KERNFORSCHUNGSZENTRUM

KARLSRUHE

Juli 1970 KFK 1209

Institut für Reaktorentwicklung

- DYSYS -

A Dynamic System Simulator for Continuous and Discrete Changes of State

E. G. Schlechtendahl

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

Juli 1970 KFK 1209

Institut für Reaktorentwicklung

- DYSYS -

A Dynamic System Simulator for Continuous and Discrete Changes of State

E. G. Schlechtendahl

Gesellschaft für Kernforschung mbH., Karlsruhe

Abstract

DYSYS is a digital computer program which allowsthe simulation of continuous and discontinuous changes of state of a system with up to 500 state variables. The program requests that the dynamic system equations be described by a Fortran IV subroutine. Other problem specifications (initial conditions, system parameters, control- and output specifications) are first submitted to extensive error checking before the simulation is executed. Variable deadtime simulation is available.

Acknowledgement

The author gratefully acknowledges the valuable contributions of Mr.C.Koepp, who implemented the variable deadtime simulation feature.

Zusammenfassung

DYSYS ist ein digitales Rechenprogramm, das die Simulation stetiger und unstetiger Zustandsänderungen in einem System von bis zu 500 Zustandsvariablen gestattet. Das Programm verlangt, daß die das System bestimmenden Gleichungen als eine Fortran IV Subroutine geschrieben werden. Andere Problemspezifikationen (Anfangsbedingungen, Parameter, Genauigkeits- und Ausgabeinformationen) werden zunächst ausführlich auf Fehler geprüft, bevor die Simulation ausgeführt wird. Auch die Simulation variabler Totzeit ist möglich.

Table of Content

Seite

1. Introduction

A wide variety of technical problems can be described as initial value problems of ordinary differential equations. This is the classical field of application of analog computers. Even many problems, which originally lead to partial differential equations, such as transient heat transfer problems, can be solved by a final difference approximation in the space domain, which reduces the problem to ordinary differential equations in the time domain.

Analog computers have a number of deficiences e.g. the variable must be scaled, the treatment of nonlinear algebraic equations is difficult. Many of these deficiences can be overcome by digital computer application, however, at the expense of computer time required. Hybrid computation is in many cases the optimum approach to the solution of systems of ordinary differential equations and is not restricted to initial value problems. The all-digital approach, however, has the following advantages: There are far more medium and large scale digital computers than hybrid computers and there are far more people who know how to use digital computers. The access methods to digital computers, both batch and interactive, allow more different problems to be solved at a time than with the hybrid computers which require a tight interaction with the hardware [1].

For these reasons a number of digital computer programs have been developped which supply with more or less comfort the service of hybrid computers. Most of these programs are block oriented similar to the original analog computer approach where each hardware unit (integrator, summer, function generator etc.) represents a block of certain type [2, 3, 4]. Block oriented computer codes are very handy for people who are used to the analog computer formulation of a problem, but they have 4 disadvantages:

- In order to solve the problem, the problem equations must first be transformed into a block structure and this block structure must be coded in a second step.

- Differential equations of similar structure - as they occur in a finite difference formulation of partial difference equations - must be coded explicitly as often as they occur.

- When the program is designed, nobody can foresee all block types which future users would like to have

- Each block represents only one type of operation throughout the whole problem solution: special blocks which simulate the switches and function generators of the analog computer, offer only a small degree of flexibility.

A completely different approach is used in the Dynamic System Simulator DYSYS. Instead of being block oriented this program is equation oriented similar to the CSMP [59]. It requires all the problem equations to be formulated in Fortran IV. The disadvantages of block oriented programs do not arise for the following reasons:

- The problem equations can immediately be written in Fortran with problem oriented names of the variables.

- Differential equations of similar structure can easily be handled by means of DO-loops.

- The full capability of Fortran is available to the user, thus offering an almost unlimited flexibility.

- The structure of the functional relation between variables may vary during the course of the problem solution, in particular the same variables may be defined by differential equations or algebraic equations in different domains of the problem variables.

- Any number of discrete (events discontinuous changes of state)may occur during the course of the continuous system simulation.

Other features incorporated in DYSYS may be found in other dynamic simulation codes e.g.

- documentation of the input data
- extensive input data testing with the aim to detect all errors at once
- steady state search before the dynamic problems solution
- built-in output of all extreme values of the integration values
- automatic stepsize control
- easy-to-read print output
- plot output
- easy input handling for parametric studies
- capacity for up to 500 differential equations

The following desirable features are not yet incorporated in DYSYS

- dynamic storage allocation for an unlimited number of differential equations
- automatic iteration of boundary value problems
- optimization with restrictions.

2. Problem formulation for DYSYS

In order to solve a dynamic problem with DYSYS a model of the problem must be set up by coding in Fortran the equations which describe the problem and by preparing the input data for each particular task.

2.1 Problem equations

The problem equations must be coded in Fortran IV as a subroutine named DYNAMØ. Therefore the first statement of this subroutine (except for comment cards) must be

```
SUBRØUTINE DYNAMØ
```

The subroutine communicates with DYSYS via 3 labeled Common blocks, namely

```
COMMØN/DATA/ for the parameter data
CØMMØN/INTVAR/ for the variables
CØMMØN/DERIV/ for the derivatives
```

If no parameters are used, i.e.if no DATA input block exists in the input to the task, the DATA-Common is not required. The INTVAR-Common contains first the independent variable, followed by the integration variables in the same sequence as in the INCØ-input block, followed by other variables. The DERIV-Common contains the derivatives of the integration variables in the corresponding sequence. All parameters, all dependent variables (but not the independent variable) and all derivatives may be modified by equations in DYNAMØ.

2.2 Simple DYNAMO example

The basic rules for coding DYNAMØ shall be illustrated in a simple example.
The movement of a ball which is released at a certain height above an elastic plate is described by the following equations:

$$\frac{d}{dt} \text{ (height) } = \text{ velocity}$$

$$\frac{d}{dt} \text{ (velocity) } = \text{ gravity}$$

If the height is equal to 0 a discrete event happers, namely the ball movement is reversed at reduced velocity. With a damping factor ε

the following equation holds for height = 0.

$$(\text{velocity})_{\text{new}} = -\varepsilon\cdot(\text{velocity})\ \text{old}$$

These equations might be coded as follows:

```
      SUBROUTINE DYNAMO
      COMMON/DATA/GRAVIT,EPSIL
      COMMON/INTVAR/TIME,HEIGHT,VELOC
      COMMON/DERIV/      DHEIGH,DVELOC
      IF(HEIGHT.GT.0.)GO TO 1
      HEIGHT = 0.
      VELOC = -EPSIL*VELOC
    1 DHEIGH = VELOC
      DVELOC = GRAVIT
      RETURN
      END
```

2.3 Input data

A detailed description of the input data is given in chapter 3. For a summary see Appendix A.

As an illustration, the input data of the sample problem used in 2.2 will be formulated more completely:

Title of the problem	= Sample problem 1	
Task 1: Initial conditions	= height = 1m	
	velocity = 0	
Printout variables	= height, velocity	
Parameters	= gravity = -9.81 m/sec^2	
	= damping factor ε = 0.8	
Control data	= minimum stepsize	= 0.00001 sec
	initial stepsize	= 0.01 sec
	maximum stepsize	= 0.05 sec
	accuracy	= 0.001
	end-of-problem time	= 10 sec
	max.number of steps between printout	= 1000
	max.number of steps	= 10000

Plot output variables - velocity from -20m/sec to + 20m/sec
- height from 0 to 1 m
both versus time from 0 to 10 sec

The input deck, which together with the DYNAMØ subroutine of 2.2 represents the complete input for the solution of the sample problem by DYSYS, is listed in Appendix B.
The complete output of the problem is also shown in Appendix B.

3. Input Handling Routines

3.1 Input reader

The name of this routine is SHFT 12. The routine is called only once in each application of DYSYS.

This routine processes all input cards one at a time and prints their content (80 columns). All cards which do not contain a $ character in column 1 are copied onto an input copy file.(Columns 1 through 71). Cards which contain the $ in column 1 are not copied. They may be used for whatever comments the user wishes to be documented as useful information about this particular application of DYSYS.

All cards beginning with the character * are copied as such from column 2 through 71. Column 1 is made blank. In column 72 through 80 a sequence number is inserted. All other cards are divided into 6 fields of 12 characters each. The information in each of these fields, if any, is shifted right until it becomes rightjustified. The cards, thus modified, are copied onto the input copy file with a sequence number inserted in columnes 72 through 80. Information contained in columns 72 through 80 of the original input cards drops out.

