CLASS COMBINEDSIMULATION

Installation Guide

This software contains the SIMULA-class `COMBINEDSIMULATION` and its extension, class `DISCO` (version March 1982).

There are 38 files containing the following information:

File	Name	Contents
1	DOC/INSTALL.pdf	Installation guide
2	DOC/USERGUIDE.pdf	User's guide to `COMBINEDSIMULATION`
3	DOC/MANUAL.pdf	Reference manual to `COMBINEDSIMULATION`
4	SRC/SIMULATION.sim	The source text of `COMBINEDSIMULATION`
5	SRC/PLOTGUIDE .pdf	User's guide to `PLOTENVIRONMENT`
6	SRC/PLOTENVIRONMENT.sim	The source text of `PLOTENVIRONMENT`
7	SRC/TEST.sim	A test program for `COMBINEDSIMULATION`
8	SRC/SYSTEM.sim	Class `SYSTEM` - a class for table-lookup
9	EXAMPLES/EX01.sim:	Prey-predator system
10	EXAMPLES/EX02.sim:	Launch of a three-stage rocket
11	EXAMPLES/EX03.sim	World model
12	EXAMPLES/EX04.sim:	World model with discrete events
13	EXAMPLES/WORLDDATA	Data for example EX03 and EX04
14	EXAMPLES/EX05.sim	Ingot heating system
15	EXAMPLES/EX06.sim	Chemical reaction process
16	EXAMPLES/EX07.sim	Higher-order differential equations
17	EXAMPLES/EX08.sim:	Van der Pol's equation
18	EXAMPLES/EX09.sim:	Exponential delays

19	EXAMPLES/EX10.sim	Ideal delays
20	EXAMPLES/EX011.sim	Collection of statistics
21	DOC/DISCOGUIDE.pdf	User's guide to `DISCO`
22	SRC/DISCO.sim	The source text of `DISCO`
23	EXAMPLES/EX12.sim	Hydraulic servo drive
24	EXAMPLES/EX13.sim	Fire-fighting
25	EXAMPLES/EX14.sim	Pilot ejection study
26	EXAMPLES/EX15.sim	`DOMINO` game
27	EXAMPLES/EX16a.sim	`PHYSBE` (version A)
28	EXAMPLES/EX16b.sim	`PHYSBE` (version B)
29	SRC/WAITGREATER.sim	Procedure `WAITGREATER`
30	SRC/NODISCO.sim	Class `NODISCO` - the user-attributes of `DISCO`
31	SRC/XSIMULATION.sim	An extended version of `COMBINEDSIMULATION`
32	DOC/XUSERGUIDE.pdf	Guide to extended version of `COMBINEDSIMULATION`
33	SRC/XDISCO.sim	Revised version of `DISCO`
34	SRC/PDEONE.sim	`PDEONE` - Solution of Partial Differential Equations
35	DOC/PDEONEGUIDE.pdf	User's guide to `PDEONE`
36	EXAMPLES/EX17.sim	Central heating of a building
37	SRC/XXSIMULATION.sim	`COMBINEDSIMULATION` - stiff differential equations
38	SRC/XXDISCO	`DISCO` - stiff differential equations

Comments:

File

2 This guide is intended to give a SIMULA-user sufficient information to express models in the terminology and notation provided by COMBINEDSIMULATION. An introduction may also be found in ref. 1, 2 and 18.

A detailed documentation of COMBINEDSIMULATION is given in file 4. The interpolation method used is described separately in ref. 6.

3 This file contains a reference manual to class COMBINEDSIMULATION.

4 COMBINEDSIMULATION is written in Common Base Language. The only machine dependence is found in the procedure MAXREAL which returns the maximum real-value expressible on the computer. This procedure (line 4) probably should be rewritten.

The class has been prefixed by the class PLOTENVIRONMENT. If the facilities of the latter class are not desired, prefixing with class SIMULATION will suffice.

