Description of a simulation system DYSIM for continuous dynamic processes

Forskningscenter Risø, Roskilde

Publication date:
1981

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Cour Christensen, P. L. (1981). Description of a simulation system DYSIM for continuous dynamic processes. (Risø-M; No. 2271).

RISØ-M-2271

DESCRIPTION OF A SIMULATION SYSTEM DYSIM FOR CONTINUOUS DYNAMIC PROCESSES

P. la Cour Christensen

Abstract. A general purpose simulation system DYSIM for continuous dynamic processes has been worked out based upon five years experience with a similar system DYSYS from Kernforschungszentrum Karlsruhe. The new system has been made in order to improve the performance by excluding unused features and including new ones, and speed up the computations by a careful programming of the essential routines for integration and administration.

INIS-descriptors: COMPUTERIZED SIMULATION; D CODES; DIFFERENTIAL EQUATIONS;FORTRAN; NUCLEAR POWER PLANTS; STEADY-STATE CONDITIONS; TRANSIENTS

UDC 621.039.5 : 681.3.06

January 1981
Risø National Laboratory, DK 4000 Roskilde, Denmark

ISBN 87-550-0740-6
ISSN 0418-6435

Risø repro 1981

CONTENTS

Page

1. INTRODUCTION

In 1975 a program DYSYS for simulation of dynamic processes was bought from the Kernforschungszentrum Karlsruhe (E.G. Schlechtendahl, 1970). In the past five years it has been used extensively for simulation of nuclear reactors and plant components. Several minor modifications have been done to adapt the program to our computer and our special demands. Major modifications could not be made due to the structure of the program, which is very complicated and difficult to work out. Therefore, it was finally decided to write a completely new program based upon the experience gained in the work with DYSYS.

For the many models which have been developed for DYSYS there has been a major demand that the old models described by Fortran subroutine and input datafiles should be usable with only minor modifications. So the new program, called DYSIM, has been developed along the same lines as DYSYS. But some features are not used in DYSIM, and new ones have been introduced. The accuracy control and steady-state search have been completely changed and the routines most essential for the computing time have been programmed carefully in order to save time at the expense of memory space. The time performance is improved by a factor 1.4-1.5 for typical examples, and the program is cut down from about 1900 Fortran lines to about 1000.

The main features are:

- The program is written in Fortran and includes the main program and a number of subroutines. It takes care of data input with test and documentation, integration of system equations, and organisation of outputs. Several administration procedures are included.

- The problem equations must be formulated in a Fortran subroutine DERIVA, which gives the derivatives for the first order differential equations.

- The structure of the model may be altered so differential equations are changed to algebraic ones and back again at any time during the calculations.

- The integration routine is a fourth-order Runge-Kutta.

- Steady-state calculations can be performed and the result stored in a disk file as initial values for the next run.

- During the transient calculation, accuracy control with stepsize adaption is carried out in a simple and fast manner.

- Output tabulation is specified in the input file and the selected values are stored intermediately in a disk file, which may be changed to a permanent file if the data is to be used to make plots later on.

- Pure time delays can be simulated by use of a separate function, DELAY, which must be included in the user program when it is needed.

- A dump and restart facility provides the possibility of dumping the state of the system in a disk file and in a later run restart the calculation at that state. This is useful for test purposes when an error occurs long into a transient.

2. PROBLEM FORMULATION

The dynamic problem is formulated in a Fortran subroutine with one parameter, DERIVA (NR), which calculates the derivatives for all state variables and performs other calculations of, e.g., algebraic or output variables. DERIVA communicates with DYSIM by three COMMON fields:

```
    COMMON/INTVAR/T, STV(NDE), ALV(NAE)
    COMMON/DERIV/DIF(NDE)
    COMMON/DATA/DATEN(NDA)
```

T is the time variable, STV the time-dependent state variables and ALV are the algebraic or special output variables. The sum of NDE and NAE is limited to 400 in the present version.

DIF contains all the time derivatives.

Variables which shift between state and algebraic variables must be placed in STV and have a derivative in DIF.
DATEN contains all input data given in the input file by the DATA block.