This process continues until a card containing only the character sequence END anywhere in columns 1 through 12 is encountered. This will be the last card to be copied. Any errors occurring will lead to an immediate stop.

3.2 Input documentation

The name of this routine is XADE. The routine is called only once for each application of DYSYS.

This routine reads all cards (80-character records) from the input copy file and prints them as a documentation of all relevant information supplied to the input processor. This documentation does not contain the comment cards (cards with $ in column 1). The sequence number of these cards, as inserted by the input reader, is printed.

The routine continues until the character sequence END is found in columns 10 through 12, with columns 1 through 9 being blank. If any error occurred in this routine, DYSYS is terminated after completion of the input documentation.

3.3 Input Processor

The name of this routine is INPUT. The routine is called once for each task of DYSYS.

The input to a task consists of a number of input blocks which may be in any sequence except for those blocks which modify previous information.
The routine reads the input blocks from the input copy file until the next END or GØØN label is encountered. Each input block consists of a block label and the block information as described below and as

summarized in Appendix A. A thorough checking of the information is made. If an error was detected in any input block for one task, this task and all its follow-on-tasks, will not be executed.

3.3.1 Initialisation

At first, a standard set of default information for all input blocks is established. (See Appendix A). If one or more of the input label types is not found, this default information will be used for the execution of the task . If the new task is a follow-on-task, i.e. if the input to this task does not start with the INPT label , all input blocks are initialized to the final input information of the preceding task. In this case, the task sequence number is incremented by 1 and title and sequence number are printed. Note: if the INPT label is encountered in the task input but not as the first label of this task, this task is still considered as a follow-on-task and the input information established so far is not modified except for the title.

Following the initialization, the input processor reads one card from the input copy file and interprets it as a block label. The block information is then expected to start in the following card. If the label cannot be interpreted, the error handling routine is initialized.

3.3.2 INPT - block

The card following the INPT-label is used as title of this task. If the INPT-block is the first block of a task input, the task sequence number is set to 1. The titel and sequence number of this task are printed as headline on the output. If this block is not the first block of a task, this task will be considered as a follow-on-task, and all previously specified information will be accepted for the new task. If this not the first block of a task, a new group of the task begins and previous information is no longer available.

3.3.3 DATA - block

The card following the DATA-label contains an integer in the first field indicating the number of data to be read starting in field one of the subsequent card. The data may be of any of the following types: real, integer or a literal.

The cards containing the data are read one at a time. Each field is checked for its content and the appropriate format is generated. If the first non-blank character in the field is numeric or+ or-, the data is assumed to be numeric. If it is a period and the subsequent character is numeric, the data is assumed to be numeric. Numeric data are assumed to be real, if a period is found in the character string. If no period is found in a numeric field, integer is assumed. In all other situations, including a completely blank field, the data is taken as alphabetic. In this case only the four last characters of the string are taken as input data of this field.

The card content and the appropriate format are written on an intermediate data file for later processing. The maximum number of data allowed is 500.

If there is no data block, the programm assumes that no data are required.

3.3.4 INCØ - block

The card following the INCØ - label contains an integer in the first field indicating the number of initial conditions to be read starting in field one of the subsequent card. The initial conditions for all differential equations must be valid real data. The maximum number of initial conditions allowed is 500. Otherwise the input handling routine is called.
If there is no INCO-block, the program assumes that there is one differential equation with an initial condition of 0.

3.3.5 PRNT - block

The card following the PRNT-Label contains in the first field the number of variables to be printed. The following cards contain three pairs of integer and title information (except for the last card of this block which may contain 1 through 3 pairs of print information). The integer indicates the sequence number of the dependent variable in the INTVAR-Common block (see 2.1), sequence number 0 representing the independent variable. The sequence numbers are not limited to the number of differential equations but must not exceed 500. Otherwise the input error handling routine is called. The title is a character string of up to 4 characters and will be printed as heading to the variable on the print output.

If there is no PRNT-block, the program assumes that the independent variable with title X and the first dependent variable with title Y are to be printed.

3.3.6 CHCK - block

The card following the CHCK-label contains 5 real data fields: the minimum allowable integration stepszize, the estimated initial integration stepsize, the maximum allowable integration stepsize, the accuracy parameter and teh final value of the independent variable.

The next card contains two integer fields: the maximum number of successful integration steps and the maximum number of steps from one printout to the next one.

If the data are not meaningful (e.g.negative) ot if the accuracy parameter is less than 10^{-7} or greater than 0.1 an error message is printed and the error handling routine is called.

3.3.7 STST - block

This block contains no additional cards. If the STST-label is found, a steady state search will be attempted for this task and all subsequent tasks until a new group of tasks begins, prior to the solution of the dynamic problem.

3.3.8 REFV - block

This block allows the arbitrary setting of minimum reference values for the accuracy control. The cards (any number is allowed) following the REFV-label should contain three data each, two integers and one real. The integers specify the range of sequence numbers of dependent variables for which the real value will be taken as a minimum reference.

If one of the integers is outside the range 1 through 500 or if the first one is greater than the second one, the block is considered to be finished and the next card to be a new label.

3.3.9 TIME - block

The card following the TIME-label contains an integer specifying the number of additional minutes CPU-time allowed for this task. If this block is not found in the first task of a task group, no time control will be performed for all tasks of this group.

3.3.10 PLOT - block

The PLOT-label is followed by a number of card groups each one specifying a complete plot in Cartesian coordinates. The first card of the group contains as an integer the number of curves on this plot. If this number is not within the range 1 through 30 the program assumes the PLOT-block to be finished and the next card to be a new label. The second card of the group contains two integers. The first integer corresponds to the format of the hardcopy of the according to the conventions of PLOTA(*). The second integer indicates the variable sequence number of the variable to be used as the abscissa. The subsequent card contains 6 real values:

the maximum value of the abscissa
the minimum value of the abscissa
the abscissa increment corresponding to 2 cm on the hardcopy
the maximum value of the ordinate
the minimum value of the ordinate
the ordinate increment corresponding to 2 cm of the hardcopy

The subsequent cards contain integers (6 on each card except for the last one, which may contain up to 6 integers) specifying the variable sequence numbers of the variables to be plotted.

The next card is read and is assumed to specify the number of curves on the next plot.
If variables are to be plotted, which were not specified for printout, they are automatically added to the list of printout variables.

3.3.11 MODV - block

The subsequent cards specify modifications to the initial conditions defined previously by the INCO-block (and possibly preceding MODV-blocks). Each card contains two pairs of an integer and a real value. The integer specifies the sequence number of the variable to be modified, the real specifies the new initial condition.

*) Special PLOT-subroutine of Kernforschungszentrum Karlsruhe

If the first integer is 0, this card is not used for modifications. If the second integer is 0, only the variable specified by the first integer will be modified. In either case the program assumes the next card to be a new label. If an integer is less than 0 or greater than the number of initial conditions specified in the preceding INCØ-block, the error handling routine is called.

3.3.12 MODD - block

The subsequent cards specify modifications to data previously defined in a DATA-block or a MODD-block. The conventions for this block are the same as for the MODV-block, except that the new data values may be any of the following types: integer, real or a 4 character long literal.
The procedure to determine the format of the new data is identical to the one described for the DATA-block.

The individual cards of this block and their appropriate format are written onto the same intermediate file as the cards of the data block for later processing.

3.3.13 GOON - block

This block has no additional cards. The occurrence of the GOON-label indicates the end of the input to one task.

3.3.14 END - block

This block has no additional cards. The occurrence of the END-label indicates the end of the input to the last task.

3.3.15 Input interpretation

After a GOON - or an END-label was found and if no error was detected during the reading of the task input, the program will prepare a complete documentation of the task input as interpreted by the program itself. In order to do so, the program first reads the information which describes the parameters from the intermediate file with their appropriate format and stores them into the DATA common. The printout of the input processor is sellexplanatory and is the exact description of the task as delegated to the steady state and transient solution routines.

3.3.16 Input error handling

If an input error is detected by the input processor, the error handling routine is called.

At first, the error handling routine attempts to read the next card and, if successful, will print this card in order to make it easier to locate the error. The program will then check the following cards, until one of them contains a block label. If so, the program will reasume input reading and input checking at this label. However, no task will be submitted for steady state or transient solution until a new group of tasks is found, which is identified by an INPT-label immediately after the GOON-label. The program is terminated when the END-label is found.