5 This guide to class PLOTENVIRONMENT is very brief, but should be sufficient for most applications.

6 PLOTENVIRONMENT is written in Common Base Language. However, some caution is in order:

(1) Temporary direct files are automatically created to be used for storing intermediate plot information. Each PRINTERPLOT-object has associated with it a direct file which contains the plot points.

"PLOT1" is the name of the first file created, "PLOT2" is the name of the second, and so on. However, the direct file associated with the system-defined PRINTERPLOT-object SYSPLOT is given the name "SYSPLOT".

Moreover, a temporary direct file named "COMPLOT" is used in case the X-values of the plot are not monotonically increasing.

These conventions for naming files may possibly cause a little trouble when the class is installed.

(2) MAXRANK denotes the maximum character rank. Its value has been set to 127. Possibly this setting is inexpedient and should be corrected (line 439).

(3) The line length of the printer device must be at least 122 characters (at least if 5 significant digits are wanted for the X-Y-values).

PLOTENVIRONMENT has been prefixed by class SIMULATION. None of the latter's facilities, however, are used in the implementation.

7 It is highly recommended to run this test program when COMBINEDSIMULATION is installed.

8 The class SYSTEM extends COMBINEDSIMULATION with facilities for table look-up.

The user has the following view of the class:

```
COMBINEDSIMULATION class SYSTEM;
begin
  class TABLE;
  begin
    procedure ADD(X,Y); real X,Y; ... ;
    procedure VAL(X); real X; VAL:= ... ;
  end;
end;
```

A function table is represented by a TABLE-object.

An entry (X,Y) is added to the table by calling ADD(X,Y). The entries can be added in arbitrary order. The X-es must be distinct, but need not be uniformly spaced.

A call VAL(X) returns the Y-value corresponding to X using linear interpolation.

If X is less than the minimum X-value of the table, XMIN, then VAL(X)=VAL(XMIN). If X is greater than the maximum X-value, XMAX, then VAL(X)=VAL(XMAX).

Example 3 (in file 11) illustrates the use of the facilities of class SYSTEM.

9 This example is a simulation of a simple ecosystem. The cycle time of the system is determined by using the procedure WAITUNTIL.

10 This is the rocket example described in the User's guide (in file 2).

11 The world model presented in this example was defined by Forrester and has been described in detail in ref. 7.

As far as possible Forrester's nomenclature has been followed. All five level equations of the model have been expressed as first-order differential equations (lines 21-65).

Alternatively, the equations could have been expressed as difference equations:

```
P.STATE    := P.LASTSTATE+DT*(BR-DR);
NR.STATE   := NR.LASTSTATE+DT*(-NRUR);
CI.STATE   := CI.LASTSTATE+DT*(CIG-CID);
POL.STATE  := POL.LASTSTATE+DT*(POLG-POLA);
CIAF.STATE := CIAF.LASTSTATE+DT*(CFIFR*CIQR-CIAF)/CIAFT;
```

where `P` denotes population, `NR` natural resources, `CI` capital investment, `POL` pollution and `CIAG` capital investment-in-agriculture fraction. The step-size, `DT`, will remain constant at `DTMAX`. Efficiency may be gained by setting `EULER` to true in this case.

Data for the simulation may be found in file 13.

12 The model of example 5 is extended to include the following discrete events:

Time-events:

- food shortage (occurs every 20 years and lasts 4 years)
- discovery of new resources (occurs in 1975)
- a world epidemic (occurs in 1980)

State-events:

- overpopulation (causes birth control)
- overpollution (causes legislation)
- resource shortage (causes conservation)

The model has been taken from ref. 8, pp. 285-304.

Data for the simulation may be found in file 13.

13 This file contains data for the world models in example 3 and 4.

14 This example of simulating the heating of steel ingots in a furnace of limited capacity has been taken from ref. 9. It can also be seen in ref. 1.

15 This example has been taken from ref. 10. As may be seen by this example `COMBINEDSIMULATION` provides a much more convenient language than GASP (which is FORTRAN-PL/I-based).

