SIMNON - An Interactive Simulation Program for Non-Linear Systems

Elmqvist, Hilding

Published in:
Simulation '77

1977

Document Version:
Peer reviewed version (aka post-print)

SIMNON - AN INTERACTIVE SIMULATION PROGRAM FOR NONLINEAR SYSTEMS

H. Elmqvist
Department of Automatic Control
Lund Institute of Technology
Lund, Sweden

ABSTRACT

Simnon is a command driven interactive program written in Fortran for simulation of systems governed by ordinary differential equations and difference equations.

The description of the system is done in a special model language or in Fortran. The user can separately describe subsystems with inputs and outputs and connect them. Each subsystem can be either a continuous time system or a discrete time system.

The program is controlled by commands. There are e.g. commands to change parameters of the model, perform simulation, plot the time response of selected variables on a display and to modify the model. A macro facility enables the user to store sequences of commands on a file for later and repeated use.

INTRODUCTION

Good man-machine communication is important for simulation. The user must be able to change parameters and modify the model easily. It should also be possible to select variables to be plotted. The availability of mini computers and advanced time sharing operating systems have made it possible to implement interactive simulation programs.

The program Simnon is an interactive simulation program. The user controls the program with commands. There are e.g. commands to change parameters of the model, perform simulation, plot results from simulation and to modify the model. The commands contain arguments which can be e.g. file names, variables, numbers and options. They have a flexible format. Default values are used for omitted arguments. The user can construct own high level commands by means of a Macro facility.

The model can be described either in a special model language or in Fortran. The model language is simple and easy to learn and permits extensive error checking. The compiler is included in the program and is working in parallel with an editor. This enables the user to correct erroneous lines immediately. It is thus easy to enter and change the model.

The notation of a subsystem is very important when modelling systems. When inputs and outputs for a subsystem are defined, it is sufficient to concider the internal behavior only. Simnon has a subsystem concept which in some sense correspond to the macro facility in CSSL-type of programs.

In traditional dynamic simulations it was frequency sufficient to describe models by ordinary differential equations only. When simulating computer controlled processes, it is naturally to describe the physical process by ordinary differential equations and the computer with its control algorithms by difference equations. For this reason Simnon allows description of both continuous time subsystems and discrete time subsystems.

A special discrete subsystem is available for optimization [4]. It has a loss function as input and parameters as outputs. Using this subsystem it is possible to solve optimal control problems. There is also a continuous subsystem for time delays.

The complete description of Simnon can be found in [1].

MODEL STRUCTURE

There are three types of systems in Simnon: continuous, discrete and connecting.

A continuous subsystem is mathematically defined as

$$\dot{x}(t) = f(x(t),t,u(t),p)$$
$$y(t) = g(x(t),t,u(t),p)$$
$$x(t_0) = x_0$$

The following notations are used

- t - time (independent variable)
- x - state variables (dependent variables)
- y - output variables
- p - parameters
- t_0 - start time for simulation
- x_0 - initial values for states
- u - input variables

A discrete subsystem is mathematically defined as

$$x(t_{i+1}) = f(x(t_i),t_i,u(t_i),p)$$
$$y(t_i) = g(x(t_i),t_i,u(t_i),p)$$
$$x(t_1) = x_0$$

The i:th sampling instance is denoted t_i. The sampling instances need not be equidistant and need not be equal in all subsystems.

The outputs of the discrete subsystems are not defined between the sampling instances and before the first sampling instance. In order to allow for connection of continuous systems and discrete systems with different sampling instances, the definition of the outputs must be extended. This is done by introducing zero order hold circuits at the outputs. The outputs are then defined by

$$y(t) = \begin{cases} y(t_i) & t_i \le t < t_{i+1} \\ y_0 & t < t_1 \end{cases}$$

where y_0 is the initial values for the outputs of the system.

