IMPROVEMENTS OF USER CONTROL IN

QUASI

A COMPUTER PROGRAM

by

KENNIE E. GARLINGTON, JR.

DEPARTMENT OF COMPUTER SCIENCE

Submitted in Partial Fulfillment of the Requirements of
the University Undergraduate Fellows Program

1983-1984

Approved by:

Dr. Bart Childs

April 1984

ABSTRACT

Improvements of User Control in

QUASI

A Computer Program (April 1984)

Kennie E. Garlington, Jr.

Faculty Advisor: Dr. Bart Childs

In early April of 1984, a project to add computer graphics to the QUASI computer simulation system was completed and implemented at Texas A & M University. QUASI is a system that can be used in several engineering fields to compute various aspects of a mathematical model of a real-world process, then compare the model with data taken from observation of the process. This program provides quick access to a comprehensive analysis of a design, and can be used to great advantage early in the design process. It was decided to add graphics to the system to provide a better format for the representation of large groups of data. The system was implemented using the Data General Dasher G300 terminal as a display unit. The advantages of the system are its ability to alert the user to errors in the model, provide fast, flexible methods for comparing models, and in general make the system more "user-friendly." The major conclusion of this project is that the enhancements greatly increase the user's productivity and interest in the system, and it is recommended that QUASI, with the graphics

enhancements, be considered by industry to be a standard tool for design and testing work.

ACKNOWLEDGEMENTS

I would like to thank Dr. Childs, for all the time (which he could ill-afford) that he spent patiently explaining the mechanics and the theory of QUASI. I would also like to acknolwledge the help provided by the workers at the Eagle lab at Texas A & M University, who suffered through constant questions and requests for manuals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

The QUASI computer modelling system was developed by Dr. Bart Childs and others at the University of Houston in the late 1960's. The system has excellent computing power and flexibility, and can be used to solve problems in several areas of product design. The system has undergone several revisions, and currently a organized review and update of the entire system is in effect. One of the initial by-products of this review was a goal to make graphics available to improve user access to the system and control of its operation.

As a result, I was asked to implement a graphics subsystem as a part of the QUASI program. This was done as a one-year project under the auspices of the University Undergraduate Fellows Program.

This project was divided into two parts:

1. Determine the feasibility of using various devices as part of the system.

2. Implement the best of these alternatives.

Final testing for the new graphics modules was completed on April 10, 1984, and the documentation was finished at approximately the same time. This document describes the QUASI system, the design criteria used for the enhancements, the process of building the

---

This document follows the style of *Communications of the ACM* system, and methods of use of the graphics subsystem.

The Graphics Subsystem

The QUASI System: How it works

QUASI is a computer simulation system that performs a mathematical analysis of collected field data to estimate a requested parameter value. To put it simply, the user constructs a mathematical "model" (set of equations) that describes some process in which he is interested. He will supply these equations to QUASI, along with a set of data collected during field tests on the process. At this point, QUASI does two things.

First, QUASI will take the set of equations from the user, and an initial estimate of the final value, and perform a numerical integration using the Runge-Kutta or stepwise techniques. The result will, hopefully, be a better approximation of the model solution. Taking the results of the first integration and feeding them back into the integration routine will continue to increase the accuracy of the solution to some user-specified bound.

Next, the system compares the solution with the field data to determine if the model was a succesful predictor of some aspects of the process's performance. If there was not a reasonably good fit, QUASI will alert the user to a possible error in his model. The user then can refine his model to increase its power.

QUASI provides the engineer with an invaluable tool for quick estimation of critical design parameters as well. This can be used to its full potential early in the design stage to single out the most promising approach to a design problem from among several alternatives.

Alternatives to Printed Output

One disadvantage of the original QUASI package was the method in which it presented its results. First, the production of large amounts of computer printouts, involving complex, dense lists of numbers and statistical summaries that often proved to be difficult to use by the inexperienced user, made engineers avoid the use of QUASI unless they could not solve their problem any other way. Also, the printed output approach made quick comparisons of differing models difficult. The users found it impossible to compare two lists of numbers and come up with a general description of the models' differences.

To overcome this problem, the decision was made to add graphics to the system, facilitating easier comparisons of models and allowing quick subjective decisions concerning the accuracy and overall "look" of the process's behavior. Three types of plotting devices were considered in implementing this new system (see also Table 1 for a summary of each unit's attributes):

1. The use of the traditional paper line printer to produce a rough graph of the results.

2. The use of a Versatec hardcopy plotter to produce paper line drawings.

3. The Data General G300 graphics terminal to produce single-screen graphs.

The Line Printer

The line printer was already being used to some extent in the old system to produce simple graphs. Although this device would be the most available of any of the devices considered, it was decided that the printer was just too limited to produce the complex graph types necessary under the proposed subsystem. Problems with this device include:

1. Its inability to be used as part of an interactive system. All plots using the line printer would be produced at the end of the job, and the user could not stop the system before its conclusion since he could not see the graphics data.

2. The resolution of the printer. At only 55 X 100 elements per page, little in the way of detailed information could be presented.

3. The low speed available. Especially with multiple users on the same printer, fifteen minute wait times for output are not uncommon.

The Versatec Plotter

The Versatec line plotter has several advantages over the other two systems. It can produce graphs of much higher resolution than the line printer, and could provide a permanent record of the data, unlike the graphics terminal. Unfortunately, the Versatec is not a high-speed plotter. This, along with the remote location of the device and the fact that there is only one such plotter attached to the Texas A & M computer, ruled out use of this device as a primary medium. However, there is a chance that a Versatec connection will be added in the near future to give the user a choice of systems.

The Data General G300 Terminal

The Data General Dasher G300 graphics terminal was the device of choice for the implementation of the plotting subsystem, for five reasons:

1. There were three of these units attached to the main computer, easily accessable by the user. This meant that testing of the system could be readily carried out, and that the use of the graphing routine would not be hampered by unavailable equipment.

2. The resolution of the terminal was good, with a screen of 240 X 640 pixels.

3. The unit was easily programmed, since a graphics command interpreter was downloaded into the device, making English-like drawing commands possible.

4. The G300 has a response time well within any user's requirements. In fact, when the system is lightly loaded, an entire graph can be drawn in less than a second.

5. Since the system can request guidance from the user at the same

terminal where the plots are being drawn, interaction with the user is highly supported.

Therefore, the graphics subsystem was implemented with the Dasher unit as the medium for the graphs.

Table 1. Comparison of Characteristics of Primary Plotting Media
The following table provides a summary of the major points considered in selecting the Data General Dasher G300 terminal over use of the line printer or Versatec line plotter.

	Interactive?	High Resolution?	High Speed?	Many Available?
Printer	No	No	No	Yes
Versatec	No	Yes	No	No
Terminal	Yes	Yes	Yes	Yes

Programming Methodology

The QUASI system is designed as a set of related program segments, written in the Data General FORTRAN 77 lanuuguage. For this project, it was necessary to make modifications to four routines as well as write a new routine, G300, which does the actual plotting. See Appendix A for a listing of the code segments.

Modifications to module QUASI

QUASI is the main-line routine which coordinates the execution of all other routines. The largest of all the system modules, this program actually required the least amount of modification. Primarily, a new variable array (IPLOT2) was added to contain the user's choices of state variables to be plotted, and two temporary files were created from this routine which were used to hold previous values of the plotting routines for the iterative plots (UNIT20 and UNIT21). The reason why the changes were so small is due to the original design of the system, which had the necessary code segments to support the addition of a graphics unit. However, in the conversion from the IBM architecture on which it was originally developed to the Data General MV/8000, the plot segments were not converted. Thus, the major part of the time spent on this routine was on the conversion process. Of course, the parameter lists for the other routines modified were changed as well.

Modifications to module ININT

The ININT routine reads in the integer parameters. Two of those parameters (see Figure 1) are used to select plotting. The IPLOT2 variable described above needed to be set in this routine, and so code was added to read in those values.

Modifications to module BUILD

The BUILD routine was set up for use with a line printer routine. The modifications to be made here were the inclusion of a title to be sent to the graphics routine, and the necessary file synchronization to store each plot matrix for later use. Also, since this routine is the one which produces the iterative plots, the maximum iterative value (XMAX) was passed to BUILD to define the size of the graph. Also, a segment of code to number the points by iteration cycle was added.

Modifications to module PLOT

The modifications to PLOT (which produces the final plot) were almost identical. The titles were, of course, different. Also, the XMAX variable was not needed, so it was not included in this routine.