16 This example illustrates how a general class of continuous processes may be defined for the solution of higher-order differential equations. Normally a user has to rewrite these equations into first-order differential equations. The class FUNCTION frees the user from this task.

The user's view of the class is:

```
class FUNCTION(N); integer N;
begin
  ref(VARIABLE) array D(0:N);
  procedure START; … ;
  procedure STOP; … ;
  Boolean procedure ACTIVE; ACTIVE:= … ;
end;
```

A FUNCTION-object represents a variable whose continuous change can be described by an equation involving an Nth order derivative. The D array of the object contains the derivatives: D(I).STATE contains the I'th derivative (D(0).STATE contains the function value itself).

D(I).RATE should not be used by the user.

It is assumed that no continuous process is given a priority less than -MAXREAL/2.

Example 8 (file 17) illustrates the use of class FUNCTION in solving a second-order differential equation.

17 This example numerically solves the second-order equation of Van der Pol:

$$\frac{d^2y}{dt^2} + E(1-y^2)\frac{dy}{dt} + y = 0$$

Using the class FUNCTION of Example 8 this equation may be written as:

```
Y.D(2).STATE:= -E*(1-Y.D(0).STATE**2)*Y.D(1).STATE-Y.D(0).STATE
```

where Y is a FUNCTION-object with parameter N=2. Note that Y must be STARTed to undergo continuous change (cf. VARIABLE-objects).

18 This example defines a class, EXPDELAY, for describing exponential delays of arbitrary order. The order of the delay, N, is the number of cascaded first-order delays that compose the delay in question.

The user's view of the class is:

```
class EXPDELAY(N,INITIALSTATE,TARGET,LAG);
integer N; real INITIALSTATE,LAG; ref(VARIABLE) TARGET;
begin
  real procedure STATE; ... ;
  procedure START; ... ;
  procedure STOP; ... ;
  Boolean procedure ACTIVE; ACTIVE:= ... ;
end;
```

The STATE of an ACTIVE EXPDELAY-object continuously approaches TARGET's STATE with the specified delay, LAG.

The initial value is a parameter, INITIALSTATE.

The TARGET may itself be moving (ACTIVE).

19 This example defines a class, IDELAY, for describing ideal delays.

The user's view of the class is:

```
class IDELAY(V,LAG); ref(VARIABLE) V; real LAG;
begin
  real procedure STATE; ... ;
  procedure START; ... ;
  procedure STOP; ... ;
  Boolean procedure ACTIVE; ACTIVE:= ... ;
end;
```

The STATE of an ACTIVE IDELAY-object lags behind the given variable V. The delay is specified through the parameter LAG. During the first LAG time units of an IDELAY-object's existence its STATE is equal to V's STATE at the creation of the IDELAY-object.

An IDELAY-object actually stores samples of V's STATE at the end of each integration step and interpolates linearly between sampled values.

20 This example defines a class, STATSIM, for collecting statistics about variables that vary with time.

The user's view of the class is the following:

```
COMBINEDSIMULATION class STATSIM;
begin
  VARIABLE class STATVAR;
  begin
    real procedure MIN; ... ;
    real procedure MAX; ... ;
    real procedure MEAN; ... ;
    real procedure SDV; ... ;
  end;
end;
```

In addition to its VARIABLE-attributes, each STATVAR-object has the property that its minimum (MIN), maximum (MAX), mean (MEAN) and standard deviation (SDV) are automatically computed during the simulation. The mean and standard deviation are estimated by trapezoidal integration.

21 Class DISCO extends class COMBINEDSIMULATION with facilities for:

- solution of higher-order differential equations
- table-lookup
- exponential and ideal delays
- formulation of implicit functions
- collection of statistics

This file contains a brief user guide to DISCO.

