View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Online Research Database In Technology

DTU

DTU Library

# DYSIM - A Modular Simulation System for Continuous Dynamic Processes

**Noauthor, Risø; Forskningscenter Risø, Roskilde; Forskningscenter Risø, Roskilde**

*Publication date:*
1986

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

Link back to DTU Orbit

*Citation (APA):*
Christensen, P. L. C., Kofoed, J. E., & Larsen, N. (1986). DYSIM - A Modular Simulation System for Continuous Dynamic Processes. Roskilde: Risø National Laboratory. Risø-M, No. 2607

**General rights**
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

DK8700004

Risø-M-2607

RISØ

Risø-M-2607

# DYSIM - A Modular Simulation System for Continuous Dynamic Processes

P. la Cour Christensen, Jan E. Kofoed and Niels Larsen

Risø National Laboratory, DK-4000 Roskilde, Denmark
September 1986

RISØ-M-2607

# DYSIM - A MODULAR SIMULATION SYSTEM FOR CONTINUOUS DYNAMIC PROCESSES

P. la Cour Christensen, Jan E. Kofoed
and Niels Larsen

Abstract. The report describes a revised version of a simulation system for continuous processes, DYSIM. In relation to the previous version, which was developed in 1981, the main changes are conversion to Fortran 77 and introduction of a modular structure. The latter feature gives the user a possibility for decomposing the model in modules corresponding to well delimited physical units, a feature which gives a better survey of the model. Furthermore, two new integration routines are included in addition to the single one used before.

September 1986
Risø National Laboratory, DK-4000 Roskilde, Denmark

ISBN 87-550-1265-5

ISSN 0418-6435

Grafisk Service Risø 1986

## CONTENTS

Page

APPENDICES

## 1. INTRODUCTION

Risø's work with simulation of dynamic systems by means of digital computers was started in 1975 with a simulation program DYSYS which was bought from Kernforschungszentrum Karlsruhe (E.G. Schlechtendahl, 1970). In the following five years it was used for simulation of nuclear power plants. Gradually several minor modifications in DYSYS were made to adapt the program to our computer and our special demands, but major modifications were difficult to introduce in a reliable way due to the program structure. So, in 1980 a complete new version was written with some new features. It was given the name DYSIM and described in the report RISØ-M-2271 (CHRISTENSEN, P. la Cour, 1981).

DYSIM was written according to the same conventions as DYSYS, so old input data files still could be used with minor modifications. Since then DYSIM has been improved by addition of new features during the years 1981-1985, and in the end of 1985 a major change was made. First the code was changed from Fortran 66 to Fortran 77, and then a modular structure for models was introduced for data specification in the input file.

The modular structure was developed and used in the years 1983-1985. It makes it possible to split up a system into individual modules describing separate physical units, to program and check these modules one by one, and then combine the modules to a system by a connecting routine which describes the input - output relations between modules. Furthermore, symbolic names were introduced for variables and parameters instead of array numbers in the input file.

DYSIM is in the present version designed to run on a Burroughs B7800 computer with batch job control and not for interactive use. It means that the whole job must be specified in the input file. This has been a natural solution on the B7800 computer, where interactive jobs are limited in CPU-time to 60 s and most simulation jobs take more time. DYSIM reads the input file and a corresponding list file, with definition of variable names and parameter names, sets up initial values and parameters for the job, and controls the job excution with accuracy control of integration and with output administration.

For easy set-up of modules a new feature using submodules has been developed. Well defined physical components, such as heat exchangers, can be preprogrammed and stored in a submodule library. Such submodules can then be incorporated in a module by macro calls which specify the connections between the global variables of the module and the local variables of the submodules. Modules with macro calls must be passed through a precompiler, called precompiler 1, which converts to Fortran 77 code. Furthermore, the precompiler is used to set-up the head of the module subroutine under control of commands that specify the module name and variables.

A subroutine, called a connecting routine, is needed to assemble modules into a complete system model and establish the connection to DYSIM. This subroutine can be written manually, but the process can be made easier and more reliable by another precompiler, called precompiler 2, because the statements in the connecting routine must follow certain rules strictly. So, they can easily be given in compact form by commands to precompiler 2. Development of the submodule concept and precompilers is carried out as part of a Ph.D. degree study and will be described in a separate report later.

Finally it should be mentioned that some auxiliary programs are used in connection with DYSIM for preparation of input and utilization of output. The list file and input file must be written according to some conventions. It can be done by a general editor, but the user can get valuable help from two programs, a list file editor and an input file editor. The output from DYSIM is for steady state calculations always in form of printer listings, but by transient calculations one can get both tables and/or a data file. This data file can be used to generate graphic representation on a screen and hard copies on a plotter using a plotter program. For a steady state situation DYSIM can be used to calculate the Jacobi matrix for the system of differential equations and give the result on a data file; and another auxiliary program can be run afterwards to interpret the matrix with calculation of eigenvalues and step size limits for Runge-Kutta integration.

Lately DYSIM has been transferred to a personal computer with some changes which make it more suitable for the operator to control the simulation. The present PC-version is described in section 11.

A diagram in Fig. 1 shows how DYSIM communicates with the input file and list file in the initial phase and with the model via the connecting routine during the main phase with steady state or transient calculations. Some output files are generated along with the calculations, others in the terminal phase.

## 2. PROBLEM FORMULATION AND MODEL STRUCTURE

Simulation of a process involves several phases:

- Investigation of the process and formulation of a physical-mathematical model and data retrieval

- Separation of the system into modules and establishment of the input-output structure

- Writing the simulation codes for modules consisting of Fortran 77 and/or macro statements and commands for pre-compiler 1

- Testing of single modules

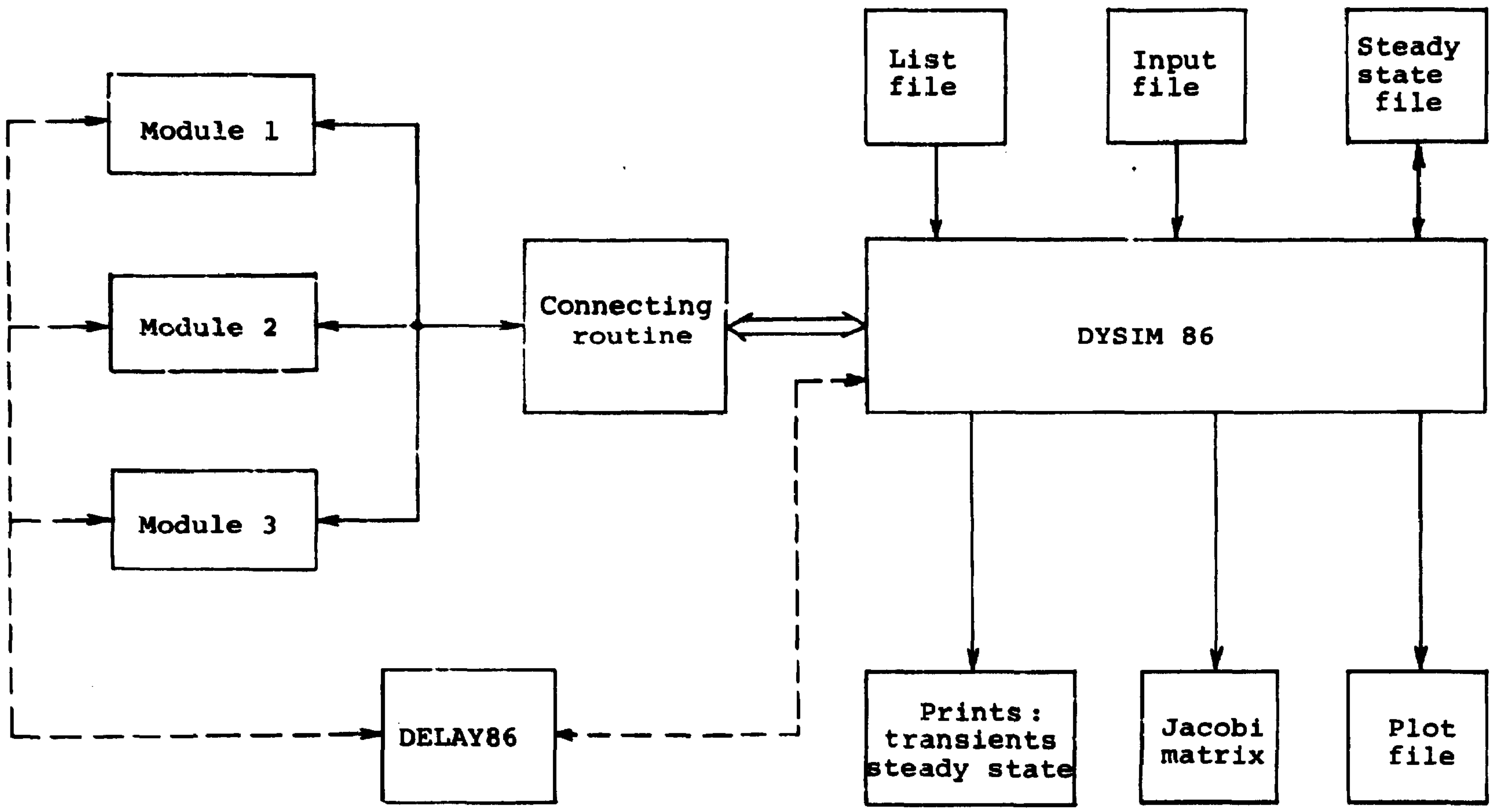

Fig. 1. DYSIM simulation system scematics.

- Creation of the connecting routine possibly with the use of precompiler 2

- Testing of the whole system including steady state and transient calculations

- And finally documentation

The first and the last point are very important and the most time consuming.

A flow diagram of the programming procedure using precompiler 1 and 2 is shown in Fig. 2. Each source module containing macro calls and precompiler commands is converted to Fortran 77 modules. Submodules from the library are inserted by precompiler 2; precompiler 1 only gives lists of submodules called from the modules. On the B7800 computer all Fortran 77 program units are compiled in one run. On the PC individual files are compiled separately and then linked together by the linker program.

Some processes are by nature discrete which means that they can be described by point models with ordinary differential equations; others are distributed in space which results in partial differential equations. The latter type is for DYSIM transferred into discrete systems by division of the space axis in small segments and with description of the small nodes by means of ordinary differential equations. So the resulting mathematical model is for all modules a mixture of ordinary nonlinear differential equations and algebraic equations for auxiliary variables.

The programmer must look carefully at the order in which the equations are written so that auxiliary variables are calculated before they are used, if possible. It is, however, not always possible because the state derivatives and auxiliary variables in principle may be functions of all state variables, derivatives, auxiliary variables and input variables. So in some cases old values for auxiliary variables from the last time step will be used. Therefore, the Fortran SAVE statement must be used in all program units to save local variables from one call

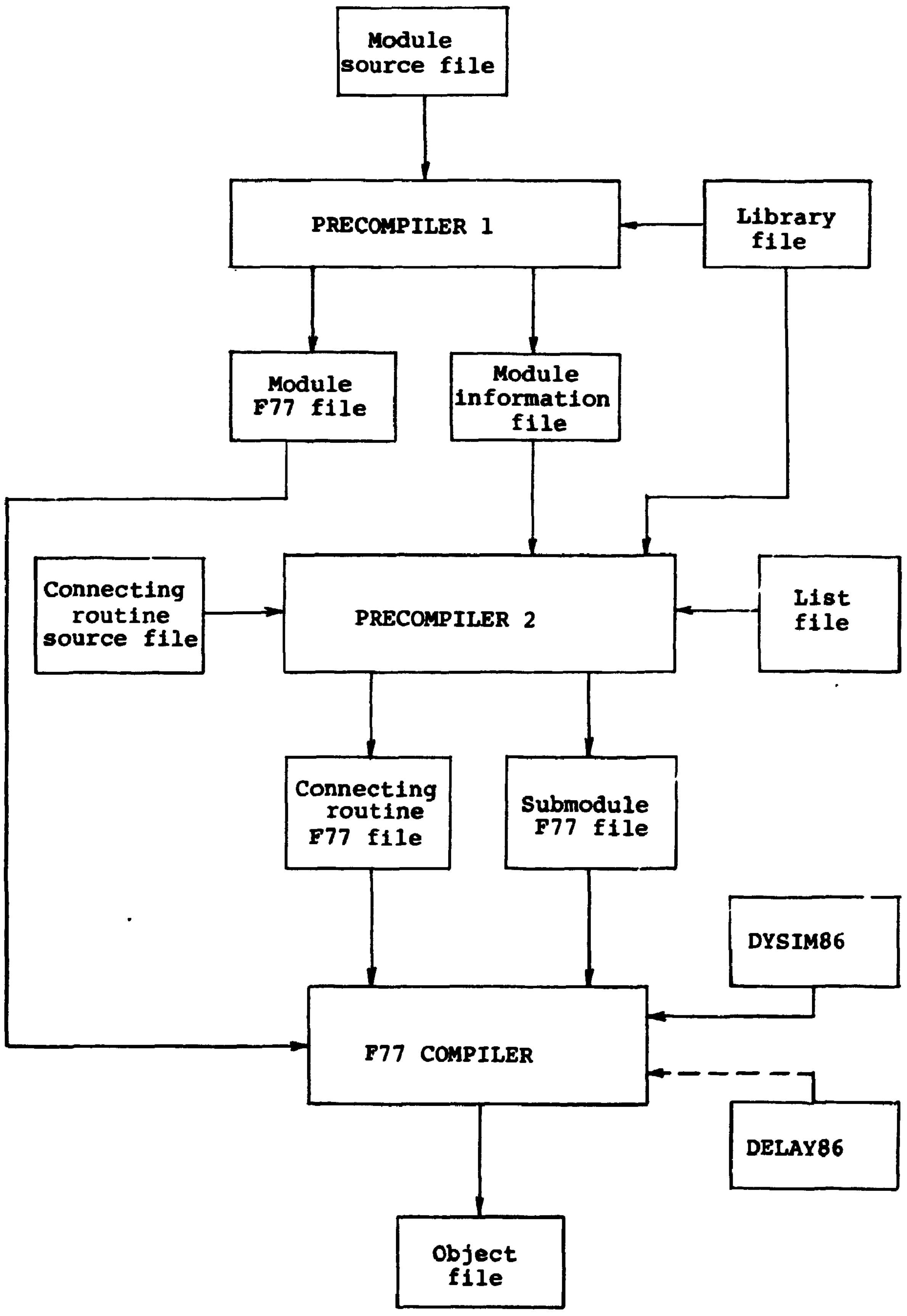

Fig. 2. The precompiler system.

to another. Furthermore, the programmer is responsible for algebraic loops which are not allowed to have a loop gain greater than 1. Otherwise the solution of the system of equations becomes unstable. For practical purposes the gain must be somewhat smaller.

