPECIFIED VARIABLE AND THE TIME AT WHICH THESE OCCUR;
C THIS SUBROUTINE IS TYPICALLY USED WHEN THE NUMBER OF CALCULATED POINTS
C TO OUTPUTTED POINTS (I.E. ISKIP) IS LARGE.  THE INPUT VARIABLES HAVE
C THE FOLLOWING MEANING...
C         ID         IDENTIFICATION NUMBER (I.E. 1,2,...  )
C         NAME       A 10H NAME USED ON THE OUTPUT
C         VALUE      CURRENT VALUE OF VARIABLE FOR WHICH THE MIN/MAX IS
C                    DESIRED
C
      DIMENSION NAMES(10),ISTART(10),TMIN(10),TMAX(10),VMIN(10),VMAX(10)
      IMAX=10
      COMMON /FINAL/IFINAL
      COMMON/TIE1/NIN,NOUT,M,ALINE
      COMMON/TIE2/TIME
      DATA IUSED / 0 /
      DATA ISTART/79*0/
C CHECK FOR END OF THE PROBLEM (I.E. OUTPUT THE MIN/MAX DATA)
      IF(ID.LE.0)GO TO 20
C CHECK FOR THE START OF THE PROBLEM
      IF(ISTART(ID).EQ.0)GO TO 10
C CHECK FOR FINAL VALUE OF THE STEP
      IF(IFINAL.NE.1)GO TO 999
C THIS IS THE NORMAL FLOW PATH.
      IF(VALUE.GT.VMIN(ID))GO TO 5
      VMIN(ID)=VALUE
      TMIN(ID)=TIME
5     CONTINUE
      IF(VALUE.LT.VMAX(ID))GO TO 999
      VMAX(ID)=VALUE
      TMAX(ID)=TIME
      GO TO 999
10    CONTINUE
C INITIALIZE THE ARRAYS.
      NAMES(ID)=NAME
      ISTART(ID)=1
      VMIN(ID)=VALUE
      VMAX(ID)=VALUE
      TMIN(ID)=TIME
      TMAX(ID)=TIME
      IUSED=1
      GO TO 999
20    CONTINUE
C CHECK TO SEE IF THE MINMAX OPTION USED FOR THIS PROBLEM.
      IF(IUSED.NE.1)GO TO 999
C OUTPUT THE MIN/MAX INFORMATION.
      WRITE(NOUT,21)
21    FORMAT(1H1,5/,26HMINIMUM/MAXIMUM DATA IS...,3/)
      WRITE(NOUT,22)
22    FORMAT(48H  VARIABLE        --------  M I N I M U M  -------,5X,
     232H--------  M A X I M U M  -------,5X,20H---- DIFFERENCE ----)
```

```
      WRITE(NOUT,23)
23    FORMAT(4X,4HNAME,15X,5HVALUE,13X,4HTIME,15X,5HVALUE,13X,4HTIME,/)
      I=0
24    CONTINUE
      I=I+1
      IF(I.GT.IMAX)GO TO 999
      IF(ISTART(I).NE.1)GO TO 24
      DIFF=VMAX(I)-VMIN(I)
      WRITE(NOUT,25)NAMES(I),VMIN(I),TMIN(I),VMAX(I),TMAX(I),DIFF
25    FORMAT(1X,A10,5X,E20.13,2X,E10.3,5X,E20.13,2X,E10.3,5X,E20.13)
      GO TO 24
      GO TO 999
999   CONTINUE
      RETURN
      END


      SUBROUTINE DELAY(ID,IUNITS,XIN,XOUT)
C
C
C SUBROUTINE DELAY STORES AND RECALLS DATA TO PROVIDE A DELAY OPERATION
C FOR PROGRAM SS.  AS PRESENTLY DIMENSIONED, UP TO 5 DISTINCT DELAY
C TIMES ARE ALLOWED.  THE RANGE OF POSSIBLE FINAL IS 2 TO 100 STEP
C SIZES; THE MINIMUM OF 2 IS SET BY THE RUNGA-KUTTA STARTER AND
C THE MAXIMUM IS SET BY DIMENSION STATEMENTS.  THE  NUMBER OF DELAY
C OPERATORS IS LIMITED BY PROGRAM DIMENSION STATEMENTS.
C         ID          IDENTIFICATION NUMBER (1,2,3,4, AND/OR 5)
C         IUNITS      THE NUMBER OF INTEGRATION STEPS (UNITS) OF DELAY
C         XIN         INPUT TO DELAY OPERATION
C         XOUT        OUTPUT OF DELAY OPERATION
C
C
      DIMENSION STORAGE(5,101),NEXT(5),IFIRST(5)
      DATA IMAX /101/
      DATA (IFIRST(I),I=1,5) /0,0,0,0,0/
      COMMON/FINAL/IFINAL
      IF(IFIRST(ID).NE.0)GO TO 5
C INITIALIZE THE ID PORTION OF STORAGE.
      DO 2 I=1,IMAX
      STORAGE(ID,I)=XIN
2     CONTINUE
      NEXT(ID)=1
      IFIRST(ID)=1
      XOUT=XIN
      GO TO 999
5     CONTINUE
C CHECK THE RANGE OF THE REQUESTED DELAY.
      IF((IUNITS.GT.1).AND.(IUNITS.LT.IMAX))GO TO 10
C FOR NEGATIVE, ZERO, OR ONE DELAY UNITS, OUTPUT THE INPUT.
      XOUT=XIN
      GO TO 999
10    CONTINUE
C GET XOUT FROM STORAGE.
      I=NEXT(ID)
      XOUT=STORAGE(ID,I)
      IF(IFINAL)999,999,20
20    CONTINUE
C STORE THE CURRENT VALUE OF XIN AND UPDATE NEXT(ID).
      I=I+IUNITS
      IF(I.GT.IMAX)I=I-IMAX
```