The parameter NR is the substep number in the Runge-Kutta routine; it goes from 1 to 4 during a time step. At the program start an extra call of DERIVA is given with NR=0 providing the possibility of making initial calculations of, e.g., constant parameter values or initial conditions that can be calculated directly. The derivatives from this first call is not used at all in DYSIM.

Other special communication possibilities are mentioned in Chapter 9.

3. INPUT FILE

The input file is divided in blocks of records each with a codework in the first record. Some blocks consist of the codeword alone; in others the codeword is followed by one or more records with data or a test string.

A codeword always consists of a "*" followed by four characters, e.g., *INCO; the rest of the record is empty. Some codewords are obligatory, others are optional.

A data record consist of maximum 72 characters divided in 6 fields of 12 characters each. The data as either integers, real numbers or text strings may be placed everywhere inside the 12-character field. The input routine performs a shift to the right side of the field before the data are interpreted.

The order of blocks is unimportant with the exception that INCO and CHCH must come before REFV.

One special record does not have a codework; that is a comment line with a $ as the first character. A comment record is normally placed as the first record but may be placed everywhere between the blocks. The second character, being either a space or a "-" is used to select a copy of the input file on the output printer. The copy indicator is initially set per default, but is reset by the "-".

Remark: if more than one comment record is used, every one will be used to reset the copy indicator for a "-".

The input file will be read and tested for errors in the subroutine INPUT. Error messages are given in clear text. No guarantee for detection of all errors can be given. If an error is found the INPUT routine will continue and try to check the rest of the input file, and the program will be terminated then.

The following sections explain all codewords and associated data records.

3.1. *INPT

An obligatory codeword followed by one record with a text string, where the first character must be a "*". The following 71 characters are used to identify the problem and are printed as a head for the output tables.

3.2. *INPD

An optional codeword without data. It gives an input documentation in a clearer form than the copy of the input file.It may be used to facilitate the search for errors in the input file.

3.3. *INCO

An obligatory codeword followed by a number of data records. The first one must have two integers: NDE, that is the number of the differential equation, and NAE, that is the number of algebraic variables in COMMON/INTVAR/. Then follows records with initial values, six per record, as many as NDE indicates, and written as real numbers.

3.4. *DATA

An obligatory codeword followed by a number of data records. The first one must have one integer, NDAT, that gives the number of parameters in the following records with six per record. These parameters may be either integers or real numbers; the input routine identifies the type by the decimal period.If no data are used for the problem, NDAT is set to zero and no data records are written. The parameters are placed in COMMON/DATA/ in the written order.

3.5. *CHCK

An obligatory codeword followed by one record with five parameters for control of the integration written as real numbers in the following order:

DTMIN:	Minimum value of the time step
DT :	Initial " " " " "
DTMAX:	Maximum " " " " "
EPSI :	The accuracy control parameter
TMAX :	Maximum value of true transient time

3.6. *TIME

An obligatory codeword followed by one record with one number, TTOT, that gives the maximum value of used computer seconds. The number may be given either as an integer or a real number.

3.7. *REFV

Obligatory when EPSI in the CHCK block is greater than zero. It is followed by a number of records each with two integers and one real number. The integers give a range of the state variables for which the real number is used as reference value in the accuracy control. The block is terminated by a record with 0 in the first field. When the routine for REFV is entered the initial values will be inserted in the reference value array. If all these values are representative no reference value need be specified and the block can be terminated by the first record. However, if some initial values are zero, reference values greater than zero must be specified for those variables.

3.8. *PRNT

An obligatory codeword followed by a number of records. The first one must contain one integer NPRI and five real numbers. NRPI gives the number of print variables including the time. The five real numbers specify the following parameters:

PDT0: Time intervals for printout from time 0 to POT1
POT1
PDT1: " " " " " " POT1 to POT2
POT2
PDT2: " " " " " " POT2 and upwards

The following records specify the variables to be printed with three per record and two fields for each. The first field must hold an integer giving the number in COMMON/INTVAR/ and the second holds a text string with maximum 6 characters, which are used as a name for the variable in the printout table.