If the end of the input data set is encountered before the END-label was found, either during the normal input reading or within the error handling routine, the program prints an error message and terminates execution

4. Problem Solution

The name of this routine is XRUNGE. It consists of two parts: the steady state iteration and the transient problem solution. After initialization of the routine, steady state iteration will be performed if requested in the input of the task.

4.1 Fundamentals

The program assumes that the problem to be solved is described by a number of first order ordinary differential equations plus, possibly, a number of algebraic equations. The maximum number of variables to be handled is 500. All variables are consecutively numbered. The first variables are primarily considered as integration variables, i.e.it is assumed that they may follow a first order differential equation at any time or they may be determined by an algebraic equation. The remainder of the variables must be described by algebraic equations.

The following set of equations is assumed as the basic problem description.

$$\frac{dy_i}{dt} = f_i(t, y_1 \ldots\ldots y_k)$$

for at least one i at
any t in the range + $0 \leqq t \leqq \text{tmax}$
with $1 \leqq i \leqq n \leqq k \leqq 500$

(1)

$$y_j = f_j(t, y_1 \ldots\ldots y_k)$$

for all $j \neq i$
with $1 \leqq j \leqq k \leqq 500$

where $y_1 \ldots y_k$ are the dependent variables, and t is the independent variable. In case of a steady state iteration request the validity of these equations is extended to include negative values of t. The steady state iteration will be performed on the following set of equations.

$$\frac{dy_i}{dt} = f_i(t, y_1 \ldots\ldots y_k)$$

(2)

for $t = t^- < 0$

$$y_j = f_j(t, y_1 \ldots\ldots y_k)$$

The program assumes that these equations are computed by a Fortran - IV-subroutine named DYNAMØ.

4.2 Steady State Iteration

The problem to be solved is given by

$$\begin{aligned} \frac{dy_i}{dt} &= f_i(t, y_1 \cdots\cdots y_k) = 0 \qquad \text{for } t = t^- < 0 \\ y_j &= f_j(t, y_1 \cdots\cdots y_k) \end{aligned} \tag{3}$$

The solution of this problem is identical to the solution of the following problem

$$\begin{aligned} \dot{y}_i = \frac{dy_i}{dt} &= a_i \cdot f_i(t^-, y_1 \cdots\cdots y_k) = 0 \\ y_j &= f_j(t^-, y_1 \cdots\cdots y_k) \end{aligned} \tag{4}$$

where a_i is an arbitrary positive factor. This modified problem is solved in the steady state iteration routine rather than the original problem (3). Here the a_i are acceleration factors which are generated by the program in order to improve the convergence behaviour.

The program cannot determine whether one or more solutions to the steady state problem exist; nor can it determine whether a steady state condition once it is found is actually the one desired. However, if there exists a stationary point in the space of the state variables y_i the program will converge to this point in most practical cases.

4.2.1 Integration

The program applies a 4th order Runge-Kutta integration method with an adjustable stepsize h to calculate a new value for all integration variables y_i according to the following algorithm.

$$\begin{aligned} & y_i^{(0)} = y_i && \text{Initial condition} \\ & \left.\begin{aligned} y_j^{(0)} &= f_j(t^-, y_1^{(0)} \cdots y_k^{(0)}) \\ \dot{y}_i^{(0)} &= f_i(t^-, y_1^{(0)} \cdots y_k^{(0)}) \end{aligned}\right\} && \begin{aligned} &\text{computed by} \\ &\text{DYNAMØ} \end{aligned} \end{aligned} \qquad (5)$$

$$\left.\begin{aligned} & k_i^{(1)} = h \cdot a_i \cdot y_i^{(0)} \\ & y_i^{(1)} = y_i^{(0)} + 0.5 \cdot k_i^{(0)} \\ & \left.\begin{aligned} y_j^{(1)} &= f_j(t^-, y_1^{(0)} \cdots\cdots y_k^{(0)}) \\ \dot{y}_i^{(1)} &= f_i(t^-, y_1^{(0)} \cdots\cdots y_k^{(0)}) \end{aligned}\right\} \quad \begin{aligned} &\text{computed by} \\ &\text{DYNAMØ} \end{aligned} \\ & k_i^{(2)} = h \cdot a_i \cdot \dot{y}_i^{(1)} \\ & y_i^{(2)} = y_i^{(0)} + 0.5 \cdot k_i^{(2)} \\ & \left.\begin{aligned} y_j^{(2)} &= f_j(t^-, y_1^{(2)} \cdots\cdots y_k^{(2)}) \\ \dot{y}_i^{(2)} &= f_i(t^-, y_1^{(2)} \cdots\cdots y_k^{(2)}) \end{aligned}\right\} \quad \begin{aligned} &\text{computed by} \\ &\text{DYNAMO} \end{aligned} \\ & k_i^{(3)} = h \cdot a_i \cdot \dot{y}_i^{(2)} \\ & y_i^{(3)} = y_i^{(0)} + k_i^{(3)} \\ & \left.\begin{aligned} y_j^{(3)} &= f_j(t^-, y_1^{(3)} \cdots\cdots y_k^{(3)}) \\ \dot{y}_i^{(3)} &= f_i(t^-, y_1^{(3)} \cdots\cdots y_k^{(3)}) \end{aligned}\right\} \quad \begin{aligned} &\text{computed by} \\ &\text{DYNAMØ} \end{aligned} \\ & k_i^{(4)} = h \cdot a_i \cdot \dot{y}^{(3)} \\ & y_i^{(4)} = y_i^{(0)} + \frac{1}{3}\left(0.5 \cdot k_i^{(1)} + k_i^{(2)} + k_i^{(3)} + 0.5 \cdot k_i^{(4)}\right) \\ & \left.\begin{aligned} y_j^{(4)} &= f_j(t^-, y_1^{(4)} \cdots\cdots y_k^{(4)}) \\ \dot{y}_i^{(4)} &= f_i(t^-, y_1^{(4)} \cdots\cdots y_k^{(4)}) \end{aligned}\right\} \quad \begin{aligned} &\text{computed by} \\ &\text{DYNAMØ} \end{aligned} \end{aligned}\right\} \qquad (6)$$

If the convergence control, which is described below, accepts the result, the values of $y_i^{(4)}$, $y_j^{(4)}$ and $\dot{y}_i^{(4)}$ are taken as $y_i^{(0)}$, $y_j^{(0)}$ and $\dot{y}_i^{(0)}$ of the subsequent step.

4.2.2 Steady State Convergence Control

According to [6] $y_i^{(4)}$ is a good approximation of the new value of y_i provided that

$$\left| \frac{k_i^{(3)} - k_i^{(2)}}{k_i^{(2)} - k_i^{(1)}} \right| < 0,05 \; . \; . \; . \; . \; . \; 0.1 \qquad (7)$$

In this case of a steady state iteration, accuracy requirements of the transient solution are not important; only the steady state which is reached asymptotically, must be accurate. It was found that the criterien

$$\left| \frac{k_i^{(3)} - k_i^{(2)}}{k_i^{(2)} - k_i^{(1)}} \right| < 0.4 \qquad (8)$$

provides adequate performance with no tendency to instability. A criterion of the type (7) or (8) is not generally applicable. According to [6] in the case of a single differential equation the following relation holds

$$\frac{k^{(3)} - k^{(2)}}{k^{(2)} - k^{(1)}} \sim h \cdot \frac{\partial}{\partial y} f(t,y) \qquad (9)$$

In a system of differential equations, however, complications arise even in very simple cases such as

$$\dot{y}_i = 0 \qquad \text{(one of the variables is constant)}$$

or

$$\left.\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -y_1 - y_2 \end{aligned}\right) \quad \text{(damped oscillation)}$$

where (8) becomes undefined either always of in some instances. Therefore, if (8) is not satisfied for a certain variable y_l out of the variables y_i the following test is performed: If

$$\left| y_l^{(4)} - y_l^{(0)} - h \cdot a_l \cdot \dot{y}_l^{(0)} \right| < \varepsilon_s \cdot y_{Rl} \qquad (10)$$

the new value $y_l^{(4)}$ will be accepted. Here ε_s is the accuracy criterion for steady state iteration and is computed from the accuracy parameter ε specified in the input according to

$$\varepsilon_s = \mathrm{Min}(0.01 \cdot \varepsilon \;) \; 5 \cdot 10^{-6}) \qquad (11)$$

Y_{Ri} is the reference value of variable y_i, which is steadily adapted to the actual value of y_i according to the following algorithm:

$$y_{Ri(new)} = 0.9 \cdot y_{Ri\ (old)} + \mathrm{Min}\ (0.1 \cdot |y_i^{(4)}|,\ 100 \cdot y_{Ri(old)})$$

$y_{Ri(new)}$ is not allowed to exceed 10^{40} or to drop below the minimum reference value specified for this variable in the input.