```
      STORAGE(ID,I)=XIN
      NEXT(ID)=NEXT(ID)+1
      IF(NEXT(ID).GT.IMAX)NEXT(ID)=1
999   CONTINUE
      RETURN
      END


      SUBROUTINE LD(XIN,XON,XOFF,MODE,LEVEL)
C
C
C THIS SUBROUTINE SIMULATES THE ACTIONS OF A LEVEL DETECTOR THAT
C HAS HYSTERSIS.  INPUT/OUTPUT TO THIS SUBROUTINE IS AS FOLLOWS...
C    XIN         ANALOG INPUT SIGNAL
C    XON         THE ANALOG LEVEL AT WHICH THE OUTPUT GOES TO THE 1 STATE
C    XOFF        THE ANALOG LEVEL AT WHICH THE OUTPUT GOES TO THE 0 STATE
C    MODE        =0 NORMAL LEVEL DETECTOR; =1 INVERTED LEVEL DETECTOR
C    LEVEL        DIGITAL OUTPUT SIGNAL (I.E. 1 OR 0)
C
C
C THE NORMAL LEVEL DETECTOR LOOKS LIKE...
C
C    1 LEVEL           ---V-----V--]--
C                         I     I
C                         I     I
C                         V     I
C                         I     I
C                         I     I
C    0 LEVEL --V--]------]---
C                       XOFF     XON
C
C
C THE INVERTED LEVEL DETECTOR LOOKS LIKE...
C
C    1 LEVEL --V--]------]---
C                         I     I
C                         I     I
C                         I     V
C                         I     I
C                         I     I
C    0 LEVEL           ---V-----V--]--
C                        XON     XOFF
C
C
      LOLD=LEVEL
      IF(MODE.NE.0)GO TO 20
C
C THIS IS THE NORMAL LEVEL DETECTOR.
C
      IF(XIN.LT.XON)GO TO 11
C LEVEL HAS VALUE 1
      LEVEL=1
      GO TO 999
11    CONTINUE
      IF(XIN.GT.XOFF)GO TO 12
C LEVEL HAS VALUE 0
      LEVEL=0
      GO TO 999
12    CONTINUE
C THE VALUE OF LEVEL IS UNCHANGED.
```

```
      GO TO 999
20    CONTINUE
C
C THIS IS THE INVERTED LEVEL DETECTOR.
C
      IF(XIN.GT.XON)GO TO 21
C LEVEL HAS VALUE 1
      LEVEL=1
      GO TO 999
21    CONTINUE
      IF(XIN.LT.XOFF)GO TO 22
C LEVEL HAS VALUE 0
      LEVEL=0
      GO TO 999
22    CONTINUE
C THE VALUE OF LEVEL IS UNCHANGED.
      GO TO 999
999   CONTINUE
C IF A LEVEL DETECTOR CHANGE OCCURS, CHANGE THE INTEGRATION METHOD.
      IF(LOLD.NE.LEVEL)CALL RESTART
      RETURN
      END


      SUBROUTINE IMPEQS(ID,Z,ITERS,EPSC,EPSJ,GAIN,IFLAG)
C
C THE PURPOSE OF THIS SUBROUTINE IS TO PROVIDE A MEANS FOR SOLVING
C IMPLICIT EQUATIONS IN PROGRAM SS.  THE USER SUPPLIES THE NONLINEAR
C ALGEBRAIC EQUATIONS VIA SUBROUTINE NAE; SUBROUTINE IMPEQS IS CALLED
C IN SUBROUTINE USER AT THE POINT WHERE ONE WISHES TO SOLVE THE IMPLICIT
C NONLINEAR ALGEBRAIC EQS.  THE INPUT PARAMETERS TO THIS SUBROUTINE ARE
C AS FOLLOWS....
C
C  ID          INTEGER IDENTIFICATION   (1,2,3,4,5) THAT TELLS THE PROGRAM
C              WHICH SET OF NONLINEAR ALGEBRAIC EQUATIONS TO SOLVE.
C  Z           ARRAY WHICH CONTAINS THE VARIABLES OF THE NONLINEAR
C              EQUATIONS.
C  ITERS       MAXIMUM NUMBER OF ITERATIONS THAT THE NEWTON-RAPHSON
C              PROCESS IS ALLOWED TO ITERATE
C  EPSC        EPSILON USED TO JUDGE CONVERGENCE
C  EPSJ        EPSILON USED TO ESTIMATE JACOBIAN
C  IFLAG       =0 IF OK; OTHERWISE, A PROBLEM HAS OCCURED
C  GAIN        GAIN OF CORRECTION TERM; TYPICALLY, GAIN=1
C
C
      DIMENSION Z(3),ERROR(3),A(3,3),F(3),CORR(3),ABSERR(3)
C CHECK FOR OUT OF RANGE CONDITIONS.
      IF((ID.LT.1).OR.(ID.GT.5))GO TO 998
      IF(ITERS.LE.0)GO TO 998
      IF((EPSJ.EQ.0.0).OR.(EPSC.LE.0.0))GO TO 998
      IF(GAIN.EQ.0.0)GO TO 998
C SOLVE FOR THE Z VECTOR SUCH THAT F(Z)=0
      CALL NAE(ID,N,Z,ERROR)
C CHECK FOR OUT OF RANGE N.
      IF((N.LT.1).OR.(N.GT.3))GO TO 998
      DO 1 K=1,ITERS
C GENERATE A NUMERICAL APPROXIMATION TO THE JACOBIAN OF F AT Z
      DO 10 I=1,N
      STORE=Z(I)
      Z(I)=Z(I)+EPSJ
```