It is sometimes practical to be able to treat the states of a subsystem as outputs when connecting the systems. In order to allow this, the states must also be defined for all t. This is done as

$$x(t) = \begin{cases} x(t_i) & t_i \le t < t_{i+1} \\ x_0 & t < t_1 \end{cases}$$

The connections of the subsystems are done in a so called connecting system. Introduce the notation X, U and Y for the concatenation of the state-, input- and output-vectors in all subsystems. The connections can then be written as

$$U(t) = h(X(t),t,Y(t),p)$$

Together with the output equations of the subsystems, this equation represents a set of nonlinear equations. If the system does not contain any algebraic loop then the equations can be solved sequentially by sorting them appropriately.

The description of a subsystem contains two parts, one for computing outputs and one for computing derivatives resp. update of states. The equations of each part are executed sequentially. The equations defining the connections between the subsystems must be written in a special order. If an output variable appears in the right hand side of such an equation then the output-section of the corresponding subsystem will be executed before evaluating the equation. The user then has to ensure that the inputs which are used in that output section have been assigned previosly. Warnings are given to aid sorting the equations. There are plans to make the program sort all the equations automatically.

The sampling instances for each discrete subsystem are specified by a special variable in the system description. This variable is updated at each sampling instance to contain the time for the next sampling. After the activation of some discrete subsystems it is thus known when the next sampling should be performed. The differential equations, which are parameterized by the discrete states and outputs, can thus be solved over the sampling interval by an ordinary integration routine. At present there are a Hammings predictor-corrector algorithm and a Runge-Kutta algorithm. Both are able to handle discontinuities in the equations. There are plans to include an algorithm for stiff equations and algebraic loops.

There are cases when one wants certain events to occur when some condition on the variables are fulfilled, i.e. the next sampling of a system should be performed when this condition is fulfilled. This feature requires modification of the integration routines and are planned to be included later.

Another facility which is sometimes required, but not implemented yet, is to reach the value of an input variable to a discrete subsystem at the time just before the sampling instance. This corresponds to placing synchronously triggered sample and hold circuits at the inputs.

MODEL LANGUAGE

Simnon includes a special language for describing subsystems and connections. The equations are entered using the assignment statement of Algol-60. The if-then-else construction has shown to be very useful and also eliminates the need for special "nonmemory operators". The model language is very simple which has made it possible to do extensive error checking. If complicated models are used there is a possibility to use Fortran subroutines for model description.

There are three types of systems, continuous, discrete and connecting. Continuous and discrete systems have the following structure.

```
system heading
declarations
INITIAL-section
OUTPUT-section
DYNAMICS-section
END
```

The system heading gives the type of the system and its name. There are declarations to specify variable types. The types are INPUT, OUTPUT, TIME, STATE, DER, NEW and TSAMP. The type DER is used in continuous systems to associate a variable as derivative for a state-variable. The NEW-declaration is used in the same way in discrete systems for the update of a state-variable. One variable of type TSAMP is used in each discrete system to specify the next sampling instance. The system description can also contain parameters and auxiliary variables, which are not declared.

Parameters are assigned by a statement of the following form

```
<parameter>: <number>
```

Initial values of state-variables can be assigned in the same way.

The INITIAL-section contains statements which are executed only before the simulation is started, e.g. parametric expressions. The OUTPUT-section is used to assign output variables and the DYNAMICS section is used to assign DER- or NEW- variables. Auxiliary variables can be used in all sections and parameter assignments can be done in all sections.

The connecting system has the following structure.

```
system heading
declaration
INITIAL-section
CONNECT-section
END
```

The CONNECT-section contains assignment statements for the INPUT-variables of the subsystems. The same identifier may be used for variables in different subsystems. The following notation is therefore used in connecting system to reference variables in the subsystems.

```
<variable>[<system identifier>]
```

The right hand part of the assignments may contain STATE- and OUTPUT-variables which are referenced in the same way.

MODEL MANIPULATION

In Simnon the model manipulation is clearly distinguished from the model itself. The model description is stored on files on mass storage. The manipulation of the models are done with commands with arguments. The commands are normally entered from the terminal but can also be read from files. The results of the simulations are plotted on a graphical display.

