CLASS DISCO

User Guide

Class `DISCO` is an extension of class `COMBINEDSIMULATION` containing facilities for:

(1) Solution of higher-order differential equations (class `FUNCTION`)

(2) Table-interpolation (class `TABLE`)

(3) Description of exponential and ideal delays (class `EXPDELAY` and class `IDELAY`)

(4) Formulation of implicit functions (procedure `IMPLICIT`)

(5) Collection of statistics (class `STATVAR`, class `TALLY`, class `HISTOGRAM`, class `ACCUMULATE` and class `COUNT`)

(6) Program testing (procedure `DUMP`).

The following description of the facilities is very brief but should be sufficient for most applications.

(1) SOLUTION OF HIGHER-ORDER DIFFERENTIAL EQUATIONS

The solution of higher-order differential equations normally requires that the user rewrites the system as first-order equations. The class FUNCTION frees the user from this task.

The user's view of class `FUNCTION` 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 `N`th order differential equation. 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.

Example:

The second-order equation of Van der Pol:

$$d^2y/dt^2 + e(1-y^2)dy/dt + y = 0$$

may be represented by:

```
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 `START`ed to undergo continuous change (cf. `VARIABLE`-objects).

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

(2) TABLE-INTERPOLATION

Class `TABLE` facilitates table-lookup.

The user has the following view of the class:

```
class TABLE;
begin
  procedure ADD(X,Y); real X,Y; ... ;
  real procedure VAL(X); real X; VAL:= ... ;
  procedure READ(N); integer N; ... ;
  procedure PRINT; ... ;
  procedure PLOT; ... ;
  procedure SETTITLE(TITLE); value TITLE; text TITLE; ... ;
end;
```

A `TABLE`-object represents a function table.

An entry (`X`,`Y`) is added to the table by calling `ADD(X,Y)`. The entries may 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 of the table, `XMAX`, then `VAL(X)=VAL(XMAX)`.

A call `READ(N)` reads `N` table entries from `SYSIN` using `INREAL`.

The procedure `PRINT` may be used to print the contents of the table.

A line printer plot of the function defined by the table may be produced by calling procedure `PLOT`. The table-entries themselves are designated with an 'O' on the curve.

A print or plot of the table may be accompanied with an explanatory title. Calling `SETTITLE(T)` sets the title to T.

(3) DELAYS

Class `EXDELAY` may be used to describe 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 class `EXPDELAY` is:

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

The STATE of an ACTIVE EXPDELAY-object continuously approaches the TARGET's STATE with the specified delay, LAG. The initial value is a parameter, INITIALSTATE. The TARGET may itself be moving.

For describing ideal delays the class IDELAY is available.

The user's view of class `IDELAY` is:

```
class IDELAY(V,LAG); ref(VARIABLE) V; real LAG;
begin
  real procedure STATE; 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` actually stores samples of `V`'s `STATE` at the end of each integration step and interpolates linearly between sampled values.

(4) IMPLICIT FUNCTIONS

In describing continuous processes it is the user's responsibility that the equations are evaluated in the correct order: variables must be updated before they are used on the right-hand side of an equation.

The user can determine the order of evaluation within each continuous process and the continuous processes themselves may be ranked by giving them a priority. Usually by these means a correct evaluation order can be achieved. In some cases, however, this is impossible: there is an "algebraic loop" in the equation system.

An algebraic loop can often be circumvented by either rewriting the equations or using the procedure `IMPLICIT`.

Procedure `IMPLICIT` is used to solve equations of the form

$$x = f(x)$$

The procedure heading is:

```
procedure IMPLICIT(X,FX,TOL);
name X,FX; real X,FX,TOL;
```

`IMPLICIT` finds an `X` satisfying the equation within the tolerance `TOL`, i.e.

```
ABS(X-FX) <= ABS(TOL)
```

`FX` is an arithmetic expression involving `X` at least once.

Wegstein's accelerated convergence algorithm is used to compute X. The algorithm is iterative and uses the value of X at procedure entry as an initial guess for X. If the tolerance criterion is not satisfied after 100 iterations, the algorithm stops with the error message:

```
IMPLICIT:  CONVERGENCE IS NOT ACHIEVED WITHIN 100 ITERATIONS
```

(5) COLLECTION OF STATISTICS

The class `STATVAR` may be used for collecting statistics about state variables that vary continuously with time.