```
      CALL NAE(ID,N,Z,F)
C ESTIMATE THE JACOBIAN
      DO 11 J=1,N
      A(J,I)=(F(J)-ERROR(J))/EPSJ
11    CONTINUE
      Z(I)=STORE
10    CONTINUE
C SOLVE FOR THE NEWTON-RAPHSON CORRECTION TERM
      CALL GAUSS(N,A,ERROR,CORR,IFLAG)
      IF(IFLAG.NE.0)GO TO 998
C UPDATE THE Z-ARRAY
      DO 31 I=1,N
      Z(I,=Z(I)-CORR(I)*GAIN
31    CONTINUE
      CALL NAE(ID,N,Z,ERROR)
C CHECK FOR CONVERGENCE
      ERRMAX=0.0
      DO 32 I=1,N
      ABSERR(I)=ABS(ERROR(I))
      IF(ABSERR(I).GT.ERRMAX)ERRMAX=ABSERR(I)
32    CONTINUE
      IF(ERRMAX.LE.EPSC)GO TO 999
1     CONTINUE
      IFLAG=2
      GO TO 999
998   CONTINUE
      IFLAG=1
999   CONTINUE
      RETURN
      END


      SUBROUTINE GAUSS(N,A,B,X,IFLAG)
C
C SUBROUTINE GAUSS SOLVES THE VECTOR EQUATION A*X=B FOR THE X VECTOR
C GIVEN THAT THE A MATRIX AND B VECTOR ARE KNOWNS AND THAT THE
C A MATRIX HAS FULL RANK.  PROBLEMS MAY OCCUR FOR NEAR-SINGULAR A
C MATRICES; IF SO, ERROR MESSAGES ARE PRINTED AND IFLAG IS
C MADE NONZERO.  A,B, AND X ARE DEFINED OVER THE FIELD OF REAL
C NUMBERS.  INPUT/OUTPUT IS AS FOLLOWS...
C         N IS THE SYSTEM ORDER
C         A IS SYSTEM MATRIX
C         B IS INPUT VECTOR
C         X IS SOLUTION VECTOR
C         NOUT IS THE LOGICAL TAPE UNIT NUMBER
C         IFLAG=0     GAUSS ELIMINATION PERFORMED
C         IFLAG=1     GAUSS ELIMINATION CAN NOT BE PERFORMED
C
C
C THIS SUBROUTINE IS TAKEN FROM COMPUTER SOLUTION OF LINEAR ALGEBRAIC
C SYSTEMS BY G. FORSYTHE AND C. B. MOLER, PRENTICE-HALL  1967, PP 68-70.
C MODIFICATIONS WERE MADE TO THIS SUBROUTINE TO CHANGE THE MANNER
C IN WHICH ERROR MESSAGES ARE HANDLED.
C
C
C TO CHANGE THE MAXIMUM SIZE MATRIX THAT ONE CAN HANDLE, CHANGE
C THE VALUE OF NMAX IN THIS SUBROUTINE AND ALL DIMENSION STATEMENTS
C IN THIS SUBROUTINE PLUS SUBROUTINES DECOMP, SOLVE. AND IMPRUV.
C
C
```

```
      NMAX=03
      DIMENSION A(03,03),UL(03,03),B(03),X(03)
      IFLAG=0
C CHECK THE VALUE OF N
      IF((N.GT.0).AND.(N.LE.NMAX))GO TO 40
      IFLAG=1
      WRITE(NOUT,14)
14    FORMAT(38HIN A CALL TO GAUSS, N IS OUT OF RANGE.)
      GO TO 999
40    CONTINUE
      IF(N.NE.1)GO TO 41
      X=B(1)/A(1,1)
      GO TO 999
41    CONTINUE
C DECOMPOSE MATRIX A INTO UPPER AND LOWER TRIANGLE MATRICES, STORE IN UL
      CALL DECOMP(N,A,UL,IFLAG)
      IF(IFLAG.NE.0)GO TO 10
C SOLVE SYSTEM OF EQUATIONS USING U AND L MATRICES.
      CALL SOLVE(N,UL,B,X)
C USE IMPROVEMENT TO CONVERGE ON TRUE ANSWER.
      CALL IMPRUV(N,A,UL,B,X,DIGITS,IFLAG)
10    CONTINUE
C
C THE ERROR PRINTOUT HAVE BEEN SUPPRESSED FOR USE IN IMPEQS.
C
      IFLAG=IFLAG+1
      GO TO(1,2,3,4),IFLAG
2     CONTINUE
C     WRITE(NOUT,11)
   11 FORMAT(54H0MATRIX WITH ZERO ROW IN DECOMPOSE.                   )
      GO TO 1
3     CONTINUE
C     WRITE(NOUT,12)
   12 FORMAT(54H0SINGULAR MATRIX IN DECOMPOSE.  ZERO DIVIDE IN SOLVE.  )
      GO TO 1
4     CONTINUE
C     WRITE(NOUT,13)
   13 FORMAT(54H0NO CONVERGENCE IN IMPRUV. MATRIX IS NEARLY SINGULAR.  )
1     CONTINUE
      IFLAG=IFLAG-1
999   CONTINUE
      RETURN
      END