Module G300 Description

G300 is a new module that does the conversion of a matrix of X versus Y values to positions on a Data General Dasher G300 terminal, and outputs them to the terminal (see Appendix A) with supporting titles, labels, etc. Briefly, the module algorithm is as follows:

- Compute maximum and minimum X and Y values in the input matrix.
- Compute units/pixel for conversion to pixel number on screen.

- Print a message to the user to allow him time to see printed output before the plot begins. An undocumented option at this point is to respond with the number '1' to the message "Enter (New Line) when ready to see plot..." This will create a dump of important program parameters, useful when debugging the module after modifications.

- Set the terminal to graph mode, set up the eight line styles (so that different state variable lines can be distinguished), and print the top headings. Part of the set-up procedure is to place the terminal in "mnemonic mode," which allows English-like commands to be sent to the terminal [2].

- Draw lines for each state variable in sequence.

- Print X label and legend for lines.

- Put terminal back into text mode and print a similar message to the opening print above to give the user time to see the graph.

Plotting Subsystem Characteristics

The user, before executing QUASI, must first provide two lines of input data describing what action the system should take regarding graphics output (see Figure 1). The integer parameter "19" selects plotting to occur after every iteration. The user can select a plotting device (currently the old line printer package or the

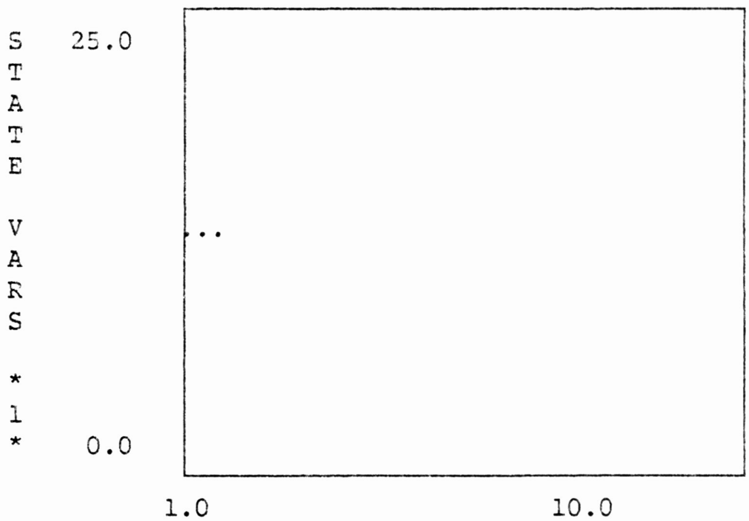
Dasher unit) and which elements of the system (1-8) he wants to see. The line labelled "21" provides similar information about plots to be generated at th end of the computations.

An example of how the graph might look on the Data General terminal is shown in Figure 2. Each variable plotted is identified by a unique line type (dotted line, dashed line, etc.) which is listed on the "Legend" line. For iterative plots, the subsystem will continually re-draw previous values to give the user a point of reference for determining model errors. All X-axis and Y-axis scaling is automatic, and all values are plotted (no variable can exceed the limits of the graph scale). The user is given the option at the end of each plot of pressing "Enter" to continue processing, or he can interrupt the program from the console if an error is indicated.

```
   19    2 ITER PLOT FROM TERM     1    2    3
   21    2 FINL PLOT FROM TERM     1    2    3
```

These lines would be entered to select plots from QUASI.
In this example, the Data General graphics terminal (device "2") was chosen to plot variables 1, 2, and 3.

Figure 1. Input Example For Plotting Subsystem



PLOT OF ITERATIVE VALUES

Line Legend: Y1 = ...

Enter (New Line) to continue...

This plot might be drawn on the first iteration of a ten-iteration cycle, plotting only 1 variable during the cycle.

Figure 2. Sample Plot from QUASI

Conclusion

QUASI is a computer package designed to aid engineers in developing and testing mathematical models of processes of interest. It provides quick, reliable, and comprehensive answers to the major items of interest concerning the model. However, it suffered in the past from a tendency to print too much information in tabular form, decreasing the ability of the user to make value judgements concerning the results. Graphics were added to the system to overcome this disadvatage.

The graphics subsystem, implemented for use with a Data General Dasher G300 graphics terminal, provides timely, accurate summaries of the important information resident within QUASI. It allows better interaction with the engineer, giving him a better look at the data and allowing him to spot more errors in his model earlier in the process.

With the addition of the graphics package, the user has a great deal of flexibility in choosing how much information he wants to see plotted. He can select plotting to occur at the end of each iteration step, at the end of all iterations (the final solution), both times, or neither time (no plotting). Up to eight different system parameters can be selected for display. The user can, upon detecting a solution that is not reasonable, stop the system at any point and start over. He can also obtain a table of the data to be plotted immediately before the plot is performed in case more accurate values are required.

Overall, the system in practice has demonstrated the following characteristics:

1. Total response time of slightly less than one second under optimal conditions. Included in this time is the interval needed to format the data, clear the screen, and produce the new graph.

2. A greater emphasis on involving the user in the computational process, by giving him more information faster. Thus, the user feels more confidence in the final results.

3. As mentioned before, more flexibility in operation than under the old package.

This enhancement of QUASI, providing powerful, easy-to-use graphics to the program user, has resulted in a system that is more "user-friendly," more accurate, less costly when errors do occur, and better able to express qualitative points about a model. The costs involved are practically non-existant, and the additional system load is so negligible, that no reasonable complaints can be made about the new system as compared to the old one. Therefore, the use of this system, along with the overall QUASI package, is highly recommended to engineers involved with design or testing of new processes and products.

REFERENCES

1. Childs, Bart and H. R. Porter. QUASI - A System Identification Code. *Lecture Notes in Computer Science*: *Codes for Boundary-Value Problems in Ordinary Differential Equations* 76 (May 14-17, 1978) 186-195