3.9. *STST

An optional codework without data. It specifies a steady state calculation.

3.10. *PLOT

An optional codeword without data. It will keep the disk file OUT7 as a permanent filc at program termination. OUT7 contains all the values of the printout variables collected during the run.They can be used later on by an independent plotter program.

3.11. *DELY

An optional codeword without data. It specifies the use of the delay simulation routine on the disk file DELAY and establishes the necessary communication between DYSIM and DELAY.DELAY must be included in the user's own program.

3.12. *DUMP

An optional codeword without data. It gives a dump at the termination time of all variables needed for a later restart. The disk file DYSIMDUMP is created as a permanent file in any case, and the file DELAYDUMP is created if the delay function is used.

3.13. *REST

An optional codeword without data. It is used to restart a calculation after a dump. The information in DYSIMDUMP and DELAYDUMP is loaded as needed.

3.14. *ENDE

An obligatory codeword without data. It indicates the end of the input file.

4. STEADY-STATE CALCULATION

The steady-state is calculated by iteration using the integration routine, but with a large negative value for the time variable and a constant time step DT equal to the maximum value DTMAX. The calculation starts with the values in the INCO block. No stability check or accuracy control is used; so it is the user's responsibility to select a reasonable value for DTMAX that gives a stable and fast iteration. Experience with advanced accelerated steady-state convergence in DYSYS indicates that only a small or no profit is obtained due to the extra administration.

The calculation will go on until the computing time reaches the limit set by the TIME block. At termination time the state variables are written in a permanent disk file FIL12 with the format 6E12.6. They can then be inserted manually in the input file in the INCO block as a new set of initial values.

During the program run the state variables will be printed out every 100 steps. The user must decide himself, when steady-state has been reached with a reasonable accuracy.

The steady-state convergence can be accelerated by the user if some variables can be derived directly or bound to other variables, e.g. the outlet temperature from a reactor core can be made use of in calculating the temperature further on in the cooling loop.

In DERIVA the user can detect a steady-state calculation on a negative time value.

Steady-state is calculated in the subroutine ICCAL that utilises the subroutine RUNGE.

5. TRANSIENT CALCULATION

The transient calculation starts normally at time zero with the initial conditions in the INCO block. The initial time step is given in the CHCK block. An accuracy control is carried out for each time step if EPSI in the CHCK block is greater than zero. The deviation between the Runge-Kutta step and a simple Euler step is calculated for each variable and compared with a reference deviation REF=EPSI*REFV, where REFV is the reference value. The following actions are taken:

DEV>3*REF:	The step is cancelled and the time step is divided by 1.5
3*REF>DEV>REF:	The step is accepted but the next time step is divided by 1.5
DEV>REF/3:	Incrementation of time step is prohibited.

These calculations are made for all state variables until the first criterion is fulfilled or to the end. The first variable which fulfills the first or second criterion is marked and a counter for that variable is incremented by one. If none of the three criteria are fulfilled for any variable the time step is multiplied by 1.5 and an incrementation counter is incremented by one.

At program termination a small list is written below the output table giving the number of DERIVA calls, the number of step increments and the number of step decrements for each variable which caused a decrement.

By every time step change the new step is limited to the range given by DTMIN and DTMAX in the CHCK block. If the routine tries to decrease the step below DTMIN when it already is equal to DTMIN the program is stopped by an announcement of step size control stop giving the number of the variable that caused the step decrement.

The deviation between the Runge-Kutta and Euler step gives a measure of the higher derivatives and is therefore a reasonable quantity to use for the step size control being both simple and fast. The more refined and complicated calculation in DYSYS is more time consuming, and the two procedures give similar time step reductions for typical problems for power plant transients.

The procedure developed for DYSIM has a further advantage in avoiding frequent cancellations of steps followed by step increments.Only large pertubations of the system give rise to step cancellations when EPSI is carefully selected (about 1.E-3). Most often the step will be decreased by criterion no. 2 in due time to avoid violating criterion no. 1.