A brief description of the most important commands are given below. Note that all facilities are not described. For a complete description see [1].

```
SYST <file name>...
```

This command compiles the system descriptions on the named files. The last file should contain the connecting system. The entire system description can, however, be done as a single continuous or discrete system with no inputs.

The compiler is working in parallel with an editor. When the compiler discovers an error an error-message is given and the user directly gets the possibility to correct the erroneous line by certain editing commands.

Some of the available editor commands are described below.

```
R <line>
```
Replaces the current line

```
C /<character string 1>/<character string 2>/
```
Changes some characers in the current line.

```
T
```
Goes to the beginning of the file.

```
N <number of lines>
```
Goes a number of lines down the file.

```
L <character string>
```
Locates a specified string.

```
I <line>
```
Inserts a line.

```
D
```
Deletes a line

```
E
```
Exits from the currently edited file.

```
STORE <variable>...
```

The simulation results can either be stored on a file for later plotting or plotted during the simulation. The command STORE is used to specify which variables should be stored. If plotting is wanted during the simulation the variables are specified in the command PLOT. In that case the scaling must be entered with the AXES-command since automatic

scaling can not be used.

SIMU <start time> <stop time>

This command demands simulation of the system defined by the previous SYST-command.

ASHOW <variable>... [(< variable>)]
SHOW <variable>... [(< variable>)]

If simulation results have been stored on a file they can be plotted by these commands. Both commands have a list of variables which will be plotted versus time or the variable within parenthesises if any. The command ASHOW finds appropriate scalings and draws axes before plotting. The display area can be divided into several independent plotting areas with the command SPLIT.

PAR <parameter> : <number>

Parameters are assigned in a special way in the system descriptions. To alter parameter values between the simulations one need not edit and recompile the system description. This can be done with the PAR-command.

INIT <state variable>:<number>

This command changes the initial value of a state variable.

SAVE <file name>

After having altered parameters and initial values the current values can be stored on a file with this command.

GET <file name>

Restores a parameter set and a set of initial values from a file.

DISP <variable>...

Displays the current value of variables.

LIST <file name>

Lists the contents of files.

INTERACTIVE FACILITIES

The program Simnon is controlled by commands. The structure of each command is flexible. Arguments can in some cases be omitted. Default values are then used. One common situation when running the program is that the same sequence of commands is given several times. The user can then difine a Macro containing the commands. This Macro is then used as a new command several times, possibly with different values of the arguments.

Repetitive loops and jumps can be introduced in a Macro. It is even possible to make a Macro which makes the program look like a question and answer program when executed. This possibility can e.g. be used to introduce Simnon to new users. The description of these facilities can be found in [3].

EXAMPLE

Some of the features of Simnon will be illustrated by an example. The system considered is the tank system shown in "Fig. 1"

The valve at the inlet is controlled by a regulator to keep the level constant. The valve at the outlet is externally manipulated. When studying this system the valve area is considered as a disturbance.

The simulation study is used to determine a suitable regulator and to find the regulator settings.

The model descriptions are shown in "Fig. 2". The tank and the valves are described in system TANK. The system has two inputs, the signal V to the inlet valve and the area of the outlet valve AOUT. The level H is the state and is also considered as output. The tank is controlled by a process computer which contains a PID-regulator. The model is called DPID.

The connections of TANK and DPID is made in REGTANK. The disturbance at the outlet is also given in REGTANK. When time=100 the outlet area is increased from 0.01 to 0.05. The tank is initially empty.