      SUBROUTINE DECOMP (NN, A, UL, IFLAG)
      DIMENSION A(03,03), UL(03,03), SCALES(03), IPS(03)
      COMMON / AA / IPS
      N = NN
C
C     INITIALIZE IPS, UL AND SCALES
      DO 5 I = 1,N
         IPS(I) = I
         ROWNRM = 0.0
         DO 2 J = 1,N
            UL(I,J) = A(I,J)
            IF(ROWNRM-ABS(UL(I,J))) 1,2,2
    1       ROWNRM = ABS(UL(I,J))
    2    CONTINUE
         IF (ROWNRM) 3,4,3
```

```
    3     SCALES(I) = 1.0/ROWNRM
          GO TO 5
    4     IFLAG=1
          GO TO 19
    5 CONTINUE
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
      NM1 = N-1
      DO 17 K = 1,NM1
          BIG = 0.0
          DO 11 I = K,N
              IP = IPS(I)
              SIZE = ABS(UL(IP,K))*SCALES(IP)
              IF (SIZE-BIG) 11,11,10
   10             BIG = SIZE
                  IDXPIV = I
   11     CONTINUE
          IF (BIG) 13,12,13
   12         IFLAG=2
              GO TO 19
   13     IF (IDXPIV-K) 14,15,14
   14         J = IPS(K)
              IPS(K) = IPS(IDXPIV)
              IPS(IDXPIV) = J
   15     KP = IPS(K)
          PIVOT = UL(KP,K)
          KP1 = K+1
          DO 16 I = KP1,N
              IP = IPS(I)
              EM = -UL(IP,K)/PIVOT
              UL(IP,K) = -EM
              DO 16 J = KP1,N
                  UL(IP,J) = UL(IP,J) + EM*UL(KP,J)
C             INNER LOOP.  USE MACHINE LANGUAGE CODING IF COMPILER
C         DOES NOT PRODUCE EFFICIENT CODE.
   16     CONTINUE
   17 CONTINUE
      KP = IPS(N)
      IF (UL(KP,N)) 19,18,19
   18 IFLAG=2
19    CONTINUE
      RETURN
      END


      SUBROUTINE SOLVE (NN, UL, B, X)
      DIMENSION UL(03,03), B(03), X(03), IPS(03)
      COMMON / AA / IPS
      N = NN
      NP1 = N+1
C
      IP = IPS(1)
      X(1) = B(IP)
      DO 2 I = 2,N
          IP = IPS(I)
          IM1 = I-1
          SUM = 0.0
          DO 1 J = 1,IM1
    1         SUM = SUM + UL(IP,J)*X(J)
    2 X(I) = B(IP) - SUM
```

```
C
      IP = IPS(N)
      X(N) = X(N)/UL(IP,N)
      DO 4 IBACK = 2,N
      I = NPI-IBACK
C        I GOES (N-1),....,1
         IP = IPS(I)
         IP1 = I+1
         SUM = 0.0
         DO 3 J = IP1,N
    3       SUM = SUM + UL(IP,J)*X(J)
    4 X(I) = (X(I)-SUM)/UL(IP,I)
      RETURN
      END


