ESM 98

SIMULATION-PAST, PRESENT AND FUTURE

EDITED BY

Richard Zobel

Dietmar Moeller

12TH EUROPEAN SIMULATION MULTICONFERENCE
as part of the 50th anniversary celebrations of the
University of Manchester, The Home of Computing

JUNE 16-19, 1998
Manchester,
United Kingdom

Sponsored by:

Systems Modeling Corporation

co-organised by:

co-sponsored by:

hosted by:

the University of Manchester, Computer Science Department

A Publication of the Society for Computer Simulation International

SIMULATION OF CONTINUOUS DYNAMIC SYSTEMS USING THE GODYS–PC LANGUAGE *

Jacek Kuraś Jacek Lembas Marek Skomorowski
Department of Computer Science, Jagiellonian University
Nawojki 11, 30–072 Kraków, Poland
E-mail: kuras@ii.uj.edu.pl

KEYWORDS
Continuous simulation, Dynamic modelling.

ABSTRACT

This paper presents the use of the GODYS–PC continuous system simulation language for teaching modelling and simulation of continuous dynamic systems. GODYS–PC runs on PCs, and provides comprehensive, interactive simulation environment.

INTRODUCTION

The graduate program in computer science at the Department of Computer Science, at the Jagiellonian University, Kraków, Poland, offers a course on modelling and computer simulation of continuous dynamic systems. The simulation language used is GODYS–PC (abbreviated from Graph Oriented Dynamic System Simulator for PCs). There are a number of continuous system simulation languages currently in use, for example: ACSL, SIMULINK, TUTSIM, **20-sim** and others. A description and comparison of some of the more popular continuous system simulation languages has been presented by Rimvall and Cellier 1985. A excellent discussion on continuous system simulation languages has been presented by Divakaruni 1985.

The GODYS–PC is a new version of the GODYS language. The GODYS continuous system simulation language was originally developed at the Institute of Computer Science, Jagiellonian University, Cracow, Poland in the mid-1970s and has been installed on Honeywell, ICL 1900 and IBM 360 computers (Jakubowski and Król 1972, Król et al. 1980). It has been developed in recent years and has been installed on PCs. GODYS–PC is the result of this development.

* This work was partially supported from ESPRIT Project 20288-13.

Using the same basic approach as GODYS, GODYS–PC provides an interactive simulation facility and a number of advanced features. GODYS–PC is consists of two basic modules. One of them is the syntax-directed translator, which generates the object program in the language of some abstract machine. This machine is implemented by the effective interpreter, which is the main part of the second module.

SIMULATION USING GODYS–PC

A GODYS-PC program consists of two parts: the model description and the runtime commands. The model description defines the model of the system being modelled. The model description must be written in a file whose name ends in *.mod*. The runtime commands exercise this model (for example, they change parameters, execute runs, specify plots, and so on). Runtime commands can be entered interactively or in a batch process. In the case of a batch process the runtime commands must be written in a file whose name ends in *.sim*. Runtime commands are read, decoded, and executed in sequence.

The model description consists of two parts: initial and dynamic sections. The initial section is the section for calculations performed once before each dynamic run. Usually some initial calculations are calculated at this section. The initial section is optional. The dynamic section comprises a set of algebraic and differential equations defining the model. The equations defining the model can be written in any order. The translators sorts the equations in the dynamic section of the program to ensure a correct sequence of calculations variables. The sort algorithm is based on the theory of functional graphs (Jakubowski and Król 1975; Jakubowski and Król 1979).

The language consists of a set of arithmetic, relational and logical operators, and standard

functions. The operators work as they do in FORTRAN. The functions consists of special GODYS–PC operators such as for example, `INTEG` (integration function), `DERIVT` (derivative function), `STEP` (step function) and so on. The language provides over fifty standard functions. The user can define his own functions in FORTRAN. GODYS–PC statements may be placed anywhere on the line. Any model source or runtime statement can be continued onto another line by ending the first line with `@`. All text after # to the end of the line is considered a comment.

Example 1. Let us consider a simple model of the following inventory control system (Coyle R.G. and Sharp J.A. 1976). Production order backlog (*pob*) depends on production order rate (*por*) and production rate (*pr*). Production rate depends on production order backlog and a time constant t_1. Inventory (*inv*) depends on consumption (*cons*), which is exogenous, and production rate. Average consumption (*avcon*) depends on consumption and averaging period t_2. Desired inventory (*dinv*) depends on average consumption and weeks cover desired – a constant r. Production order rate depends on inventory, desired inventory, average consumption and an inventory correction – a constant t_3. The influence diagram of a model of the system is shown in Fig. 1.

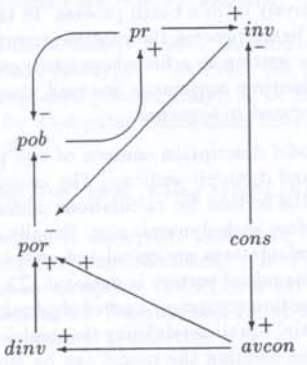

Fig. 1. The influence diagram (Example 1).