2. Data General Corporation. *Dasher G300 Display Terminal Users Manual*. Rev. 0, April 1981

APPENDIX A

CODE LISTINGS

```
C
C*********************************************************************
C****************************               ****************************
C****************************    QUASI      ****************************
C****************************               ****************************
C*********************************************************************
C
C      THE STYLE OF PROGRAM-WRITING INCORPORATED INTO THIS SOURCE CODE
C      IS BASED ON THE ARTICLE 'FORTRAN POISONING AND ANTIDOTES' BY
C      BRIAN T. SMITH, FROM PP. 178-256 OF VOLUME 57 OF LECTURE NOTES
C      IN COMPUTER SCIENCE, 1977.
C
C-----FORM CONTAINS THE ARRAY FRMT, WHICH IS USED FOR RUN-TIME
C      FORMATTING.
       CHARACTER FRMT*80, IAF*4(2)
       COMMON/FORM/  FRMT,  IAF
C
C-----IBLK CONTAINS THE INTEGER INPUT VARIABLES
       INTEGER  IOUT, LIN, NDDE, LITER,NORMRD, NBC, NEQ, NCN,
     X          NWRITE, IVSAVE, ITRACE, IEXTRA, IXTPL, IDIF, NCQBC,
     X          IPLT, ISETNL, IXXX, NGRAD, MOUT, ICOP, ISTAT,
     X          IDOIT, NEXT6, NEXT7
       COMMON/IBLK/  IOUT,     LIN,      NDDE,     LITER,    NORMRD,
     X               NBC,      NEQ,      NCN,      NWRITE,   IVSAVE,
     X               ITRACE,   IEXTRA,   IXTPL,    IDIF,     NCQBC,
     X               IPLT,     ISETNL,   IXXX,     NGRAD,    MOUT,
     X               ICOP,     ISTAT,    IDOIT,    NEXT6,    NEXT7
C
C-----JBLK CONTAINS INTEGER IDENTIFIERS TO BE PASSED TO CERTAIN
C      SUBROUTINES.
       INTEGER  NVECT, IXTRA, JXTRA, KXTRA, NOUT, NL, NSPACE,
     X          NCSD, NRSD, NRYD, I, II, ICOUNT, IN1, IN2, INTKEY,
     X          IPUNT, IQ, IS, ISUB, ITEMP, ITER, J, JJ, JQ, JSUB,
     X          K, KK, KOUNT, KOUT, KQ, KTEM, IDONE, NBCM, NCOL, NRANKC,
     X          NROWS, ISHTT
       COMMON/JBLK/  NVECT, IXTRA, JXTRA, KXTRA, NOUT, NL,
     X               NSPACE, NCSD, NRSD, NRYD, I, II, ICOUNT, IN1,
     X               IN2, INTKEY, IPUNT, IQ, IS, ISUB, ITEMP, ITER,
     X               J, JJ, JQ, JSUB, K, KK, KOUNT, KOUT, KQ, KTEM,
     X               IDONE, NBCM, NCOL, NRANKC, NROWS, ISHTT
C
C-----KBLK CONTAINS INTEGER ARRAYS RELATED TO INITIAL CONDITIONS
C      AND BOUNDARY CONDITIONS
       INTEGER  XACTIV(20), IPTR(20), QBC(201), XACTBC(201)
       COMMON/KBLK/  XACTIV, IPTR, QBC, XACTBC
C
C-----FBLK CONTAINS FLOATING POINT IDENTIFIERS TO BE PASSED TO
C      CERTAIN SUBROUTINES
       REAL  ZERO, DZERO, SMALL, DPTRB, DETT, RN1, RN2, RN3, H, T,
     X       TAVG, TL, TN, DN1, TBMT, TEM, TEMP, TTEM, SUMBV,
     X       DTEMP, TRACE
       COMMON/FBLK/  ZERO, DZERO, SMALL, DPTRB, DETT, RN1, RN2, RN3, H,
     X               T, TAVG, TL, TN, DN1, TBMT, TEM, TEMP, TTEM, SUMBV,
     X               DTEMP, TRACE
C
C-----RBLK CONTAINS THE REAL INPUT VARIABLES
       REAL  DELT, SPTRB, PNORM, TSTART, TSTOP, DET, CONV, WEIGHT,
     X       ALPHA, TOUT, AUX(2000)
       COMMON/RBLK/  DELT,     SPTRB,    PNORM,    TSTART,   TSTOP,
     X               DET,      CONV,     WEIGHT,   ALPHA,    TOUT,
     X               AUX
C
C-----REG CONTAINS VARIABLES USED IN SUBROUTINE STANAL
       REAL  PB(20), YCALC(20), SAVE(202,22)
       COMMON/REG/   PB, YCALC, SAVE
C
```

```
C-----RK4P CONTAINS INFORMATION USED IN NUMERICAL INTEGRATION
      REAL  STO(20,3), JACOB(20,20,3), SDY(20), YYK(20), YYP(20)
      COMMON/RK4P/  STO, JACOB, SDY, YYK, YYP
C
C-----SBLK CONTAINS THE SA = D MATRIX, S, AND BOUNDARY VALUE
C     AND BOUNDARY CONDITION INFORMATION
      REAL  S(202,22), BV(201), TBC(202), CQBC(20)
      COMMON/SBLK/  S, BV, TBC, CQBC
C
C-----YBLK CONTAINS INFORMATION ABOUT Y
      REAL  IIV(20), IV(20), Y(20,21), YRK4(80), PIV(20), PTRB(20),
     X      UPPER(20), LOWER(20), YPTRB(20), DY(20)
      COMMON/YBLK/  IIV, IV, Y, YRK4, PIV, PTRB, UPPER, LOWER,
     X              YPTRB, DY
C
      INTEGER  IAP2,  ICONV, IENDFL, INDX, IP, IPEXCT, IPLSTQ,
     X         ISWITC, ISWIT2, NAUX, NCSH, NEQOUT, NMEXCT, NMLSTQ,
     X         NR, NRW
C
      REAL  ANORM, SNORM, SVNEXT, TBCK, TNORM, TOUTIM
C
      INTEGER  IBIG(25), IPLOT(50,8), IPLOT2(8)
C
      REAL  DIV(50), GRID(50), RBIG(10)
C
        CHARACTER  RTITLE(10,20)*8
C
C     THE ARRAY IBIG(25) IS EQUIVALENCED TO THE VARIABLES IN THE
C     COMMON BLOCK, IBLK, TO MAKE THE INPUT OF THESE VALUES EASIER.
C     SIMILARLY, THE FIRST 10 VARIABLES IN THE COMMON BLOCK, RBLK,
C     ARE EQUIVALENCED TO RBIG(10) FOR EASE OF INPUT.
      EQUIVALENCE (IOUT,IBIG(1)),(DELT,RBIG(1))
C
      REAL ABS
C
C     AMONG THE IDENTIFIERS INITIALIZED BELOW IS ISWITC, WHICH
C     IS USUALLY SET TO ZERO; IF TIMESHARING IS DESIRED, SET IT TO 1.
      CHARACTER*4 IAST/'****'/
      ISWITC = 0
      IAP2 = 25
      IENDFL = 0
C     File 20 is used to hold succesive iterations
C     File 21 holds the variable values at the end of a cycle
      OPEN (20,FILE='QUASI.UNIT20.LS',FORM='UNFORMATTED',STATUS='FRESH')
      OPEN (21,FILE='QUASI.UNIT21.LS',FORM='UNFORMATTED',STATUS='FRESH')
C
C-----------------------------------------------------------------
C     ****  THIS IS THE START OF THE 'OUTER' LOOP  ****
C-----------------------------------------------------------------
C     AFTER ALL INTEGRATION IS DONE FOR THE CURRENT SET OF DATA,
C     AND IDONE = 1, RETURN HERE FOR THE NEXT SET.  PRINT OUT THE
C     PROGRAM'S NAME.
   10 CONTINUE
        WRITE(12,20)
   20   FORMAT('1 PARTICULAR SOLUTION PERTURBATION METHOD SYSTEM',
     X    ' VERSION 3.1 INCLUDES STAT ANALYSIS 4/1/82')
C
C       INPUT THE TITLE.
        CALL INTITL( IAST,IAP2,ISWITC,RTITLE,INDX,IENDFL )
C
C       IENDFL = 1 MEANS THERE IS NO MORE DATA TO READ, SO END THE
C       PROGRAM'S EXECUTION.
        IF( IENDFL .NE. 1 )  THEN
C
C       INPUT THE INTEGER PARAMETERS.
        CALL ININT( IOUT,IBIG,IP,IPLOT,IPLOT2,GRID,CORG,IENDFL )
```

```
C
      IF( IENDFL .NE. 1 )  THEN
C
C       INPUT THE REAL PARAMETERS.
        CALL INREAL( IOUT,NAUX,RBIG,AUX )
C
C       INPUT THE BOUNDARY CONDITION INFORMATION.
        CALL INBOUN( IOUT,NEQ,TBC,BV,QBC,XACTBC )
C
C       INPUT INITIAL VALUE INFORMATION.
        CALL INIV( IOUT,ZERO,PIV,IIV,PTRB,XACTIV,UPPER,LOWER,SPTRB )