      SUBROUTINE IMPRUV (NN, A, UL, B, X, DIGITS, IFLAG)
      DIMENSION A(03,03), UL(03,03), B(03), X(03), R(03), DX(03)
C     USES ABS(), AMAX1(), ALOG10()
      DOUBLE PRECISION SUM
      N = NN
C
      EPS = 2.**(-47)
      ITMAX = 29
C     +++  EPS AND ITMAX ARE MACHINE DEPENDENT. +++
C
      XNORM = 0.0
      DO 1 I = 1,N
    1    XNORM = AMAX1(XNORM,ABS(X(I)))
      IF (XNORM) 3,2,3
    2    DIGITS = -ALOG10(EPS)
         GO TO 10
C
    3 DO 9 ITER = 1,ITMAX
         DO 5 I = 1,N
            SUM = 0.0
            DO 4 J = 1,N
    4          SUM = SUM + A(I,J)*X(J)
            SUM = B(I) - SUM
    5    R(I) = SUM
C        +++ IT IS ESSENTIAL THAT A(I,J)*X(J) YIELD A DOUBLE PRECISION
C            RESULT AND THAT THE ABOVE + AND - BE DOUBLE PRECISION.  +++
         CALL SOLVE (N,UL,R,DX)
         DXNORM = 0.0
         DO 6 I = 1,N
            T = X(I)
            X(I) = X(I) + DX(I)
            DXNORM = AMAX1(DXNORM,ABS(X(I)-T))
    6    CONTINUE
         IF (ITER-1) 8,7,8
    7       DIGITS = -ALOG10(AMAX1(DXNORM/XNORM,EPS))
    8    IF (DXNORM-EPS*XNORM) 10,10,9
    9 CONTINUE
C     ITERATION DID NOT CONVERGE
      IFLAG=3
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NAE(ID,N,Z,F)
C
C THE PURPOSE OF THIS SUBROUTINE IS TO PROVIDE A PLACE FOR THE NONLINEAR
C ALGEBRAIC EQUATIONS (NAE) TO BE INPUTTED TO IMPEQS:  SUBROUTINE
C IMPEQS SOLVES THE IMPLICIT EQUATIONS.  THE EQUATIONS TO BE SOLVED ARE
C ASSUMED TO BE IN THE FORM SUCH THAT A SOLUTION VECTOR Z MAKES F(Z)
C EQUAL TO THE NULL VECTOR, THAT IS: F(Z)=0, WHERE F, Z, AND 0 ARE
C VECTORS.  AS PRESENTLY SETUP, UP TO FIVE SETS OF NONLINEAR EQUATIONS
C MAY BE SOLVED IN ONE SS PROBLEM; EACH SET OF EQUATIONS MAY HAVE 1, 2,
C OR 3 EQUATIONS.  THE INPUT VARIABLES HAVE THE FOLLOWING MEANING....
C
C  ID         INTEGER IDENTIFICATION NUMBER (1 THROUGH 5) THAT TELLS THE
C             PROGRAM WHICH SET OF NONLINEAR ALGEBRAIC EQUATIONS TO SOLVE
C  N          NUMBER OF EQUATIONS IN SET
C  Z          ARRAY WHICH CONTAINS THE VARIABLES OF THE NONLINEAR EQS.
C  F          ARRAY F EVALUATED AT Z
C
C WHEN USING THE IMPLICIT EQUATION OPTION, ONE MUST SUPPLY N
C AND THE VECTOR FUNCTION F FOR EACH SET OF EQUATIONS IN SUBROUTINE
C NAE.  IN ADDITION, ONE MUST HAVE A CALL TO SUBROUTINE IMPEQS IN
C SUBROUTINE USER FOR EACH SET OF EQUATIONS IN SUBROUTINE NAE.
C
C
      DIMENSION Z(3),F(3)
      IF((ID.LT.1).OR.(ID.GT.5))GO TO 999
      GO TO(100,200,300,400,500),ID
C
C SET OF EQUATIONS NUMBER ONE.
100   CONTINUE
      N=0
      F(1)=0.0
      F(2)=0.0
      F(3)=0.0
      GO TO 999
C
C SET OF EQUATIONS NUMBER TWO.
200   CONTINUE
      N=0
      F(1)=0.0
      F(2)=0.0
      F(3)=0.0
      GO TO 999
C
C SET OF EQUATIONS NUMBER THREE.
300   CONTINUE
      N=0
      F(1)=0.0
      F(2)=0.0
      F(3)=0.0
      GO TO 999
C
C SET OF EQUATIONS NUMBER FOUR.
400   CONTINUE
      N=0
      F(1)=0.0
      F(2)=0.0
      F(3)=0.0
      GO TO 999
C
C SET OF EQUATIONS NUMBER FIVE.
```

```
500   CONTINUE
      N=0
      F(1)=0.0
      F(2)=0.0
      F(3)=0.0
      GO TO 999
999   CONTINUE
      RETURN
      END


      SUBROUTINE USER(MODE,N,T,X,XDOT)
C
C
C THE VARIABLES USED BY PROGRAM SS ARE AS FOLLOWS...
C       MODE     SWITCH USED BY PROGRAM SS TO SELECT VARIOUS PARTS
C                OF SUBROUTINE USER.
C       NIN      TAPE UNIT NUMBER FOR READING USER DEFINED INPUT
C       NOUT     TAPE UNIT NUMBER FOR ECHOING USER DEFINED INPUT
C       N        DIMENSION OF THE STATE VECTOR X
C       M        NUMBER OF VARIABLES TO BE OUTPUTTED
C       T        CURRENT VALUE OF TIME
C       X        STATE VECTOR---THESE VARIABLES ARE THE RESULT OF THE
C                DIGITAL INTEGRATION.
C       XDOT     CURRENT VALUE OF THE TIME DERIVATIVE OF X EVALUATED
C                AT THE CURRENT TIME T.
C       Y        OUTPUT VECTOR---THESE VARIABLES WILL BE OUTPUTTED.
C
C NOTE:  EVERYTHING IN SECTIONS 300 AND 400 IS REQUIRED.  EVERY-
C        THING IN SECTIONS 100, 200, AND 500 IS OPTIONAL.
C
      DIMENSION X(20),XDOT(20),Y(31),LABEL(80)
      COMMON/TIE1/NIN,NOUT,M,ALINE
      COMMON/TIE3/Y
      COMMON/TIE4/LABEL
      GO TO(100,200,300,400,500)MODE
C
C
C THE USER PLACES ALL OF HIS CODING BETWEEN THE TWO + LINES.
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C
100   CONTINUE
C
C THE USER INSERTS USER DEFINED INPUT READ/WRITE STATEMENTS HERE.
C THE INPUT TAPE UNIT NUMBER MUST BE NIN AND THE OUTPUT TAPE UNIT
C NUMBER MUST BE NOUT.
      READ(NIN,101)A,B,C
101   FORMAT(3E10.3)
      WRITE(NOUT,101)A,B,C
      GO TO 999
200   CONTINUE
C
C ONE CAN DO ONE-TIME PRECALCULATIONS AND OUTPUT LABELLING IN
C THIS SECTION.
C
      D=SQRT(A+B)
C OVERWRITE THE STANDARD OUTPUT LABEL HERE.  AN EXAMPLE IS...
C     LABEL(1)=10HOUTPUT   1
```

