DCSIM

SIMULATION OF A CONTINUOUS SYSTEM CONTROLLED BY A DIGITAL CONTROLLER

by
Aslaug Haraldsdottir

Technical Report No. UM-MEAM-85-32

INTRODUCTION

This document describes how to use the simulation program package DCSIM. The program simulates the time response of a direct digital control system with a continous time plant. It is written in FORTRAN and set up to run under the DOMAIN operating system on an Apollo computer system. DCSIM is a version of the previously developed program SIMULA (1), adopted to run on the Apollo computers.

Figure 1 shows the structure of the control system that is simulated in DCSIM. The plant equations in general can be nonlinear and time varying but need to be expressed as a set of first order differential equations. The controller is described as a set of first order difference equations. The digital to analog (DAC) and analog to digital (ADC) converters have gain and saturation characeristics, as well as a sampling time for conversion. Disturbances can act on the plant and a reference input signal can be applied to the controller.

PROBLEM FORMULATION

DCSIM is a general purpose simulation program, so the user must supply the appropriate control and plant equations for a particular problem on a standard form. Also, the parameters of the DAC and ADC must be specified.

1. Continous plant state equations. These must be converted to the form

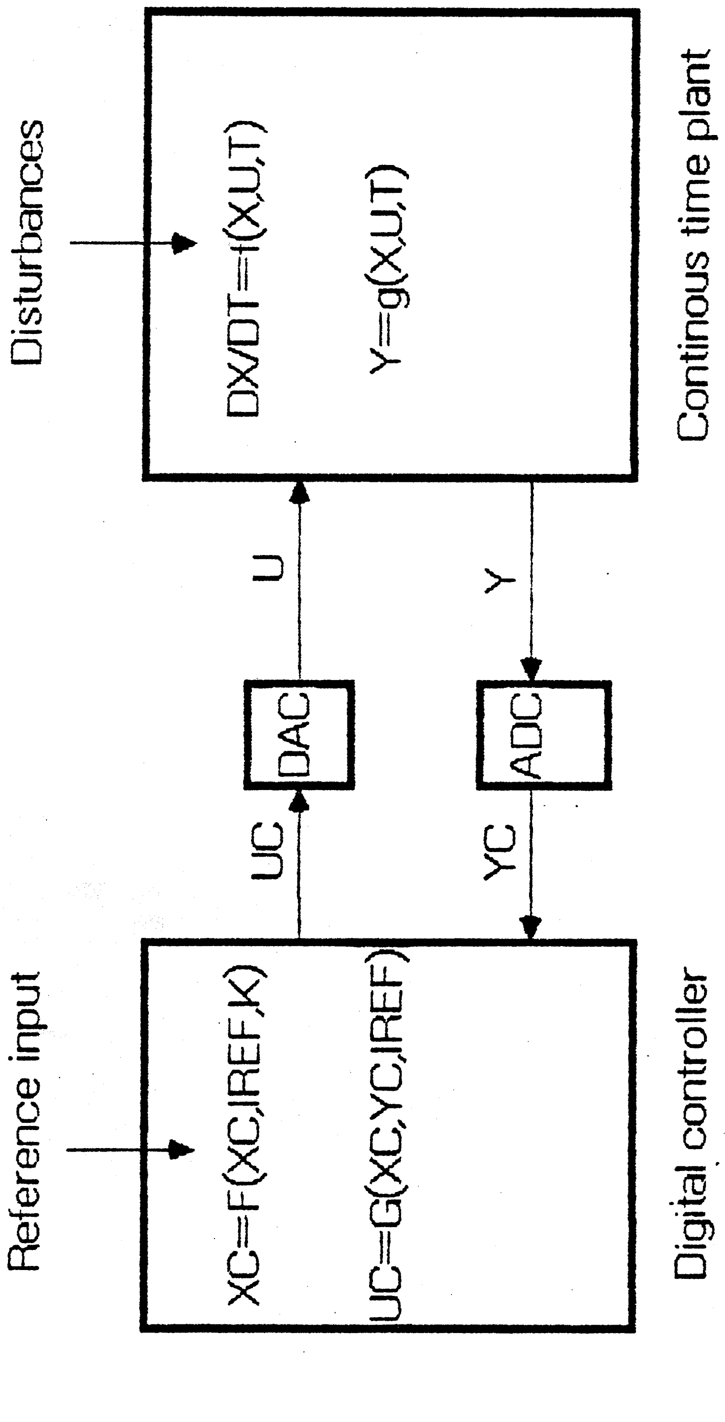
$$DX/DT = f(X,U,T)$$

$$Y = g(X,U,T)$$

X - plant state vector of dimension NEQN
U - manipulated plant input vector of dimension NIN
Y - plant output vector of dimension NOUT
f,g - nonlinear, time varying vector functions.

2. Controller state equations. These must be supplied in the form

$$IXC(k+1) = F(IXC,IYC,IREF,k)$$



Reference input
XC=F(XC,IREF,K)
UC=G(XC,YC,IREF)
Digital controller
UC
DAC
U
YC
ADC
Y
Disturbances
DX/DT=f(X,U,T)
Y=g(X,U,T)
Continous time plant

Figure 1

$$IUC(k) = G(IXC, IYC, IREF, k)$$

IXC	- controller state vector of dimension NCEQN
IYC	- converted value of plant output vector, dimension NOUT
IUC	- digital controller output, dimension NIN
IREF	- controller reference input vector, dimension NOUT
k	- sampling interval
F,G	- nonlinear, time varying vector functions.

The functions F and G can be evaluated using real arithmetic, but IXC, IYC, IUC and IREF have only integer values.

3. Digital to analog conversion. The control signal IUC calculated in the control algorithm is converted to a real value, then multiplied by a gain and sent to the plant after the sampling interval. The DAC also has a saturation limit.