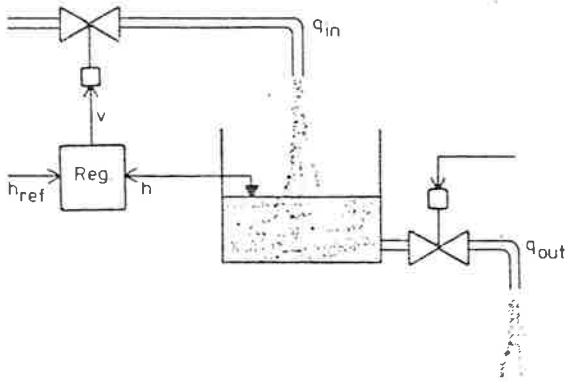

Fig 1

The interaction with Simnon is shown below. Comments are written after ".

```
>SYST TANK DPID REGTANK
>STORE H HREF QIN QOUT
>" TRY A PROPORTIONAL REGULATOR
>" WITH GAIN=1.
>PAR DT:5 " SET SAMPLING INTERVAL
>SIMU 0 200
>ASHOW H HREF QIN
>" SEE FIG. 3
>
>" ELIMINATE THE STATIC ERROR
>" WITH A PI-REGULATOR.
>PAR TI:50
>SIMU
>ASHOW H HREF QIN
>" SEE FIG. 4
>
>" TOO LARGE OVERSHOT, TRY CONDITIONAL INTEGRATION.
>SAVE PIPAR " SAVE PARAMETERS
>SYST TANK DPID REGTANK - EDIT
   EDIT
  >E    " NO CHANGES IN TANK
   EDIT
  >L DT/TI
NINTE = INTE + DT/TI*E
  >R NINTE=INTE+(IF ABS(E)<ELIM THEN DT/TI*E ELSE 0)
  >I ELIM:1
  >E
   EDIT
  >E    " NO CHANGES IN REGTANK
>
>GET PIPAR "RESTORE PARAMETERS
>STORE H HREF QIN QOUT
>SIMU
>ASHOW H HREF
>" SEE FIG. 5
>
>" STUDY THE EFFECTS OF
>" DIFFERENT SAMPLING INTERVALS.
>MACRO SAMP
  >FOR P=5 TO 25 STEP 10
  >PAR DT:P
  >SIMU
  >SHOW H
  >NEXT P
  >END
>
>AXES H 0 200 V 0 3
>SAMP
>"SEE FIG. 6
```

```
CONTINUOUS SYSTEM TANK

INPUT V AOUT
STATE H
DER DH

DYNAMICS
VALVE = IF V<0 THEN 0 ELSE IF V>1 THEN 1 ELSE V
QIN = QMAX*VALVE
QOUT = AOUT*SQRT( 2*G*MAX(H,0))
DH = (QIN - QOUT)/AREA

QMAX:1
G:9.81
AERA:10

END


DISCRETE SYSTEM DPID

INPUT Y YREF
OUTPUT U
STATE INTE YOLD
NEW NINTE NYOLD
TIME T
TSAMP TS

OUTPUT
E = YREF - Y
U = GAIN*(E + INTE + TD/DT*(YOLD-Y))

DYNAMICS
NINTE = INTE + DT/TI*E
NYOLD = Y
TS = T + DT

GAIN:1
TI:1E10
TD:0
DT:1

END


CONNECTING SYSTEM REGTANK

TIME T

AOUT[TANK] = IF T<100 THEN A1 ELSE A2
A1:0.01
A2:0.05
YREF[DPID] = HREF
HREF:2
Y[DPID] = H[TANK]
V[TANK] = U[DPID]

END
```