```
      LABEL(1)=10HSTATE NO 1
      LABEL(2)=10HSTATE NO 2
      LABEL(3)=10HSTATE NO 3
      LABEL(4)=10H XDOT(3)
      GO TO 999
300   CONTINUE
C
C THIS SECTION COMPUTES THE XDOT VECTOR GIVEN N, T, AND THE X-VECTOR.
C
C CALCULATE AN INTERMEDIATE VARIABLE WHICH IS A FUNCTION OF THE STATES.
      Z=-C*X(3)+X(1)**2-X(2)**2-D
      CALL MINMAX(1,10H XDOT(3)   ,Z)
      IF(Z.GT.15.0)CALL STOP
C CALCULATE THE TIME DERIVATIVES OF THE STATE VARIABLES.
      XDOT(1)=-0.5*X(1)
      XDOT(2)=-A*X(2)
      XDOT(3)=Z
      GO TO 999
400   CONTINUE
C
C THE USER SPECIFIES THE VARIABLES THAT WILL BE OUTPUTTED IN THIS
C SECTION----THE OUTPUT VARIABLES ARE PLACED IN THE Y-VECTOR; THE
C Y VECTOR IS OF LENGTH M, WHERE M IS SPECIFIED IN THE INPUT
C DECK SSIN.
C
      Y(1)=X(1)
      Y(2)=X(2)
      Y(3)=X(3)
      Y(4)=Z
      CALL XYPLOT(1,4,2)
      GO TO 999
500   CONTINUE
C
C THIS SECTION IS PROVIDED FOR POST PROCESSING OF THE FINAL TIME DATA.
C
C CALCULATE THE SUM OF THE THREE STATES AT THE FINAL TIME
      SUM=X(1)+X(2)+X(3)
      WRITE(NOUT,501)SUM
501   FORMAT(1H1,5/,6HSUM = ,E10.3)
      GO TO 999
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
999   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE XYPLOT(ID,IXX,IYY)
C
C SUBROUTINE XYPLOT ALLOWS THE USER TO X-Y PLOT ANY DATA IN THE Y
C OUTPUT ARRAY.  AS PRESENTLY DIMENSIONED, UP TO 10 X-Y PLOTS
C ARE ALLOWED.  THE INPUT VARIABLES HAVE THE FOLLOWING MEANING...
C       ID      IDENTIFICATION NUMBER (1 TO 10)
C       XAXIS  NUMBER OF ELEMENT IN Y ARRAY ONE WISHES PLOTTED ON X AXIS
C       YAXIS  NUMBER OF ELEMENT IN Y ARRAY ONE WISHES PLOTTED ON Y AXIS
C
C CAUTION:  THE TWO VARIABLES FOR WHICH ONE WISHES AN X-Y PLOT
C           MUST APPEAR IN THE Y OUTPUT ARRAY IN SUBROUTINE USER.
C
      COMMON/XYI/IX(10),IY(10)
C MMAX IS THE MAXIMUM NUMBER OF OUTPUTS FOR WHICH PROGRAM SS IS DIMENSIONED.
      DATA MMAX / 30 /
      IF((ID.LE.0).OR.(ID.GT.10))GO TO 999
      IF(IX(ID).NE.0)GO TO 999
      IF((IXX.LE.0).OR.(IYY.LE.0))GO TO 999
      IF((IXX.GT.MMAX).OR.(IYY.GT.MMAX))GO TO 999
      IX(ID)=IXX
      IY(ID)=IYY
999   CONTINUE
      RETURN
      END