C
C       IF THERE ARE NO BOUNDARY CONDITIONS, THERE ARE NOT ANY
C       TO SORT OR TEST, SO ADVANCE TO STATEMENT 30.
        IF( NBC .GT. 0 )  THEN
C
C         SORT THE BOUNDARY CONDITION INFORMATION IN ORDER OF INCREASING
C         TBC VALUES.
          CALL SORTBC( NBC,TBC,BV,QBC,XACTBC,TSTOP,TSTART )
C
C         IF THE PROBLEM BEING SOLVED IS NONLINEAR AND NUMERICAL
C         DIFFERENTIATION IS NOT DESIRED, CHECK TO BE SURE THE USER
C         HAS REMEMBERED TO PROGRAM THE LINEARIZED VERSION OF THE NON-
C         LINEAR PROBLEM.  IF THE USER HAS NOT DONE THIS, NUMERICAL
C         DIFFERENTIATION IS FORCED.
          IF( LIN .EQ. 0 .AND. NDDE .NE. 0 )
     X      CALL CKPROG( NEQ,TBC,IAST,NDDE )
C-------------------------------------------------------------------
C       INITIALIZE IV AND OTHER VARIABLES
C-------------------------------------------------------------------
C
C       THE FOLLOWING VARIABLES ARE INITIALIZED IN THIS SECTION OF CODE:
C
C       NVECT:    = NEQ + NCN, WHERE NEQ IS THE NUMBER OF EQUATIONS
C                 IN THE SYSTEM WITH NONTRIVIAL RIGHT HAND SIDES AND
C                 NCN IS THE NUMBER OF EQUATIONS WITH TRIVIAL RIGHT
C                 HAND SIDES.  BOTH WERE INTEGER INPUTS.
C       IV:       INITIAL VALUES VECTOR; INITIALIZED TO IIV.
C       XACTIV:   INITIALIZED IN SUBROUTINE INIV--DOUBLE
C                 CHECKED HERE
C       TNORM:    INITIALIZED TO 1 HERE; TNORM IS THE NORM FOR THE MOST
C                 RECENT CHANGES IN IV.
C       SNORM:    INITIALIZED TO PNORM, A REAL INPUT GIVING THE MAXIMUM
C                 LENGTH OF CHANGE IN IV ALLOWED; IF ZERO, UNRESTRICTED
C                 CHANGE IS ALLOWED; SNORM IS USED IN SUBROUTINE MODIFY.
C       ITER:     ITERATION STEP COUNTER; INITIALIZED TO ZERO
C       IDONE:    INITIALIZED TO ZERO; A FLAG WHICH, IF SET TO 1,
C                 INDICATES THAT THE LAST ITERATION IS BEING PERFORMED.
C
C       IF TIME-SHARING IS BEING USED, IOUT, ISWITC AND IAP2 ARE
C       ASSIGNED NEW VALUES AND ISWIT2 IS SET TO ZERO.
C
        END IF
C       IF LITER = 0 OR 1, THIS IMPLIES THAT THE PROBLEM IS A SIMPLE
C       INITIAL VALUES ONE; FOR PROGRAMMING EFFICIENCY, DON'T ALLOW
C       NEEDLESS NUMERICAL DIFFERENTIATION.
        IF( LITER .LE. 1 )  NDDE = 1
        NVECT = NEQ + NCN
C
        DO 40 I = 1,NVECT
          IV(I) = IIV(I)
C         CHECK FOR INPUT ERRORS FOR XACTIV.
          IF( NBC .LE. 0 .AND. XACTIV(I) .GE. 0 )  XACTIV(I) = -1
C         THIS IS TO NEGATE THE 4 POSSIBLY SUBTRACTED IN THE MAIN LOOP.
          IF( XACTIV(I) .LT. -1 )  XACTIV(I) = XACTIV(I) + 4
   40   CONTINUE
```

```
C
        TNORM = 1.
        SNORM = PNORM
        ITER = 0
        IDONE = 0
C
C       IF TIME-SHARING IS USED SET, RE-SET, OR CHECK THE FOLLOWING
C       VARIABLES.
        IF( IOUT .GE. 10 )  THEN
          IOUT = IOUT - 10
          ISWITC = 1
          IAP2 = 10
C         ISWIT2 IS A FLAG WHICH LIMITS OUTPUT ON THE TERMINAL IN
C         SUBROUTINE TIMESH.
          ISWIT2 = 0
          END IF
C-----------------------------------------------------------------
C
C       ECHO THE INPUT DATA, PERFORM AN ERROR CHECK, CALCULATE NMLSTQ,
C       THE NUMBER OF INEXACT BOUNDARY CONDITIONS, AND NMEXCT, THE
C       NUMBER OF EXACT BOUNDARY CONDITIONS PLUS 1, AND INITIALIZE
C       YCALC AND PB, ARRAYS USED IN SUBROUTINE STANAL.
        CALL ECHO( ISWITC,NMEXCT,NMLSTQ,NVECT,NAUX )
C
C-----------------------------------------------------------------
C       ****  THIS IS THE START OF THE 'MIDDLE' LOOP  ****
C-----------------------------------------------------------------
          DO 315 ITER = 1,LITER
C         IF ISWITC = 1, INDICATING TIMESHARING IS DESIRED, CALL
C         SUBROUTINE TIMESH.
          IF( ISWITC .EQ. 1 )
     X    CALL TIMESH(IENDFL,ISWIT2,NVECT,IV,PTRB,DELT,TSTOP,LITER,IOUT)
C
C         IENDFL = 1 MEANS THE USER HAS INDICATED IN SUBROUTINE TIMESH
C         THAT EXECUTION SHOULD BE ENDED.
          IF( IENDFL .NE. 1 )  THEN
C
C         BUMP THE ITERATION COUNTER
C         NROWS, NRW, SUMBV, AND NEQOUT ARE INITIALIZED:
C         NROWS:   NUMBER OF ROWS IN THE S MATRIX
C         NRW:     VARIABLE USED IN OTHER SUBROUTINES--COUNTS THE
C                  NUMBER OF ROWS OF INEXACT BOUNDARY CONDITIONS THAT
C                  HAVE BEEN PLACED IN MATRIX S.
C         SUMBV:   SUM OF THE BOUNDARY VALUE DISSATISFACTIONS
C                  (SUM OF THE DIFFERENCES BETWEEN THE CALCULATED
C                  BOUNDARY VALUES AND THE REAL ONES)
C         NEQOUT:  VARIABLE USED IN CONJUNCTION WITH MOUT TO DETERMINE
C                  WHICH STATE VECTOR ELEMENTS ARE TO BE PRINTED OUT
C                  AT CERTAIN TIMES
          NROWS = NBC + 1
          NRW = 0
          SUMBV = 0.
          NEQOUT = NEQ
          IF( MOUT .GT. 0 )  NEQOUT = MOUT
C
C         PDOIT IS A SUBROUTINE DESIGNED FOR THE PROFICIENT USER OF
C         THIS PROGRAM (SEE USER'S MANUAL). IDOIT IS AN INTEGER INPUT
C         WHICH, WHEN NONZERO, INDICATES PDOIT IS TO BE CALLED.
          IF( IDOIT .NE. 0 )  CALL PDOIT( SNORM )
C
C         IF PDOIT WAS JUST CALLED OR THIS IS THE FIRST ITERATION,
C         CALL SUBROUTINE INIT WHICH DOES THE FOLLOWING:
C         IF THERE ARE ANY BOUNDARY CONDITIONS THAT SPECIFY INITIAL
C         VALUES, INIT TRANSFERS THE BOUNDARY CONDITION INFORMATION
C         TO IV AND XACTIV, CHANGES XACTBC TO INDICATE THE CHANGES,
C         AND THEN DECREMENTS NROWS, THE NUMBER OF ROWS IN S.
```

```
C          KOUNT, DIV, PTRB, IPTR, NRANKC, AND NCOL ARE INITIALIZED,
C          AND TWO ERROR CHECKS ARE PERFORMED ON THE SYSTEM.
           IF( IDOIT .NE. 0 .OR. ITER .EQ. 1 )
     X       CALL INIT( DIV,NMEXCT,NMLSTQ,ISWIT2 )
C
C          WRITE A LINE OF ASTERICKS.
           WRITE(12,70)  (IAST,I=1,IAP2)
   70      FORMAT(/,5X,25A4,/)
C
C          FILL MATRIX Y AND ARRAYS YPTRB, YCALC, AND PB.
           CALL FILLY
C
C-----------------------------------------------------------------
C          SET UP VARIABLES FOR INTEGRATION
C-----------------------------------------------------------------
C
C          KOUT IS A FLAG THAT, IF SET TO 1, INDICATES THAT A LOT OF
C          INTEGRATION INFORMATION IS TO BE OUTPUT.  IT IS SET TO ZERO