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

```
VARIABLE class STATVAR;
begin
  real procedure MIN; MIN:= ... ;
  real procedure MAX; MAX:= ... ;
  real procedure MEAN; MEAN:= ... ;
  real procedure SDV; SDV:= ... ;
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.

For obtaining statistics about discrete variables the following four data collection devices are provided:

```
class TALLY
class HISTOGRAM
class ACCUMULATE
class COUNT
```

The user has the following view of these classes:

```
class TALLY(TITLE); value TITLE; text TITLE;
begin
  procedure UPDATE(V); real V; ... ;
  procedure REPORT; ... ;
  procedure RESET; ... ;
end;

class HISTOGRAM(TITLE,LOWER,UPPER,NCELLS);
value TITLE;
text TITLE; real LOWER,UPPER; integer NCELLS;
begin
  procedure UPDATE(V); real V; ... ;
  procedure REPORT; ... ;
  procedure RESET; ... ;
end;

class ACCUMULATE(TITLE); value TITLE; text TITLE;
begin
  procedure UPDATE(V); real V; ... ;
  procedure INTEGRATE(V); real V; ... ;
  procedure REPORT; ... ;
  procedure RESET; ... ;
end;

class COUNT(TITLE); value TITLE; text TITLE;
begin
  procedure UPDATE(V); real V; ... ;
  procedure REPORT; ... ;
  procedure RESET; ... ;
end;
```

The four classes share the following attributes:

`TITLE`: is a user-supplied descriptive text parameter, cut off at 12 characters if initially longer.

`UPDATE`: is used to record the observations. The call `UPDATE(V)` records the value `V`.

`REPORT`: may be used to print the current status of the object.

`RESET`: may be used to re-initialize the object so that data collection can start afresh with a new time period.

Class TALLY

A `TALLY`-object is used to record a profile of a time-independent variable, for example the number of items bought by customers in a supermarket. The statistics collected (by calling `UPDATE`) are (1) the number of observations, (2) the average, (3) the estimated standard deviation, (4) the minimum, and (5) the maximum.

Class HISTOGRAM

A `HISTOGRAM`-object maintains the same statistics as a `TALLY`-object, but in addition it gives a graphic representation of the frequency distribution. Each HISTOGRAM-object requires at its generation: a text `TITLE`, a lower bound, `LOWER`, an upper bound, `UPPER`, and also the number of recording cells, `NCELLS`. The range from `LOWER` to `UPPER` will be divided into `NCELLS` number of cells, each of the same width, `(UPPER-LOWER)/NCELLS`. When a value V is recorded the appropriate cell incidence count is incremented by one. Underflow values (`V<LOWER`), and overflow values (`V>UPPER`) are recorded separately.

Class ACCUMULATE

An `ACCUMULATE`-object is used to record a profile of a time-dependent variable, for example the length of a queue. The variable in question is assumed to have maintained a constant value between the recording times (i.e. the `UPDATE`-points). For keeping statistics on variables that vary continuously between events, the procedure-attribute `INTEGRATE` is provided. `INTEGRATE` performs the same function as procedure UPDATE except that the last value and the current value are used to integrate the time dependent variable. Combined with the use of class `REPORTER`, the trapezoidal integration used by `INTEGRATE` yields better estimates for continuous variables than does `UPDATE`. Statistics collected are (1) the number of observations, (2) the average, (3) the estimated standard deviation, (4) the minimum, and (5) the maximum.

Class COUNT

A `COUNT`-object is used to record incidences only, for example the number of ingots produced by a furnace. A `COUNT`-object records the sum of its input sequence.

When a simulation is ended the statistics collected by all user-created `TALLY`-, `HISTOGRAM`-, `ACCUMULATE` and `COUNT`-objects are automatically reported.

At any time during the simulation the user may obtain a report on the current status of all such data collecting objects. This is accomplished by calling the global procedure `REPORT`.

The automatic reporting at the end of the simulation may be switched off by calling the global procedure `NOREPORT`.

A HISTOGRAM example

The program below

```
DISCO
begin
  ref(HISTOGRAM) H; integer I,U;
  U:=7913;
  H:-new HISTOGRAM("Normal distribution",0.0,2.0,20);
  for I:=1 step 1 until 1000 do H.UPDATE(NORMAL(1.0,0.5,U));
end;
```

will produce the following output:

```
**********************************************************************
*                                                                    *
*                            R E P O R T                             *
*                                                                    *
**********************************************************************



                         H I S T O G R A M S
                         *******************

                            S U M M A R Y

TITLE       /   (RE)SET/   OBS/  AVERAGE/EST.ST.DV/  MINIMUM/  MAXIMUM
NORMAL DISTR      0.000   1000    1.0180    0.5174    -0.740    2.8330