The transient calculation is done in the subroutine INTEG that utilizes the subroutine RUNGE.

6. PRINTOUT FACILITY

The variables to be printed in the output table as well as the time intervals are specified in the PRNT block.

In order to hit the correct printout times DYSIM will adapt the time step if needed, but the original step is reinserted so that loss of time with a gradually step increase is avoided.

Besides the automatically actuacted printout, the user can demand a printout at the starting time of any step. This is done with the statement CALL PRINT, which can be given at any value of the substep number NR.

The variables at time zero are printed as the first line, and at program termination as the last line.

The maximum value of variables to be printed is 53 plus the step number which is inserted automatically in the first column.

The data collected during the integration are stored in a buffer 540 words long giving space for 10 sets of values between transfer to the disk file OUT7. When a smaller number of print variables is specified the buffer will be used in an economic way so more sets can be stored between disk transfer.

The collection of data is done in subroutine INTEG which places all data in file OUT7 in unformatted form with record length = 540. A set of variables consists of NPRI+1 data, and a record has 540/(NPRI+1) sets (truncated integer). At program termination the subroutine OUTPUT is called. It reads OUT7 and arranges the data in sides and columns for the output printer. The variable names specified in the PRNT block are used as headings.

When the program stops, OUT7 will be purged unless PLOT is specified. In that case the file is extended with one record which contains the integer NPRI+1 followed by two arrays. The first one with the variable names is 54 words long independent of NPRI. The second one with the problem identification from the INPT block is 12 words long.

7. TIME-DELAY SIMULATION

A function DELAY has been developed for simulation of time delays as found in a flow through a tube with a uniform velocity, which may change, but without sign shift. The function is not included in DYSYS, so it must be included in the user source file when it shall be used. The separation is done to save core memory when DELAY is unused, as it has a relative large data buffer.

Taking the above-mentioned example of a tube with given inlet temperature TI(t), velocity v(t) and tube length L, the outlet temperature TO(t)=TI(t-τ) is found in the following way:

- Introduce an extra state variable

$$X = \int_0^t v \, dt, \ \dot{X} = v, \ X(0) = 0$$

- Find the time delay τ by the statement TAU=TRNSTM('TAU',X,X-L), where the first parameter is an identifier of that call, the second is the new value of the position X, and the last one is the delayed value of X to look for in a buffer. The value returned for TAU is not in terms of seconds, but in units of buffer positions, i.e. time steps, with linear interpolation between buffer values.

- Find the delayed temperature by the statement TO=DEADTM('TOUT',TI,TAU), where TOUT is an identifier and TI is the new inlet temperature. TO is found by linear interpolation between buffer values.

Using TAU in units of buffer positions means that every time-delay must be found by separate call of TRNSTM. If TAU were calculated in seconds other delays with fixed relations to TAU could be calculated by algebraic equations.

The method used here will often require some extra call of TRNSTM and extra buffer space, but the single calls of both TRNSTM and DEADTM are considerably faster. With simulations carried out on our Burroughs 6700 computer saving computer time is more important than saving core space.

For each function call characterized by an identifier an input and an output buffer is used each 360 words long. When the input buffers are full they are transferred to the disk file DYSIMDELAY, where the output routine can find the data when not present in any of the two buffers. By program termination DYSIMDELAY will be purged.

The actual insertion of new values in the buffers is made by a call from DYSIM after each accepted time step. It means that only the last one for NR=4 is stored.

Separate actions are taken when the function is called by the first call of DERIVA. The identifiers are inserted in a name buffer for use at later calls. Coinciding identifiers are not allowed and will cause the program to stop with an error message. And use of identifiers not used in the first call will have the same effect.

Adoption of buffer length in order to save core space or to get space for more function calls is easily done by alteration of array dimensions as explained in comments in the head of DELAY. The present version has space for 20 function calls.

8. DUMP AND RESTART FACILITY

When an error occurs long into a calculation it may be difficult to find the cause if the calculation has to be started at time zero for every test run. Therefore a dump and restart facility has been developed so the user can run the problem once to near the point where the error occurs, and dump the state of the problem for later restart.