Due to the complicated system of equations which often is obtained no automatic sorting of equations has been attempted. Instead, much attention has been devoted to establishing a module/submodule structure to make program development easier and more reliable.

A module is a program unit that describes a well delimited physical unit. It is described by a set of differential equations which by integration gives the state variables. The auxiliary variables are divided into local variables and algebraic output variables. The state of the module is determined by a set of input parameters, normally constant during a calculation, and a set of input variables, normally connected to other modules or generated by the connecting routine. So a module is from the outside characterized by 5 vectors: State variables, derivatives, input variables, parameters, and algebraic variables stored in one Fortran COMMON block.

Appendix A gives an example of a small module. It shows how such a subroutine can be built with the COMMON block, CRK, containing the external variables, some internal data specifications and the program body which calculates derivatives. No algebraic variables are used here.

DYSIM integrates all state variables in the whole system at the same time and works with one long vector for all state variables, one for derivatives, one for parameters and one for algebraic variables.

Each vector is stored in one Fortran COMMON block as

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

The index numbers NDE, NAE, NDA are the total number of state variables, algebraic variables and parameters respectively.

The integration is carried out with one of three possible choices among integration routines: a fourth order Runge-Kutta routine, a second order Runge-Kutta routine called HEUN, and a routine selected from the ODEPACK collection of differential equation solvers.

The system model consists of the modules combined by the connecting routine, which is a Fortran subroutine with the fixed name CON and the parameter NR, where NR is a substep number for the integration routine. The connecting routine takes care of input-output connections between modules and the connection between DYSIM and the model. The connections are established by transmission of variables between COMMON blocks.

Appendix B gives an example of a connecting routine for a medium sized system with 6 modules as it looks when it is written by hand. It shows in detail how variables are transferred between program units and how the modules are called. Note the four COMMON blocks for DYSIM each with up to 6 arrays and the six COMMON blocks for the six modules each with up to five arrays. Also note that the order of modules in the four DYSIM COMMON blocks must be the same as used in the input file SYST command described in section 4. The entry point ALVAR is used to transfer parameters from DYSIM to the modules and algebraic variables backwards and forwards controlled by the parameter NO. ALVAR is called from DYSIM when necessary.

Appendix B can be used as a model for setting up connecting routines. The precompiler 2 makes this much easier. It releases the user for the troublesome work with conversion of symbolic names to array indices for the input-output connections. Besides it has commands for definition of the system by modules and automatic setting up of COMMON blocks and statements for transmission of variables. It uses the information contained in the list file with all symbolic names.

By call of the module subroutines the integration substep number NR is transferred to the modules together with the time as subroutine parameters.

For steady state calculations DYSIM sets the time to a negative value and this can be utilized by the programmer for insertion of special steady state sections in the modules in order to speed up the calculations. The substep number has the value 0 at the very first call; afterwards it takes values from 1 to 4 dependent of the integration routine. It can be used to control calculation of internal parameters in the modules; e.g. constant parameters can be calculated for NR = 0 and slowly changing parameters for NR = 1.

Initial values are given in the input file for both state variables and algebraic variables. The reason for using initial values for the latter type is that they sometimes must be used as input to one module before they can be calculated in another one, e.g. when two modules form a loop of algebraic variables.

The concept with algebraic variables and the way they are used sometimes introduces small time delays by input-output connections when a variable is used in one module before it is calculated in another one. It is an inevitable consequence of the module structure. Therefore the user must be careful to specify the best possible order of module calls.

## 3. THE LIST FILE

The list file concept is a result of the module structure development. For the first models which we developed by use of the module structure it was found necessary to write a list of variables by hand. Later the list was written on a disk file, and now the list format has been standardized so information can be extracted by a subroutine. The list is used by DYSIM for interpretation of the input file and by precompiler 2 for generation of the connecting routine. It is also used by the input file editor to set up an input file.

One single list file is used for a system model with a section for each module that may be included in the model, but not all modules need to be used. So the same list file can be used for different system configurations.

Each module has a section with a head that gives the module name which is limited to 3 characters. Then four groups with information about variables follow. The four groups are state variables, algebraic variables, input variables and parameters in that order. One line for each variable gives the symbolic name, an index for the COMMON block vectors, a physical dimension and a text describing the variable. Each variable may be a simple variable or a onedimentional array. Names may be up to 6 characters long, and the same local name may be used in different modules. Simple array elements are referenced by the array name followed by the index number limited to two digits e.g. TEMP17 for element TEMP(17). DYSIM uses composite names consisting of the local name followed by a "." and the module name, it means that the global name is limited to 12 characters.

Precompiler 2 also uses the list file. It takes the module names as subroutine names and constructs COMMON block names and vector names by addition of characters to the module name.

Precompiler 1 does not use the list file because a normal procedure will be to set up the modules before the list file. Here the module names and variable names are given in compiler commands, and the precompiler generates names for the subroutines and COMMON blocks in the same way as precompiler 2.

Appendix C gives as an example the list file for the model, for which the connecting routine is shown in appendix B. Parameters will nearly always be present in the connecting routine as they are used to initiate transients via input variables to the modules. Sometimes it can be an advantage to introduce an integration variable for generation of inputs. For that purpose the list file structure must give a possibility for definition of state variables for the connecting routine. Therefore it was chosen to standardize and describe the connecting routine exactly as a module.

The format of the list file is rather fixed, only a little amount of freedom is allowed. Therefore it is recommended to use the list file editor which will create the list with the correct format.

## 4. THE INPUT FILE

The input file is used to specify the system configuration, the initial state and the job execution in detail. It is read by the subroutine INPUT which checks the input file for errors and gives error messages in a clear text. If an error is found DYSIM tries to continue until the end of the file, but some errors are prohibitive for continuation. An error will always stop DYSIM after reading the input file. While it is being read the corresponding list file must be ready as it is used in connection with the input file.

The input file consists of records with comments, commands or data. Comment lines begin with an "&" as the first character and may be inserted everywhere in the file. Command lines begin with a "*" as the first character and are always followed by 4 characters; the rest of the record is ignored so it cannot contain information. Some commands stand alone, some must be followed by data records, and others can have data records optionally. The B7800 version of DYSIM uses 18 commands and the PC-version one more. Some commands are obligatory while others are optional. The order of the commands is not important with a few exceptions which are mentioned below. A data record consists of maximum 72 characters, in most cases divided into 6 fields of 12 characters each. The data are either integers, real numbers or text strings. The position inside the fields is unimportant as DYSIM performs a shift of significant characters to the left or right according to the interpretation as text strings or numbers. Below the commands will be described one by one with explanation of the data records following the commands and the effect in DYSIM during job execution.

### *INPT

An obligatory command followed by one record with a text string with up to 72 characters which is used to identify the job. It is used as heading in prints and plots.

### *INPD

An optional command without data. It specifies input documentation on prints in a more clear form than just printing of the input file which always is used.

### *SYST

An obligatory command used to specify the system. It is followed by two or more data records. The first must give the system name, SYNA, with up to 4 characters and the number of modules including the connecting routine, which by INPUT is treated as a module. The next and following records give the module names with 6 per record. The last one must be the connecting routine with the fixed name CON.

Ater reading the system configuration INPUT reads the list file SYNA/LIST, where SYNA is the system name. It checks the list file according to the module names specified and reads information about state and algebraic variables and parameters which is used for interpretation of data records for following commands.

The order of the module names gives the order in which state variables are stored in COMMON INTVAR independent of the order in the list file. It gives a possibility to change the configuration of the system's Jacobi matrix, a feature which may be useful for some integration methods.

Remember that the order of module vectors in DYSIM's four COMMON blocks written in the connecting routine must be the same as used here.

The SYST command must be one of the first commands; to make it simple place it just after the INPT and INPD commands.

<u>*INCO</u>

An obligatory command followed by groups of data records, one for each module. Each group starts with a record with the module name followed by the number of state variables and algebraic variables in agreement with the numbers given in the list file. Then follow two subgroups of numbers which give initial values for state variables and for algebraic variables, with 6 numbers per record in the same order as used in the list file. The number format is free, it is read by Fortran's G12.0 format.

If the connecting routine CON has state or algebraic variables it must appear as the last module, otherwise it can be omitted. The sequence of other modules is free, independent of the sequence of module names in the SYST block.

Specification of initial values for the first computation is a troublesome work. It is recommended to use the input file editor. Reasonable values for the state variables should be used, while the values for algebraic variable in most cases can be set to zero. After the first successful steady state run DYSIM gives an output file with new and better values which can be used to substitute the old ones.

The INCO command must come before the REFV and STIC commands mentioned later.

***DATA**

An obligatory command followed by a number of data records terminated by the codeword DATA.END. It is used to specify parameters to the modules. Each record can contain up to 3 parameters written in two fields with the full name followed by "=" in the first field and the parameter value in the next field e.g. as NAME1.M03= 713.5. The character "=" may be placed everywhere in the empty positions after the name in the first field or as the first character in the next field (used when the full name is 12 characters long. The number format is free as for the INCO command. It is allowed to use empty fields (2·12 blank characters) so records need not contain 3 parameters for each record. The order of parameters is completely free and a mixing of parameters from different modules is allowed. It is not necessary to give values for all parameters; parameters not specified have the default value 0.

***CHCK**

An obligatory command followed by one or two data records. The first record has 5 or 6 specifications which are

| | |
|---|---|
| DTMIN: | Minimum value of the integration step |
| DT : | Initial - - - - - |
| DTMAX: | Maximum - - - - - |
| EPSI : | An integration accuracy control parameter |
| TMAX : | Maximum value of the true transient time |
| INTE : | Name of an integration routine |

The following integration routines can be used: RUNGE, HEUN, or ODEPAK. They are discussed in section 6. If no name is given (blank field) RUNGE is used. If ODEPAK is used one record more may be specified with 6 integers which are discussed also in section 6.

***TIME**

An obligatory command in the B7800 version followed by one record with one number, TTOT, that gives the maximum value of processor time in seconds. TTOT may be written in free format. If a transient is stopped by the CPU-time a special message is given after the output tables. For steady state calculations the computing time is given by TTOT. In the PC-version the command has no effect and may be omitted.

***REFV**

An obligatory command when EPSI > 0 in the CHCK command. REFV is used to give reference values for the integration accuracy control which is discussed in section 6. The command is followed by a number of data records terminated by a record with the codeword REFV.END. The data records may be omitted, but the terminating codeword REFV.END must always be present. Each record has two state variable names in the first two fields and a reference value in the third field. The names must be full names and the number format is free. The number is inserted as reference values for all state variables from the first to the second name both included. Note that the order of the system state variables are given by the SYST command.

When the REFV command is entered default values are inserted for all state variables before the data records are read. The default values are the initial values, therefore the INCO command must come before the REFV command. If an initial value is 0 the number 1 is used instead to avoid trouble with the accuracy control. The reference values given in the data records substitute the default values. The accuracy control works with adjustable reference values and the values loaded in the initial phase are used both as initial values and as minimum values.

***PRNT**

An obligatory command followed by a number of data records terminated by the codeword PRNT.END. The first record gives 5 numbers that control table printout and/or storage of data in a plot file. The numbers are

```
PDTO: Time  intervals  for  printout  from  time  0  to  POT1
POT1
PDT1: -     -          -    -         -     -     POT1 to POT2
POT2
PDT2: -     -          -    -         -     -     POT2 and upwards
```

The same time intervals are used for plot data as for printout. Table printing can be suppressed by placing the number "1" in the last field after the five numbers above. This option is only valid for the B7800 version; the PC-version does not need that option because printing is directed to a file and printing on paper must be initiated by the user afterwards.

The following records contain up to 6 variable names (state or algebraic variables) per record, given by full names; they specify those variables that are printed and/or stored for plotting later. Empty fields are accepted and the order of variables are completely free. The integration step number and the time are inserted automatically as the first two variables.

***STST**

An optional command that specifies a steady state calculation. One or two data records may follow optionally in free order. One gives a file name for a steady state file with state and algebraic variables written in the same format as used in the INCO command, so a direct substitution is possible. The default file name is SYNA/IC where SYNA is the system name. Another record can be used to specify the number of integration steps between printout of state variables; the default value is 100. Two extra numbers may be given optionally in the last mentioned record for the PC-version as discussed in section 11.