4.2.3 Control of acceleration factor a_i

The acceleration factors a_i remain constant within the algorithm (6). They are initiated to unity at the beginning of the task. If neither (8) nor (10) is satisfied for $i = l$, the acceleration factor a_l of the appropriate variable y_l is reduced to 50% of its present value. If the acceleration factor a_l would drop below 2^{-24}, it will be set to 2^{-24} and the stepsize h will be reduced to 50% of its value, while all other acceleration factors a_i for $i \neq l$ are doubled, thus effectively reaching the same purpose. If the stepsize would fall below the minimum stepsize specified in the input, the task will be abnormally terminated. Otherwise the algorithm (6) would be repeated starting from the old values of all $y_i^{(0)}$, and doubling of the acceleration factor a_l will not be allowed in the next steps until a doubling request was issued at least n times (where n = number of differential equations) for this variable.

If a variable y_l was accepted according to (8) or (10), a test will be made whether the acceleration factor a_l may be doubled. A doubling request for a_l will be issued if

either

$$\left| \frac{k_l^{(3)} - k_l^{(2)}}{k_l^{(2)} - k_l^{(1)}} \right| < 0.1 \tag{12}$$

or

$$\left| y_l^{(4)} - y_l^{(0)} - h \cdot a_l \cdot \dot{y}_l^{(0)} \right| < 0.1 \cdot \varepsilon_s \cdot y_{Rl} \tag{13}$$

The doubling request will be acknowledged if there was no reduction of a_l in the preceding steps (see above <u>and</u> if $k_l^{(1)}$, $k_l^{(2)}$ and $k_l^{(3)}$ are not all equal to 0.

If the doubled acceleration a_l would exceed 2^{24}, it will not be doubled, but rather all <u>other</u> acceleration factors will be halfed and the stepsize h will be doubled. If, in this case, one of the acceleration factors a_l would drop below 2^{-24}, an error message (TIMECONSTANT OF VARIABLE l TOO SMALL) is printed and the task is terminated.

4.2.4 Special treatment of abnormal integration variables

Whenever an integration variable y_i ($1 \leqq i \leqq n$) is outside the validity range of the corresponding differential equation, the program assumes that the correct value of y_i is computed by DYNAMO according to an equation of the type $y_i = f_i(t, y_1 \ldots y_k)$. If a new value of y_i is computed in DYNAMO, it will in most cases be different from the one existing prior to the entry to DYNAMO. In order to detect this after each DYNAMO-call the program checks all integration variables y_i, whether they were modified. If a modification is found, this variable is considered as normal integration variable and is treated according to the following rules:

The modified value is accepted.

The variable is marked as being abnormal in this step of the Runge-Kutta integration.

The corresponding k_i - value (see (6)) is set to 0.

If the modification occurred in the last DYNAMO-call of the algorithm (6), the new value of $y_i^{(4)}$ is used to compute a fictitious derivative $\dot{y}_i^{(0)}$ such that the convergence criterion (10) will be satisfied, according to

$$\dot{y}_i^{(0)} = (y_i^{(4)} - y_i^{(0)})/(h \cdot a_i) \qquad (14)$$

If no modification is found between $y_i^{(r-1)}$ and $y_i^{(r)}$ for a variable y_i which had previously been marked abnormal in the same Runge-Kutta-step, the following rules apply:

A fictitious derivative $\dot{y}_i^{(0)}$ is computed according to

$$\dot{y}^{(0)} = (y_i^{(r)} - y_i^{(0)})/(h \cdot a_i) \qquad (15)$$

The variable is marked normal.

4.2.5 Steady State Accuracy Control

After every accepted Runge-Kutta-step a test is made to check whether steady state was already reached or not. The program assumes that steady state was reached if

$$\left| y_i^{(4)} - y_i^{(0)} \right| < \hat{\varepsilon}_s \cdot 0.5 \cdot \left| y_i^{(4)} + y_i^{(0)} \right| \qquad (16)$$

is satisfied for all integration variables $1 \leqq i \leqq n$, where

$$\hat{\varepsilon}_s = \text{Max}\ (0.05 \cdot \varepsilon, 10^{-5}) \qquad (17)$$

where ε is the accuracy parameter specified in the input. If the test is not satisfied, the $y_i^{(4)}$ are used as new starting values $y_i^{(0)}$ for the algorithm (6). However, for more than one differential equation steady state will not be assumed immediately. But rather, the program requests that the steady state test (16) was satisfied at least $^n/2$ times, where n is the number of differential equations.

If steady state was not reached with 20000 DYNAMO calls, an error message is printed and the task is terminated.

4.3 Transient solution

4.3.1 Integration

The initial values of the integration values for t=0 are taken either from the input specification, or if a steady state iteration was performed successfully, they are taken from the result of the steady state iteration. The same method of integration is applied as for the steady state iteration, the accuracy control, however, is different. The 4th order Runge-Kutta-algorithm applied is as follows:

$$\left.\begin{array}{ll}
\left.\begin{array}{l}
t^{(0)} = t_p \\
y_i^{(0)} = y_i(t_p)
\end{array}\right\} & \begin{array}{l}\text{initial condition}\\ \text{for } p = 0\end{array} \\[2ex]
\left.\begin{array}{l}
y_j^{(0)} = f_j(t^{(0)}, y_1^{(0)} \cdots\cdots y_k^{(0)}) \\
\dot{y}_i^{(0)} = f_i(t^{(0)}, y_1^{(0)} \cdots y_k^{(0)})
\end{array}\right\} & \begin{array}{l}\text{computed by}\\ \text{DYNAMO}\end{array}
\end{array}\right\} \qquad (18)$$

$$\left.\begin{array}{ll}
t^{(1)} = t^{(0)} + 0.5 \cdot h & \\
k_i^{(1)} = h \cdot \dot{y}_i^{(0)} & \\
y_i^{(1)} = y_i^{(0)} + 0.5 \cdot k_i^{(1)} & \\
\left.\begin{array}{l}
y_j^{(1)} = f_j(t^{(1)}, y_1^{(1)} \cdots\cdots y_k^{(1)}) \\
\dot{y}_i^{(1)} = f_i(t^{(1)}, y_1^{(1)} \cdots\cdots y_k^{(1)})
\end{array}\right\} & \begin{array}{l}\text{computed by}\\ \text{DYNAMO}\end{array}
\end{array}\right]$$

$$\left.\begin{array}{l}
t^{(2)} = t^{(0)} + 0.5\cdot h \\
k_i^{(2)} = h\cdot \dot{y}_i^{(1)} \\
y_i^{(2)} = y_i^{(0)} + 0.5\cdot k_i^{(2)} \\
\left.\begin{array}{l}
y_j^{(2)} = f_j(t^{(2)}, y_1^{(2)} \cdots y_K^{(2)}) \\
\dot{y}_i = f_i(t^{(2)}, y_1^{(2)} \cdots y_K^{(2)})
\end{array}\right\} \text{computed by DYNAMO} \\
\hdashline
t^{(3)} = t^{(0)} + h \\
k_i^{(3)} = h\cdot \dot{y}_i^{(2)} \\
y_i^{(3)} = y_i^{(0)} + k_i^{(3)} \\
\left.\begin{array}{l}
y_j^{(3)} = f_j(t^{(3)}, y_1^{(3)} \cdots y_K^{(3)}) \\
\dot{y}_i^{(3)} = f_i(t^{(3)}, y_1^{(3)} \cdots y_K^{(3)})
\end{array}\right\} \text{computed by DYNAMO} \\
\hdashline
t^{(4)} = t^{(0)} + h \\
k_i^{(4)} = h\cdot \dot{y}_i^{(4)} \\
y_i^{(4)} = y_i^{(0)} + \frac{1}{3}\left(0.5\cdot k_i^{(1)} + k_i^{(2)} + k_i^{(3)} + 0.5\cdot k_i^{(4)}\right) \\
\left.\begin{array}{l}
y_j^{(4)} = f_j(t^{(4)}, y_1^{(4)} \cdots y_K^{(4)}) \\
\dot{y}_i^{(4)} = f_i(t^{(4)}, y_1^{(4)} \cdots y_K^{(4)})
\end{array}\right\} \text{computed by DYNAMO}
\end{array}\right\} \quad (19)$$

If the accuracy control, which is described below, accepts these values, the Runge-Kutta-step number p is completed with:

$$t_{p+1} = t^{(4)} \qquad (20)$$

$$y_i(t_{p+1}) = y_i^{(4)}$$

and the values of $t^{(4)}$, $y_i^{(4)}$, $y_j^{(4)}$ and $y_i^{(4)}$ will be taken as the new values of $t^{(0)}$, $y_i^{(0)}$ $y_j^{(0)}$ and $y_i^{(0)}$ of the next step.