When the dump is actuated the content of the two COMMON fields INTVAR and DERIV is stored in the file DYSIMDUMP, which is made permanent. If DELAY is used, the file DYSIMDELAY is extended with some records in which the content of the delay buffers and relevant pointers are stored; the file name is changed to DELAYDUMP, and the file is made permanent. An announcement with the time for dump is given just after the output tables.

When a restart is activated the data stored in DYSIMDUMP and DELAYDUMP is used to reestablish the problem state prior to the dump, and the integration continues from there.

The two disk files remains permanent so the restart can be repeated.They must be removed manually when no longer used.

9. OTHER FACILITIES

9.1. Program termination

The calculation may be terminated by the user at the end of any integration step by the statement CALL TERM. For special purposes, e.g. search for errors, a more abrupt stop can be made by the statements CALL NSTOP; RETURN. It results in an immediate integration stop regardless of the substep number NR and gives a special messages after the output table.

9.2. Repetetion of integration step

The integration step in progress can be immediately interrupted and repeated with a smaller one (divided by 1.5) by the statements CALL REPET; RETURN regardless of the value of the substep number NR. The step decrease may be repeated several times, e.g. in order to hit a certain value for one of the state variables. At the first normal step after CALL REPET the original value of the time step will be reinserted. A permanent step decrease cannot be called open by the user.

9.3. Activation of special user routines

Special user routines can be activated by the statement CALL RECAL, when the integration step in progress has been accepted; it may be given by any value of the substep number NR. It results in execution of the statement CALL DYNAMS in DYSIM. DYNAMS must be the name of a user subroutine or an entry point without parameters.

Just before the program stops with a termination message DYSIM gives a CALL YOUT. YOUT must be supplied by the user in all programs as a subroutine or entry point without parameters. It can be used, e.g. to give a more detailed description of the system state at termination time than obtained by the output table.

10. EXPLANATION OF AN EXAMPLE

A listing of a small test example with input file is given on page 25 and the output print on page 26-28.

DELAYTEST is the user's program file. Four state variables and two output variables are used. X1 and X2 are water inlet temperatures to two tubes, Y1 and Y2 are water particle positions relative to time zero used for time delay simulation, and Z1 and Z2 are the tube outlet temperatures. The inlet temperature derivatives X1P and X2P, the water velocities Y1P and Y2P, and the tube lengths are given by the input data D(1)-D(6), respectively. Lines 160-190 shows the time delay function calls needed to find Z1 and Z2. Lines 220-250 shows that DYNAMS and YOUT are defined as empty routines not being used. The file DELAY is included by line 270. DYSIM is bound by the compiler statements in line 30-50.

IN5D is the input file.

The INPT block gives a text string as identification of the problem.

The INCO block defines the number of differential and algebraic variables as 4 and 2, and the initial values as 0 for all state variables.

The DATA block has 6 data that specifies: X1 and X2 increases with constant rates equal to 1 and 0.5 ^{o}C/sec, the water flows with constant velocities equal to 1 and 0.5 m/sec, and the tube lengths are 2 m for each. So the time delays becomes 2 and 4 sec respectively.

The CHCK block gives the initial time step equal to the maximum time step = 0.1 sec, and EPSI=0, so no accuracy control will be performed. The maximum transient time is 40 sec.

The TIME block gives the maximum computer time as 60 sec.

The PRNT block asks for 7 print variables beginning with time intervals of 0.2 sec, increasing to 1 sec at T>10 sec, and continuing at 1 sec for T>20 sec. The variable numbers and names are given in line 250-270. Note that the time is called by no. 0.

The DELY codeword specifies the use of the delay function.