***STIC**

An optional command that can be used to ask DYSIM to take a set of initial values from a file instead of those given in the INCO block. A data record with a file name can follow the command. The default file name is SYNA/IC as for *STST. As the IC-data given by the STIC command are used to substitute those given by the INCO command *STIC must come after *INCO, which still must be present with correct format. *STIC must come before *REPV; it is recommended to place it just after the INCO block.

***PLOT**

An optional command used to specify storage of data in a file for plotting later. The time intervals and the variables are specified by the PRNT command. The file name may be specified in an optional record. The default name is SYNA/OUTPUT where SYNA is the system name. Actually the file is the same as a file used for temporary collection of print data during integration. When PLOT is specified the file becomes permanent and is supplied with information about the stored data as described in section 7.

***DELY**

An optional command without data records used to tell DYSIM that the model contains pure time delay variables which are calculated by the delay function DELAY. The effect is that DYSIM during job execution calls the delay routine at every time step for the purpose of administration. The file containing the delay routine must be included in the code by the user.

***JACO**

An optional command that actuates a numerical calculation of the Jacobi matrix. The result is written on a data file with the default name SYNA/JACO, where SYNA is the system name. One data record may be used optionally, it can contain two numbers in free format. The first number is an iteration number for calculation of the derivatives for each pertubation of a state variable; it must have a value between 1 and 25. The second number specifies the relative pertubation of state variables; only values between 1.E-3 and 1.E-10 are accepted, else the value 1.E-5 is used. This number may be omitted, then the default value 1.E-5 is used. By the calculation it is assumed that the model is in a steady state.

<u>*DUMP</u>
An optional command without data records. It gives a dump at termination time of all variables needed for a later restart. The disk file SYNA/DUMP is created as a permanent file in any case, while the file SYNA/DELAYDUMP is created only when time delays are used.

<u>*REST</u>
An optional command without data records used to restart a calculation after a DUMP and continue from the dump time. The start conditions are loaded from the files SYNA/DUMP and SYNA/DELAYDUMP as needed.

<u>*TEST</u>
An optional command without data records used to run a test of the input file and list file only.

<u>*REAL</u>
An optional command with one optional data record. It can only be used for the PC-version. It gives a real time simulation if possible with a fixed time step given by the initial time step in the CHCK block. During the calculation variables in the PRNT block are transferred to either a disk file or the output port (RS-232-C) as described in section 11.

The destination is given by a filename in the data record. If the filename is COM1 the destination is the output port, else the file named. The default file name is COM1.

<u>*ENDE</u>
An obligatory command without data records indicating the end of the file.

To ensure a correct syntax by creation of input files it is recommended to use the input file editor.

An example of an input file is given in appendix D for the same model as the connecting routine in appendix B.

## 5. STEADY STATE CALCULATIONS

The steady state is calculated by iteration using the integration routine HEUN if it is specified in the CHCK block, otherwise RUNGE is used if either no specification is given or if RUNGE or ODEPAK is specified because the latter routine is not suitable for steady state search. The time step is constant equal to the initial step DT in the CHCK block. No accuracy or stability check is used; it is the users responsibility to select a reasonable value for DT.

For the B7800 version the calculation will continue until the processor time reaches the value given in the TIME block. For the PC-version the calculation is controlled in a different way. At termination time the state and algebraic variables are written in the steady state output file (SYNA/IC) grouped in modules ready to substitute the data records in the INCO block. By repeated steady state calculations the values in the steady state file can be used as initial values for the next calculation with the STIC command. When a satisfactory state is found it is recommended to transfer the steady state file to the INCO block.

A steady state calculation is characterized by a negative time value which can be detected and utilized in the modules for acceleration of steady state search in two ways. Sometimes it is possible to use direct algebraic calculations of state variables when the steady state is characterized by given working conditions, and such calculations are allowed in DYSIM. Another possibility is to reduce large time constants giving a faster convergence. Both methods must be used with care as it is possible to change a stable system to an unstable one in this way.

During the program run state variables are printed at regular intervals, which can be given in the STST block. If no value is given the printing is made for every 100 steps. At the end of the calculation both state variables, algebraic variables and derivatives are printed.

## 6. TRANSIENT CALCULATIONS

By transient calculations three integration methods can be used as mentioned in the description of the CHCK command: HEUN, RUNGE, and ODEPAK. The two first are explicit Runge-Kutta methods while the last one gives a choice between Adam's methods and backwards difference methods, all of implicit type.

Implicit methods have a much better stability than explicit methods; therefore implicit methods allow larger steps when the fast transients have died out in a stiff system, but each step require much more calculation work. According to our experience with a limited number of models, but all stiff systems, it is not possible to save processor time with the implicit methods, and moreover these methods are more difficult to use because integration parameters must be chosen very carefully which requires much experience. So it is recommended in general to use HEUN or RUNGE. Heun is about 35% faster than RUNGE, but RUNGE is more precise than HEUN.

The two types of methods are discussed in the following two subsections.

## 6.1. HEUN and RUNGE

The problem to be solved is given by a set of differential equations as

$$\dot{y} = f(t,y,x,a), \qquad y(0)=y_o$$

where y is the state vector, x is an input vector, a an algebraic vector and t is the time. In the literature the general notation is

$$\dot{y} = f(t,y)$$

where the dependence of x and a is embedded in the dependence of time t.

Both HEUN and RUNGE are Runge-Kutta methods, HEUN of second order and RUNGE of fourth order. The formulas are

HEUN : $y_{n+1} = y_n+(k_1+k_2)/2$

where $k_1 = f(t_n,y_n)\Delta t$

$k_2 = y(t_n+\Delta t,y_n+k_1)\Delta t$

where $\Delta t$ is the time step

RUNGE: $y_{n+1} = y_n+(k_1+2k_2+2k_3+k_4)/6$

where $k_1 = f(t_n,y_n)\Delta t$

$k_2 = f(t_n+\Delta t/2,y_n+k_1/2)\Delta t$

$k_3 = f(t_n+\Delta t/2,y_n+k_2/2)\Delta t$

$k_4 = f(t_n+\Delta t,y_n+k_3)\Delta t$

The accuracy control for HEUN uses the difference between $k_1$ and $k_2$ and for RUNGE the difference between $k_2$ and $k_3$

HEUN : $DEV = ABS(k_1-k_2)$

RUNGE: $DEV = ABS(k_2-k_3)$

The difference DEV is compared with a reference deviation

$REF = EPSI \cdot RF$

where EPSI is given in the CHCK block and RF is a dynamic reference value for the particular variable. It is calculated in each step by the formula

$$RF = MAX((ABS(y_n)+99 \cdot RF)/100,REFV)$$

It means that RF follows the state variable slowly with a lower limit REFV, which is given by the REFV block or by the initial value of the state variable $y_n$ as discussed for the REFV command.

The accuracy control routine takes the following actions

| | |
|---|---|
| DEV>3·REF: | The step is cancelled and the time step is divided by 2.0 |
| 3·REF>DEV>REF: | The step is accepted but the next time step is divided by 1.5 |
| DEV>REF/3: | Increase of the 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 (with priority for the first one) 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.

By every 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 name of the variable that caused the step decrement.

The deviation used for accuracy control gives a measure of higher derivatives and seems to be a reasonable quantity. The procedure developed for DYSIM has an advantage in avoiding frequent cancellations of steps followed by step increments.

Only large pertubations of the system give rise to step cancellations when EPSI has a reasonable value (about $10^{-3}$). Most often the step will be decreased by criterion No. 2 in due time to avoid fulfilling criterion No. 1.

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

The maximum step length can be found by experiments, but it can be found more precisely by calculation of the Jacobi matrix and then by the eigenvalue calculation as discussed in section 10. The maximum step length is for HEUN

$$\Delta t_{max} = 2/\lambda_{max}$$

and for RUNGE

$$\Delta t_{max} = 2.78/\lambda_{max}$$

where $\lambda_{max}$ is $\max\{|Re(\lambda)|\}$

When a new simulation is started with a steady state calculation one must select a step size by experience with similar types of models, or start with a small value and gradually increase it until an upper limit is found. Too large steps make the system unstable, which means that variables diverge outside the number limits for the computer or for standard functions so that the simulation breaks down.

Both the HEUN and RUNGE routine has a feature which allows algebraic calculation of state variables. For every substep it is checked whether the variable has been changed during calculation of derivatives. If that is the case the new value is accepted, no integration step is taken and no accuracy control is performed for that variable.

## 6.2. ODEPAK

DYSIM offers the possibility to use the implicit multistep algorithms from the Odepack collection of solvers of ordinary differential equations, developed by Dr. A Hindmarsh and Dr. L. Petzold at Lawrence Livermore National Laboratory. These routines are only used for transient calculations.

In all 17 choices of method are available. These choices fall into three major groups. The first group consists of six versions of a variable order, variable step size Adams predictor/corrector method of orders 1-12, suited for nonstiff problems. The second group contains six versions of a variable order, variable step size backward differentiation methods (BDF) of order 1-5, suited for globally stiff problems. The third group is a collection of 5 variable order, variable step size methods. This last group is a combination of the two former groups with automatic stiffness detection and switching between Adams (non-stiff) and BDF (stiff) methods.

Methods of the three groups are identified by the METHOD number:

```
    METHOD = 0 : Adams/BDF auto-stiff method
           = 1 : Adams method
           = 2 : BDF-method
```

These implicit methods require a method for corrector iteration, specified by the MITER-number:

```
    MITER = 0 : functional iteration (no Jacobian)
          = 1 : user supplied, full Jacobian
          = 2 : internally computed, full Jacobian
          = 3 : internally computed diagonal approximation to
                the Jacobian
          = 4 : user supplied, banded Jacobian
          = 5 : internally computed, banded Jacobian
```

If METHOD = 0 (Adams/BDF auto-stiff) is used, MITER = 0 cannot be used.

If a user supplied Jacobian is to be used, (MITER = 1 or 4) the source program text must include a subroutine JAC for calculation of the Jacobian. For further details consult the source text. Otherwise the Jacobian is generated internally by difference quotients.

In order to use the Odepack integration routines, the file DYS/ODEPACK must be included in the compilation by the B7800, and the integration name ODEPAK must be written in the CHCK field of the input file. One more record with six integers may be specified.

The 1st integer is the method flag MF, specifying the integration and iteration method. MF is determined as

$$MF = 10 \cdot METHOD + MITER$$

(for example: BDF method with internally generated, banded Jacobian, set

$$MF = 10 \cdot 2 + 5 = 25.$$

The 2nd and 3rd integer in the record (ML and MU) give the lower and upper half-band width of a banded Jacobian, excluding the main diagonal (used only if MITER = 4 or 5). The band is defined by the matrix locations (i,j) with

$$i-ML \leq j \leq i+MU$$

where ML and MU must satisfy

$$0 \leq ML, MU \leq NDE-1$$

NDE being the number of differential equations.

The 4th number (MXORDS) specifies the maximal order using BDF-methods.

The 5th number (IDIF) gives the number of calls to subroutine CON,calculating derivatives and algebraic variables. These repeated calculations are performed in order to stabilize possible algebraic loops in the model.

Giving a non-zero value to the 6th number (INFO) in the record causes DYSIM to generate detailed information about the progress of integration when Odepack is used.

If no record is written or zero is specified for individual numbers in the record, default values will be used. The default values are: MF=2 (unless ML or MU are written in which case MF=5); ML=MU=0; MXORDS=5, IDIF=1, and INFO=0.

When the Odepack routines are used printout is sent only at the approximate print times, given in the PRNT field. Thus an integration step will be completed and the integration variable may exceed the printout time by a fraction of the current step size. This implies that the maximum integration step DTMAX should not exceed the minimum printout time interval.

At each integration step an estimated local error vector E(I) in the state variables y(I) is calculated and the error control is passed if

$$\frac{|\,E(I)\,|}{|EWT(I)|} \leq 1 \qquad\qquad I = 1,NDE$$

where EWT(I) is a vector of error weights given by

$$EWT(I) = EPSI \cdot (0.9ABS(y(I))+0.1REFV(I))$$

where EPSI is given in the CHCK block.

A detailed description of the methods is given by (BYRNE, G.D. and HINDSMARSH, A.D. 1975).

It is not easy to give a clear recommendation for the most suited Odepack solver, neither how to set up the CHCK field optimally. Many simulation models make up a set of stiff differential equations. Stiffness arise in models whose components have a large variety of time constants. The time constants of the model in the steady state can be investigated using the auxiliary program DYS/JACOBI (cf. section 10.5). It is recommended to use this program before choosing the Odepack set-up. The program gives a survey of the couplings between state variables (the structure of the Jacobian) and the time constants of the model. If the problem is stiff a BDF-method should be used. Stiffness is detected by initial use of the Adams/BDF autostiff method or given from the stiffness ratio (>100), calculated by DYS/JACOBI.

Difficulties also arise for the Odepack routines when discontinuities appear in the derivatives. Substantial computational effort is required passing the discontinuity because of automatical stepsize reductions, integrating up to the discontinuity.

The Odepack routines to be used in connection with DYSIM are designed for solving the initial value problem of the form

$$\dot{y} = f(t,y); \qquad y(0) = y_0$$

whereas simulation models in our context represent differential equations of the form

$$y = f(t,y,a(t,y)); \qquad y(0) = y_0$$

where a are algebraic or delay calculated variables as functions of time and state. These variables being functions of time and state are only updated at each integration step. This causes problems both for the Newton iteration to the corrector and for the numerical evaluation of the Jacobian.

If many of the derivatives depend strongly on algebraic variables a large (2-5) value of IDIF should be used whereas if the dependence is weak IDIF=1 will possibly work thereby saving much computer time.