22 This file contains the source text of DISCO.

23 This hydraulic servo drive simulation example is taken from ref. 11. The equations to be simulated are:

```
yd=w-y
u =c1*yd

    1; (u>1)
x = u; (-1<=u<=1)
   -1; (u<-1)

s = c2*sign(p)*p

v1=  1+c3*abs(x)+c4*x**2;  (abs(x)<0.07)
     c5+c6*abs(x)+c7*x**2; (abs(x)>=0.07)

v2=  1+c8*abs(x)+c9*x**2;  (abs(x)<0.07)
               0         ;  (abs(x)>=0.07)

v = c10*sign(x)*(v1*sqrt(1-s)-v2*sqrt(1+s))
F = c11*p-c12*y'-Fl

    c13; (F>c13)
z =   F; (-c13<=F<=c13)
   -c13; (F<-c13)

    c13; (y'>0)
cf=   z; (y'=0)
   -c13; (y'<0)

p'=c14*v-c15*y'-c16*sign(p)*sqrt(abs(p))
y"=c17*(F-cf)
```

24 This example is a simulation of a fire station. The model is formulated by R.W. Sierenberg, Delft University of Technology, The Netherlands.

A small city owns one fire-station with three fire-engines. Fire alarms are given randomly at exponential distributed intervals with a mean of six hours.

Each house on fire contains a certain amount of inflammable material. When a fire is discovered, it already has a certain size. This size increases with a rate which is proportional to the size itself as long as no extinguishing activity takes place. At the moment an alarm is given, one engine (if possible) will be sent to the fire. When a fire-engine reaches the fire and finds out that its capacity is smaller than the rate with which the size increases, it will request assistance by sending an alarm for the same fire. If all inflammable material is burnt up, the fire will stop and the house is lost.

25 This example is a parametric study of an aircraft's pilot ejection system. The objective is to determine safe ejection as a function of aircraft altitude and velocity. The specific model and experiment is extracted from ref. 12.

26 This example has been suggested by F.E. Cellier (ref.13) as a benchmark problem which can be used to test the capability of a variable structure simulation, that is a simulation in which the number of differential equations varies with time.

Fifty five identical stones of the DOMINO game are placed upright in a sequence with a distance of D space units between any two stones. If the first stone is pushed, all the stones fall flat.

The aim of the simulation is to determine the distance (D) between successive stones which maximises the velocity (V) of the chain.

27 This file contains an implementation of PHYSBE.

PHYSBE is an acronym for Physiological Benchmark Simulation Experiment, and is a benchmark problem which has been widely used in comparing continuous simulation languages. It is a mathematical model of the human cardiovascular system and relates pressure, volume and flow.

The example demonstrates exploitation of SIMULA's class concept for "structured" modelling. A good description of the model is given in ref. 14.

28 This file contains another implementation of PHYSBE.

This implementation, however, cannot be recommended! Not only is it "unstructured", but, what is worse, numerical problems may arise because discrete changes are executed directly by a continuous process (the use of the MAX-function makes the second-order derivatives of the volumes discontinuous).

29 This file contains the procedure `WAITGREATER`.

The call `WAITGREATER(EXPR1,EXPR2,TOL)`, where `EXPR1` and `EXPR2` are arithmetic expressions, corresponds to the call `WAITUNTIL(EXPR1>=EXPR2)`. However, the state-event may take place a little bit earlier, namely when `EXPR1>=EXPR2-TOL`, or a little bit later, namely when `EXPR2<=EXPR1<= EXPR2+TOL`, where `TOL` is a positive tolerance. `WAITGREATER(EXPR1,EXPR2,0)` is fully equivalent to `WAITUNTIL(EXPR1>=EXPR2)`.

Weakening a state-condition by using `WAITGREATER` instead of `WAITUNTIL` may reduce the execution time significantly if the simulated system involves comparatively many state-events (as in example 15).

`WAITGREATER` is made available through the class `XDISCO`, an extension of class `DISCO`. The code is a bit tricky as it exploits intimate knowledge of `COMBINEDSIMULATION`'s algorithm for state-event detection. It is assumed that no state-event is given a priority less than `-MAXREAL/2`.