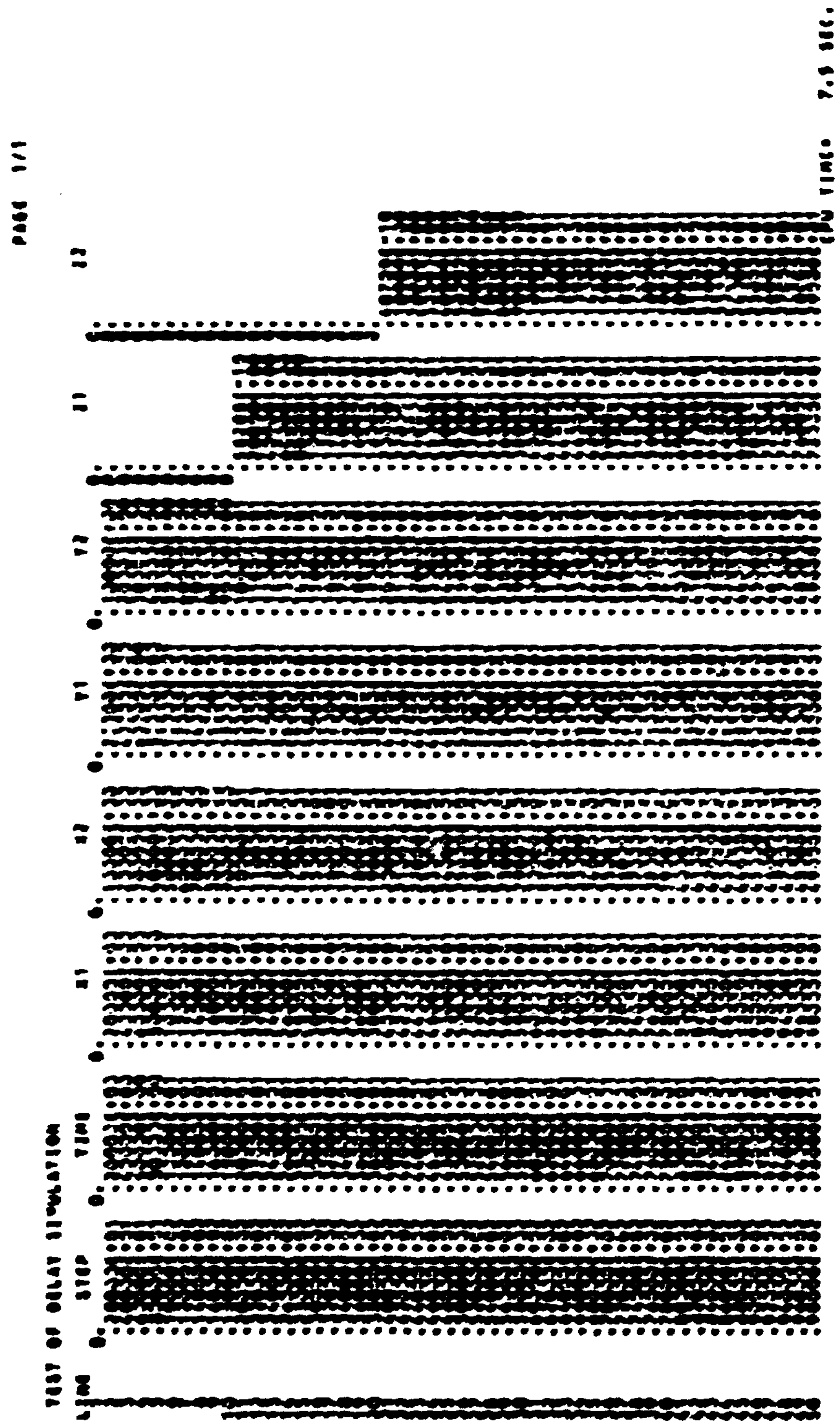
The DUMP codeword specifies a dump at program termination (here at T=40 sec).