4.3.2 Accuracy control

In order to check the accuracy of the new values $t^{(4)}$, $y_i^{(4)}$, $y_j^{(4)}$ a second numerical integration method is applied. The results of the two methods are compared for each y_i. If the difference is small for all integration variables, the values of the Runge-Kutta-method will be accepted. The second method, which is used as a check, is based upon a Hermite extrapolation [7]. The extrapolation starts with a first order extrapolation from the values y_i at time t_p (the beginning of step p) and the derivatives y_i at this time. If this test is not satisfied, the values $y_i(t_{p-1})$, $y_i(t_{p-2})$ and $y_i(t_{p-3})$ are succesively included until the order of extrapolation matcher the order of the Runge-Kutta integration (namely 4). If $y_{i,p+1}^{H,q}$ is the value of $y_i(t_{p+1})$ evaluated by Hermite extrapolation of order q, the value will be accepted if for q = 1 or 2 or 3 or 4:

$$\left| y_{i,p+1}^{H,q} - y_i^{(4)} \right| < \varepsilon \cdot y_{Ri} \qquad (21)$$

where y_{Ri} is the reference value of variable y_i which is steadily adapted to the actual value of y_i according to:

$$y_{Ri(new)} = 0.9 \cdot y_{Ri(old)} + \mathrm{Min}(0.1 \cdot \left| y_i^{(4)} \right|, \; 100 \cdot y_{Ri(old)}) \qquad (22)$$

$y_{Ri(new)}$ is not allowed to exceed 10^{40} or to drop below the minimum reference value specified for this variable in the input.

If criterion (21) is violated by one of the integration variables, the integration step is not accepted, if the following additional test fails. This test is made in order to avoid unnecessary stepsize reductions caused by a variable which remains practically constant but does not satisfy the criterion. If

$$\left| y_i^{(4)} - y_i^{(0)} \right| < 0.1 \cdot \varepsilon \cdot \left| y_i^{(4)} + y_i^{(0)} \right| \qquad (23)$$

and $$\mathrm{sgn}\left(y_i^{(4)} - y_i^{(0)} \right) = \mathrm{sgn}\, \dot{y}_i^{(0)} \qquad (24)$$

and $$\left| y_i^{(4)} - y_i^{(0)} \right| > h \cdot \dot{y}_i^{(0)} \qquad (25)$$

then $$y_i^{(4)} = \dot{y}_i^{(0)} + h \cdot \dot{y}_i^{(0)} \qquad (26)$$

and step repetition at reduced stepsize is not requested. An increase of stepsize. which might be requested from the control of other variables will not be allowed. If (21) and any one of the conditions (23), (24) or (25) is violated, the program requests repetition of the integration step at half stepsize. The step repetition will be accounted to the particular variable and a summary of this account will be printed at the end of the dynamic problem solution. After completion of the repeated step, variables which had acceptable accuracy at a larger stepsize will not be tested again. If the accuracy control requests reduction of the stepsize to a value below the minimum stepsize specified in the input, this request will not be acknowledged and the step will not be repeated. However, if in a series of consecutive steps the number of such requests exceeds 1/3 of the number of accepted steps (in particular: if 3 such requests are issued in a sequence), the program will print an error message (STEPSIZE CONTROL STOP) and the task will be terminated.

At the beginning of the accuracy control phase the program assumes that the stepsize may be doubled in the subsequent step. If this assumption still holds, each integration variable after having been successfully tested according to (21), will be tested whether stepsize doubling would be allowed for this particular variable. The test criterion is similar to (21). The assumption that the stepsize may be doubled,will be accepted if for any extrapolation order q=1 or 2 or 3 or 4:

$$\left| y_{i,p+1}^{H,q} - y_i^{(4)} \right| < \varepsilon \cdot 0.8 \cdot 0.5^{q} \cdot y_{Ri} \qquad (27)$$

Otherwise the assumption that stepsize doubling is allowed, will be negated. If, after having tested all integration variables, the assumption is still valid, the stepsize will be doubled for the next step, unless the doubled stepsize would exceed the maximum stepsize value specified in the input.

Besides this automatic stepsize control the user may request repetition of the step at half stepsize at any time by means of a CALL REPET statement in DYNAMO.

4.4 Data buffer for Plot

If the input to the task specifies some variables to be plotted, the most recent values of these variables will be stored in a 500 word buffer after completion of a successful step. If the buffer is filled to such an extent that an additional set of plot variables would cause it to overflow, the buffer is written on an intermadiate file and cleared.

4.5 Printout control

Those variables which were specified in the input for printout and the step sequence number will be submitted to the printout routine

- as initial conditions of the problem
- upon termination, of the task

and whenever

- more than the maximum allowable number of steps is passed between two printouts
- printout is requested in DYNAMO by means of a CALL PRINT statement
- a "noticeable" extremum occured in one of those integration variables, which are to be printed.

The last one of these criteria is specified in detail as follow. A printout request is issued if

$$sgn\left(y_i(t_{p+1}) - y_i(t_p)\right) \neq sgn\left(y_i(t_p) - y_i(t_{p-1})\right) \tag{28}$$

and

$$\left|y_i(t_{p+1}) - y_i(t_p)\right| + \left|y_i(t_p) - y_i(t_{p-1})\right| > \varepsilon_{pr} \cdot y_{RPi} \tag{29}$$

where

$$\varepsilon_{pr} = \text{Min } (0.2,\ 10{*}\varepsilon) \tag{30}$$

$$y_{RPi} = \text{Min } \left(\left|y_i(t_{p+1})\right|, \left|y_i(t_p)\right|, \left|y_i(t_{p-1})\right|\right) \tag{31}$$

4.6. Termination control

The task is normally terminated, if

- the maximum number of successful steps (as specified in the input) is reached
- the maximum value of the independent variable (as specified in the input) is reached. In this case the maximum value of the independent variable is exactly reached,
- termination is requested in DYNAMO by means of a CALL TERM statement

Before conclusion of the dynamic problem solution the program requests printout of all data which might be stored internally in the printout routine, and the plot buffer is cleared on the intermediate file for later treatment. A summary of the step size reduction account and the total number of DYNAMO calls during the dynamic problem solution is printed.

5. Printout routine

The name of this routine is OUTPUT.
Prior to the steady state and dynamic problem solution the printout routine is initialized. If less than 9 variables (plus the step sequence number) are to be printed, the printout routine stores the variables in a 500 word buffer corresponding to one page. Whenever 50 lines of printout are available, or when the task is to be terminated, the contents of the buffer will be printed.
If more than 9 variables are requested for printout, they would not fit on one page at 133 characters per line. In this case the buffer is written on an intermediate file before buffer overflow occurs. Whenever 50 lines are ready or when the task is to be terminated, the buffer content is written on file and all records are reread from file in the order required to fill the buffer with one page of printout

The first page will show the step sequence number and 9 variables, the second page will show variable 10 through 19, etc.

6. Plot output routine

The name of this routine is XPLOT.
In this routine the variables which had been written on file during the dynamic problem solution, are sorted and supplied to the PLOTA routine which is special Assembler routine for plotting at the computer center of the Gesellschaft für Kernforschung, Karlsruhe. If more than 500 Steps were computed the program skips as many steps as are required to limit the total number of points to be plotted to less or equal 500. If NTOT is the number of steps and

$$NP = 1 + (NTOT-1)/500$$

every NPth step will be used as a point of the plot.

7. Dynamic control

Besides the control of accuracy and printout which is available in the CHCK input block, the user can dynamically control the execution of his problem by issuing appropriate requests in the DYNAMO routine. The following operations are available

CALL TERM-execution of the dynamic problem solution will be terminated after successful completion of the present integration step.

CALL REPET-the present integration step will be repeated at half stepsize unless the minimum allowable stepsize is reached.

CALL PRINT-the state of the problem as of the beginning of the present integration step will be submitted to the printout routine.