Fig 2

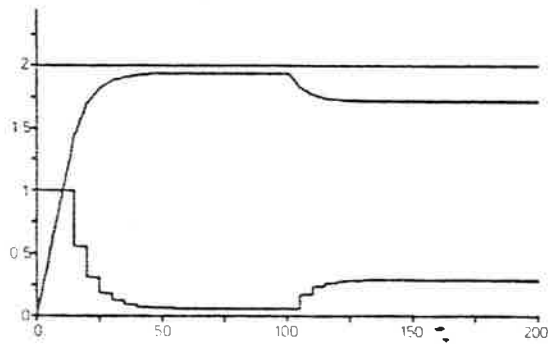

Fig 3

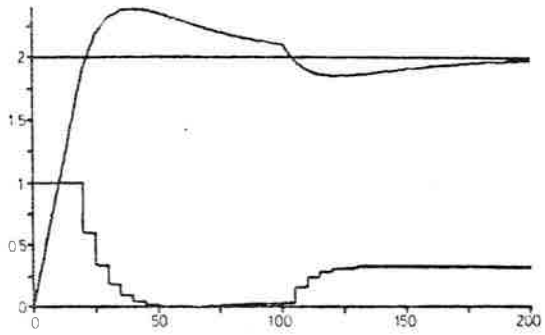

Fig 4

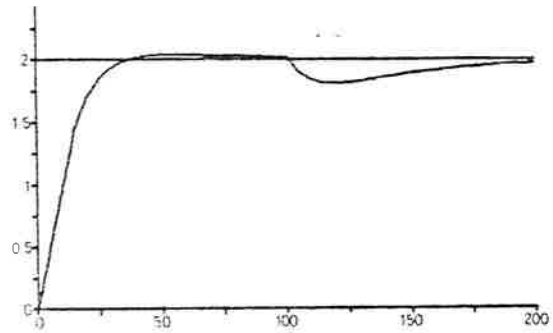

Fig 5

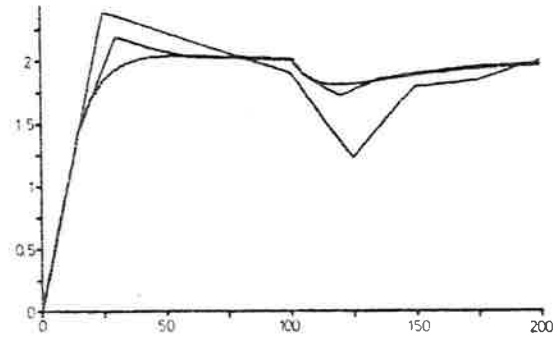

Fig 6

IMPLEMENTATION

Simnon is implemented in Fortran. The installation dependent parts for character handling, file handling and plotting are concentrated to small, well defined subroutines.

The compiler consists of two parts. The first is installation independent and translates the model to a pseudo code which is stored in an integer array. The second part is installation dependent and translates the pseudo code into machine instructions. It is not necessary to implement the second part because Simnon contains an interpretation routine for the pseudo code, which can be used during the simulations.

The interactive facilities are handled by a general interactive module [3] which can be used in other programs. The editor is also a separate module.

The program was originally implemented on a PDP-15 computer with 32 k 18 bits words. It has then also been installed on Univac-1108, Dec-10, and Honeywell H6000.

CONCLUSIONS

Simnon has been used extensively since 1974 both for research, education and to solve industrial control and simulation problems. It has been found that it is easy to learn how to run the program and the interactivity makes it possible to do fast simulation studies. The simple model language and the extensive error checking makes Simnon well suited for use in education. The possibility to include subsystems in Fortran makes it possible to use complicated models.

The model class in Simnon is sampled data systems i.e. systems governed by both ordinary differential equations and difference equations. This is important as the use of digital computers for process control increases.

The transfer of the program to different installations requires a moderate effort because of the previous mentioned structuring of the program.

ACKNOWLEDGEMENTS

The author wants to thank Prof Karl Johan Åström and Dr Johan Wieslander for many valuable ideas and stimulating discussions.

The work has been supported by the Swedish Institute of Applied Mathematics (ITM).

REFERENCES

1. H. Elmqvist: "SIMNON - An Interactive Simulation Program for Nonlinear Systems - USER's MANUAL", Report TFRT-3091, Lund Institute of Technology, Department of Automatic Control, April 1975.

2. H. Elmqvist: "Implementation of Simnon" Report, Lund Institute of Technology, Department of Automatic Control, (to appear)

3. J. Wieslander, H. Elmqvist: "INTRAC - An Interactive Monitor - Reference Manual", Report, Lund Institute of Technology, Department of Automatic Control, (to appear)

4. T. Glad: "Constrained Optimization using Multiplier Methods with Applications to Control Problems", Ph.D. Thesis, Report TFRT-1011, Lund Institute of Technology, Department of Automatic Control, April 1976.