The output print shows first the copy of the input file. No special input documentation is asked for. The result of the calculation is shown on the pages marked PAGE 1/1 and PAGE 2/1 with the variables in columns as specified by PRNT. If more than 10 columns were specified extra pages would be printed marked as PAGE 1/2 and PAGE 2/2, and so on. Below the tables each side has information about the spent computer time corresponding to the last step at that page.

```
DELAYTEST.                                                  CREATION: 80/12/19,
$RESET LIST
$RESET FREE
$SET LIBRARY
$SET AUTOBIND
$HOST IS OBJECT/DYSIM
C
      SUBROUTINE DERIVA(NR)
      COMMON /INTVAR/ T,X1,X2,Y1,Y2,Z1,Z2
      COMMON /DERIV/ X1P,X2P,Y1P,Y2P
      COMMON /DATA/ D(6)
C
      X1P=D(1)
      X2P=D(2)
      Y1P=D(3)
      Y2P=D(4)
      TAU1=TRNSTM('TAU1',Y1,Y1-D(5))
      Z1=DEADTM('DEL1',X1,TAU1)
      TAU2=TRNSTM('TAU2',Y2,Y2-D(6))
      Z2=DEADTM('DEL2',X2,TAU2)
      RETURN
C
      ENTRY DYNAMS
      RETURN
      ENTRY YOUT
      RETURN
      END
$INCLUDE 'DELAY'
```

```
IN5D.                                                       CREATION: 80/12/19,
$ DATA FILE FOR DELAYTEST.
*INPT
* TEST OF DELAY SIMULATION
*INCO
4          2
0.         0.          0.          0.
*DATA
6
1.         .5          1.          .5          2.0         2.0
*CHCK
.01        .1          .1          0.          40.
*TIME
60
*PRNT
7          .2          10.         1.          20.         1.
0          TIME        1           X1          2           X2
3          Y1          4           Y2          5           Z1
6          Z2
*DELY
*DUMP
*ENDE
```

```
$ DATA FILE FOR DELAYTEST.
*INPT
* TEST OF DELAY SIMULATION
*INCO
4           2
0.          0.          0.          0.
*DATA
6
1.          .5          1.          .5          2.0         2.0
*CHCK
.01         .1          .1          0.          40.
*TIME
60
*PRNT
7           .2          10.         1.          20.         1.
0           TIME        1           X1          2           X2
3           Y1          4           Y2          5           Z1
6           Z2
*DELY
*DUMP
*ENDE
```