C          UNLESS IOUT (AN INTEGER INPUT CONTROLLING OUTPUT) IS EQUAL
C          TO 3 OR 4 OR UNLESS THIS IS THE LAST ITERATION.
C          IF IOUT = 4, WRITE OUT A MESSAGE.
           KOUT = 0
           IF( IOUT .GT. 2 .OR. IDONE .EQ. 1 )  KOUT = 1
           IF( IOUT .GE. 4 )  WRITE(12,80)
   80      FORMAT(1X/10X,'REPEATED TIMES ARE PARTICULAR SOLUTIONS'/)
C          TWO POINTERS USED IN BUILDING THE S MATRIX AND THE FIRST ROW
C          OF THE S MATRIX ARE INITIALIZED.  IPEXCT POINTS TO THE ROW
C          IMMEDIATELY BEFORE THE NEXT ROW IN MATRIX S TO BE FILLED IN
C          WITH AN EXACT BOUNDARY CONDITION, AND IPLSTQ POINTS TO THE ROW
C          IMMEDIATELY BEFORE THE NEXT ROW IN S TO BE FILLED IN WITH AN
C          INEXACT BOUNDARY CONDITION.  THE FIRST ROW OF S IS SET TO 1'S.
           IPEXCT = 1
           IPLSTQ = NMEXCT
C
           DO 90 I = 1,NCOL
             S(1,I) = 1.0
   90      CONTINUE
C
C          TL IS INITIALIZED TO TSTART; TL IS THE LAST VALUE OF THE
C             INDEPENDENT VARIABLE THAT WAS INTEGRATED TO.
C          TOUT IS THE INTERVAL OF THE INDEPENDENT VARIABLE THAT
C               SEPARATES INTEGRATION INFORMATION OUTPUT.
C          TOUTIM IS THE NEXT VALUE OF THE INDEPENDENT VARIABLE
C                 AT WHICH INTEGRATION INFORMATION OUTPUT IS DESIRED.
C                 IF THE FIRST BOUNDARY CONDITION DOES NOT OCCUR AT
C                 TSTART (I.E. TSTART < TBC(1)), SET TOUTIM TO TSTART
C                 SO THAT INTEGRATION OUTPUT WILL OCCUR AT THE START
C                 OF THE 'NORMAL INTEGRATION ROUTE'.  IF THE FIRST
C                 BOUNDARY CONDITION DOES OCCUR AT TSTART (I.E. TSTART =
C                 TBC(1)), SET TOUTIM TO TSTART + TOUT SO THAT, AFTER ONE
C                 FLOW THROUGH THE 'NORMAL INTEGRATION ROUTE', THE
C                 'BOUNDARY CONDITION ROUTE', BEFORE WHICH INTEGRATION
C                 OUTPUT WILL OCCUR, WILL BE TAKEN.
           TL = TSTART
           TOUTIM = TSTART
           IF( IDONE .EQ. 1 )  WRITE(12,100)
  100      FORMAT(1H1)
C
C          IF THIS IS THE FINAL ITERATION, A TITLE EXISTS, AND
C          ISWITC = 0, WRITE THE TITLE.
           IF( IDONE .EQ. 1 .AND. INDX .GT. 0 .AND. ISWITC .EQ. 0 )
     X       CALL TITLE( RTITLE,INDX,IAST,IAP2 )
C
C          WRITE OUT THE ITERATION NUMBER.
           WRITE(12,110)  ITER,LITER
  110      FORMAT(20X,'ITER',I4,'  OF',I4,/,10X,'TIME',14X,'SOLUTION')
```

```
C          INTKEY IS A FLAG WHICH INDICATES WHETHER A BOUNDARY CONDITION
C                 HAS BEEN REACHED OR NOT.
C          IN1 IS USED IN SUBROUTINE RKFOUR.
C               IF THIS IS THE LAST ITERATION, SET IN1 = 1; OTHERWISE
C               SET IN1 = NRANKC.
C          K POINTS TO THE NEXT BOUNDARY CONDITION TO BE REACHED.
C          TBC(K) POINTS TO THE VALUE OF THE INDEPENDENT VARIABLE AT
C                 THE NEXT BOUNDARY CONDITION TO BE REACHED.
           INTKEY = 0
           IN1 = NRANKC
           IF( IDONE .EQ. 1 )  IN1 = 1
           K = 1
           TBCK = TBC(K)
C-----------------------------------------------------------------------
C          ****  THIS IS THE START OF THE 'INNER' LOOP  ****
C-----------------------------------------------------------------------
C
C///       BRANCHES TO THIS POINT CAN BE MADE FROM THE END OF EITHER
C          THE 'NORMAL INTEGRATION ROUTE' OR THE 'BOUNDARY CONDITION
C          ROUTE'.
  120      CONTINUE
C
C          CHECK FOR THE END OF INTEGRATION FOR THIS LOOP.
           IF( TL .LE. TSTOP )  THEN
C
C          TEST FOR THE 'NORMAL INTEGRATION ROUTE' OR THE 'BOUNDARY
C          CONDITION ROUTE' BY TESTING WHETHER INTKEY = 0 OR 1,
C          RESPECTIVELY.
           IF( INTKEY .NE. 1 )  THEN
C
C-----------------------------------------------------------------------
C          ****  NORMAL INTEGRATION ROUTE  ****
C-----------------------------------------------------------------------
C
C            IF TL >= TOUTIM AND EITHER KOUT = 1 OR WE'RE AT AN
C            INITIAL CONDITION, THEN CALL SUBROUTINE INTOUT, FOR
C            INTEGRATION INFORMATION OUTPUT, AND UPDATE TOUTIM.
C            THE INITIAL CONDITION CONDITION IS INCLUDED IN CASE
C            TSTART < TBC(1), IN WHICH CASE THE PRINTING OF THIS
C            INFORMATION IS DESIRED ON THE FIRST RUN THROUGH THIS CODE.
             IF( TL .GE. TOUTIM .AND. ( KOUT .NE. 0 .OR.
     X         ABS(TL-TSTART) .LE. ZERO ) )  THEN
               CALL INTOUT( NEQOUT,ZERO,TSTART,TL )
               TOUTIM = TL + TOUT
               END IF
C
C            TL IS THE LAST VALUE OF THE INDEPENDENT VARIABLE TO
C            WHICH INTEGRATION HAS TAKEN PLACE.  LET H BE EQUAL TO
C            DELT, A REAL INPUT, UNLESS THE NEXT BOUNDARY CONDITION,
C            AT TBCK, IS WITHIN THE NEXT INTERVAL OF LENGTH DELT,
C            IN WHICH CASE SET H TO BE TBCK - TL.  IN THIS LATTER
C            CASE, ALSO SET INTKEY = 1 TO FLAG THE BOUNDARY CONDITION
C            AND SET SVNEXT TO TL + DELT, SO THAT, AT A LATER
C            INTEGRATION STEP, THE ORIGINAL INDEPENDENT VARIABLE
C            GRID MAY BE RECOVERED.
             IF((TBCK .GT. TL+DELT) .OR. (NBC .EQ. 0 ))  THEN
               H = DELT
               ELSE
               H = TBCK - TL
               INTKEY = 1
               SVNEXT = TL + DELT
                END IF
C
C            SET T TO BE TL + H.  PERFORM RUNGA-KUTTA INTEGRATION
C            ON Y FOR THE INTERVAL TL TO T.  UPDATE TL.
             T = TL + H
```

```
      CALL RKFOUR
      TL = TL + H
C
      GO TO 120
C
C-----------------------------------------------------------------
C    ****  BOUNDARY CONDITION ROUTE  ****
C-----------------------------------------------------------------
C
C     IF KOUT = 1 OUTPUT INTEGRATION INFORMATION AND BUMP
C     TOUTIM IF NECESSARY.
      END IF
      IF( KOUT .NE. 0 )  THEN
        CALL INTOUT( NEQOUT,ZERO,TSTART,TL )
        IF( TL .GE. TOUTIM )  TOUTIM = TL + TOUT
        END IF
C     ASSIGN IQ THE QBC VALUE FOR THE KTH BOUNDARY CONDITION FOR
C     USE BY SUBROUTINE ROWOFS IN COMPUTING THE NEXT ROW OF S
C     TO BE FILLED.
C     IF THE BOUNDARY CONDITION IS AN EXACT ONE, BUMP IPEXCT
C     & ASSIGN THIS BUMPED VALUE TO NR, THE POINTER TO THE