      SUBROUTINE PLOTS(X,Y,TITLES,NUMPTS,NAME,NOUT,ISIZE)
C
C THIS SUBROUTINE GENERATES THE LINE PRINTER PLOTS FOR PROGRAM SS.
C
C DEFINITION:  A RELATIONSHIP IS A SET R OF ORDERED (X,Y) PAIRS.
C DEFINITION:  A FUNCTION IS A SET F OF ORDERED (X,Y) PAIRS WITH
C              THE PROPERTY THAT IF (X,Y1) AND (X,Y2) ARE CONTAINED
C              IN F, THEN Y1=Y2.
C THUS, FUNCTIONS ARE A PROPER SUBSET OF RELATIONSHIPS.  PROGRAM SS
C USES SUBROUTINE PLOTS TO PLOT BOTH FUNCTIONS (I.E. VARIABLE VERSUS
C TIME) AND RELATIONSHIPS (I.E. X-Y PLOTS).
C
C THE INPUT VARIABLES ARE...
C    X        X-AXIS ARRAY OF THE (X,Y) PAIRS
C    Y        Y-AXIS ARRAY OF THE (X,Y) PAIRS
C    TITLES   ARRAY USED TO STORE THE TITLE CARDS WHICH ARE PRINTED
C             AT THE TOP OF THE PLOT
C    NUMPTS   NUMBER OF (X,Y) PAIRS.  NUMPTS MUST BE GREATER THAN 1
C             AND LESS THAN OR EQUAL TO MAXPTS.
C    NAME     PLOT LABEL---MUST BE A10 OR 10H FORMAT.
C    NOUT     LOGICAL NUMBER OF OUTPUT TAPE UNIT
C    ISIZE    =0 MEANS FULL SIZE PLOTS (50X100); OTHERWISE, ONE
C             GETS THE REDUCED-SIZED PLOTS (30X40).
C
C CAUTION:  THE X AND Y ARRAYS WILL BE MODIFIED BY SUBROUTINE PLOTS.
C
C WHEN PLOTS ARE REQUESTED, THE PROGRAM RUNTIME WILL TYPICALLY
C BE DOMINATED BY THE TIME TO OUTPUT WRITE STATEMENTS 111,113,215,
C AND 235.  THEREFORE, IT IS IMPORTANT THAT THESE WRITE STATEMENTS
C BE AS EFFICIENT AS POSSIBLE; USE THE FORMS INDICATED BELOW (I.E.
C NOT AN IMPLIED DO).
C
C
```

APPENDIX C. WRITING TRANSFER FUNCTIONS AS FIRST-ORDER DIFFERENTIAL EQUATIONS

When a system is modeled, a portion of the total system is often described by a transfer function. This appendix describes how a transfer function in the normalized form can be directly converted to a set of first-order differential equations. In particular, the coefficients of the transfer function are used, without any algebraic manipulations, directly in the differential equations. The normalized form of the transfer function is:

$$G(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \ldots \ldots + a_1 s + a_o}{s^n + b_{n-1} s^{n-1} + \ldots \ldots + b_1 s + b_o} ,$$

where $m < n$ and b_n equals one. For physically realizable systems, it is required that $m \leq n$ (i.e., no feed forward of derivatives of the input). Transfer functions with $m = n$ can be put in the normalized form by expanding the transfer function into two parts: a feed-forward gain term and a transfer function with $m < n$ (see Example Two below for an illustration of this technique). The above normalized transfer function has the block diagram shown in Fig. C-1, where u is the input, y is the output, and it is assumed that $m = n - 1$.

For a transfer function with denominator of order n, n integrators are required. This will result in a set of n first-order, ordinary differential equations. These n equations can be written directly from this block diagram in terms of the a and b coefficients. The differential equations are:

$$\dot{x}_1 = x_2 ,$$

$$\dot{x}_2 = x_3 ,$$

$$\dot{x}_i = x_{i+1} \quad (i = 1, \ldots, n - 1) ,$$