4. Analog to digital conversion. The plant output Y is multiplied by a gain, then converted to an integer number and sent on to the controller. Saturation is also taken into account.

Both the DAC and the ADC operations have a gain and an offset accounted for. This must be taken into account when calculating reference inputs and controller outputs.

USER SUPPLIED ROUTINES

After the problem has been formulated as described in Chapter 2, the routines describing the control system must be modified. The file USER.FTN contains the simulation main program and three subroutines. Most of the necessary code is already there, the locations that need modification are marked with

```
C ********** Instructions **********
```

where 'Instructions' tell what changes to make.

1. Main program. The dimensions of U, X, XP, IXC, IYC, IUC, IREF and WORK need to be changed as explained in the instructions. This needs to be done **only in the main program**, not in any of the subroutines. The vector dimensions NIN, NOUT, NEQN, and NCEQN need to be defined also as explained in the instructions.

2. Subroutine F. Enter your plant equations here on the form explained above (XP=DX/DT).

3. Subroutine YEQUNS. Enter your plant output equations here (g).

4. Subroutine CONTRL. Enter your controller equations here. For example, the difference equations

$$xc_1(k+1) = 2xc_1(k) + 3$$

$$uc_1(k+1) = uc_1(k) + 2xc_1(k+1)$$

become

```
IXC(1) = 2*IXC(1) + 3

IUC(1) = IUC(1) + 2*IXC(1)
```

Note that the index k becomes unnecessary, the first state variable is calculated using its old value and the control output is calculated using the old output and the new state variable value.

RUNNING DCSIM

It will be assumed here that the user has had some exposure to the Apollo DOMAIN system. A beginners guide (2) is available at the workstations and it describes the most commonly used capabilities, such as editing a file, compiling, linking, running a program, using a floppy disk etc. After going through the beginners guide the user should have no trouble using DCSIM.

The first step is then to obtain your copy of the program package. The following files are needed and available at the CAEN Apollo network on North Campus and in the East Engineering building:

/progs/dcsim/user.ftn	:	user supplied routines
/progs/dcsim/dcsim.bin	:	object code of simulation program
/progs/dcsim/pltpac.bin	:	object code of graphics routines
/progs/dcsim/simpar	:	input file containing default simulation parameters.