It is the experience that although the implicit methods use relatively long step sizes compared to the explicit Runge-Kutta methods, integration speed is not increased due to extensive computational efforts calculating Jacobians and inverting matrices. In many cases, though, it is sufficient to generate a rather narrow banded Jacobian to give a crude approximation thus saving substantial computer time. This is done specifying ML and MU and use MITER = 4 or 5. If the Jacobian is banded the Runge-Kutta and Odepack routines become comparable in speed for large systems (>20 differential equations). However, for small systems Odepack is superior in speed.

The performance of the Odepack routines is very sensitive to the choice of good values of DTMIN, DT, DTMAX, EPSI, ML, MU and IDIF. Generally DTMIN and DT should be several orders of magnitude smaller than for Runge-Kutta methods, whereas EPSI normally should be set one order of magnitude lower. An extensive experimental work must be performed adjusting DTMIN, DT, DTMAX, EPSI, ML, MU, and IDIF to optimize the step size and reducing the CPU time for a given Odepack routine. Doing this it is advised to set INFO=1. At each printing time DYSIM will report integration time, step number, number of CON calls (derivative and algebraic variable evaluations), step size last used, integration method used, order, number of Jacobian evaluations and CPU time consumed.

## 7. PRINT AND PLOT FACILITY

During transient calculations variables are stored in a temporary file according to the list and the time intervals given in the PRNT block. The integration step number and the time are stored in position number one and two automatically. The maximum number of variables are 72 at present, time and step numbers included.

In order to hit the correct printout times DYSIM adapts the time step if needed; but the original step is reinserted afterwards so that loss of processor time by a gradual step increase is avoided.

In addition to the input actuated printout the user can demand a printout of variable values taken 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. This facility cannot be used in connection with ODEPAK integration.

The values at time zero are printed as the first set and the values at terminal time as the last set.

The data collected during the integration are stored in a buffer 720 words long giving space for 10 sets of values between transfer to the output file. When a smaller number of print variables is specified the buffer will be used in an economic way so that more sets can be stored between disk transfer.

The collection of data is done in subroutine INTEG which places all data in the output file in unformatted form with record length = 720. A set of variables consists of NPRI data, and a record has 720/NPRI sets (truncated integer). At program termination the subroutine OUTPUT is called. It reads the file and arranges the data in pages and columns for the output printer. The variable names specified in the PRNT block and physical dimensions from the list file are used as headings.

When the program stops, the file will be deleted unless PLOT is specified. In that case the file is extended with one record which contains two integers, two text arrays and one text string. The first integer is the maximum number of variables in a set of variables (72),the second integer is the actual number NPRI. The two arrays contain variable names and dimensions; each element is 12 characters long. The last text string is 72 characters long and contains the identification text in the INPT block. This information is used in the plotter program described in section 10.

DYSIM also gives some messages with information about the simulation job. Some of these always go to the printer and are printed on the last page. Others are written on file 6 which for a remote controlled job is the screen and for a batch job the printer. In the latter case these messages are written on the first page of the printer output.

The messages give information about the following items:

- Errors in the list file and input file.
- Type of job: steady state, Jacobian or transient calculation.
- Restart from a dump file.
- Loading of initial condition from an IC-file.
- The integration routine and related statistics about step size changes.
- An error in the time delay function.
- Program termination by CALL NSTOP or CALL TERM.
- A user message related to the previous item.
- Program stop due to too small integration step.
- Program stop due to CPU time limitation.
- Program stop due to generation of too much table printing.
- Dump of the state on a dump file.
- Steady state saved on an IC-file.

## 8. TIME DELAY SIMULATION

A function DELAY has been developed for simulation of pure time delays as the one found in a flow through a tube with a uniform velocity, which may change, but without sign shift. The function is not part of the file DYS/DYSIM86, so it must be included by the user from file DYS/DELAY86 when it is to be used. The separation is done in order to save core memory when DELAY is unused, as it has a relatively large data buffer.

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

- Introduce an extra state variable

  $\dot{X} = v; \quad X(0) = 0$

- Find the time delay $\tau$ by the statemennt

  TAU=TRNSTM('TAU',X, X-L),

  where the first parameter is an identifier of the function 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 then other delays with fixed relations to TAU could be calculated by algebraic equations. The methods used here will often require some extra calls of TRNSTM and extra buffer space, but the calls of both TRNSTM and DEADTM are very fast.

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

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 substep value is stored even if the delay function is called by every substep number.

Separate actions are taken when the function is called the first time at substep number zero. 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. Use of identifiers not given in the first call will have the same effect by subsequent calls.

Adaption 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.

## 9. OTHER FACILITIES

### 9.1. Dump and restart

When an error occurs long into a transient 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 that 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 contents of the COMMON fields INTVAR, and ALGVAR are stored in the file SYNA/DUMP, which is made permanent. If DELAY is used, the file SYNA/DELAY is transferred to another file SYNA/DELAYDUMP extended with some records in which the content of the delay buffers and relevant pointers are stored, and the file is made permanent. An announcement with the time for dump is given after the output tables.

When a restart is activated the data stored in SYNA/DUMP and SYNA/DELAYDUMP is used to reestablish the problem state prior to the dump, and the integration continues from there. The two disk files remain permanent so that the restart can be repeated. They must be removed manually when no longer used.

A dump can be activated by the DUMP command or by a statement CALL DUMP in the user's program. When the latter possibility is used the calculations may also be stopped by one of the two calls (TERM or NSTOP) mentioned below. The statement CALL DUMP does not stop the calculations.

## 9.2. Program termination

The calculation of both transients and steady states can be terminated by the user at the end of any accepted integration step by the statement CALL TERM(TEXT). For special purposes, e.g. search for erors, a more abrupt stop can be made by the statements CALL NSTOP(TEXT); RETURN. It results in an immediate integration stop regardless of the substep number NR. However, the calculations in the following modules called from the connecting routine are carried out until program control is returned to DYSIM. By steady state the iteration is terminated without storing the state in the IC-file, but printing is carried out as normally.

Both TERM and NSTOP must have a text string as parameter; it is printed by DYSIM after the output tables together with a message which tells what type of stop call the user has given.

## 9.3. Repetition of an integration step

The integration step in progress can be interrupted and repeated with a smaller time step (divided by 2.0) by the statements CALL REPET; RETURN regardless of the value of the substep 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 upon by the user. Note that for systems with several modules the calculation in the following modules will continue until program control is returned to DYSIM. This feature is not valid for ODEPAK integration.

## 9.4. Activation of special user routines

The statement CALL RECALL can be used to activate a special user routine, when the integration step in progress has been accepted; it may be given at 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. This feature is not valid for ODEPAK integration.

Just before the program stops DYSIM gives a CALL YOUT. YOUT must be supplied by the user in all programs as a subroutine or an 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 tables. Precompiler 2 can be used to create the entry points YOUT and DYNAMS.

# 10. INDEPENDENT AUXILIARY PROGRAMS

## 10.1. The plotting program DYS/PLOT

This program is used to show transients as functions of time on a graphic screen and make hardcopies on a side plotter or a remote plotter in either the control room for the B7800 computer or a local terminal room. The program is strongly dependent on the B7800 installation as it is based upon RISØ's general graphic display system called RIGS. Another more elaborate version will be developed later for the PC-version of DYSIM.

When the program starts the user is asked to select plot device in one or two stages. The first selection is between side plotter or remote plotter (answer SIDE for side plotter and REMOTE for remote plotter). If the choice is REMOTE the next selection is between central or local (answer CENTRAL for the central Benson plotter and LOCAL for the local line printer).

Then the program asks for the name of the data file, which previously has been created by DYSIM.

The program now enters a repetitive section and asks for a variable name, which must be the full name with a ".", and module name. The user can get a list of possible names by the command LIST instead of a variable name. If the name is valid the program gives the maximum time and the minimum and maximum values for the variable. Then the user must specify the time axis defined by the size and number of time intervals between marks; he must further select between linear and spline interpolation and between automatic or manual scaling of the y-axis for the variable. The scale must be specified by the zero point and the size and number of intervals between marks. Both scales are always linear. The picture will now appear on the screen, and the user can after observation decide to make a plot or delete the picture as an answer of a new question. After that the program returns to the beginning of the repetitive section and asks for a new variable.

When some pictures are generated and stored for plotting the user can give the command PLOT instead of a variable name which brings the program to the plot section. It starts to ask if two picture shall be put together on one A4 page. If the answer is YES it further asks which variables the user wants to compose, and finally it sends the necessary data to the central plotting system on B7800 or to a side plotter. Upon termination the control goes back to the repetitive section with the picture buffer cleared. The user can now proceed with a new group of pictures, select a new data file, or stop the program with the command STOP. A HELP command can also be used to give a list of valid commands.

## 10.2. The list file editor

The list file editor is used to create a new list file from scratch or compose a new one using modules from old files with or without modifications, or add new modules to an existing file. Small corrections in existing files can easier be made with the computer's general editor if only the user is aware of the fixed format which clearly appears in a listing of the file.

By program start the user must give the system name (with maximum 4 characters) upon demand, whereupon the program constructs the file name as SYNA/LIST. Then, and only then, it is possible to copy an old version of the list file without changes before entrance to the main repetitive program section which always starts with a menu. The possible commands are

| | |
|---|---|
| MAKE XXX | : Make a new module with the name XXX. |
| GET XXX | : Get module XXX. |
| LOAD XXX | : Load module XXX made by precompiler 1. |
| COPY XXX | : Copy module XXX from an old version of the file. |
| TAKE XXX FROM YYYY: | Take module XXX from the list file for system YYYY. |
| INFO (YYYY): | Find and show all module names in the old file or the list file for system YYYY. YYYY = *NEW gives the modules saved in the new list file |
| OMIT | : Remove the new file and keep the old one; cancel the job and terminate. |
| TEST | : Close the new file, make a formal test and terminate. |
| END | : Close the new file and terminate. |

The MAKE command guides the user through creation of a list for a new module. It poses questions and checks answers and asks the user to write information about variables inside frames. The user must go through the whole procedure writing information about state variables, algebraic variables, input variables and parameters.

The LOAD command loads a list file for module XXX belonging to the system defined at program start. The file is created by precompiler 1, but without dimension and text fields. The user is asked to fill in these fields inside given frames, and he must go through all variables as for the MAKE command.

The commands MAKE, LOAD, GET, and TAKE leave a list for one module in a buffer with space for 200 lines and give the user a small menu with only three commands:

SAVE : Save the buffer in the new file and return to the main menu.

EDIT : Call an editor subroutine which can be used to correct errors or change the list in the buffer or just inspect it.

REM : Clear the buffer, regret and return to the main menu.

Having used and left the editor called by EDIT the program goes back to the small menu so that the user can save or remove the list in the buffer. The EDIT routine is described below in section 10.3.

The list file is created with records 80 characters long and a record number in columns 73-80. On B7800 the file is of type SEQ with record number increment equal to 100.

## 10.3. The local editor EDIT

This editor is of a more general type; it is used to edit a text field contained in a buffer which is connected to the subroutine via a parameter in the call. When it is used from the list file editor or the input file editor the user himself must take care of the format as mentioned above in connection with the use of the computer's normal editor.

EDIT works with lines which are 80 characters long and assumes that columns 73-80 are reserved for line numbers. It starts by renumbering all lines with consecutive numbers from 1 and upwards using column 76-78; column 73-75 and 79-80 are filled with blanks and later used by EDIT to set marks for the corrections given by the user. By an update command all line numbers are updated, and when EDIT is terminated a new updating is given with number increment of 100 using columns 73-80 as in the list file editor.

EDIT presents a menu for the user when it is called. The menu can always be called by the H command. EDIT uses a prompt character ">" to ask for a command from the user. The menu looks like

```
L n,m    : List m lines from no. n without corrections.

S n,m    : Show m lines from no. n with corrections if present.

D n,m    : Delete m lines from no. n

C n,m    : Change m lines from no. n.

R n,m    : Renew m lines from no. n, i.e. cancel all correc-
           tion marks.

A        : Append new lines at the end.

I        : Insert new lines before no.  .

P   FROM XX n-m:
           Insert lines no. n to m from file XX before line
           no.  .

U        : Update, i.e. insert all corrections introduced and
           make a new line numbering in columns 76-78.

E        : Update if corrections are present, make a new
           line numbering in columns 73-80 and exit.

H        : Give a menu list.
```

Command strings with n,m may be written in several forms which for the L command are:

```
L

L n

L n,m

L *

L *,m
```

The "*" gives the present value of a line counter containing the last used line number +1. When m is absent the command refers to line no. n only. When both n and m are absent the whole buffer is referenced.

EDIT works with a correction buffer with space for 100 lines. The routine checks for violation of buffer limits. If the correction buffer is full an update command must be given; if the list buffer runs full due to insertion of lines no further insertion can be made.

The commands C, I, and A require that the user writes new lines. EDIT helps the user by presenting lines to be corrected, or neighbouring lines by insertion and appending; it furthermore creates a frame inside which the user must write the new line. If the user writes outside the frame EDIT will repeat presentation of line and frame and expect a new text string. The C, I, and A mode can be terminated by the character ">" in the first column as the only character accepted outside the frame.

## 10.4. The input file editor DYS/INPUT

To assist the user in writing a correct and complete input file the auxiliary program DYS/INPUT is available. Developing this program serves several purposes: the user does not have to worry about the syntax of the input file, the program presents all obligatory and optional fields/commands thereby ensuring the user that all necessary fields are present and forcing the user to select among all optional commands which to be used. Furthermore, the program gives a short explanation to each field presented. The program works selfexplaining and interactively with the user and precautions have been taken against damaging already existing input files.