The model of the system can be written mathematically as follows:

$$\frac{d(inv)}{dt} = pr - cons, \qquad \frac{d(pob)}{dt} = por - pr$$

$$\frac{d(avcon)}{dt} = \frac{cons - avcon}{t_2}, \qquad pr = \frac{pob}{t_1}$$

$$por = avcon + \frac{dinv - inv}{t_3}, \qquad cons = f(t)$$

$$dinv = r * avcon$$

Let us consider that consumption (*cons*) is a step function. Real inventory (*realinv*) should be greater or equal to zero. Production rate (*pr*) should be bounded from above. The model description is the following:

```
MODEL invent
PREPARE cons, relinv, pr
PARAMT t1, t2, t3, inv0, pob0, @
   avcon0, a1, a2, t0, b, r
DYNAMIC
inv=INTEG(pr-cons; inv0)
relinv=MAX(inv, 0)
pob=INTEG(por-pr; pob0)
avcon=INTEG((cons-avcon)/t2; avcon0)
pr=BOUND(pob; 1, b)/t1
por=avcon+(dinv-inv)/t3
cons=a1*STEP(t)+a2*STEP(t-t0)
dinv=r*avcon
END
```

Runtime commands are the following:

```
LOAD invent
DATA t1=4, t2=4, t3=4, inv0=1000, @
   pob0=400,avcon0=100, a1=100, @
   a2=25, b=600, r=10, t0=5
EXECUTE(dt=0.1, tmax=70, comdel=2, @
   method=trapez)
PRPLOT(cons=(0,400), @
   relinv=(0,1500),  pr=(0,200))
FINISH
```

Now for some explanations about program itself. Elements shown in upper case can be written in lower case and vice versa. The statement `MODEL` identifies the model. The statement `PREPARE` specifies which variables are to be collected for latter printing or plotting. The statement `PARAMT` declares identifiers of parameters of the model. The statement `DYNAMIC` identifies the beginning of the dynamic section. The dynamic section comprises a set of equations defining the model. The statement `DYNAMIC` must be accompanied the matching statement `END`. In the example the dynamic section contains the following standard functions: `INTEG`, `MAX`, `BOUND` and `STEP`.

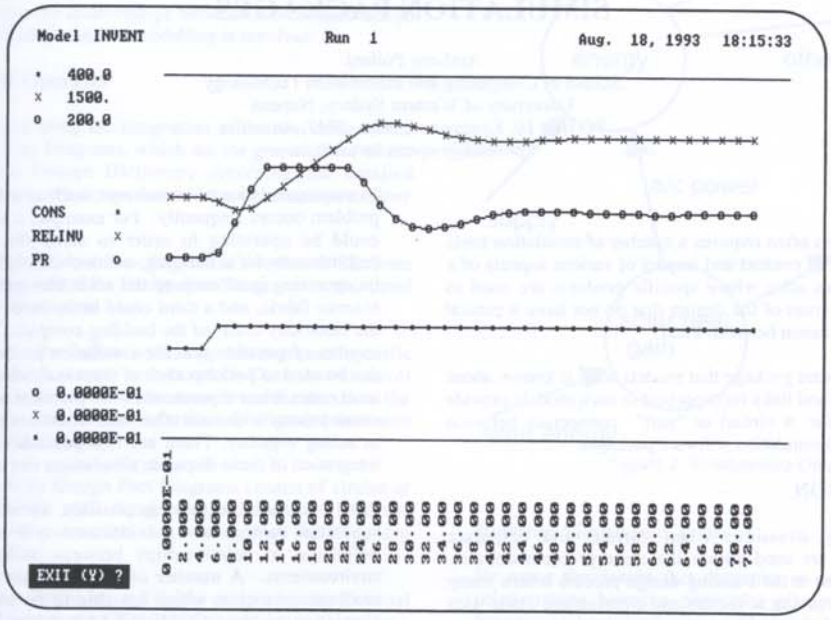

Fig. 2. Output of the PRPLOT statement (Example 1).

The statement **LOAD** identifies the beginning of runtime commands. The statement **LOAD** must be accompanied the matching statement **FINISH**. The statement **DATA** is used to assign values to parameters of the model. The statement **EXECUTE** initiates the run. The parameter **dt** is the integration step size. The parameter **tmax** specifies the end of the simulation time. The parameter **comdel** is the interval size for printing and plotting simulation results. The parameter **method** specifies the name for the integration algorithm. GODYS–PC provides five fixed step algorithms and a variable step algorithm. The statement **PRPLOT** generates graphs of variables against **t** (time). The output of the statement PRPLOT is shown in Fig. 2.

CONCLUSION

GODYS–PC is a CSSL based simulation system. It meets requirements needed for a good simulation language and is a powerful tool simulation of continuous systems from a wide range of engineering and scientific disciplines. GODYS–PC provides an integrated development environment and an interactive simulation facility running on MS–DOS with 640 K of RAM and 2 MB on the hard disk.

REFERENCES

Rimvall M. and Cellier F. 1985. "Evolution and perspectives of simulation languages following the CSSL standard", *Model. Ident. Contr.* 6, no. 4.

Divakaruni S.M. 1985. "Perspectives in software design for dynamic process simulation", *Model. Ident. Contr.* 6, no. 4.

Jakubowski R. and Król J. 1972. *Implementation of the simulation language based on functional graphs*, *Podstawy sterowania* 2, no. 2.

Król J., Kuraś J., Lembas J., Ślusarek M. 1980. "Implementation of the language for the simulation of continuous systems with discontinuities", *Podstawy sterowania* 10, no. 1.

Jakubowski R. and Król J. 1975. "General algorithm of complex dynamic systems simulation", *Systems Science* 1, no. 1.

Jakubowski R. and Król J. 1979. "Extended functional graphs in modelling and simulation of systems", *Podstawy sterowania* 9, no. 2.

Coyle R.G. and Sharp J.A. 1976. "System dynamics problems and cases", University of Bradford.