After signing on and obtaining a process window and a $ prompt at the bottom of it, you can copy a file to your directory using the command CPF (copy-file), f.ex. type

```
CPF /progs/dcsim/user.ftn yourfl.ftn
```

to obtain the Fortran code of user.ftn in the file yourfl.ftn. Similarly, obtain copies of dcsim.bin, pltpac.bin and simpar.

Now, edit your version of user.ftn and make the necessary changes. When this is done, use the following command to run the Fortran compiler:

```
FTN yourfl.ftn -L
```

If there are errors reported to you, read the file yourfl.lst, created by the compiler, to locate the errors. Then edit yourfl.ftn to correct them. After your code compiles without any errors, use the following to link all routines together and store the run code in the object file RUN:

```
BIND yourfl.bin dcsim.bin pltpac.bin -binary run
```

You should get the message "All globals are resolved" and a $ prompt once again. Now you are ready to execute DCSIM, do that by typing the name of your run file:

```
RUN
```

and the simulation is under way.

First you will be asked to choose a name for a file that contains simulation parameters. These are constants having to do with integration time steps, length of simulation etc. and are explained later. The file SIMPAR contains the default values and you can use it to begin with.

Then you will be asked for a file to contain simulation results for the continous plant, which will be T, X, U, Y. Type in a name and hit return, the program will create the file if it doesn't exist or write over its contents if

3. **Max ADC**: The maximum digital value of the ADC. The minimum digital value is zero.

4. **DAC Gain**: The gain of the DAC. This is found by dividing the voltage range by the digital range. For example, if the voltage range is -5.12 volts to 5.12 volts and the digital range is 0 to 4095 (12 bit converter) then the gain is 10.24/4095= .0025 volts/count. This value should be consistent with Min DAC and Max DAC.

5. **DAC Offset**: The offset of the digital value. In the above example the offset is 2048. This is the ditial value that corresponds to a zero voltage.

6. **Min DAC**: This is the minimum voltage output. In our example this is -5.12.

7. **Max DAC**: This is the maximum voltage output. In our example this is 5.12.

The default parameter values displayed are contained in the file **SIMPAR** which is read by the program. You can either change them during the run, as just described, or edit SIMPAR before the run and make changes to the default values. **Make sure to observe the format in SIMPAR and enter your changes in accordance with that.**

The program now goes into the graphics selection mode. It will ask you if you want to plot the results. If you don't, enter NO and it asks you whether to repeat the whole simulation. If you don't want that either, enter NO and the execution is terminated. If you choose to repeat the simulation, you will be given a choice to use the final states as your new initial conditions.

If you enter YES or Y to the plotting prompt, you will be asked how many curves you want on your plot. You can superimpose up to 9 curves. Then you are asked to input the index of the independent variable, which is the x-axis on the plot. The variables are indexed in the following order, both for the plotting and printing to the output files:

T, X(1),X(2),...,X(NEQN),U(1),...,U(NIN),Y(1),...,Y(NOUT),

XC(1),...,XC(NCEQN),IUC(1),...,IUC(NIN),IYC(1),...,IYC(NOUT)

Make yourself a little table to index all your variables, e.g.

1	2	3	4	5	6	7	8	9	10
T	X(1)	X(2)	U(1)	Y(1)	IXC(1)	IXC(2)	IXC(3)	UC(1)	IYC(1)

for a second order plant, single input, single output and three controller states. If you want to plot X(1) AND IXC(1) v.s. time, you enter 1 for independent variable, 2 and 6 for the dependent variables. The program will only remember the variables in the order that you selected them: e.g. curve 1 is variable 2 and curve 2 is variable 6. So make a note of the order.

The program now asks whether you want to have all curves plotted using the same scale or each curve with its own scale. If the same scale is used for all curves, the maximum and minimum of all the data are found and those determine the scale. This may result in some of the curves being almost zero on the plot and therefore not visible. So choose this option in accordance with your data.