30 Class NODISCO only contains the user attributes of class DISCO. Class NODISCO may, for example, be used for the syntactical checking of DISCO-programs.

31 This file contains a new extended version of class COMBINEDSIMULATION. The most important extension is the provision of a predictor-corrector method, namely Adams method with variable step size and variable order (ref. 15).

32 A brief description of the extended version of class COMBINEDSIMULATION.

33 This file contains a revised version of class DISCO to be used together with the extended version of class COMBINEDSIMULATION (file 31). In relation to the original version of DISCO (file 22), only minor corrections in procedure DUMP have been made.

34 The source text of PDEONE - a class for solving Partial Differential Equations in ONE space dimension. PDEONE is a SIMULA implementation of the methods presented in reference 16.

35 User's guide to PDEONE.

36 This example demonstrates the use of PDEONE in a combined simulation involving partial differential equations. The example has been taken from reference 17 and has to do with the central heating of a building.

The building is modelled by a stick of length one. The left side of the stick represents the centre of the building where the heating takes place. The right side represents the walls of the building.

The temperature distribution of the building is modelled using the following heat diffusion equation:

$$\frac{dU}{dt} = 0.5 * \frac{d^2U}{dX^2}$$

The heating of the central room is modelled by the equation (left boundary condition)

$$U = 30 * Z$$

where Z is described by the ordinary first order differential equation

$$\frac{dZ}{dt} = 4(1-Z) \qquad \text{, if the heating is on}$$

$$\frac{dZ}{dt} = -4Z \qquad \text{, if the heating is off.}$$

At night time (between 7 p.m. and 7 a.m.) the heating is always off. During the day the heating is on when the wall temperature falls below 19.5 degrees centigrade, and is turned off as soon as the wall temperature goes above 22.5 degrees centigrade. The simulation starts at 6 a.m. with zero degrees centigrade throughout the whole building.

The model is far away from physical reality, but has been chosen intentionally since it demonstrates the use of partial differential equations, ordinary differential equations, time-events, as well as state-events in one single model.

37 This file contains an extended version of `COMBINEDSIMULATION` (file 31). The extension implements Fowler and Warten's second-order explicit integration method for the solving of "stiff" differential equations (ref. 19).

The user selects this method merely by setting the Boolean variable STIFF to true. Initially the value of `STIFF` is `false`. Step-size is variable and error control is the same as in the Runge-Kutta-England method and in the Adams method.

38 This is a version of class `DISCO` that corresponds to the extended version of `COMBINEDSIMULATION` in file 37.

The use of semi-discretization to solve partial differential equations usually gives rise to stiffness. It is therefore recommended to use this version of `DISCO` as a prefix for class `PDEONE` (file 34).

REFERENCES

1 Helsgaun, K.:
"COMBINEDSIMULATION - a SIMULA-class for combined continuous and discrete simulation".
Proceedings of the sixth SIMULA users' Conference,
Lisboa, 1978.

2 Helsgaun, K.:
"COMBINEDSIMULATION. Speech given at the sixth SIMULA users' Conference, Lisboa 1978".
Roskilde University Center, Roskilde, september 1978.

3 Helsgaun, K.:
"COMBINEDSIMULATION. User's manual". (in Danish)
Roskilde University Center, Roskilde, november 1978.

4 Helsgaun, K.:
"COMBINEDSIMULATION. Documentation". (in Danish)
Roskilde University Center, Roskilde, january 1979.

5 Helsgaun, K.:
"On interpolation in class COMBINEDSIMULATION". (in Danish)
Roskilde University Center, Roskilde, october 1978.

6 England, R.:
"Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations".
Computer Journal, Vol. 12, 1969, pp. 116-170.

7 Forrester, J.W.:
"World Dynamics".
Cambridge, Mass.: Wright-Allen Press, Inc., 1971.

8 Pritsker, A. A. B., and R. E. Young:
"Simulation with GASP_PL/1".
John Wiley & Sons, New York 1975.

9 Sim, R.:
"CADSIM. User's Guide and Reference Manual".
Imperial College. Publ. no. 75/23. London 1975.

10 Hurst, N. R., and A. A. B. Pritsker:
"Simulation of a Chemical Reaction Process Using GASP IV".
SIMULATION, Vol. 21, September 1973, pp. 71-75.