C     NEXT ROW OF S TO BE FILLED.  IF THE BOUNDARY CONDITION
C     IS INEXACT, BUMP IPLSTQ AND ASSIGN THIS VALUE TO NR,
C     REMEMBERING TO BUMP NRW, A COUNTER OF THE NUMBER OF
C     INEXACT BOUNDARY CONDITIONS FOR USE IN OTHER SUBROUTINES.
 170  IQ = QBC(K)
C
      IF( XACTBC(K) .NE. 0 )  THEN
        IPLSTQ = IPLSTQ + 1
        NRW = NRW + 1
        NR = IPLSTQ
        ELSE
        IPEXCT = IPEXCT + 1
        NR = IPEXCT
      END IF
C
C     FILL IN THE NR'TH ROW OF S WITH THE K'TH BOUNDARY
C     CONDITION INFORMATION.
      CALL ROWOFS( NR,TBCK )
C
C     BUMP K.  CALCULATE SUMBV AND PRINT IT IF IT IS COMPLETED.
      K = K + 1
      IF( K-1 .LE. NBC )
     X  SUMBV = SUMBV + ABS( S(NR,1) - S(NR,NCOL) )
      IF( K .GT. NBC )  WRITE(12,200)  SUMBV
  200 FORMAT(///,' SUM OF BV DISSATISFACTIONS IS',G15.7,///)
C
C     IF ALL THE BOUNDARY CONDITIONS HAVE BEEN CALCULATED AND THIS
C     IS NOT THE FINAL ITERATION, DON'T INTEGRATE OUT TO
C     TSTOP > TBC(NBC), BUT GO PREPARE FOR THE NEXT ITERATION.
      IF( K .LE. NBC .OR. IDONE .NE. 0 )  THEN
C
C     IF K > NBC AND THIS IS THE FINAL ITERATION, SET TBCK
C     TO A FAKE VALUE OF TSTOP + 2 * DELT SO THAT INTEGRATION
C     WILL CONTINUE UNTIL TSTOP IS REACHED.  OTHERWISE, SET
C     TBCK TO BE TBC(K).
      TBCK = TSTOP + 2 * DELT
      IF( K .LE. NBC )  TBCK = TBC(K)
C
C     IF WE'RE AT A MULTIPLE BOUNDARY CONDITION, GO TO 170 TO
C     BUILD ANOTHER ROW OF S.
      IF( ABS( TBCK-TBC(K-1) ) .LE. ZERO )  GO TO 170
C
C     IF TBCK, THE VALUE OF THE INDEPENDENT VARIABLE AT THE
C     NEXT BOUNDARY CONDITION, IS ALSO IN THE SAME INTERVAL
C     OF LENGTH DELT, I.E., TBCK IS LESS THAN SVNEXT, SET H
```

```
C             TO BE THE DIFFERENCE BETWEEN TBCK AND TL, AND LET INTKEY
C             REMAIN AS 1, INDICATING WE'RE STILL AT A BOUNDARY CONDITION.
C             IF TBCK IS NOT <= SVNEXT, LET H EQUAL SVNEXT - TL, SO AS
C             TO GET THE INDEPENDENT VARIABLE BACK ON THE ORIGINAL
C             GRID, AND SET INTKEY TO ZERO, INDICATING WE'RE NOT
C             AT A BOUNDARY CONDITION.
              IF( TBCK .GT. SVNEXT )  THEN
                H = SVNEXT - TL
                INTKEY = 0
                ELSE
                H = TBCK - TL
                END IF
C
C             INCREMENT T, INTEGRATE FROM TL TO T AND UPDATE TL.
              T = TL + H
              CALL RKFOUR
              TL = TL + H
C
              GO TO 120
C
C-----------------------------------------------------------------
C           ****  THIS IS THE END OF THE 'INNER' LOOP  ****
C-----------------------------------------------------------------
C
C           FLOW COMES HERE AT THE END OF AN ITERATION, IF TL > TSTOP
C           OR IF K > NBC AND IDONE = 0.
             END IF
             END IF
C
C           BUILD DATA FOR A CONVERGENCE PLOT IF DESIRED.
            IF( ICOP .GT. 0 .AND. IDONE .EQ. 0)
     X        CALL BUILD( ICOP,NEQ,IPLOT2,IPLT)
C
C           CHECK FOR THE FINAL ITERATION.
            IF( IDONE .NE. 1 )  THEN
C           IF IOUT IS NONZERO, PRINT S AT THIS TIME.
            IF( IOUT .NE. 0 )  THEN
              WRITE(12,240)
  240         FORMAT(1X// 2X,'SAVE MATRIX'/
     X          5X,'BOUNDARY VALUE,UNPTRBD SOLN, PTRBD SOLNS.....')
C
              DO 260 I = 1,NROWS
                WRITE(12,250)  I,S(I,NCOL),(S(I,J),J=1,NRANKC)
  250           FORMAT(2X,I3,1P7G14.6/(5X,1P7G14.6))
  260         CONTINUE
C
C           SAVE S IN SAVE AND PASS DET TO GJRWLS IN DETT.  GJRWLS
C           SOLVES THE LINEAR SYSTEM REPRESENTED IN S; THE ANSWER
C           COEFFICIENTS ARE RETURNED IN THE LAST COLUMN OF S AND
C           THEN PRINTED.
          END IF
            DETT = DET
C
            DO 280 J = 1,NCOL
C
              DO 280 I = 1,NROWS
                SAVE(I,J) = S(I,J)
  280       CONTINUE
C
            CALL GJRWLS( S,NRSD,NCSD,NROWS,NRANKC,NMEXCT,Y,NCSD,
     X                   NCSD,DETT,ZERO,IOUT,IPUNT,0,PB,YCALC )
C
            WRITE(12,290)  (S(I,NCOL),I=1,NRANKC)
  290       FORMAT(1X/3X,'CONSTANTS  ',1P7G15.7/(14X,1P7G15.7))
C           KQ IS USED IN OTHER SUBROUTINES.
            KQ = NRANKC - 1
```

```
C
C          CALCULATE THE NEW IV VALUES
           CALL NEWIV( ANORM,TNORM,SNORM,DIV )
C
C          CHECK FOR CONVERGENCE.  ICONV = 1 INDICATES CONVERGENCE.
           ICONV = 1
           DO 300 I = 2,NRANKC
             IF( ABS( S(I,NCOL) ) .GE. CONV )  ICONV = 0
  300      CONTINUE
           IF( ICONV .NE. 0 )  THEN
C            IF CONVERGENCE HAS OCCURRED, CALL STANAL IF ISTAT > 0,
C            SET IDONE TO ONE TO FLAG THE FINAL ITERATION AND RETURN
C            TO 60 FOR THE FINAL ITERATION.
C            NCSH IS NEEDED BY SUBROUTINE STANAL
             NCSH = NCSD / 2 + 1
             IF( ISTAT .GT. 0 )
     X         CALL STANAL( SAVE,NRSD,NCSD,NROWS,NRANKC,NMEXCT,YCALC,
     X         NRYD,PB,DETT,ZERO,IOUT,S(1,1),S(1,NCSH),ALPHA )
             IDONE = 1
C            IF CONVERGENCE HAS NOT OCCURRED, CHECK FOR WHETHER THE
C            LITER'TH SOLUTION HAS BEEN REACHED.  IF IT HAS NOT,
C            RETURN TO 60 FOR THE NEXT ITERATION.  IF IT HAS BEEN
C            REACHED, CALL STANAL IF ISTAT < 0, SET IDONE TO 1 TO
C            FLAG THE FINAL ITERATION AND RETURN TO 60 FOR THE
C            FINAL ITERATION.
             ELSE IF (ITER .GT. LITER) THEN
               NCSH = NCSD / 2 + 1
               IF( ISTAT .LT. 0 )
     X           CALL STANAL( SAVE,NRSD,NCSD,NROWS,NRANKC,NMEXCT,YCALC,
     X           NRYD,PB,DETT,ZERO,IOUT,S(1,1),S(1,NCSH),ALPHA )
               IDONE = 1
               END IF
             ELSE
             GO TO 325
             END IF
             ELSE
             GO TO 335
             END IF
 315         CONTINUE
C-------------------------------------------------------------------
C          ****  THIS IS THE END OF THE 'MIDDLE' LOOP  ****
C-------------------------------------------------------------------
C
C        IF THE LAST ITERATION WAS JUST DONE, CALL SUBROUTINE PLOT
C        IF IPLOT > 0, PRINT THE TITLE IF IT EXISTS, AND LOOP BACK
C        TO STATEMENT 10 TO PROCESS THE NEXT SET OF DATA OR END
C        EXECUTION.
 325     IF( IPLT .GT. 0 .AND. IDONE .GT. 0 )
     X      CALL PLOT( NEQOUT,IPLOT,IP,IPLT,GRID )