Now, the maximum and minimum data values are displayed and you are given a chance to change them. If you want the maximum to be a 'nice' value, you probably need to change these. Try out different things until you get a feel for how the graph is constructed. If the minimum values are negative, the graph will be made symmetric so that zero is midway between min and max. You need to make a note of the minimum and maximum values used, as no text appears on the plot.

Then you will be asked if you want a grid on the plot. If you don't want a grid, you get tick marks on the axes and a box around the graph. You will be asked for a filename to store the plot; the program will create a file if it doesn't find it in your directory, otherwise it writes over the contents of the existing file. The plot will now be displayed on the screen, hit return when you are done observing it. It will be available for copying to the printer in the file that you specified.

You can obtain hard copies of all results after you are done running the simulation by using the command PRF (print-file). To print the plot,

```
PRF filename -PLOT
```

and the specifier -PLOT will set the plot mode on the printer. For the other files, omit -PLOT and you get regular print.

The Fortran source code of DCSIM and PLTPAC are available for you to copy from /progs/dcsim/ just as with the object codes.

Take your time to get familiar with all the options of DCSIM, you will soon be able to use it to test your control system designs.

EXAMPLE

To clarify the descriptions in the preceding chapters, a simple example will be presented. This is the example that will be set up in the files user.ftn and simpar on /progs/dcsim.

Consider a continous time linear plant to be controlled described by

$$d^2x/dt + 10dx/dt + 34x = u(t)$$

Define the state variables

$$x_1 = x$$

$$x_2 = dx_1/dt$$

and obtain from the above differential equation a first order form

$$dx_1/dt = x_2$$

$$dx_2/dt = -34x_1 - 10x_2 + u(t)$$

Assume that there is one output, $y = x_1$. Apply proportional control to the plant so that

$$u(t) = (r(t) - y(t))k_p$$

where r(t) is the reference input to the closed loop system. Let the input be a sine wave

$$r(t) = \sin(3t)$$

This leads to the following formulation for DCSIM:

```
Subroutine F:  XP(1) = X(2)

               XP(2) = U(1) - 10.*X(2) - 34.*X(1)

Subroutine YEQUNS:  Y(1) = X(1)
Subroutine CONTRL:  IREF(1) = 2048 + 2048*SIN(3*T)
                    IE = IREF(1) - IYC(1)
                    IXC(1) = KP*IE
                    IXC(2) = IE
                    IXC(3) = IREF(1)
                    IUC(1) = IXC(1) + 2048

MAIN:  NEQN = 2
       NIN = 1
       NOUT = 1
       NCEQN = 3
```

Notice particularly the offset of 2048 added to IUC(1) and IREF(1), due to the operation of the DAC. The controller states are used here to enable printed and plotted output of the error, IE, and the reference signal, IREF(1).

Appendix A contains a listing of the file USER.FTN for this example. Figure 2 shows a sample output of the simulation program.

REFERENCES

1. Lauderbaugh, L.K., and A.G. Ulsoy, 'Simula - Digital Controller Simulation Package', Technical Report No. UM-MEAM-84-22, University of Michigan, Ann Arbor, Michigan.
2. 'Getting Started with your DOMAIN System', Order No. 002348, Revision 01, Software Release 7.0, Apollo Computer Inc., 1983.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the work of L.K.Lauderbaugh, whose simulation program SIMULA was the model for the version adopted for the Apollo computers. Valuable assistance was given by the staff of the Computer Aided

Engineering Network. Thanks also to A.G.Ulsoy for his guidance in developing the program. Financial support was provided by the Mechanical Engineering and Applied Mechanics Department and the College of Engineering at the University of Michigan.