11 Halin, H. J.:
"Integration across discontinuities in ordinary differential equations".
SIMULATION, Vol. 32, February 1979, pp. 33-45.

12 SCi Simulation Software Committee:
"The SCi Continuous System Simulation Language (CSSL)".
SIMULATION, Vol. 9, December 1967, pp. 281-303.

13 Cellier, C.F.:
"Combined continuous discrete simulation by use of digital computers. Techniques and tools".
Ph.D. Thesis, The Swiss Federal Institute of Technology Zurich,
Zurich, 1979.

14 Benham, R.D.:
"An ISL-8 and ISL-15 study of the physiological simulation benchmark experiment".
SIMULATION, Vol. 18, no. 4, March 1972, pp. 152-156.

15 Shampine, L.F. and Gordon, M.K.:
"Computer solution of ordinary differential equations".
W.H. Freeman and Company, San Francisco, 1975.

16 Sincovec, R.F and Madsen, N.K.:
"Software for Nonlinear Partial Differential Equations".
ACM Transactions on Mathematical Software,
Vol. 1, No. 3, September 1975, pp. 232-260 and 261-263.

17 Cellier, F.E. and Blitz, A.E.:
"GASP-V: A universal simulation package".
Simulation of Systems, L. Dekker (editor),
North-Holland, 1976, pp. 391-402.

18 Helsgaun, K.:
"DISCO - a SIMULA-based language for combined continuous and discrete simulation".
SIMULATION, Vol. 34, no. 7, July 1980, pp. 1-12.

19 Fowler, M.E. and Warten, R.M.:
"A Numerical Intergration Technique for Ordinary Equations with Widely Separated Eigenvalues".
IBM J. Res. Develop. 11, September 1967, pp. 537-543.

RECOMMENDED INSTALLATION PROCEDURE

(1) Rewrite the procedure MAXREAL (line 3) of COMBINEDSIMULATION.

(2) Change the PLOTENVIRONMENT-prefix of COMBINEDSIMULATION to SIMULATION.

(3) Compile COMBINEDSIMULATION.

(4) Run the test program of file 7. The following message should result:

```
*** COMBINEDSIMULATION
*** NO ERRORS FOUND
```

If necessary, make the proper corrections.

(6) Change the prefix of COMBINEDSIMULATION back to PLOTENVIRONMENT.

(7) Compile COMBINEDSIMULATION together with PLOTENVIRONMENT.

(8) Compile class DISCO together with COMBINEDSIMULATION.

(9) Run some of the examples.

If your SIMULA system has double precision capabilities, it may be advantageous to change the type specifications of some of the real-variables into "long real". In class COMBINEDSIMULATION it is recommended specifying the following attributes as "long real"-variables:

STATE, OLDSTATE and EPSSTATE in class VARIABLE

If the "hidden-protected" facility is available, non-user attributes can be made inaccessible to the user. In class COMBINEDSIMULATION the following attributes should be hidden from the user:

OLDSTATE, EPSSTATE, DS, DSH, A1, A2, A3, A4, A5, PREDVAR,
SUCVAR (and PHI, if present) in class VARIABLE

FRQ, REPTIME, PREDREP, SUCREP and EXECUTE in class REPORTER

PRI, PREDCONT, SUCCONT and EXECUTE in class CONTINUOUS

THEMONITOR and the classes MONITOR, CONTROLLER1, CONTROLLER2 and WAITNOTICE in the main program.

If you have problems with installation, you are welcome to contact:

K. Helsgaun
Department of Computer Science, 20.1,
Roskilde University,
4000 Roskilde
Denmark

E-mail: keld@ruc.dk

You are also invited to send any proposal for improvement of the software found on this tape.