8. Deadtime Simulation

As additional feature an exact variable deadtime simulation facility was incorporated into DYSYS by C.Koepp*. A detailed description of the techniques applied is given in [8]. The user has the possibility to specify up to 20 equations of the type

$$y_\ell(t) = f_\ell(t-\tau_\ell(t), y_1(t-\tau_\ell(t)),\dots y_k(t-\tau_\ell(t))) \tag{32}$$

provided that $t-\tau_1(t)$ is an always increasing function of t. If there is a value t_m such that

$$t_m-\tau_\ell(t_m) > t -\tau_\ell(t) \quad for \quad 0 \leqq t_m \leqq t \tag{33}$$

the variable deadtime facility will always supply the most recent value, namely

$$y_\ell(t) = f_\ell(t_m, y_1(t_m)\dots y_k(t_m)) \qquad \text{rather than} \tag{34}$$

$$y_\ell(t) = f_\ell(t-\tau_\ell(t), y_1(t-\tau_\ell(t)),\dots y_k(t-\tau_\ell(t)))$$

The basic procedure is as follows: After each successful Runge-Kutta-step the value of $f_1(t,y_1(t)....y_k(t))$ is stored in an incore memory which is dumped onto a direct access storage device upon overflow. When retrieval of a deadtime variable is requested, a search is started for the appropriate value first in the incore memory and then on the external device. If the deadtime variable sequested lies between two Runge-Kutta-steps-which is the general case,-a first order interpolation is carried out between these steps and the result is returned to DYNAMO.

In order to use the variable deadtime facility the user must obey the following conventions:

1) In order to retrieve a deadtime variable DYNAMO must issue a Fortran function call of the type

   DEADTM (n_1,f_1,τ_1)

*) C.Koepp, Institut fuer Reaktorentwicklung, Kernforschungszentrum Karlsruhe, delegated from Euratom

where n_1 is an identification or block number for this particular deadtime relation. This is an integer.

f_1 is the real variable expression the value of which is to bl retriered at a later time

τ_1 is the real expression which determines the present value of the deadtime.

Example:

Let

(Q)	=	Q	=	a slowly vorying coolant mass flow
(ρ)	=	RHØ	=	the collant density
(A)	=	AREA	=	the cross section of a pipe
(l)	=	XL	=	the length of the pipe
(ϑ_o)	=	TØUT	=	the outlet temperature of the coolant
(ϑ_i)	=	TI	=	the inlet temperature of the coolant
		103	=	the (arbitrary) identification number of the coolant transportequation

In order to obtain the outlet temperature according to

$$\vartheta_o = \vartheta_i (t - \ell \cdot \rho \cdot A/Q) \qquad (35)$$

the following statement should appear in DYNAMØ:

TOUT = DEADTM(103,TI,XL*RHØ*AREA/Q)

As a matter of fact, relation (35) is only on approximate solution of the coolant transport problem described and is applicable only for slow coolant velocity changes. A precise formulation would have to solve the integral equation:

$$\int_0^t \frac{Q}{\rho A} dt - \int_0^{t-\tau} \frac{Q}{\rho A} dt = \ell \qquad (36)$$

and then $\vartheta_o = \vartheta_i (t - \tau)$ (37)

However, for moderate changes of the coolant velocity in the time period from t-τ to t equation (35) is a practical approximation.

2) One identification number n_1 must not occur more than once.

3) The user must make sure, that all DEADTM function calls in DYNAMØ are executed exactly once during the first execution of DYNAMØ.

4) Sufficient space must be allocated on a direct access device. See Files of DYSYS.

Files of DYSYS

	Purpose	Type of records	length of records	Remark
1	Reservation of complete input information of task input for future use	unformatted	variable, up to 4589 words	required
2	terminal output of transient problem solution	formatted	variable, up to 116 characters	required only for TCP-terminal output, blank carriage control character included
3	input copy right justified	formatted	constant, card image	required
4	Print output buffer	unformatted	constant, 501 words	required only if more than 9 variables are requested for printout
5	standard input	formatted	constant, card image	required
6	standard output	formatted	variable, up to 133 characters	required, carriage control characters included
7	plot output,written by PLOTA-routine(Assembler)	non-Fortran	constant, card image	required only if plot output is requested
10	Plot buffer and copy of DATA and MODD-information together with appropriate formats	unformatted formatted	constant, 501 words variable,up to 80 characters	required, because of the double purpose variable records of up to 501 words length must be accomodated
11	Deadtime simulation	direct access unformatted	1815 words	Space for 1600 records must be allocated on direct access device only if this feature is used

References

[1] W. Frisch, G. Wilhelmi, Dynamische Simulatoren in der Reaktorentwicklung. Ein Vergleich. Ext.Rep.8/69-1 Gesellschaft für Kernforschung, Karlsruhe (1969)

[2] H. Trauboth, Programmsystem zur Simulierung allgemeiner Regelsysteme auf einem Digitalrechner, Regelungstechnik 14, 1, p. 22,(1966)

[3] W.M.Syn, R.N.Lineberger, DSL/90 . A Digital Simulation Program for Continuous System Modeling, Proc.AFIPS Conf.28, P.165, Spartan Books, Washington D.C.,(1966)

[4] J.C.Strauss, D.C.Augustin, M.S.Fineberg, B.B. Johnson, R.N.Lineberger, F.J.Sansom, The SCi Continuous System Simulation Language (CSSL), Simulation 9, 6, p.281,(1967)

[5] CSMP-Continuous System Modeling Program, Appl.Prog. Unser's Manual, H20-0367, IBM Techn.Publ.Dep.,New-York

[6] R. Zurmühl, Praktische Mathematik für Ingenieure und Physiker, 4.Aufl.,p.413, Springer, Berlin-Göttingen-Heidelberg,(1963)

[7] A.Ralston, A First Course in Numerical Analysis,p.60, McGraw Hill, New-York (1965)

[8] C. Koepp, DAS-2-Ein dynamischer Simulator mit TOTZEIT-Gliedern für Digitalrechner, KFK 1142, EUR 3695 d.Ges. für Kernforschung, Karlsruhe (1970)

[9] H.d'Hoop, R.Monterosso, SAHYB-2: A Programme for the Solution of Differential Equations using an Analogue-Oriented Longuage, EUR 3622 e, Ispra, 1967

Appendix A

Summary of Input Specification

The input consists "comment cards" and of a number of "blocks", each one consisting of one or more "cards" (records of 80 characters length). 6 "fields" of the length 12 (e.g.12 columns or 12 character)positions followed by a field of length 8.

The "comment cards" have the following format:

$ in column 1,
79 columns of arbitrary literal content

Example:

```
$ THIS IS A COMMENT CARD
```

The "blocks" have all similar structure. They begin with a "label" card, which in most cases, is followed by "information" cards. All of these cards (except one)contain 6 "fields" of length 12 plus one field of length 8. This latter field may contain any information e.g. sequence number; it has no effect upon DYSYS. For some of the blocks default options are provided, when they are completely missing

block	card	field	type	content	default option
title	1	1	literal	INPT	not applicable
	2	1-6	"	*In column 1, title in columns 2 through 72,columns 2 through 40 will appear on plot output	
para-meters	1	1	literal	DATA	no parameters
	2	1	integer	Total number of parameters in the following cards.The parameters are stored in the DATA-labeled common in the same sequence as listed in these cards. For literal parameters only the 4 last characters will be accepted. Literals are not to be enclosed in ' ', e.g.TEXT is valid, while'TEXT' is invalid	
	3	1 2 : 6	integer, real or literal		
	4	1 : 6	. . . .		
	:	:	.		
initial con - dition	1	1	literal	INCØ	one initial condition value
	2	1	integer	Total number of initial condition values	
	3	1 : 6	real :	The initial condition values are stored in the INTVAR-labeled common in the same sequence as listed in these cards	value = 0
	4 :	1 :	:		
check	1	1	literal	CHCK	$1.\cdot 10^{-45}$
	2	1	real	Minimum stepsize	1.
		2	real	Proposed initial stepsize	$1.\cdot 10^{20}$
		3	real	Maximum stepsize	
		4	real	Accuracy parameter	0.001
		5	real	Final value of independent variables	$1.\cdot 10^{40}$
		6			-
	3	1	integer	not used Maximum number of integration steps	100
		2	integer	Maximum number of unprinted steps between printouts	1
		3 : 6		not used	
Print	1	1	literal	PRNT	
	2	1	integer	Total number printout variables specified in the following cards	2
	3	1	integer	Sequence number of the variable in the INTVAR-labeled common	0