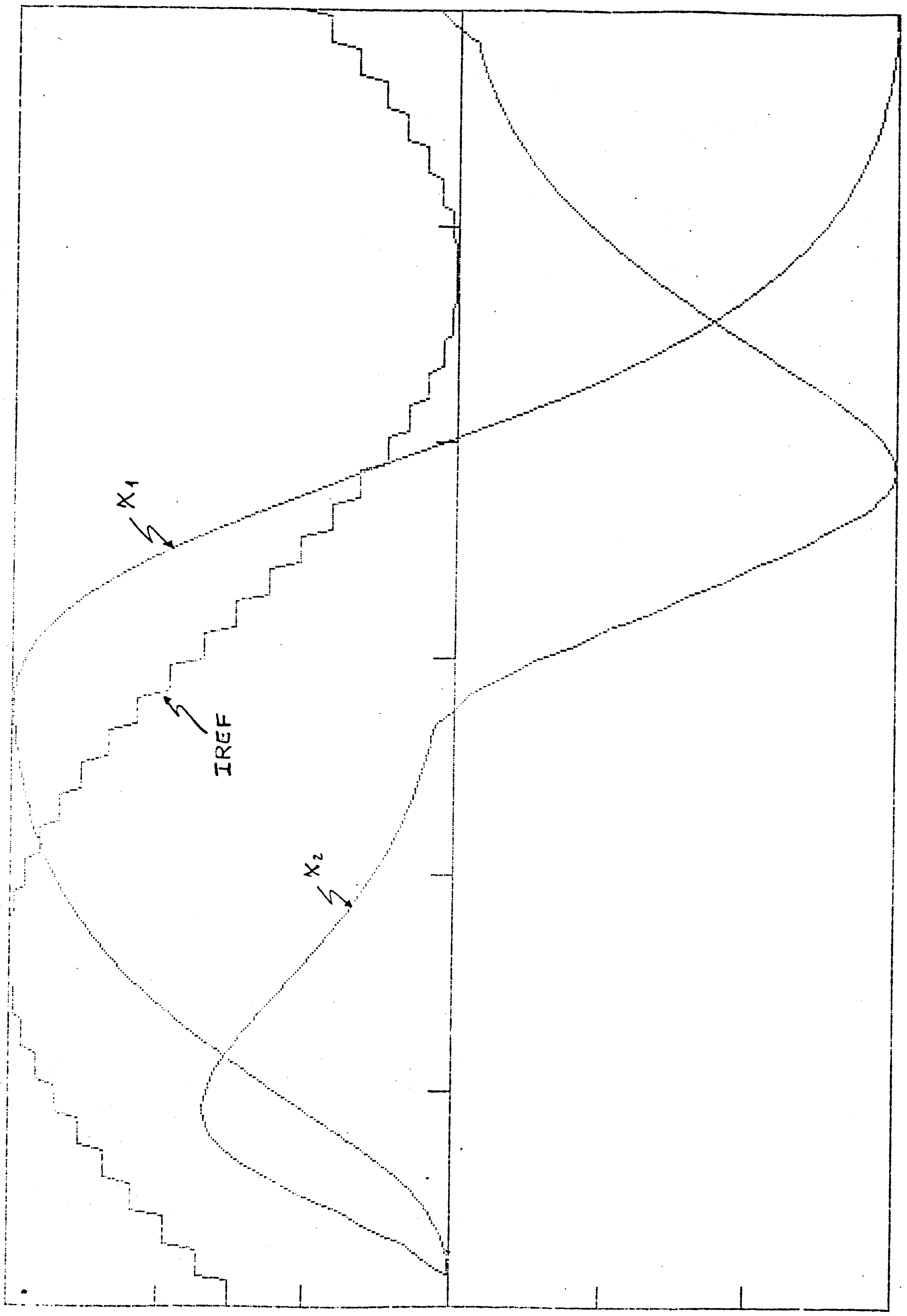


Figure 2.