CELL/LOWER LIM/    N/   FREQ/  CUM %
                                       I------------------------------
   0 -INFINITY    25    0.02    2.50   I*********
   1     0.000    11    0.01    3.60   I****
   2    0.1000    29    0.03    6.50   I**********
   3    0.2000    30    0.03    9.50   I***********
   4    0.3000    30    0.03   12.50   I***********
   5    0.4000    26    0.03   15.10   I*********
   6    0.5000    51    0.05   20.20   I******************
   7    0.6000    69    0.07   27.10   I************************
   8    0.7000    51    0.05   32.20   I******************
   9    0.8000    85    0.08   40.70   I******************************
  10    0.9000    70    0.07   47.70   I*************************
  11    1.0000    80    0.08   55.70   I****************************
  12    1.1000    81    0.08   63.80   I*****************************
  13    1.2000    69    0.07   70.70   I************************
  14    1.3000    73    0.07   78.00   I**************************
  15    1.4000    49    0.05   82.90   I*****************
  16    1.5000    52    0.05   88.10   I******************
  17    1.6000    35    0.03   91.60   I************
  18    1.7000    17    0.02   93.30   I******
  19    1.8000    23    0.02   95.60   I********
  20    1.9000    12    0.01   96.80   I****
  21    2.0000    32    0.03  100.00   I***********
                                       I------------------------------
```

(6) PROGRAM TESTING

The procedure `DUMP` may be used for program testing.

`DUMP` may be called at any time during a simulation. Calling `DUMP` causes information about the current state of the model to be printed. Below is given a example of output produced by `DUMP`.

```
**********************************************************************
*                                                                    *
*                        D I S C O - D U M P                         *
*                                                                    *
**********************************************************************


                                 TIME
                             150.0000
```

```
         DTMIN          DTMAX    MAXRELERROR    MAXABSERROR
     1.000&-03         1.0000      1.000&-05      1.000&-05

            DT       LASTTIME  NEXTEVENTTIME NEXTREPORTTIME
     7.420&-02       149.9258       150.0000       150.0000

                 WAITPRIORITY      WAITPRIOR
                        0.000          FALSE

                     INTEGRATION METHOD: RKE


**********************************************************************
*                                                                    *
*            A C T I V E   V A R I A B L E - O B J E C T S           *
*                                                                    *
**********************************************************************

          STATE       RATE  LASTSTATE   RELERROR   ABSERROR
   1:    0.2205     -0.170     0.2334  1.000&-05  1.000&-05
   2:    0.1210     -0.105     0.1290  1.000&-05  1.000&-05
   3:  782.7003   -156.575   795.7096  1.000&-05  1.000&-05

**********************************************************************
*                                                                    *
*          A C T I V E   C O N T I N U O U S - O B J E C T S         *
*                                                                    *
**********************************************************************

           PRIORITY
   1:         0.000

**********************************************************************
*                                                                    *
*            A C T I V E   R E P O R T E R - O B J E C T S           *
*                                                                    *
**********************************************************************

                             - NONE -

**********************************************************************
*                                                                    *
*                       T I M E - E V E N T S                        *
*                                                                    *
**********************************************************************

             EVTIME
   1:      150.0000 - CURRENT - MAIN
   2:      150.3517
   3:      150.5794

**********************************************************************
*                                                                    *
*                      S T A T E - E V E N T S                       *
*                                                                    *
**********************************************************************

           PRIORITY
   1:        0.1240
   2:         0.000
   3:         0.000
```

ERROR MESSAGES

In case of illegal use of the facilities described, an error message is output, and the program is terminated.

The possible error messages are listed below:

```
ADD:  TWO TABLE ENTRIES WITH EQUAL X-VALUES

VAL:  TABLE IS EMPTY

NEW EXPDELAY( , ,TARGET, ):  TARGET == NONE

NEW EXPDELAY( , , ,LAG):  LAG <= 0

NEW IDELAY(V, ):  V == NONE

NEW IDELAY( ,LAG):  LAG < 0

IMPLICIT:  CONVERGENCE IS NOT ACHIEVED WITHIN 100 ITERATIONS

NEW HISTOGRAM( ,LOWER,UPPER, ):  LOWER >= UPPER

NEW HISTOGRAM( , , ,NCELLS):  NCELLS < 1
```

ACKNOWLEDGEMENT

The classes `TALLY`, `HISTOGRAM`, `ACCUMULATE` and `COUNT` are taken from the SIMULA-class `DEMOS`, written by Graham Birtwistle, University of Bradford, England (ref. 1).

REFERENCE

1. Birtwistle, G:
   "A System for Discrete Event Modelling on SIMULA",
   Macmillan Computer Science Series,
   London 1979.