block	card	fild	type	content	default option
Print		2	literal	Heading, e.g.Y(1), not to be enclosed in ' '	X
		3	integer	Sequence number	1
		4	literal	Heading	Y
		5	integer	Sequence number	
		6	literal	Heading	
	4 ⋮	1 ⋮	integer ⋮	- - -	
Plot	1	1	literal	PLOT	no plot
	2	1	integer	Number of curves to be plotted	
	3	1	integer	Ratio of diagram extension in x-direction to-Y-direction may be 1,2,3 or 4	
		2	integer	Sequence number of variable to be taken as abscissa	
		3 6		((not used (	
	4	1	real	maximum abscissa value	
		2	real	minimum " "	
		3	real	abscissa increment equivalent to 2 cm on the plot	
		4	real	maximum ordinate value	
		5	real	minimum ordinate value	
		6	real	ordinate increment equivalent to 2 cm on the plot	
	5	1 ⋮ 6	integer	sequence n numbers of the variables to be plotted plotted	
	6 ⋮	1 ⋮			
	More diagrams may be specified by repeating the card sequence starting at card 2				
	Last	1	integer	an integer less than 1 or greater than 30	
steady state	1	1	literal	STST	no steady state
Modify initial condition	1	1	literal	MODV	not applicable
	2	1	integer	sequence number of the variable to be modified	
		2	real	new value	
		3	integer	sequence number of the variable to be modified	
		4	real	new value	
		Repeat card 2 until field 1 or 3 contains 0			

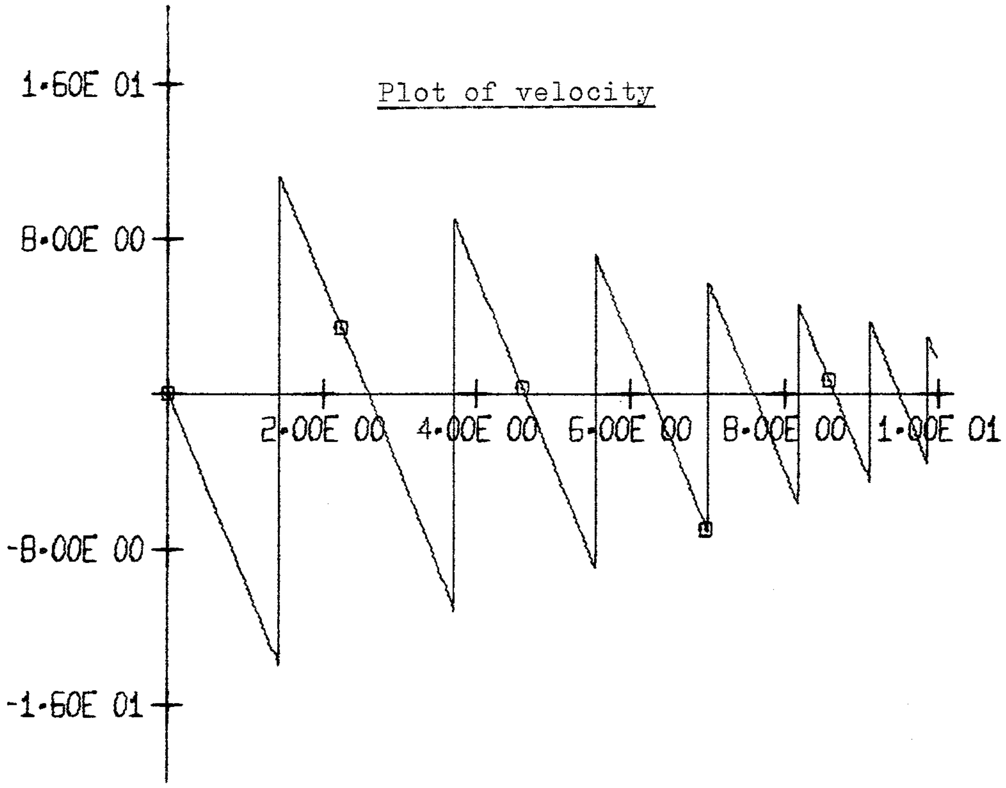
block	card	field	type	content	default option
Modify parameter	1	1	literal	MODD	
	2	1	integer	sequence number of parameter to be modified	not applicable
		2	integer real or literal	new value of the parameter	
		3	integer	sequence number of parameter to be modified	
		4	integer real or literal	new value of the parameter	
	Repeat card 2 until field 1 or 3 contains 0				
time	1	1	literal	TIME	
	2	1	integer	time(processor time) allowed for execution of next task in minutes	no limit
end of task input	1	1	literal	GOON, except for last task, where END is required	not applicable

Appendix B

Sample Problem Output

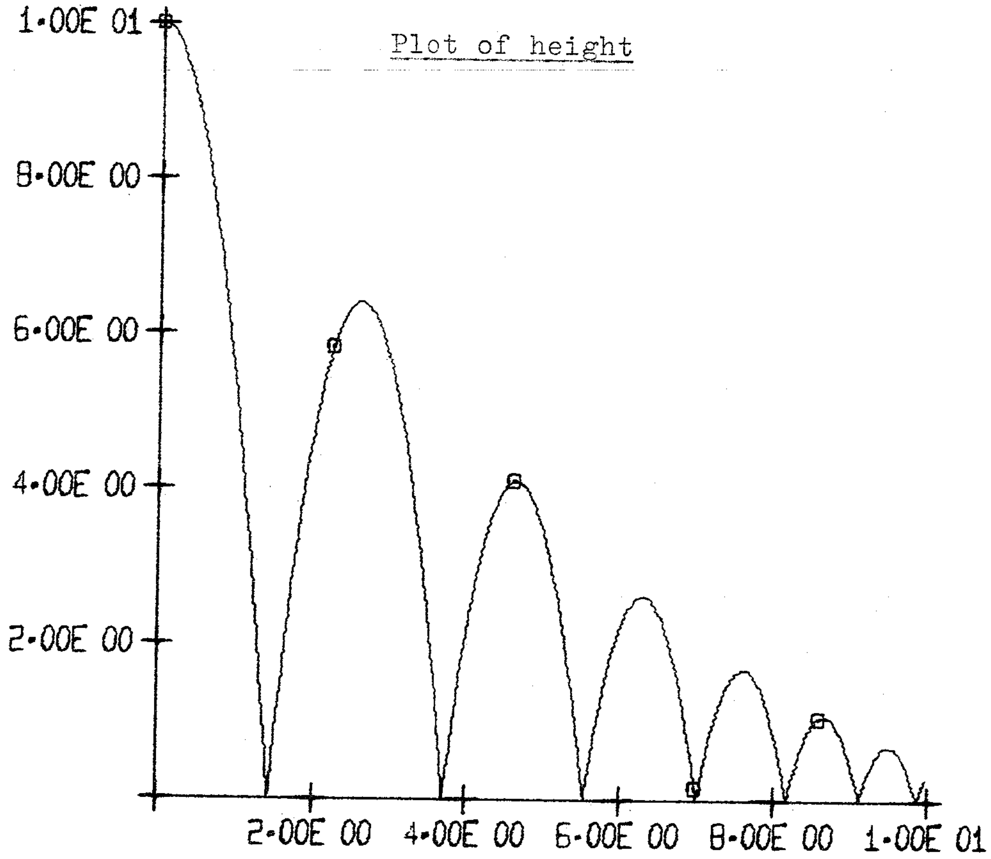
```
DDDDDDDDD       YY        YY     SSSSSSSSSS     YY        YY     SSSSSSSSSS
DDDDDDDDDD       YY      YY     SSSSSSSSSSSS     YY      YY     SSSSSSSSSSSS
DD       DD       YY    YY      SS        SS      YY    YY      SS        SS
DD        DD       YY  YY       SS                 YY  YY       SS
DD        DD        YYYY        SSS                 YYYY        SSS
DD        DD         YY          SSSSSSSSSS          YY          SSSSSSSSSS
DD        DD         YY           SSSSSSSSSS         YY           SSSSSSSSSS
DD        DD         YY                  SSS         YY                  SSS
DD        DD         YY                   SS         YY                   SS
DD       DD          YY         SS        SS         YY         SS        SS
DDDDDDDDDD           YY         SSSSSSSSSSSS         YY         SSSSSSSSSSSS
DDDDDDDDD            YY          SSSSSSSSSS          YY          SSSSSSSSSS


      DATE = 08.05.70                                  TIME = 12.44.29
```

```
INPUT CARC IMAGE
|          ||          ||          ||          ||          ||           |
INPT                                                                     00000420
* SAMPLE PRCBLEM 1                                                       00000430
INCO                                                                     00000440
2                                                                        00000450
1C.           0.                                                         00000460
DATA                                                                     00000470
2                                                                        00000480
-9.81          C.8                                                       00000490
CHCK                                                                     00000500
C.CCCC1     C.C1        C.C5        0.001        10.                     00000510
1CCCC       1C00                                                         00000520
REFV                                                                     00000530
2           2            C.1                                             00000540
C           C          C                                                 00000545
PRNT                                                                     00000550
3                                                                        00000560
C           TIME        1            HGHT          2           VELO      00000570
FLCT                                                                     00000580
1                                                                        00000590
1           C                                                            00000600
1C.         C.          2.C         2C.         -20.         8.          00000610
2                                                                        00000620
1                                                                        00000630
1            C                                                           00000640
1C.          C.          2.C         1C.           0.           2.       00000650
1                                                                        00000660
C                                                                        00000670
END                                                                      00000680
|          ||          ||          ||          ||          ||           |
```