In order to create a new input file the list file holding variable names for all modules must be present (i.e. the list file editor DYS/LIST should be used in advance). The list file is identified to DYS/INPUT by the user giving the system name SYNA when required by the program. The list file is supposed to have the name SYNA/LIST.

The program offers four facilities working with the input file:

a) write new input file

b) edit an existing input file

c) continue writing the input file after a break (i.e. it is possible to break sessions, save what has been written so far and continue writing the input file later).

d) replace the entire INCO field in an existing input file by new steady state values produced previously by DYSIM under the STST command.

Re a:
Making an input file for a simulation model it is assumed that the list file holds variable names for all modules making up the model. However, it is often desirable for test purposes to run a simulation using only one or more modules from the entire model. Writing an input file for this application it is possible to use the list file for the whole model, simply selecting the modules to be used. Of course the correct connecting system must be written.

Furthermore it is possible to make an input file for a simulation using the modules in a changed sequence. This may be useful in Jacobian calculations if the couplings between state variables could be arranged such that the Jacobian turn into a nearly banded or triangular structure (c.f. section 10.5).

When writing values for the INCO, DATA, REFV, and PRNT blocks it is possible to copy values from an existing input file, either for the whole system or for single modules.

Re b:
The editor described in the previous section is used. Of course also the computer's standard editor could be used. Writing a new input file the local editor can be called at each completion of a field or before saving the input file.

## 10.5 The Jacobian service routine DYS/JACOBI

In order to get a survey of the couplings between state variables and time constants of the simulation model the auxiliary program DYS/JACOBI is available.

To use the program the Jacobian of the system of differential equations must be calculated. This is done by DYSIM when the JACO command is written in the input file. The Jacobian is calculated by difference quotients in the system state given by the INCO field (normally a steady state), and written on the file SYNA/JACO; SYNA stands for the system name. Running DYS/JACOBI this file is read together with the input file.

Four commands can be accomplished:

a) A table of the Jacobian can be printed. This table could be quite extensive for large systems and should only be used when values of the matrix elements are required.

b) A compact image of the couplings between state variables can be printed. This printing, called a non-zero image, shows all non-vanishing elements of the Jacobian. This picture holds information about which state variables influence a given derivative. The modular structure of the model is easily read from the picture. A possible rearrangement of the order of sequence of the modules might appear from a study of this picture, thereby changing the structure of the Jacobian into a banded or nearly triangular form. The possibility of specifying a banded structure to the Odepack integration routines saves much computational effort. The actual half-band width of the Jacobian shown (MU and ML) are written. However, in most cases the band can be made more narrow for the Odepack usage.

c) Linear approximations to differentials of the derivatives can be printed; i.e. when the NDE differential equations of the model are written as

$$\frac{dy_i}{dt} = f_i(t,y_j); \quad y_i(0) = y_{oi}; \quad i,j = 1,N$$

a table is written giving

$$df_i = \sum_{j=1}^{j=NDE} \frac{\delta f_i}{\delta y_j} dy_j \quad ; \quad i=1,NDE$$

where $\dfrac{\delta f_i}{\delta Y_j}$

is printed as the value of the (i,j)-matrix element of the Jacobian. Only non-vanishing terms of the series are printed. This gives an insight into the strength of the couplings, when the magnitudes of the state variables are known.

d) Tables of eigenvalues, eigenvectors, decay times and suggested stepsizes to be used can be printed. The table of the eigenvectors is extensive for large systems and is only printed on request. Eigenvalues and eigenvectors are calculated using the LINPACK library of numerical algorithms. An analysis of the eigenvalues is performed: A special message is written for components of the solution having positive real parts of the eigenvalue as this leads to unstable systems. For the components having negative real parts $\lambda_R$ and non vanishing imaginary parts $\lambda_I$ of the eigenvalues, i.e. exponentially decreasing oscillatory components, a suggested time step for use in multistep methods is written. This maximum time step h is calculated from the condition

$$h|\lambda_I| < \frac{2\pi}{8}$$

thus representing the oscillatory solution in at least 8 points in one period. A decay time T(HALF) is written for all decaying components giving the time for the component to reach half the initial value.

Suggested time steps for the Runge-Kutta methods are calculated using

| | |
|---|---|
| $h \cdot \max\{|\lambda_R|\} < 2$ | 2. order |
| $h \cdot \max\{|\lambda_R|\} < 2.78$ | 4. order |

Spectral radius $\max\{|\lambda|\}$ and stiffness ratio

$$\frac{\max\{|\lambda_R|\}}{\min\{|\lambda_R|\}}$$

are calculated. For stiff systems (large stiffness ratios) the Odepack-BDF methods may be usefull.

Details concerning step size limitations and stiffness ratio can be found in chapter 8 in (J.D. LAMBERT, 1973).

## 11. A PC-VERSION OF DYSIM

Recently DYSIM has been transferred to a personal computer in a preliminary form with the introduction of some new features mentioned below. The calculations are controlled in a more interactive way and the user has a possibility to follow the progress of calculations. In the near future a more elaborate user interface will be developed based upon graphic representation.

For the present version some points shall be mentioned.

a) The TIME command is no longer significant as the processor time is unlimited in principle. The steady state calculations are controlled by the user.

b) A new subroutine by name SERVER is introduced; it is called both by transient and steady state calculations at intervals determined by the user. It presents itself by a message giving either the time for transients or the integration step number by steady state. The prompt character ":" is given to ask the user for a command which must be one of the following 4 commands

- VARIABLENAME which gives the actual value of either a state variable, an algebraic variable or a parameter. If a parameter is chosen the user gets the opportunity to give a new value; the old value is retained by a CR.

- T or S gives the time for transients and the step number for steady state.

- C n results in continuation of calculations n seconds for transients or n steps for steady state. After that the program control is passed over to SERVER again.

- E means exit. The calculations are terminated in a normal way.

The server routine can be called by the user at any time during the calculations typing CTRL C at the keyboard. For that case the continuation command C can be used without the number n.

c) Steady state calculations can be controlled by one to three numbers in the STST data record. The first number gives the number of integration steps between printout as for the B7800 version, the second one gives the number of steps between output to the screen, and the third one is the number of steps between call of SERVER. Default values are used if not all numbers are specified. If the data record is skipped the default values are 100, 10, 100. If only the first number, N1, is given (as for B7800 version input files) the default values for the two next are N2 = N1/10 and N3 = N1. The output to the screen consists of one line with the integration step and the first 6 variables in the PRNT block.

d) Transient calculations run until the terminal time given in the CHCK block is reached; then program control goes to the SERVER routine. The user can now continue integration in time segments as wanted. At present nothing is shown at the screen during integration. This procedure gives the user the possibility to prepare an input file set-up with parameter values without pertubations and with a small terminal time (e.g. 1 or 10 s). By the first call of SERVER (or by sequential calls) the user can introduce a transient disturbance and integrate as long as wanted. This means that the same input file can be used for many transients.

e) An extra command, *REAL, can be used to specify syncronization between real time and integration time if the model is sufficiently small. No accumulation of print or plot data is made. The integration step is fixed equal to the initial value DT given in the CHCK block. The two time intervals given in the PRNT block as PDT1 and PDT2 (number 1 and 3 in the first data record) are used as time intervals for two outputs. The first one controlled by PDT1 writes the values of all print variables (with time included as the first one) out on the standard port RS-232-C or on a disk file. The second output controlled by PDT2 goes to the screen with one line containing the time followed by the first 6 variables in the print list. When a file name is given in connection with the REAL command the first output goes to that file unless the file name is COM1, which is a reserved name for the port RS-232-C. Without a file name the output goes to the port. The format for port output is with Fortran notation (1P,(X,7E11.4)), which means that each record contains 7 numbers in the 1PE11.4 format. For disk file output a set of variables is written as

```
((time)
(name var)(name var)(name var)
---
)
```

and all sets are enclosed by brackets, "("as the first and ")" as the last character of the file, written in separate records. The text name is a 12 character long variable name and var is the numerical value in 1PE11.4 format. The time is given by the value without a text.

For the near future the intention is to transfer the whole simulation package with precompilers and auxiliary programs to the PC and develop graphics for display of results both during and after the calculation.

Furthermore, it has been discussed to develop some graphic programs to guide an unexperienced user through the process of creating models and running simulations. A graphic system can possibly also be used to present a model for the user in a form more clear and easier to use than tables and diagrams on paper.

## REFERENCES

BYRNE, G.D. and HINDMARSH, A.D. (1975). A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations. ACM Trans. Math. Software, Vol. 1, 71, 1975.

CHRISTENSEN, P. la Cour (1981). Description of a Simulation System DYSIM for Continuous Dynamic Processes, Risø-M-2271.

LAMBERT, J.D. (1973). Computational Methods in Ordinary Differential Equations, John Wiley & Sons.

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

APPENDIX A

<u>Program listing of a small module in fortran code</u>

```
      SUBROUTINE FC(NR,TID)
C *   FEEDWATER CONTROL BY THE AUXILIARY SYSTEM
C
C     COMMON /CFC/ CONTAINS:
C     S: STATE VARIABLES    : 6
C     D: DERIVATIVES        : 6
C     X: INPUT VARIABLES    : 6
C     C: INPUT CONSTANTS    : 6
C     A: ALGEBRAIC VARIABLES: 0
C
      COMMON /CFC/
     S WFE,XH(3),V42,V23,
     D WFEP,XHP(3),V42P,V23P,
     X WLEV,PE,WLR,SS,ASKH,P12,
     C GL,GW,WFER
C
      REAL YH(6)
      REAL KV4,KV42,KV23,KHF
C
      DATA WLERO,TC1,TV42,TV23 /4.1, 10., 30., 45./
      DATA KV4,KV42,KV23,KHF /5.63E-3, 5.63E-5, 7.E-4, 8.E-4/
      DATA A34,B34,C34 /-.101853E-2,.325180E-2,25.74E0/
      DATA A12,B12,C12 /-.394141E-2,.647404E-1,76.3186/
      DATA WLEVS,SST / 0., -1. /
C
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
C
      IF(NR.EQ.0) THEN
C *     INITIALIZATIONS AT START
        SST=-1.
        WLEVS=0.
      END IF
C
      IF(SS.GT.0.) THEN
C *   CHANGE OF WATER LEVEL SETPOINT BY SCRAM
C *     SST: EVENT TIME
        IF(SST.LT.0.) SST=TID
        WLEVS=-.4
        IF(TID-SST.GE.600.) WLEVS=-.4*AMAX1(0.,1.-(TID-SST-600.)/120.)
      END IF
C
C *   AUXILIARY FEEDWATER FLOW CALCULATION
C *   YH1: LEVEL SIGNAL
      YH(1)=ALIM((WLEV-2.5)/3.5,1.,0.)
C *   XH1: VARIABLE IN COMPENSATION CIRCUIT
      XHP(1)=(YH(1)-XH(1))/TC1
C *   YH2: COMPENSATED LEVEL SIGNAL
      YH(2)=ALIM(YH(1)-XH(1),.05,-.05)+YH(1)
C *   YH3: LEVEL ERROR SIGNAL
*!NB! YH3 IS MODIFIED WITH AN EXTRA FEEDBACK SIGNAL (WFE-WLR)
*     REPRESENTING THE FLOW ERROR
*     GL IS THE LEVEL ERROR GAIN BEFORE THE CONTROLLER
*     GW -  -   FLOW  -     -     -     -   -
      YH(3)=GL*((WLERO+WLEVS-2.5)/3.5-YH(2))-GW*(WFE-WLR)/100.
C *   PI-CONTROLLER. INPUT YH3, OUTPUT YH4
      XHP(2)=20.*YH(3)/60.
      IF(XH(2).LE.0..AND.XHP(2).LT.0.) XHP(2)=0.
      IF(XH(2).GE..72.AND.XHP(2).GT.0.) XHP(2)=0.
      XH(2)=ALIM(XH(2),.72,.00)
      YH(4)=ALIM(XH(2)+20.*YH(3),.72,.00)
      IF(WFER.GT.0.) YH(4)=AMIN1(WFER/100.,.72)
C *   CONTROL VALVE SERVO (V4).  XH3:VALVE POSITION
C *   WFE: AUXILIARY FEEDWATER FLOW
```

```
      WFES=WFE/100.
      YH(5)=ALIM(20.*(YH(4)-WFES),1.,-1.)
      XH(3)=ALIM(XH(3),1.,1.E-10)
      XHP(3)=YH(5)/30.
      IF(XH(3).LE.1.E-10.AND.XHP(3).LT.0.) XHP(3)=0.
      IF(XH(3).GT.1..AND.XHP(3).GT.0.) XHP(3)=0.
C *   BYPASS VALVE OPENING (V42), ON-OFF CONTROL BY WATER LEVEL
      V42=ALIM(V42,1.,1.E-10)
      V42P=0.
      IF(WLEV.LT.2.9.AND.V42.LT.1.) V42P=1./TV42
      IF(WLEV.GT.4.2.AND.V42.GT.1.E-10) V42P=-1./TV42
C *   CONTAINMENT VALVE (V2,V3) CONTROLLED BY INPUT ASKH
      V23=ALIM(V23,1.,1.E-10)
      V23P=0.
      IF(ASKH.GT.0..AND.V23.LT.1.) V23P=1./TV23
      IF(ASKH.EQ.0..AND.V23.GT.1.E-10) V23P=-1./TV23
C *   FRICTION COEFF.FOR V4 AND V42 INDIVIDUALLY
      HV4=KV4/XH(3)**2
      HV42=KV42/V42**2
C *   FRICTION COEFF.FOR V4 PARALLEL WITH V42
      HV442=HV4*HV42/(SQRT(HV4)+SQRT(HV42))**2
C *   FRICTION COEFF.FOR V2/V3
      HV23=KV23/V23**2
C *   FLOW CALCULATION WITH PUMPS P3/P4 + P1/P2 PUMPING
C *   AGAINST PE+3 WITH FRICTION COEFF. HV442,HV23 AND KHF
      A=A34+P12*A12-(HV442+HV23+KHF)
      B=B34+P12*B12
      C=C34+P12*C12-PE-3.
      W=(-B-SQRT(B*B-4.*A*C))/(2.*A)
C *   CALCULATED FLOW IS FILTERED WITH .2 S TIME LAG
      WFEP=(W-WFE)/.20
      RETURN
      END
```