         IF( INDX .GT. 0 )  CALL TITLE( RTITLE,INDX,IAST,IAP2 )
         GO TO 10
C-------------------------------------------------------------------
C        ****  THIS IS THE END OF THE 'OUTER' LOOP  ****
C-------------------------------------------------------------------
C
 335     END IF
         END IF
      STOP
      END
```

```
C-----------------------------------------------------------------
C****  SUBROUTINE ININT  ****
C-----------------------------------------------------------------
      SUBROUTINE ININT( IOUT,IBIG,IP,IPLOT,IPLOT2,GRID,CQBC,IENDFL )
C
      INTEGER  IENDFL, IOUT, IP, IP2
      INTEGER  IBIG(25), IPLOT(50,8), IPLOT2(8)
      REAL  CQBC(20), GRID(50)
C
C     THIS SUBROUTINE INPUTS THE INTEGER PARAMETERS NEEDED FOR
C     THIS PROGRAM.
C
C     INPUTS  ARE:  THERE ARE NONE.
C
C     OUTPUTS ARE:
C       IBIG:     THE COMMON BLOCK IBLK CONTAINS 25 INTEGER VARIABLES
C                 WHICH ARE EQUIVALENCED TO THE 25 ELEMENT ARRAY IBIG.
C                 THE DESCRIPTIONS FOR THESE VARIABLES ARE INCLUDED
C                 IN THE USER'S MANUAL.
C       IOUT:     THE FIRST ELEMENT OF IBIG, WHICH CONTROLS OUTPUT
C                 IN THE PROGRAM.
C       IP:       THE NUMBER OF ROWS IN IPLOT AND THE NUMBER OF ELEMENTS
C                 IN GRID.
C       IPLOT:    A MATRIX INDICATING WHICH ELEMENTS OF THE STATE
C                 VECTOR ARE TO BE PLOTTED.
C       IPLOT2:   A matrix incidating which elements of the state
C                 vector are to be plotted in a convergence plot
C       GRID:     AN ARRAY OF INTERVALS USED FOR EXPANDING/ CONTRACT-
C                 ING PLOTS.
C       CQBC:     AN ARRAY THE ELEMENTS OF WHICH ARE USED IN REPRE-
C                 SENTING A LINEAR COMBINATION OF THE STATE VECTOR
C                 ELEMENTS AS A BOUNDARY CONDITION.
C       IENDFL:   A FLAG WHICH, WHEN SET TO 1, INDICATES THE PROGRAM'S
C                 EXECUTION SHOULD CEASE.
C
C     IN GENERAL, IN1 INDEXES THE ELEMENT OF IBIG THAT IS BEING
C     INITIALIZED TO THE VALUE IN2.  IHOLER IS A CHARACTER STRING
C     OF COMMENTS.
C
C     IN1 = 0 INDICATES A BLANK CARD HAS BEEN READ, WHICH SIGNALS
C     THE END OF INTEGER INPUTS.
C
C     THE '(IPLOT(IP,IQ),IQ=1,8),GRID(IP)' PART OF THE READ STATEMENT
C     ONLY HAS MEANING IF IN1 = 16, IN WHICH CASE THE IPTH ROW OF
C     THE IPLOT MATRIX AND THE IPTH ELEMENT OF THE GRID ARRAY ARE
C     READ IN.
C
C     If IN1 = 21, then IPLOT2(IP,IQ) is used in place of IPLOT
C     to select convergence plot state vectors.
C
C     IF IN1 = 15, IN2 ELEMENTS OF ARRAY CQBC ARE READ FROM THE
C     IMMEDIATELY FOLLOWING DATA RECORDS, 10 VALUES TO A RECORD.
C
C     IBLK IS INITIALIZED TO 'DEFAULT' VALUES IN THE BLOCK DATA.
C
C     LOCAL VARIABLES:
      INTEGER  I, IN1, IN2, IP9, IQ, J
      INTEGER  IHOLER(5)
C
      IP = 1
   10 CONTINUE
      READ(9,20,END=70)  IN1,IN2,IHOLER,(IPLOT(IP,IQ),IQ=1,8),GRID(IP)
   20 FORMAT(2I5,5A4,8I5,G10.3)
C     BUMP IP IF PARTS OF IPLOT AND GRID WERE JUST INITIALIZED.
      IF( IN1 .EQ. 16 )  IP = IP + 1
C     Store vector in IPLOT2 if convergence plot requested
```

```
      IF ( IN1 .EQ. 21) THEN
         DO 25 IQ = 1,8
25          IPLOT2(IQ) = IPLOT(IP,IQ)
      ENDIF
C
C     IF IN1 = 0 A BLANK CARD HAS JUST BEEN READ, SIGNALLING THE
C     END OF INTEGER INPUT.  IF IOUT < 10 THEN PRINT IHOLER.
      IF( IN1 .EQ. 0 .AND. IOUT .LT. 10 )  WRITE(12,30)  IHOLER
   30 FORMAT(15X,'BLANK CARD',5X,5A4,//)
C
      IF( IN1 .GT. 0 )  THEN
C
C     IN1 > 0.  IF IOUT < 10, ECHO THE INTEGER INPUTS JUST READ.
C     INITIALIZE THE IN1TH ELEMENT OF IBIG.
      IF( IOUT .LT. 10 )  WRITE(12,40)  IN1,IN2,IHOLER
   40 FORMAT(' INPUT DATA',2I5,5X,5A4)
      IBIG(IN1) = IN2
C
C     IF IN1 = 15 AND IN2 > 0, READ THE VALUES INTO CQBC THAT ARE
C     TO REPRESENT A LINEAR COMBINATION OF Y ELEMENTS AS A
C     BOUNDARY CONDITION.
      IF( IN1 .NE. 15 .OR. IN2 .LE. 0 )  GO TO 10
C
        DO 60 I = 1,IN2,10
           IP9 = I + 9
           READ(9,50)  (CQBC(J),J=I,IP9)
   50      FORMAT(10G8.0)
   60   CONTINUE
C
      GO TO 10
   70 IENDFL = 1
      END IF
      RETURN
      END
```

```
      SUBROUTINE BUILD(IPT,NEQ,IPLOT2)
C ---------------------------------------------------------------------
C  THIS IS THE ROUTINE WHICH BUILDS THE MATRIX OF INITIAL VALUES
C  WHICH ARE PLOTTED  BY THE CONVERGENCE PLOT REQUEST
C   (INTEGER PARAMETER 21)
C---------------------------------------------------------------------
      DIMENSION TL(300,9),A(50),IPLOT2(8)
      CHARACTER*40 TITLE
      COMMON/JBLK/ID1(23),ITER,ID2(10),MITER
      COMMON/IBLK/IOUT,LIN,NDDE,LITER
      DATA TITLE/'ITERATION CYCLES WITHIN ITERATION LOOP   '/
      I = 1
      K = 1
      XMAX = LITER
      REWIND 20
C     Set matrix size
      DO 3 IQ=1,8
3        IF (IPLOT2(IQ) .NE. 0) I = I+1
C     Read input values and set in array
    5 READ(20,END=200)ID,T,(A(J),J=1,NEQ)
      TL(K,1)=T
      DO 6 IQ=1,8
         IND = IPLOT2(IQ)
    6    IF (IND .GT. 0) TL(K,IQ+1) = A(IND)
      K=K+1
      IF(K.LE.300.AND.K.LE.KK) GO TO 5
  200 KK=K-1
C     Set file for later appends by INTOUT
      BACKSPACE 20
      WRITE(12,310)IPT
  310 FORMAT(//,10X,'CONVERGENCE PLOT OF Y(',I2,')')
      IF (IPT .LE. 1) THEN
         CALL Y8VSX(TL,KK,I,0.0)
      ELSEIF (IPT. EQ. 2) THEN
         CALL G300(TL,KK,I,XMAX,TITLE)
      ELSEIF (IPT. EQ. 3) THEN
         CALL VPLT(TL,KK,I,XMAX,TITLE)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE PLOT(NEQ,IPLOT,IP,IPLT,GRID)
C-----------------------------------------------------------------------
C  THIS ROUTINE DOES ALL THE SET UP WORK FOR PLOTTING
C     CALLED AT END OF INTEGRATION
C-----------------------------------------------------------------------
      DIMENSION IDUM(300),TL(300,9),A(50),IPLOT(50,8),GRID(50)
      CHARACTER*40 TITLE
      DATA TITLE/'INDEPENDENT VARIABLE-END OF INTEGRATION '/
C     SET XMAX TO 0 FOR AUTO SCALING
      XMAX = 0.0
      IP1=IP-1
      REWIND 21
      DO 190 I=1,IP1
      DO 100 J = 1,300
      READ(21,END=140)IDUM(J),T,(A(K),K=1,NEQ)
      TL(J,1)=T
      DO 110 IJ=1,8
      IND=IPLOT(I,IJ)
      IF(IND.GT.0) THEN
      TL(J,IJ+1) = A(IND)
      ELSE
      GOTO 100
      END IF
 110  CONTINUE
 100  CONTINUE
      CALL ERRORS(-6)