$$\dot{x}_n = - \sum_{i=1}^{n} b_{i-1} x_i + u ,$$

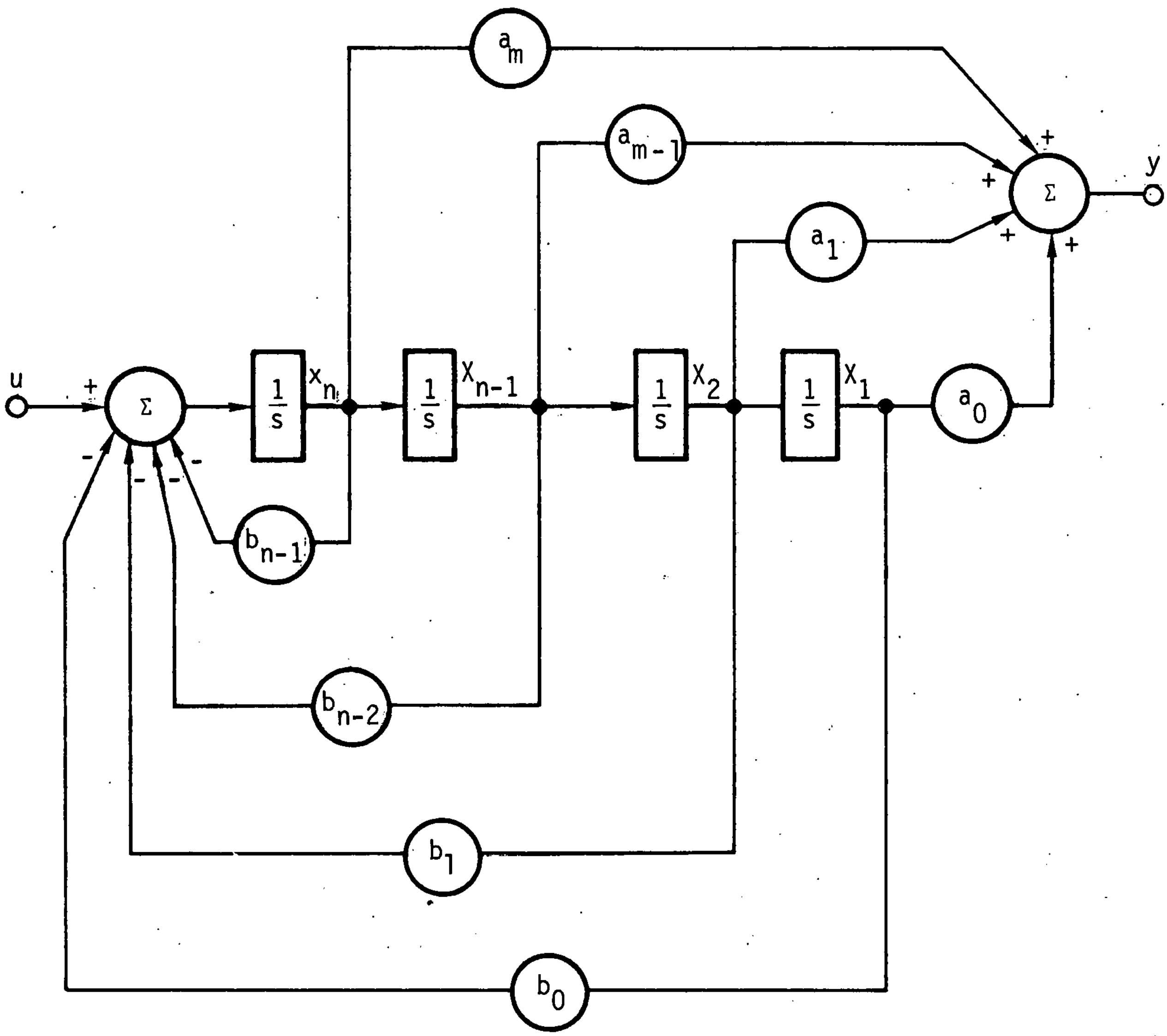

Fig. C-1. Block diagram of normalized transfer function with m = n - 1.

where, by definition, the initial values of the state variables are zero for transfer functions. Note that the constants in this set of differential equations directly use the coefficients of the normalized transfer function. The output of the transfer function, y, is given by:

$$y = \sum_{i=1}^{n} a_{i-1} x_i \ .$$

The above realization of G(s) and other possible forms can be found in a book by C. A. Desoer.[1]

Example One — Write the first-order ordinary differential equations for:

$$G(s) = \frac{10s + 2}{3s^2 + 9s + 6} \ .$$

Put this in the normalized form:

$$G(s) = \left(\frac{1}{3}\right) \frac{10s + 2}{s^2 + 3s + 2} \ .$$

The 1/3-gain term is handled as a separate gain in series with the normalized transfer function, as shown in Fig. C-2. One writes the differential equations directly from the normalized transfer function:

$$\dot{x}_1 = x_2 \ ,$$

$$\dot{x}_2 = -2x_1 - 3x_2 + u \ ,$$

where x_1 and x_2 have initial values of zero. The output is:

$$(2x_1 + 10x_2)/3 .$$

The forms are suitable for use in Program SS.

[1]Charles A. Desoer, *Notes for a Second Course on Linear Systems* (Van Nostrand Reinhold, New York, 1970), pp. 99-104.

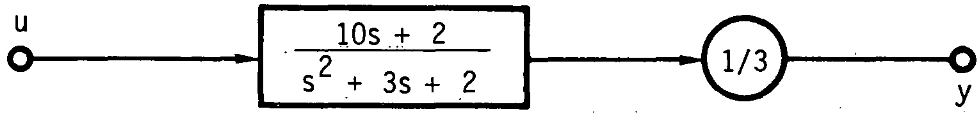

Fig. C-2. Block diagram for normalized transfer function of Example One.

Example Two — Write the differential equations for:

$$G(s) = \frac{12s^2 + 46s + 26}{3s^2 + 9s + 6}$$

Note that m = n. One must expand G(s) so that m = n - 1. If one divides the numerator by the denominator, one gets:

$$G(s) = \frac{12}{3} + \frac{C_1 s + C_2}{3s^2 + 9s + 6} ,$$

where C_1 and C_2 must be determined. Observe that the second term is the remainder after one division, and that the first term is the ratio of a_n and b_n coefficients. By placing the expanded G(s) over its common denominator and comparing the original numerator with this one, one can solve for C_1 and C_2. In this case they are 10 and 2, respectively, and the equation becomes

$$G(s) = 4 + \frac{10s + 2}{3s^2 + 9s + 6} .$$

Observe that this second term is the same transfer function as in Example One. The differential equations and output equations are:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -2x - 3x_2 + u ,$$

where x_1 and x_2 have initial values of zero and

$$y = \frac{1}{3}(2x_1 + 10x_2) + 4u .$$

The 4u term in y accounts for the direct feed-through of the input to the output. The block diagram for this transfer function is shown in Fig. C-3.

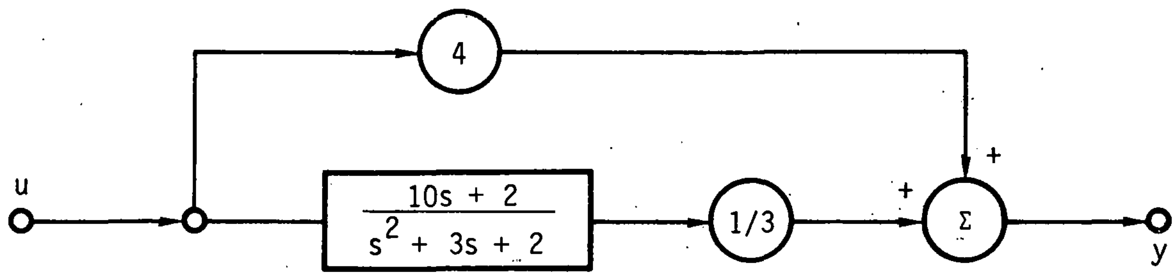

Fig. C-3. Block diagram for transfer function of Example Two.

FW/11/vt/mla

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $2.25

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages: add $4.50 for each additional 100 page increment over 1,000 pages.

Technical Information Department
LAWRENCE LIVERMORE LABORATORY
University of California | Livermore, California | 94550