## APPENDIX B

<u>Connecting routine for a system with 6 modules written in fortran code</u>

```
      SUBROUTINE CON(NR)
*     MAIN PROGRAM FOR BARSEBAECK LOW POWER MODEL
*     THE MODEL HAS THE FOLLOWING MODULES:
*     RE  : THE REACTOR MODEL
*     CR  : CONTROL ROD ROUTINE
*     FC  : FEEDWATER CONTROL MODEL
*     PC  : PRESSURE CONTROL MODEL
*     FL  : NEUTRON FLUX MONITOR AND LIMIT CONTROLS
*     RK  : RESIDUAL HEAT REMOVAL MODEL
*
*
      COMMON /INTVAR/ TID,
     # SVRE(63),SVCR(3), SVFC(6), SVPC(9), SVFL(3), SVRK(4)
      COMMON /DERIV/
     # DERE(63),DECR(3), DEFC(6), DEPC(9), DEFL(3), DERK(4)
      COMMON /ALGVAR/
     # AVRE(60),AVCR(30),AVFL(30)
      COMMON /DATA/
     # PARE(12),PACR(12),PAFC(6), PAPC(6), PAFL(6), PARK(6),
     # ASKH,XSKD,PER,WREK,DUMP,A314,
     # I314,WLX,WLXT,P12,DRAIN
*
      COMMON /CRE/  SRE(63),DRE(63),XRE(30),PRE(12),ARE(60)
      COMMON /CCR/  SCR(3), DCR(3), XCR(12),PCR(12),ACR(30)
      COMMON /CFC/  SFC(6), DFC(6), XFC(6), PFC(6)
      COMMON /CPC/  SPC(9), DPC(9), XPC(12),PPC(6)
      COMMON /CFL/  SFL(3), DFL(3), XFL(6), PFL(6), AFL(30)
      COMMON /CRK/  SRK(4), DRK(4), XRK(6), PRK(6)
*
      REAL I314
*
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
*
*     INSERT STATE VARIABLES
      DO 110 J=1,63
 110  SRE(J)=SVRE(J)
      DO 120 J=1,3
 120  SCR(J)=SVCR(J)
      DO 130 J=1,6
 130  SFC(J)=SVFC(J)
      DO 140 J=1,9
 140  SPC(J)=SVPC(J)
      DO 150 J=1,3
 150  SFL(J)=SVFL(J)
      DO 160 J=1,4
 160  SRK(J)=SVRK(J)
*
*
*     CONTROL RODS :
*     XCR(1):QN
      XCR(1)=ARE(34)
*     XCR(2):PE
      XCR(2)=SRE(56)
*     XCR(3):WGE
      XCR(3)=ARE(37)
*     XCR(4):SS
      XCR(4)=AFL(16)
*     XCR(5):CRS
      XCR(5)=0.
      IF(AFL(17)+AFL(18).GT.0.) XCR(5)=1.
*     XCR(6 :DBT
      XCR(6)=AFL(24)
```

```
*     XCR(7):RT
      XCR(7)=SRE(49)
      CALL CR(NR,TID)
*
*     REACTOR :
*     XRE(1):WLR
      XRE(1)=SPC(8)
*     XRE(2):WFE
      XRE(2)=SFC(1)
*     XRE(3):XSKD
      XRE(3)=XSKD
      IF(AFL(2).GT.0.) XRE(3)=0.
*     XRE(4):TREK
      XRE(4)=SRK(1)
*     XRE(5):WREK
      XRE(5)=WREK
*     XRE(6):WLOS
      XRE(6)=0.
*     IF WATER LEVEL > 4.2 AND DRAIN > 0  DRAIN WATER FROM TANK MANUALLY
      IF(DRAIN.GT.0..AND.SRE(53).GT.4.2) XRE(6)=5.
*     XRE(7:30):PCCR(1:24)
      DO 310 J=1,24
  310 XRE(J+6)=ACR(J)
      CALL RE(NR,TID)
*
*     FEEDWATER CONTROL
*     XFC(1):WLEV
      XFC(1)=SRE(53)
*     XFC(2):PE
      XFC(2)=SRE(56)
*     XFC(3):WLR
      XFC(3)=SPC(8)
*     XFC(4):SS
      XFC(4)=AFL(16)
*     XFC(5):ASKH
      XFC(5)=ASKH
      IF(AFL(2).GT.0.) XFC(5)=0.
*     XFC(6):P12
      XFC(6)=P12
*     START PUMPS 1-2 WHEN P>20 BAR IN CASE 104.
      IF(PRE(1).EQ.104..AND.SRE(56).GT.20.) XFC(6)=1.
      CALL FC(NR,TID)
*
*     PRESSURE CONTROL
*     XPC(1):PE
      XPC(1)=SRE(56)
*     XPC(2):QN
      XPC(2)=ARE(34)
*     XPC(3):PER
      XPC(3)=PER
      IF(ACR(27).GE.8.) XPC(3)=ACR(27)
*     XPC(4):XSKD
      XPC(4)=XSKD
      IF(AFL(2).GT.0.) XPC(4)=0.
*     XPC(5):DUMP
      XPC(5)=DUMP
      IF(AFL(6).GT.0.) XPC(5)=0.
*     XPC(6):A314
      XPC(6)=A314
      IF(AFL(2)*AFL(19)+AFL(3)+AFL(6).GT.0.) XPC(6)=1.
*     XPC(7):I314
      XPC(7)=I314
      IF(AFL(2)*AFL(19)+AFL(3)+AFL(6).GT.0.) XPC(7)=1.
*     XPC(8):WLX
      XPC(8)=AMIN1(WLX*TID/WLXT,WLX)
      CALL PC(NR,TID)
*
```

```
*     FLUX MONITORS AND LIMIT CONTROLS
*     XFL(1):FLUX
      XFL(1)=ARE(44)
*     XFL(2):WLEV
      XFL(2)=SRE(53)
*     XFL(3):PE
      XFL(3)=SRE(56)
*     XFL(4):DBT
      XFL(4)=ARE(45)
*     XFL(5):XXX
      XFL(5)=1.
*     XFL(6):YYY
      XFL(6)=1.
      CALL FL(NR,TID)
*
*     RESIDUAL HEAT REMOVAL SYSTEM
*     XRK(1):TRI
      XRK(1)=ARE(36)
*     XRK(2):WREK
      XRK(2)=WREK
      CALL RK(NR,TID)
*
*     RETURN STATE VARIABLES AND DERIVATIVES
 1000 CONTINUE
      DO 1010 J=1,63
      SVRE(J)=SRE(J)
 1010 DERE(J)=DRE(J)
      DO 1020 J=1,3
      SVCR(J)=SCR(J)
 1020 DECR(J)=DCR(J)
      DO 1030 J=1,6
      SVFC(J)=SFC(J)
 1030 DEFC(J)=DFC(J)
      DO 1040 J=1,9
      SVPC(J)=SPC(J)
 1040 DEPC(J)=DPC(J)
      DO 1050 J=1,3
      SVFL(J)=SFL(J)
 1050 DEFL(J)=DFL(J)
      DO 1060 J=1,4
      SVRK(J)=SRK(J)
 1060 DERK(J)=DRK(J)
      RETURN
*
*
      ENTRY ALVAR(NO)
      IF(NO.EQ.0) THEN
*       INSERT INPUT DATA FOR NR=0
        DO 10 J=1,12
 10     PRE(J)=PARE(J)
        DO 20 J=1,12
 20     PCR(J)=PACR(J)
        DO 30 J=1,6
 30     PFC(J)=PAFC(J)
        DO 40 J=1,6
 40     PPC(J)=PAPC(J)
        DO 50 J=1,6
 50     PFL(J)=PAFL(J)
        DO 60 J=1,6
 60     PRK(J)=PARK(J)
      END IF
*
      IF(NO.EQ.1) THEN
*        TAKE ALGEBRAIC VARIABLES FROM COMMON /ALGVAR/
         DO 1110 J=1,60
 1110    ARE(J)=AVRE(J)
         DO 1120 J=1,30
 1120    ACR(J)=AVCR(J)
```

```
         DO 1130 J=1,30
 1130    AFL(J)=AVFL(J)
      END IF
      IF(NO.EQ.2) THEN
*       RETURN ALGEBRAIC VARIABLES TO COMMON /ALGVAR/
        DO 1210 J=1,60
 1210   AVRE(J)=ARE(J)
        DO 1220 J=1,30
 1220   AVCR(J)=ACR(J)
        DO 1250 J=1,30
 1250   AVFL(J)=AFL(J)
      END IF
      RETURN
*
*
      ENTRY YOUT
      CALL YOREAC
      CALL YOCROD
      RETURN
*
*
      ENTRY DYNAMS
      RETURN
      END
```