Plot of velocity

ABB·00001 SAMPLE PROBLEM 1 08·05·70 / 12·44·29



Plot of height

ABB·00002 SAMPLE PROBLEM 1 08·05·70 / 12·44·29

```
DYSYS   RELEASE CATE = C2.09.69              DATE = 08.05.70                    TIMF = 12.44.2^

INFLT CCNTRCL

                                                                                                          0 MIN  1.14 SFC
  SAMFLE PRCBLEM 1                                                              2   TASK NR.  1
NO ERRCR HAS BEEN FOUND. THIS TASK WILL BE EXECUTED

INPUT CATA FCR CYNAMC
   1         -S.81COC            0.800000
INITIAL CONDITION
   1          1C.CCCOC          0.0
MINIMUM REFERENCE VALUES
   1         C.C                0.9999996E-01
NMAX=      1CC00  NSCHRP=       1000  DX  =   0.9999998F-02 DXMAX=  0.5000000F-01 DXMIN=  0.1000000F-04
EPS  =  0.9999999E-C3     XEND=  0.1000000E 02
PRINT
   1         C   TIME            1   HGHT            2   VELO
PLCT
KUGELSCHREIBER  AUSFUEHRUNG
         1   XMIN=   0.0             XMAX=   0.1C000C0E 02   SX=   0.2000000F 01
             YMIN=  -0.200C0CCE C2   YMAX=   0.2000000E 02   SY=   0.8000000E 01   FORMAT  Y/X=1/  1
             X=   C   Y=   2
         2   XMIN=   C.C             XMAX=   0.1C000C0E 02   SX=   0.2000000F 01
             YMIN=   0.0             YMAX=   0.1000000E 02   SY=   0.2000000E 01   FORMAT  Y/X=1/  1
             X=   C   Y=   1
STEACY STATE NOT REQUESTED
                                                                                                          0 MIN  1.96 SEC
```

```
                                                                                                   SETTF    1/   1
   SAMPLE PRCBLEM 1                                                               2

LINE   STEP          TIME          HGHT          VELO
  1  C.C           0.0           C.1000E 02  0.0
  2  C.45CCE C2    C.1428E 01    C.1846E-C3 -C.1401E 02
  3  C.46CCE C2    C.1428E 01    C.3648E-04  0.1121E 02
  4  C.118CE C3    C.3712E C1    C.1483E-C3 -0.1121E 02
  5  C.119CE C3    C.3712E C1    C.2918E-C4  0.3964E 01
  6  C.179CE C3    C.554CE C1    C.8002E-04 -0.8964E 01
  7  C.18CCE C3    C.554CE C1    C.7003E-04  C.7171E 01
  8  C.233CE C3    C.7CC2E C1    C.1321E-03 -C.7171E 01
  9  0.234CE C3    C.7C02E 01    0.1867E-04  0.5737E 01
 1C  C.279CE C3    C.8172E C1    C.1657E-04 -0.5737E 01
 11  C.28CCE C3    C.8172E C1    C.8964E-C4  0.4589E 01
 12  C.302CE C3    C.8651E 01    C.1073E C1 -0.1193E 00
 13  C.32CCE C3    C.9107E C1    C.8252E-C4 -0.4589E 01
 14  C.321CE C3    C.91C7E C1    C.1195E-C4  0.3671E 01
 15  C.34CCE C3    C.9467E C1    C.6860E 0C  C.1400E 00
 16  C.36CCE C3    C.9856E C1    C.3736E-04 -0.3671E 01
 17  C.361CE C3    C.9856E C1    C.9561E-C5  0.2937E 01
 18  0.374CE C3    C.9976E C1    C.2818E 0C  0.1760E 01
 19  C.375CE C3    C.10C0E C2    C.3218E 00  0.1521E 01
                                                                                                   0 MIN  6.37 SEC


NUMBER OF DYNAMC-CALLS=    1906
NUMBER OF STEPSIZE REDUCTIONS/VARIABLE
*  128/  1 *

PLOT  TERMINATED
                                                                                                   0 MIN  7.15 SEC
```

```
                                                                                                       0 MIN  0.78 SEC
INFUT DATA RIGHT JUSTIFIEC,FIELD LENGTH=12
        INPT                                                                   1
  SAMFLE PROBLEM 1                                                             2
        INCO                                                                   3
           2                                                                   4
         10.          0.                                                       5
        DATA                                                                   6
           2                                                                   7
       -9.81         C.8                                                       8
        CHCK                                                                   9
     C.CCOC1        C.01        0.05       0.001         10.                  10
       10CCC        1C00                                                      11
        REFV                                                                  12
           2           2         0.1                                          13
           CC          0                                                      14
        PRNT                                                                  15
           3                                                                  16
           C        TIME           1        HGHT           2        VELO      17
        PLOT                                                                  18
           1                                                                  19
           1           0                                                      20
         10.          0.         2.0         20.        -20.          8.      21
           2                                                                  22
           1                                                                  23
           1           0                                                      24
         1C.          0.         2.C         10.          0.          2.      25
           1                                                                  26
           C                                                                  27
         ENC                                                                  28
```

Appendix C

DYSYS calls for a few routines which are not included in the source deck. These routines are described in the following.

1) Subroutine DATUM
   The call is CALL DATUM (DDAT, DZEIT)
   with REAL*8 DDAT, DZEIT
   As a result of this subroutine call DDAT will contain the date and DZEIT will contain the time of the run as 8 characters which are printed each with format A8 on the first pages of the output.

2) Subroutine FSPIE
   The call is CALL FSPIE
   This routine merely improves the error diagnostic printout in case of a severe error occurring during program execution.

3) Function ZEIT

```
The call is C = ZEIT(0.)
            B = ZEIT(C)
with REAL*4 A,B,C
```

   As a result of the first function call the time accounting routine is initialized. Subsequent calls supply (in B) the computing time elapsed since time C in seconds.

4) Subroutine PLOTA
   The call is CALL PLOTA(XWERT,YWERT,N,3,NP,1,1,LP,INDZ,XMAX, XMIN,SX,YMAX,YMIN,SY,KOMENT,KBILD,0,0,0)

The arguments have the following meoning:

XWERT = array containing the abscissa values

YWERT = array containing the ordinate values

N = number of points

NP = integer, characterizing the symbol (point, circle etc) which will be used to mark every LPth point of the curve on the plot

LP = integer (LP=(N+4)/5) computed such that each curve will be marked with 5 points

INDZ = integer, which indicates the format of the plot for the first curve and is set equal to 0 for each subsequent curve on the same plot

XMAX = maximum value of the abscissa X

XMIN = minimum value of the abscissa X

SX = increment in X corresponding to 1/100 inch

YMAX = maximum value of the ordinate Y

YMIN = minimum value of the ordinate Y

SY = increment in Y corresponding to 1/100 inch

KOMENT = Title of the plot, consisting of the first 40 characters of the title card of this task followed by data and time

KBILD = figure number within this task

5) Subroutine ERRSET

The call is CALL ERRSET $(a_1, a_2, a_3, a_4, a_5)$

The effect of the two ERRSET calls in the subroutine OUTPUT is the following:

Before an attempt is made to write a record on file 2 (which may be directed to the TCP-Terminal) the standard error handling routines are modified by the first ERRSET call in a manner, that a missing DD-card for file 2 does not lead to a program interrupt. The second ERRSET call restores the standard setting of the error handling routines. This feature eliminates the need for specification of file 2 for users who do not want to use DYSYS from one of the terminals.