TEST OF DELAY SIMULATION
PAGE 1/1
LINE
STEP
TIME
X1
X2
Y1
Y2
Z1
Z2
END TIME= 7.5 SEC.

```
                                                                                                        PAGE  2/1
  TEST OF DELAY SIMULATION

 LINE       STEP          TIME          X1            X2            Y1            Y2            Z1            Z2
   1     .10000E+03   .10000E+02   .10000E+02   .50000E+01   .10000E+02   .50000E+01   .80000E+01   .30000E+01
   2     .11000E+03   .11000E+02   .11000E+02   .55000E+01   .11000E+02   .55000E+01   .90000E+01   .35000E+01
   3     .12000E+03   .12000E+02   .12000E+02   .60000E+01   .12000E+02   .60000E+01   .10000E+02   .40000E+01
   4     .13000E+03   .13000E+02   .13000E+02   .65000E+01   .13000E+02   .65000E+01   .11000E+02   .45000E+01
   5     .14000E+03   .14000E+02   .14000E+02   .70000E+01   .14000E+02   .70000E+01   .12000E+02   .50000E+01
   6     .15000E+03   .15000E+02   .15000E+02   .75000E+01   .15000E+02   .75000E+01   .13000E+02   .55000E+01
   7     .16000E+03   .16000E+02   .16000E+02   .80000E+01   .16000E+02   .80000E+01   .14000E+02   .60000E+01
   8     .17000E+03   .17000E+02   .17000E+02   .85000E+01   .17000E+02   .85000E+01   .15000E+02   .65000E+01
   9     .18000E+03   .18000E+02   .18000E+02   .90000E+01   .18000E+02   .90000E+01   .16000E+02   .70000E+01
  10     .19000E+03   .19000E+02   .19000E+02   .95000E+01   .19000E+02   .95000E+01   .17000E+02   .75000E+01
  11     .20000E+03   .20000E+02   .20000E+02   .10000E+02   .20000E+02   .10000E+02   .18000E+02   .80000E+01
  12     .21000E+03   .21000E+02   .21000E+02   .10500E+02   .21000E+02   .10500E+02   .19000E+02   .85000E+01
  13     .22000E+03   .22000E+02   .22000E+02   .11000E+02   .22000E+02   .11000E+02   .20000E+02   .90000E+01
  14     .23000E+03   .23000E+02   .23000E+02   .11500E+02   .23000E+02   .11500E+02   .21000E+02   .95000E+01
  15     .24000E+03   .24000E+02   .24000E+02   .12000E+02   .24000E+02   .12000E+02   .22000E+02   .10000E+02
  16     .25000E+03   .25000E+02   .25000E+02   .12500E+02   .25000E+02   .12500E+02   .23000E+02   .10500E+02
  17     .26000E+03   .26000E+02   .26000E+02   .13000E+02   .26000E+02   .13000E+02   .24000E+02   .11000E+02
  18     .27000E+03   .27000E+02   .27000E+02   .13500E+02   .27000E+02   .13500E+02   .25000E+02   .11500E+02
  19     .28000E+03   .28000E+02   .28000E+02   .14000E+02   .28000E+02   .14000E+02   .26000E+02   .12000E+02
  20     .29000E+03   .29000E+02   .29000E+02   .14500E+02   .29000E+02   .14500E+02   .27000E+02   .12500E+02
  21     .30000E+03   .30000E+02   .30000E+02   .15000E+02   .30000E+02   .15000E+02   .28000E+02   .13000E+02
  22     .31000E+03   .31000E+02   .31000E+02   .15500E+02   .31000E+02   .15500E+02   .29000E+02   .13500E+02
  23     .32000E+03   .32000E+02   .32000E+02   .16000E+02   .32000E+02   .16000E+02   .30000E+02   .14000E+02
  24     .33000E+03   .33000E+02   .33000E+02   .16500E+02   .33000E+02   .16500E+02   .31000E+02   .14500E+02
  25     .34000E+03   .34000E+02   .34000E+02   .17000E+02   .34000E+02   .17000E+02   .32000E+02   .15000E+02
  26     .35000E+03   .35000E+02   .35000E+02   .17500E+02   .35000E+02   .17500E+02   .33000E+02   .15500E+02
  27     .36000E+03   .36000E+02   .36000E+02   .18000E+02   .36000E+02   .18000E+02   .34000E+02   .16000E+02
  28     .37000E+03   .37000E+02   .37000E+02   .18500E+02   .37000E+02   .18500E+02   .35000E+02   .16500E+02
  29     .38000E+03   .38000E+02   .38000E+02   .19000E+02   .38000E+02   .19000E+02   .36000E+02   .17000E+02
  30     .39000E+03   .39000E+02   .39000E+02   .19500E+02   .39000E+02   .19500E+02   .37000E+02   .17500E+02
  31     .40000E+03   .40000E+02   .40000E+02   .20000E+02   .40000E+02   .20000E+02   .38000E+02   .18000E+02
                                                                                                      CPU TIME=  26.2 SEC.

DUMP ON FILE "DYSIMDUMP" AT TIME:  40.000 SEC.

DYSIM IS TERMINATED
```

REFERENCES

SCHLECHTENDAHL, E.G. (July 1970). DYSYS - A Dynamic System Simulator for Continuous and Discrete Changes of State, Institut für Reaktorentwiklung, Karlsruhe, KFK 1209.

Risø National Laboratory **Risø - M -** 2271

Risø - M -

Title and author(s)	Date January 1981
Description of a simulation system DYSIM for continuous dynamic processes P. la Cour Christensen	Department or group Dept. of Reactor Technology
	Group's own registration number(s)
27 pages + tables + illustrations	
Abstract	Copies to

A general purpose simulation system DYSIM for continuous dynamic processes has been worked out upon five years experience with a similar system DYSYS from Kernforschungszentrum Karlsruhe. The new system has been made in order to improve the performance by excluding unused features and including new ones, and speed up the computations by a careful programming of the essential routines for integration and administration.