## APPENDIX C

<u>List file for the same system as used in Appendix B</u>

```
 10 MODULE:      RE    REACTOR , SUBROUTINE RE
 20 ==========================================
 30 STATE VARIABLES   63
 40 --------------------
 50 NAME         SRE     DIMENSION    TEXT
 60 --------------------------------------------------------------------------
 70 XQN(3)       1-3     MW           DECAY HEAT VARIABLES
 80 TC(10)       4-13    C            WATER TEMP. IN THE CORE
 90 ALF(10)      14-23                VOID FRACTION IN THE CORE
100 TU(10)       24-33   C            FUEL TEMP.
110 CN1(10)      34-43   CMe-3        CONCENT.OF DELAYED NEUTRON EMITTERS
120 TLP1         44      C            TEMPERATURE IN LOWER PLENUM 1
130 TLP2         45      C            -           -  -    -     2
140 TBP1         46      C            -           -  CORE BYPASS 1
150 TBP2         47      C            -           -   -    -    2
160 TR           48      C            -           -  RISER
170 TT           49      C            -           -  TOP WATER VOLUME
180 TB           50      C            -           -  FEEDWATER CHAMBER
190 TDC          51      C            -           -  DOWNCOMER
200 TPU          52      C            -           -  RC-PUMPS
210 WLEV         53      M            WATER LEVEL IN TANK
220 PP           54      BAR/S        DP/DT FILTERED BY TAU=0.2
230 PR           55      BAR          RISER PRESSURE
240 PE           56      BAR          STEAM DOME PRESSURE
250 ALFR         57                   RISER VOID FRACTION
260 WRC          58      KG/S         COOLANT FLOW FROM RC-PUMPS
270 YHC1         59      Me3          VARIABEL FOR DELAY SIMULATION
280 YHC2         60      Me3          -        -   -     -
290 TTW1         61      C            TANK WALL TEMP.AT STEAM VOLUME
300 TTW2         62      C            -    -    -     -  WATER -
310 RTVTF        63      PCM          REACTIVITY FILTERED, 1/(1+S)
320
330 ALGEBRAIC VARIABLES   60
340 ------------------------
350 NAME         ARE     DIMENSION    TEXT
360 --------------------------------------------------------------------------
370 TCA(10)      1-10    C            CANNING TEMPERATURE
380 NPD(10)      11-20   MW           CORE AXIAL POWER WITH RESIDUAL HEAT
390 XEN(10)      21-30   CMe-3        XENON CONCENTRATION
400 TRCI         31      C            TEMP. IN RC-LINE AT TANK INLET
410 TCS          32      C            SATURATION TEMP.IN THE CORE
420 TES          33      C            -           -    -  STEAM DOME
430 QND          34      MW           NUCLEAR POWER WITHOUT RESIDUAL HEAT
440 MVOID        35                   CORE AVERAGE VOID
450 TPUI         36      C            INLET TEMP.TO HC-PUMPS
460 WGE          37      KG/S         STEAM FLOW TO STEAM DOME
470 WGR          38      KG/S         STEAM FLOW IN RISER
480 WFR          39      KG/S         WATER FLOW -  -
490 WR           40      KG/S         MASS FLOW  -  -
500 WT           41      KG/S         WATER FLOW TO TOP VOLUME
510 RTVT         42      PCM          REACTIVITY
520 CNSF         43                   ITERATION FACTOR FOR POWER
530 FLUX         44      CMe-2 Se-1   MEAN NEUTRON FLUX,CALIBRATED
540 DBT          45      S            NEUTRON FLUX DOUBLING TIME
550 REACTI       46      PCM          REACTIVITY WITHOUT NEUTRON SOURCE
560 TCM          47      C            CORE COOLANT MEAN TEMP.
```

```
 570 EMPTY       48                     NOT USED
 580 PHI(12)     49-60   CMe-2 Se-1  AXIAL NEUTRON FLUX
 590
 600 INPUT VARIABLES   30
 610 -------------------
 620 NAME         XRE     DIMENSION    TEXT
 630 ---------------------------------------------------------------------
 640 WLR          1       KG/S         TOTAL STEAM LOAD
 650 WFE          2       KG/S         FEEDWATER FLOW
 660 XSKD         3                    STEAM STOP VALVE OPENING
 670 TREK         4       C            TEMP.OF SHUTDOWN COOLING WATER
 680 WREK         5       KG/S         FLOW IN SHUTDOWN COOLER CIRCUIT
 690 WLOS         6       KG/S         SMALL BREAK FLOW OUT FROM RC-LINE
 700 PCCR(24)     7-3C    %            POSITION OF CONTROL RODS (24 GROUPS)
 710
 720 PARAMETERS   12
 730 ---------------
 740 NAME         CRE     DIMENSION    TEXT
 750 ---------------------------------------------------------------------
 760 STI          1                    STEADY STATE INDICATOR
 770 QSR          2       MW           -     -     POWER SETPOINT
 780 CQN          3                    -     -     ITERATION FACTOR,STI=1
 790 FCAL         4                    FLUX CALIBRATION FACTOR
 800 SOUR         5                    NEUTRON SOURCE TERM
 810 TAUR         6       S            REACTIVITY FILTER TIME CONSTANT
 820 QRES         7       MW           CONSTANT RESIDUAL HEAT
 830 ORP          8                    RC-PUMP VELOCITY NORMALIZED
 840 TFE          9       C            FEEDWATER TEMPERATURE
 850 XENON        10      %            XENON STEADY STATE POWER LEVEL
 860 EMPTY(2)     11-12                NOT USED
 870
 880
 890
 900
 910 MODULE:      CR    CONTROL RODS , SUBROUTINE CR
 920 ==========================================
 930 STATE VARIABLES   3
 940 ------------------
 950 NAME         SCR     DIMENSION    TEXT
 960 ---------------------------------------------------------------------
 970 XCR          1       %            CONTROL ROD DISPLACEMENT
 980 TREF         2       C            TEMP.REFERENCE VALUE FOR CASE 104
 990 XQ           3       MW           VARIABLE IN TEMP.CONTROLLER CASE 104
1000
1010 ALGEBRAIC VARIABLES   30
1020 -----------------------
1030 NAME         ACR     DIMENSION    TEXT
1040 ---------------------------------------------------------------------
1050 PCR(24)      1-24    %            CONTROL ROD POSITIONS, OUT=100 %
1060 SUMCR        25      %            SUMMA CONTROL RODS OUT, 0-10900 %
1070 PREF         26      BAR          SATURATION PRESSURE FOR TREF
1080 PER          27      BAR          SETPPOINT FOR PRESSURE CONTROLLER
1090 EMPTY(3)     28-30                NOT USED
1100
1110 INPUT VARIABLES   12
1120 ------------------
```

```
1130 NAME         XCR     DIMENSION   TEXT
1140 ------------------------------------------------------------------------
1150 QN           1       MW          NUCLEAR POWER WITHOUT RESIDUAL HEAT
1160 PE           2       BAR         REACTOR PRESSURE
1170 WGE          3       KG/S        STEAM PRODUCTION IN REACTOR
1180 SS           4                   REACTOR SCRAM SIGNAL
1190 CRS          5                   ROD WITHDRAW STOP INDICATOR
1200 DBT          6       S           DOUBLING TIME
1210 RT           7       C           REACTOR TOP TEMP.
1220 EMPTY(5)     8-12                NOT USED
1230
1240 PARAMETERS    12
1250 ---------------
1260 NAME         CCR     DIMENSION   TEXT
1270 ------------------------------------------------------------------------
1280 STI          1                   STEADY STATE AND TRANSIENT INDICATOR
1290 QSR          2       MW          -     -     POWER SETPOINT
1300 WLSR         3       KG/S        -     -     STEAM LOAD SETPOINT
1310 CQCR         4                   -     -     ITERATION FACTOR,STI=2
1320 CPCR         5                   -     -     -          -     ,STI=3
1330 NGR          6                   TRANSIENT PAR.,NUMBER OF GROUPS
1340 GRP          7                   -         -  ,GROUP NO.
1350 FCR          8       %           -         -  ,% CR-WITHDRAWING
1360 VCR          9       %           -         -  ,CR-SPEED AS % OF MAX.
1370 TG1          10      C/H         TEMP. RATE FOR P<PTG, CASE 104
1380 TG2          11      C/H         -     -     -  P>PTG, CASE 104
1390 PTG          12      BAR         PRESSURE LIMIT FOR TEMP.RATE,CASE 104
1400
1410
1420
1430
1440 MODULE:      FC    FEEDWATER CONTROL SYSTEM , SUBROUTINE FC
1450 ====================================================
1460 STATE VARIABLES   6
1470 -------------------
1480 NAME         SFC     DIMENSION   TEXT
1490 ------------------------------------------------------------------------
1500 WFE          1       KG/S        FEEDWATER FLOW
1510 XH(3)        2-4                 CONTROLLER VARIABLES,XH(3)=VALVE POS.
1520 V42          5                   BYPASS VALVE OPENING
1530 V23          6                   CONTAINMENT VALVE OPENING
1540
1550 ALGEBRAIC VARIABLES   0
1560 -----------------------
1570
1580 INPUT VARIABLES   6
1590 -------------------
1600 NAME         XFC     DIMENSION   TEXT
1610 ------------------------------------------------------------------------
1620 WLEV         1       M           REACTOR WATER LEVEL
1630 PE           2       BAR         -       PRESSURE
1640 WLR          3       KG/S        -       STEAM LOAD
1650 SS           4                   REACTOR SCRAM SIGNAL
1660 ASKH         5                   INDICATOR FOR ISOLATION VALVE OPEN
1670 P12          6                   -          -   PUMPS 1-2 RUNNING
1680
```

```
1690
1700 PARAMETERS   6
1710 -------------
1720 NAME         CFC     DIMENSION   TEXT
1730 ------------------------------------------------------------------------
1740 GL           1                   GAIN IN LEVEL LOOP
1750 GW           2                   -    -  FLOW FEEDBACK LOOP
1760 WFER         3       KG/S        FEEDWATER SETPOINT FOR MAN.CONTROL
1770 EMPTY(3)     4-6                 NOT USED
1780
1790
1800
1810 MODULE:      PC    PRESSURE CONTROL SYSTEM , SUBROUTINE PC
1820 ===========================================================
1830 STATE VARIABLES   9
1840 -------------------
1850 NAME         SPC     DIMENSION   TEXT
1860 -------------------------------------------------------------------------
1870 XG(3)        1-3                 COARSE CONTROLLER VARIABLES
1880 XH           4                   HYDRAULIC  -      -
1890 XBPV         5                   DUMP VALVE OPENING
1900 XBL(2)       6-7                 BLOWDOWN CONTROL VAR.,XBL(2)=VALVE POS.
1910 WSL          8       KG/S        TOTAL STEAM FLOW
1920 PES          9       BAR         PRESSURE CONTROLLER SETPOINT
1930
1940 ALGEBRAIC VARIABLES   0
1950 ----------------------
1960
1970 INPUT VARIABLES   12
1980 ------------------
1990 NAME         XPC     DIMENSION   TEXT
2000 -------------------------------------------------------------------------
2010 PE           1       BAR         REACTOR PRESSURE
2020 QN           2       MW          NUCLEAR POWER WITHOUT RESIDUAL HEAT
2030 PER          3       BAR         REACTOR PRESSURE SETPOINT
2040 XSKD         4                   STEAM ISOLATION VALVE OPENING
2050 DUMP         5                   DUMP CONTROL INDICATOR
2060 A314         6                   BLOWDOWN  -  -
2070 I314         7                   START 314 BLOWDOWN SYSTEM FOR I314>0
2080 WLX          8       KG/S        STEAM FLOW DISTURBANCE
2090 EMPTY(4)     9-12                NOT USED
2100
2110 PARAMETERS  6
2120 ------------
2130 NAME         CPC     DIMENSION   TEXT
2140 ------------------------------------------------------------------------
2150 PRATE        1       BAR/S       MAX RATE FOR PRESSURE SETPOINT
2160 EMPTY(5)     2-6                 NOT USED
2170
2180
2190
2200 MODULE:      FL   FLUX MONITORS AND LIMIT CONTROLS, SUBROUTINE FL
2210 ==================================================================
2220 STATE VARIABLES   3
2230 ------------------
2240 NAME         SFL     DIMENSION   TEXT
```

```
2250 ------------------------------------------------------------------------
2260 X1           1                    FILTER VARIABLE
2270 PRMF         2        %           PRM-SIGNAL FILTERED
2280 DBTR         3        1/S         DOUBLING TIME RECIPROCAL FILRERED
2290
2300 ALGEBRAIC VARIABLES   30
2310 -----------------------
2320 NAME         AFL      DIMENSION   TEXT
2330 ------------------------------------------------------------------------
2340 SS4          1                    SS4 INDICATOR
2350 SS5          2                    SS5 -
2360 SS6          3                    SS6 -
2370 SS7          4                    SS7 -
2380 SS8          5                    SS8 -
2390 SS11         6                    SS11 -
2400 E2           7                    E2 INDICATOR
2410 E3           8                    E3 -
2420 S4           9                    S4 INDICATOR
2430 S5           10                   S5 -
2440 S6           11                   S6 -
2450 S7           12                   S7 -
2460 S8           13                   S8 -
2470 S10          14                   S10 -
2480 S11          15                   S11 -
2490 SS           16                   SS INDICATOR
2500 E            17                   E  -
2510 S            18                   S  -
2520 PRML         19                   INDICATOR FOR PRM > 5 %
2530 SRM          20       C/S         SRM SIGNAL
2540 IRM          21       %           IRM -
2550 PRM          22       %           PRM -
2560 IRMC         23                   IRM CHANNEL
2570 DBTF         24       S           DOUBLING TIME AFTER FILTER (1/DBTR)
2580 SRMLG        25                   LOG OF SRM
2590 EMPTY(5)     26-30                NOT USED
2600
2610 INPUT VARIABLES   6
2620 ------------------
2630 NAME         XFL      DIMENSION   TEXT
2640 ------------------------------------------------------------------------
2650 FLUX         1        CMe-2 Se-1  NEUTRON FLUX
2660 WLEV         2        M           REACTOR WATER LEVEL
2670 PE           3        BAR         REACTOR PRESSURE
2680 DBT          4        S           NEUTRON FLUX DOUBLING TIME
2690 XXX          5                    INDICATOR FOR SRM DETECTOR INSERTED
2700 YYY          6                    -         -    IRM -        -
2710
2720 PARAMETERS   6
2730 -------------
2740 NAME         CFL      DIMENSION   TEXT
2750 ------------------------------------------------------------------------
2760 STI          1                    STEADY STATE INDICATOR
2770 ARS          2                    INDICATOR FOR AUTO IRM SCALING
2780 TAUD         3        S           DOUBLING TIME FILTER TIME CONSTANT
2790 NOSS         4                    INDICATOR FOR SS-BYPASS
2800 NOS          5                    -         -    S-BYPASS
```

```
2810 SSD         6                     -         -   SS MANUAL DEMAND
2820
2830
2840
2850
2860 MODULE:     RK   RESIDUAL HEAT REMCVAL MODEL, SUBROUTINE RK
2870 =======================================================
2880 STATE VARIABLES   4
2890 -------------------
2900 NAME        SRK     DIMENSION   TEXT
2910 ----------------------------------------------------------------------
2920 TRO         1       C           PRIMARY COOLANT OUTLET TEMP.
2930 THO         2       C           SEAWATER OUTLET TEMP.
2940 THC         3       C           INTERMEDIATE COOLANT TEMP.COLD SIDE
2950 THH         4       C           -             -       -    HOT  -
2960
2970 ALGEBRAIC VARIABLES   0
2980 -----------------------
2990
3000 INPUT VARIABLES   6
3010 -------------------
3020 NAME        XRK     DIMENSION   TEXT
3030 ---------------------------------------------------------------------
3040 TRI         1       C           PRIMARY COOLANT INLET TEMP.
3050 WREK        2       KG/S        REACTOR COOLANT FLOW
3060 EMPTY(4)    3-6                 NOT USED
3070
3080 PARAMETERS   6
3090 --------------
3100 NAME        CRK     DIMENSION   TEXT
3110 ----------------------------------------------------------------------
3120 THI         1       C           SEAWATER INLET TEMP.
3130 NHEX        2                   NUMBER OF ACTIVE LOOPS (0, 1, 2)
3140 EMPTY(4)    3-6                 NOT USED
3150
3160
3170
3180 MODULE:      CON    CONNECTING SYSTEM , SUBROUTINE CONSYS
3190 =====================================================
3200 STATE VARIABLES   0
3210 -------------------
3220
3230 ALGEBRAIC VARIABLES   0
3240 -----------------------
3250
3260 INPUT VARIABLES   0
3270 -------------------
3280
3290 PARAMETERS  12
3300 --------------
3310 NAME        CCON    DIMENSION   TEXT
3320 ----------------------------------------------------------------------
3330 ASKH        1                   INDICATOR FOR FEEDWATER STOP VALVE
3340 XSKD        2                   -         -   STEAM LINE -    -
3350 PER         3                   PRESSURE SETPOINT
3360 WREK        4                   RESIDUAL HEAT COOLANT FLOW
```

```
3370 DUMP          5                    INDICATOR FOR STEAM DUMP CONTROL
3380 A314          6                    -          -  -      BLOWDOWM -
3390 I314          7                    -          -  STEAM BLOWDOWN
3400 WLX           8                    STEAM LOAD DISTURBANCE
3410 WLXT          9                    -    -    -             RAMP TIME
3420 P12           10                   INDICATOR FOR FEEDWATER PUMP 1-2
3430 DRAIN         11                   DRANAIGE FROM REACTOR VESSEL
3440 EMPTY         12                   NOT USED
```