  140 NDPS=J-1
C     Set file for later appends by INTOUT
      BACKSPACE 21
      N=IJ
      NN=N-1
      NGRID=0
      IF(IPLT.LE.1) THEN
         WRITE(6,150)(IPLOT(I,IK),IK=1,NN)
  150    FORMAT(///,25X,'++++++ T I M E (DOWN)  VS   Y ( ',8(I2,','))
         WRITE(6,160)
  160    FORMAT(1H+,85X,') ++++++')
         CALL Y8VSX(TL,NDPS,N,GRID(I))
      ELSEIF(IPLT.EQ.2) THEN
         CALL G300(TL,NDPS,N,XMAX,TITLE)
      ELSEIF(IPLT.EQ.3) THEN
         CALL VPLT(TL,NDPS,N,XMAX,TITLE)
      END IF
  190 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE G300(TL,NDPS,N,XMAX,TITLE)
C     This routine is used to send plotted output to the
C     DASHER G300 graphics terminal.
C
C     Input parameters are:
C
C     TL - A matrix of the following:
C          Column one is the independent variable values.
C          Columns 2-9 contain up to 300 dependent values.
C     NDPS - The number of rows (values) in TL
C     N - The number of columns (dependent vars+1) in TL
C     XMAX - If not zero, defines max indep. range (for iterations)
C     TITLE - Title to print on bottom
C
      REAL TL(300,9),XPDS,YPDS,XMIN,YMIN,XMAX,YMAX
      INTEGER XPST,YPST,XPEN,YPEN,XPNT,YPNT,DEBUG,CHT,CLEN
      CHARACTER*40 TITLE
C     Set up device-dependent parameters
C     XPST,XPEN - Start & end points for x points
C     YPST,YPEN - Start & end points for y points
C     CHT - Height of a text char in pixels
C     CLEN - Length of a text character in pixels
C     XOFF - offset from X axis in pixels
C     YOFF - offset from Y axis in pixels
C
      PARAMETER (XPST=100, XPEN=635, YPST=50, YPEN=438)
      PARAMETER (CHT=16,CLEN=16,XOFF=2,YOFF=4)
C
C     If XMAX not supplied (0) set XMAX to largest point. Compute XMIN.
      IF (XMAX .EQ. 0) THEN
         XMIN=TL(1,1)
         XMAX=TL(NDPS,1)
      ELSE
         XMIN = 1.0
         DO 5 I=1,NDPS
5           TL(I,1)=I
      ENDIF
C
C     Compute maximum and minimum Y values
      YMIN = 1.0E+75
      YMAX = -1.0E+75
      DO 10 I = 1,NDPS
         DO 10 J = 2,N
            IF (TL(I,J) .GT. YMAX) YMAX = TL(I,J)
10          IF (TL(I,J) .LT. YMIN) YMIN = TL(I,J)
C
C     Avoid scale duplication if YMAX = YMIN
      IF (YMAX .EQ. YMIN) THEN
         YMAX = YMIN + 1
      ENDIF
      IF (XMAX .EQ. XMIN) THEN
         XMAX = XMIN + 1
      ENDIF
C
C     Compute units/pixel for X & Y axes.
      XPDS = (XMAX-XMIN)/(XPEN-XPST)
      YPDS = (YMAX-YMIN)/(YPEN-YPST)
      IF (XPDS .EQ. 0) XPDS = 1
      IF (YPDS .EQ. 0) YPDS = 1
C     Allow user time to read screen before plot
      PRINT *,'Enter (New Line) when ready to see plot:'
      READ(10,2,END=20) DEBUG
C     If DEBUG requested, print TL matrix
      IF (DEBUG .GT. 0) THEN
         PRINT *,'***Debug requested***'
         PRINT *,'        TL matrix'
```

```
          PRINT *,'       ---------'
          PRINT *,'Indep.      State variable(s)'
          PRINT *,' var.'
          PRINT *,'------   ----------------------'
          PRINT *
          DO 15 I = 1,NDPS
15           PRINT *,(TL(I,J),J=1,N)
          PRINT *,'XMIN,XMAX,YMIN,YMAX = ',XMIN,XMAX,YMIN,YMAX
          PRINT *,'XPDS,YPDS = ',XPDS,YPDS
          PRINT *,'------------------------------'
          PRINT *,'Press (New Line) to continue...'
          READ(10,2,END=20) I
       ENDIF
C
C      Perform set-ups for putting to G300.
C      Enter mnemonic mode
20     PRINT *,'<36>G<42>1'
C      Erase screen
       PRINT *,'ERASE'
C      Set drawing color to green
       PRINT *,'COLOR 1'
C      Reset text attributes
       PRINT *,'TEXT RESET'
C      Put y-title
       PRINT *,'TEXT EXTENT -90 CELL 90'
       PRINT *,'TEXT ',0,' ',YPEN-CHT,' STATE VARIABLES | *',N-1,'*'
C      Put top and bottom points
       PRINT *,'TEXT RESET'
       PRINT *,'TEXT ',CLEN+2,' ',YPEN-CHT,' ',YMAX
       PRINT *,'TEXT ',CLEN+2,' ',YPST,' ',YMIN
C      Define the line styles for the eight line types
       PRINT *,'LSTYLE DEFINE 0 1000000010000000 1'
       PRINT *,'LSTYLE DEFINE 1 1000100010001000 1'
       PRINT *,'LSTYLE DEFINE 2 1111100011111000 1'
       PRINT *,'LSTYLE DEFINE 3 1010101010101010 1'
       PRINT *,'LSTYLE DEFINE 4 1111000011110000 1'
       PRINT *,'LSTYLE DEFINE 5 1110000011100000 1'
       PRINT *,'LSTYLE DEFINE 6 1100110011001100 1'
       PRINT *,'LSTYLE DEFINE 7 1100000011000000 1'
C      Draw chart axes
       PRINT *,'LINE ',XPST,' ',YPST,' ',XPST,' ',YPEN
       PRINT *,'LINE ',XPST,' ',YPST,' ',XPEN,' ',YPST
C      Draw N-1 lines.
       IF (DEBUG .NE. 0) THEN
          PRINT *,'---PLOT DEBUG: '
          PRINT *,'   X RANGE = ',XMIN,XMAX
          PRINT *,'   Y RANGE = ',YMIN,YMAX
          PRINT *,'   X START,INC = ',XPST,XPDS
          PRINT *,'   Y START,INC = ',YPST,YPDS
       ELSE
          CONTINUE
       ENDIF
       DO 200 J = 2,N
C         Plot initial point for this col
          XPNT = INT(XPST+(TL(1,1)-XMIN)/XPDS)+XOFF
          YPNT = INT(YPST+(TL(1,J)-YMIN)/YPDS)+YOFF
          PRINT *,'POINT ',XPNT,' ',YPNT
C         Plot rest of points if necessary
          IF (NDPS .GT. 1) THEN
             PRINT *,'LSTYLE ',J-2,' 1 0'
             DO 100 I = 2,NDPS
                IF (DEBUG .NE. 0) THEN
                   PRINT *,'   *** X=',TL(I,1),'   Y=',TL(I,J)
                ENDIF
                XPNT=INT(XPST+(TL(I,1)-XMIN)/XPDS)+XOFF
                YPNT=INT(YPST+(TL(I,J)-YMIN)/YPDS)+YOFF
```

```
               PRINT *,'LSLINE LAST ',XPNT,' ',YPNT
100         CONTINUE
         ELSE
            CONTINUE
         ENDIF
200   CONTINUE
C     Print x labels
      PRINT *,'TEXT ',XPST,' ',YPST-CHT,' ',XMIN
      PRINT *,'TEXT ',XPEN-2*CLEN,' ',YPST-CHT,' ',XMAX
C     Print line types
      PRINT *,'TEXT ',XPST,' ',YPST-2*CHT,' ',TITLE
      PRINT *,'TEXT 0 ',YPST-3*CHT,' LINE LEGEND: '
      PRINT *,'LAST TEXT'
      DO 300 I = 1,(N-1)
      PRINT *,'TEXT LAST " Y',I,' "'
      PRINT *,'LSTYLE ',I-1,' 1 0'
300   PRINT *,'LSLINE TEXT ',I*50+XPST,' ',YPST-3*CHT
C     Return to normal mode
      PRINT *,'<36>G<42>0'
C     Wait for user to see graph before continuing.
      PRINT *,'Graph finished...Press (New Line) to continue'
      READ (10,1,END=999,ERR=999) I
999   RETURN
C     Formats
1     FORMAT(A1)
2     FORMAT(I1)
      END
```