APPENDIX A

```
C
C   General Description:  This is a driver main program for
C       the simulation program
C
C   External References:  SIMULA
C
C   Adapted to the Apollo computer system by Aslaug Haraldsdottir
C   from a program by L.K.Lauderbaugh
C
C   Declarations
C
      CHARACTER*1 ANSW,NO,YES
C
C********** Change dimensions here **********
C
C   THE DIMENSION OF X AND XP = NEQN
C   THE DIMENSION OF IXC = NCEQN
C   THE DIMENSION OF Y, IYC AND IREF = NOUT
C   THE DIMENSION OF U AND IUC = NIN
C   THE DIMENSION OF WORK = 5*NEQN
C
      REAL*4 U(1)
      DIMENSION X(2),XP(2),Y(1),IXC(3),IYC(1),IUC(1),IREF(1)
      DIMENSION WORK(10)
C
C   Common blocks
C
      COMMON /SIZE/ NEQN,NIN,NOUT,NCEQN
      COMMON /CLOCK/ H,TEND,TPRINT,DELT,DELDST,DELAY
      DATA NO,YES/'N','Y'/
C
C   Set up constants for simulation
C
C
C********** Change sizes here **********
C
C   NIN = # OF PLANT INPUTS
C
      NIN=1
C
C   NOUT = # OF PLANT OUTPUTS
C
      NOUT=1
C
C   NEQN = # OF PLANT STATE EQUATIONS
C
      NEQN=2
C
C   NCEQN = # OF CONTROLLER STATE EQUATIONS
C
      NCEQN=3
C********** Editing in main complete **********
C
C   Initialize variables and input simulation parameters
C
 50   CALL START(X,IXC,U,IUC,IREF)
C
C   Perform the simulation
C
 10   CALL SIMULA(X,XP,Y,U,IXC,IYC,IUC,IREF,WORK)
```

```
C
C   Check if user wants to plot results
C
 9980 PRINT *,'DO YOU WISH TO PLOT THE RESULTS? (Y/N)'
      READ (5,20)ANSW
      IF (ANSW.EQ.NO) GO TO 9990
      IF (ANSW.NE.YES) GO TO 9980
C
C   Plot the results
C
      CALL PLOTTER
      GO TO 9980
C
C   Check for a repeat
C
9990  PRINT *,'DO YOU WISH TO REPEAT THE SIMULATION? (Y/N)'
      READ (5,20)ANSW
 20   FORMAT(A1)
      IF(ANSW.EQ.YES) GO TO 40
      IF(ANSW.NE.NO) GO TO 9990
      CLOSE (UNIT=2)
      CLOSE (UNIT=3)
      CLOSE (UNIT=4)
      GO TO 30
C
 40   PRINT *,'DO YOU WISH TO START WITH THE LAST STATES AS YOUR NEW IC'S'
      READ (5,20) ANSW
      IF(ANSW.EQ.YES) GO TO 10
      IF(ANSW.NE.NO) GO TO 40
      CLOSE (UNIT=2)
      CLOSE (UNIT=3)
      CLOSE (UNIT=4)
      GO TO 50
 30   STOP
      END
C
C
      SUBROUTINE F(T,X,XP,U)
C
C********** Do not change these dimensions **********
C
      REAL*4 T,X(1),XP(1),U(1)
C
C********** Delete this set of plant equations and enter your's **********
C
      XP(1)=X(2)
      XP(2)=U(1) - 10.*X(2) - 34.*X(1)
C
      RETURN
      END
C
C
      SUBROUTINE YEQUNS (X,Y,U)
C
C********* Do not change these dimensions *********
C
      DIMENSION X(1),Y(1),U(1)
C
C********* Delete this set of equations and enter your's *********
C
```

```
      Y(1)=X(1)
C
      RETURN
      END
C
C
      SUBROUTINE CONTRL(T,IXC,IYC,IUC,IREF)
C
C********** Do not change these dimensions **********
C
      DIMENSION IXC(1),IYC(1),IUC(1),IREF(1)
C
C********** Delete this set of control equns. and enter your's ******
C
      KP=2
      IREF(1)=2048+2048*SIN(3*T)
C
C   Calculate the error
C
      IE=IREF(1)-IYC(1)
      IXC(1)=KP*IE
      IXC(2)=IE
      IXC(3)=IREF(1)
C
C   Offset
C
      IUC(1)=IXC(1)+2048
      RETURN
      END
C
```