## APPENDIX D

Input file for the same system as used in Appendix B

```
100 & DATA FOR BARSEBAECK LOW POWER MODEL.
110 *INPT
120  BSO-MODEL. CASE 105. TEST ON PC COMPUTER.
130 *SYST
140  BSO          7
150 RE           CR             FC             PC             FL             RK
160 CON
170 *INCO
180  RE          63             60
190  6.80000E+01 6.80000E+01 6.80000E+01 2.77299E+02 2.78057E+02 2.79025E+02
200  2.79889E+02 2.80496E+02 2.80906E+02 2.61051E+02 2.81071E+02 2.81062E+02
210  2.81043E+02 1.00000E-20 1.00000E-20 9.39502E-03 3.66348E-02 7.55238E-02
220  1.13070E-01 1.50814E-01 1.84858E-01 2.08845E-01 2.22072E-01 2.89841E+02
230  3.01692E+02 3.12658E+02 3.18735E+02 3.18936E+02 3.14953E+02 3.10099E+02
240  3.06616E+02 3.00167E+02 2.91944E+02 8.86739E-02 1.68632E-01 2.57664E-01
250  3.16213E-01 3.21243E-01 2.84418E-01 2.38989E-01 2.08568E-01 1.52880E-01
260  8.40151E-02 2.76898E+02 2.76898E+02 2.76898E+02 2.77638E+02 2.80798E+02
270  2.80742E+02 2.76866E+02 2.76822E+02 2.76898E+02 4.10000E+00 0.00000E+00
280  6.51949E+01 6.50000E+01 1.96827E-01 1.77036E+03 0.00000E+00 0.00000E+00
290  2.74438E+02 2.70631E+02-2.18603E-07
300  2.78719E+02 2.80745E+02 2.81818E+02 2.82195E+02 2.82318E+02 2.82296E+02
310  2.82201E+02 2.82112E+02 2.81922E+02 2.81630E+02 3.37802E+00 6.39165E+00
320  9.45815E+00 1.12415E+01 1.12667E+01 1.00268E+01 8.54501E+00 7.49333E+00
330  5.56369E+00 3.13518E+00 4.78510E+13 7.10908E+13 8.70883E+13 9.37310E+13
340  9.29699E+13 8.75408E+13 8.04268E+13 7.49664E+13 6.37867E+13 4.42604E+13
350  2.76898E+02 2.80940E+02 2.80742E+02 6.80000E+01 1.00121E-01 2.76822E+02
360  2.58896E+01 2.57431E+01 1.74462E+03 1.77036E+03 1.74429E+03 9.63496E-07
370  9.70000E-01 3.30311E+12 1.02508E+05-2.63952E-05 2.79990E+02 0.00000E+00
380  1.07961E+11 2.69249E+12 5.12668E+12 7.87820E+12 9.82469E+12 1.02081E+13
390  9.24004E+12 7.93852E+12 7.05028E+12 5.24238E+12 2.90393E+12 1.56943E+11
400  CR           3             30
410  0.00000E+00 2.80742E+02 6.80000E+01
420  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
430  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
440  6.63592E+01 1.00000E+02 1.00000E+02 3.00000E+01 8.00000E+01 1.00000E+02
450  1.00000E+02 1.00000E+02 1.00000E+02 1.00000E+02 0.00000E+00 0.00000E+00
460  5.50544E+03 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
470  FC           6             0
480  2.58895E+01 4.57143E-01 2.58895E-01 3.46134E-01 1.00000E-10 1.00000E+00
490  PC           9             0
500  6.27284E-02-1.22647E-10 6.27284E-02 6.27284E-02 1.20697E-01 0.00000E+00
510  0.00000E+00 2.58896E+01 6.50000E+01
520  FL           3             30
530  3.96056E+00 3.96056E+00 9.75531E-06
540  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
550  0.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
560  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
570  0.00000E+00 1.00000E+06 1.93432E+01 3.96056E+00 1.10000E+01 1.02508E+05
580  6.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
590  RK           4             0
600  2.00000E+01 2.00000E+01 2.00000E+01 2.00000E+01
610 *DATA
620  STI.RE=     105.          QSR.RE=      0.            CON.RE=      1.E-3
630  FCAL.RE=    .485          SOUR.RE=     45.           TAUR.RE=     1.
640  QRES.RE=    8.5           ORP.RE=      .20           TFE.RE=      20.
```

```
 650  XENON.RE     0.
 660  STI.CR=      105.         QSR.CR=      68.0         WLSR.CR=     0.
 670  CQCR.CR=     1.           CPCR.CR=     1.E-2        NGR.CR=      0
 680  GRP.CR=      16           FCR.CR=      00.          JCR.CR=      000.
 690  TG1.CR=      0.           TG2.CR=      0.           PTG.CR=      0.
 700  GL.FC=       2.           GV.FC=       1.           WFER.FC=     0.
 710  PRATE.PC=    .05
 720  STI.FL=      105.         ARS.FL=      1.           TAUD.FL=     1.
 730  NOSS.FL=     0            NOS.FL=      0            SSD.FL=      0
 740  THI.RK=      17.          NHEX.RK=     0
 750  ASKH.CON=    1.           XSKD.CON=    1.           PER.CON=     65.
 760  WREK.CON=    0.           DUMP.CON=    1.           A314.CON=    0.
 770  I314.CON=    0            WLX.CON=     0.           WLXT.CON=    1.
 780  P12.CON=     1.           DRAIN.CON=   0
 790  DATA.END
 800 *CHCK
 810  .001         .200         .200         1.E-3        10.          HEUN
 840 *REFV
 850  XQN1.RE      XQN3.RE      50.
 860  ALF1.RE      ALF10.RE     .25
 870  CN11.RE      CN110.RE     .20
 880  PP.RE        PP.RE        1.
 890  ALFR.RE      ALFR.RE      .25
 900  YHC1.RE      YHC2.RE      10.
 910  RTVTF.RE     RTVTF.RE     200.
 920  XCR.CR       XCR.CR       10.
 930  TREF.CR      XQ.CR        100.
 940  WFE.FC       WFE.FC       10.
 950  XH1.FC       V23.FC       1.
 960  XG1.PC       XBL2.PC      1.
 970  WSL.PC       PES.PC       10.
 980  X1.FL        PRMF.FL      2.
 990  DBTR.FL      DBTR.FL      .10
1000  REFV.END
1010 *PRNT
1020  1.0          60.          1.           120.         5.
1030  QND.RE       PR.RE        PE.RE        WRC.RE       WR.RE        WSL.PC
1040  WGE.RE       WFE.FC       ALFR.RE      TRCI.RE      TB.RE        TR.RE
1050  TES.RE       WLEV.RE      FLUX.RE      SUMCR.CR     DBT.RE       SRM.FL
1060  IRM.FL       IRMC.FL      PRM.FL       RTVTF.RE     REACTI.RE    DBTR.FL
1070  SRMLG.FL     MVOID.RE     TCM.RE
1080  PRNT.END
1090 *DELY
1100 *STST
1105  200
1110 *ENDE
```

## APPENDIX E

### List of files used by DYSIM

Program files: Main program unit and subroutines

| | |
|---|---|
| DYS/DYSIM86 | : Main program and subroutine for integration and input-output |
| DYS/DELAY86 | : Time delay function |
| DYS/ODEPACK | : Subroutine for integration by ODEPACK methods |
| DYS/SYST | : Subroutine for interpretation of a list file |
| DYS/LIST | : List file editor |
| DYS/INPUT | : Input file editor |
| DYS/EDIT | : Local editor for use in DYS/LIST and DYS/INPUT |
| DYS/JACOBI | : The Jacobian service routine |
| DYS/PLOT | : Display and plotting program for transient time functions |
| DYS/PRE | : Precompiler 1 and 2 |

DYSIM works with the following files:

```
2 : SYNA/JACO      : File with the Jacobian matrix
3 : SYNA/INPUT     : Input file for the model
5 : Used for user communication
6 : -    -    -    -
8 : SYNA/DELAY     : Buffer file for time delay function
9 : SYNA/DELAYDUMP: Dump file for time delay function
10: Printer file
11: SYNA/OUTPUT    : Output buffer for DYSIM and default plot
                     file
12: SYNA/IC        : Initial condition output file
15: -              : -       -          input   -
13: SYNA/DUMP      : Dump file for DYSIM output
14: -              : -    -     -    -     input
16: SYNA/LIST      : List file for the model
```

SYNA stands for the system name. The precompilers also use some input-output files. No names shall be given here as the work is incomplete at present and will be described separately.

A system model may consist of files as:

```
SYNA/CONSYS: Connecting routine
SYNA/MOD1   : Module 1
SYNA/MOD2   : -      2
....
SYNA/MODn   : -      n
SYNA/INPUT  : Input file
SYNA/LIST   : List file
```

If the precompilers are used the module files and the connecting routine are present in a source (macro) version; in any case they must occur in a Fortran 77 version.

## APPENDIX F

### Reserved subroutines and function names

DEADTM
DELAY
DUMP
FILNAM
HEUN
ICCAL
INTEG
INPUT
JACOBI
MANA
NAMES
NSTOP
ODEPAK
OUTENT
OUTPUT
READ3
RECALL
REPET
PRINT
RUNGE
SERVER
SHIL
SHIR
STEPCO
SYST
TERM
TRNSTM
VADR

If Odepack is used the following names are also reserved

CFODE
EWSET
F
INTDY
JAC
LSODA
LSODE
ODEERR
PREPJ
PRJA
RSCMA
RSCOM
SAXPY
SGBFA
SGBSL
SGEFA
SGESL
SOLSY
SSCAL
STODA
STODE
SUCMA
SVCOM
XERRMV
XSETF
XSETUN

## APPENDIX G

<u>List of system calls available for the user</u>

| | | |
|---|---|---|
| CALL TERM(TEX) | : | Stop calculations at the end of first accepted integration step. |
| CALL NSTOP(TEX) | : | Stop calculations at once independent of substep number |
| CALL REPET | : | Cancel the step in progress and make a new one with half the step size |
| CALL DUMP | : | Dump system state at termination time |
| CALL RECALL | : | Ask DYSIM to execute the statement CALL DYNAMS at the end of first accepted step. DYNAMS must be a user supplied subroutine. |
| CALL PRINT | : | Print variable values as they were at the beginning of this time step |

Note: CALL YOUT is executed by DYSIM at the end of any calculation. The user must supply the subroutine YOUT.

## APPENDIX B

**Definition of concepts used in DYSIM**

| | |
|---|---|
| System: | The whole model simulated by use of DYSIM. |
| Module: | A well delimited unit or part of a system described in an independent subroutine with input-output interface to the rest of the system. |
| Submodule: | A smaller well delimited unit programmed beforehand and stored in a library. |
| Connecting routine: | A subroutine which establishes the input-output connections between the modules and links the whole system to DYSIM. Perturbation of the system is also programmed here. |
| State variables: | Variables given by differential equations. |
| State derivatives: | Derivatives of state variables. |
| Algebraic variables: | Auxiliary variables given by algebraic equations and available for output. |
| Input variables: | Input variables to the modules either given as output variables from other modules or calculated in the connecting routine. |
| Parameter: | Constant parameters for the modules and the connecting routine given in the input file. |
| Output variable: | A state variable or an algebraic variable. |

| | |
|---|---|
| Input file: | A formatted text file with specifications for the system and the particular calculation. It also contains initial values for the state and algebraic variables. |
| List file: | A formatted text file with names and definitions of all variables and parameters in the system available for the user. The list is set up in groups, one group for each module. |
| Macro call: | A statement in the source code which is transferred to a set of statements in the parent language, here Fortran 77. The macro statement normally has a special character as the first one followed by the macro name and some parameters in one or more lines. |
| Source module (file): | A module (file) written with use of macro statements. It must be passed through a compiler (here precompiler 1 or 2) which creates a module (file) in the parent language. |

**Risø National Laboratory** **Risø – M – 2607**

| | |
|---|---|
| **Title and author(s)**<br>DYSIM - A MODULAR SIMULATION SYSTEM FOR CONTINUOUS DYNAMIC PROCESSES<br><br>P. la Cour Christensen<br>J.E. Koefoed<br>N. Larsen | **Date** 1 December 1986 |
| | **Department or group**<br>Energy Technology |
| | **Groups own registration number(s)** |
| | **Project/contract no.** |
| **Pages** 72 **Tables** **Illustrations** 2 **References** 4 | **ISBN** 87-550-1265-5 |

**Abstract (Max. 2000 char.)**

The report describes a revised version of a simulation system for continuous processes, DYSIM. In relation to the previous version, which was developed in 1981, the main changes are conversion to Fortran 77 and introduction of a modular structure. The latter feature gives the user a possibility for decomposing the model in modules corresponding to well delimited physical units, a feature which gives a better survey of the model. Furthermore, two new integration routines are included in addition to the single one used before.

**Descriptors** - INIS:
COMPUTERIZED SIMULATION; D CODES; DIFFERENTIAL EQUATIONS; DYNAMICS; FORTRAN; INDUSTRIAL PLANTS; NUCLEAR POWER PLANTS; STEADY-STATE CONDITIONS; TRANSIENTS

**Available on request from Risø Library, Risø National Laboratory, (Risø Bibliotek, Forskningscenter Risø), P.O. Box 49, DK-4000 Roskilde, Denmark.
Telephone 02 37 12 12, ext. 2262. Telex: 43116, Telefax: 02 36 06 09**

RR950000 99