Simulation in Industry

ESS 96

EDITED BY

Agostino G. Bruzzone

Eugene J.H.Kerckhoffs

8th European Simulation Symposium

Volume I

OCTOBER 24-26, 1996

Genoa, Italy

SCS

A Publication of the Society for Computer Simulation International

SiE

GODYS–PC: AN INTERACTIVE CONTINUOUS SYSTEM SIMULATION 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 GODYS–PC interactive continuous system simulation language for PCs. The basic structure of the language follows the CSSL standard (Strauss et al. 1967). It is derived from the continuous system simulation language GODYS, which has been installed on Honeywell, ICL 1900 and IBM 360 computers. GODYS–PC provides comprehensive, interactive simulation environment which allows the user full control of the simulation of realistic situations. Examples are provided to illustrate the use of the language.

INTRODUCTION

GODYS (an acronym for graph oriented dynamic system simulator) continuous system simulation language was originally developed in the mid-1970s and has been installed on Honeywell, ICL 1900 and IBM 360 computers (Jakubowski and Król 1972). It has been developed in recent years and has been installed on PCs. GODYS–PC is the result of this development. Using the same basic approach as GODYS, GODYS–PC provides an interactive simulation facility and a number of advanced features. GODYS–PC has been developed for the purpose of modelling of dynamic systems described by a set of algebraic and differential equations. Output of the results may be either in tabulated or graphical form.

GODYS–PC is written in FORTRAN and consists of two 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 an effective interpreter, which is the main part of the second module (Król et al. 1980).

Typical areas in which GODYS–PC is currently applied come from a wide diversity of realistic situations in engineering, control system design, economics and biology.

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 1968; 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.

To illustrate the use of GODYS–PC let us consider a model of the spring damping example shown in Fig. 1.

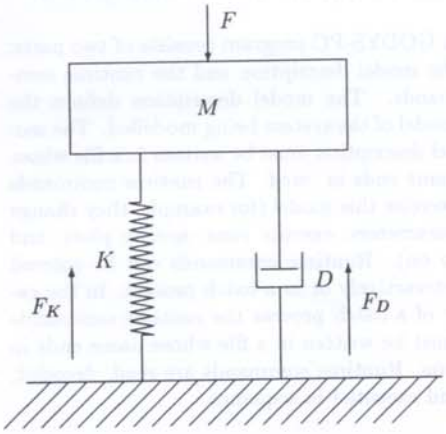

Fig. 1. The spring damping example

The model of the spring damping example can be written mathematically as follows:

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = -\frac{K}{M}x - \frac{D}{M}v + \frac{F}{M}$$

The program written in GODYS–PC for the model is the following:

```
MODEL spring
PREPARE x,v
PARAMT k,d,m,a,t0
DYNAMIC
x=INTEG(v;0)
v=INTEG((a/m)*STEP(t-t0)- @
   (d/m)*v-(k/m)*x;0)
END

LOAD spring
DATA k=1,d=0.3,m=1,a=-1,t0=10
EXECUTE(dt=0.1,tmax=50,comdel=0.1)
PLOTXY(t=(0,50),x)
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 used here the dynamic section contains standard functions: **INTEG** and **STEP**.

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 paremeter **method** specifies the name for the integration algorithm. GODYS–PC provides five fixed step algorithms and a variable step algorithm. The statement **PLOTXY(t=(0,50),x)** generates a graph of **x** against **t** (time) for values of **t** between **0** and **50**. The output of the statement **PLOTXY(t=(0,50),x)** is presented in Fig. 2.

PARAMETER OPTIMIZATION

In the case of parameter optimization a finite number of parameters has to be determined such that a cost function of these parameters is minimal (Amyot and Blokland 1987). At most 8 parameters can be optimized in a cost function in GODYS–PC. The language provides four function minimization algorithms.

To illustrate the use of parameter optimization in GODYS–PC let us consider the system block diagram shown in Fig. 3 (Amborski and Marusak 1978).

The parameter $k \in (1, 25)$ has to be determined such that the following cost function

$$f = \int_0^\infty e^2(t)dt$$

is minimal. The description of the model in GODYS–PC is the following:

```
MODEL contr
PREPARE x,y,e,f
PARAMT k,t0
DYNAMIC
x=STEP(t-t0)
e=x-y
u=k*REALPL(e;8,0)
y=REALPL(0.01*INTEG(u;0);2,0)
f=INTEG(e*e;0)
END
```

In the example used here the dynamic section contains the folowing standard function:

$$\mathtt{y} = \mathtt{REALPL}(\mathtt{x};\ \mathtt{a},\ y_0)$$

which produces first order lag where output y is related to input x through the following transfer function:

$$\frac{x}{y} = \frac{1}{as+1}$$

where $y_0 = y(0)$.

Runtime commands with parameter optimization of the model are the following:

```
LOAD contr
DATA k=5, t0=20
EXECUTE(tmax=175,dt=0.1,comdel=1,@
   opt=(f(k=(5,25)),alg=mgs,lim=10))
PLOTXY(t,e)
FINISH
```

The parameter `opt` initiates parameter optimization in the run. The parameter `alg` specifies the method for the function minimization algorithm. Each function evaluation involves a simulation run. The parameter `lim` specifies the number of runs during parameter optimization. Fig. 4 presents the results of parameter optimization. The minimal value of the cost function $F = 9.958$ has been found for $k = 17.36$. Optimization time of the example used here measured on a 50 MHz 486DX microprocessor is 13 seconds. Each function evaluation involves a simulation run: 10 runs during parameter optimization and 1 run with the optimal value of the parameter k.

The statement `PLOTXY(t,e)` generates a graph of `e` against `t`.

CONCLUSION

GODYS–PC continuous simulation language for PCs was described. It meets requirements needed for a good simulation language and is a powerfull tool simulation of continuous systems from a wide range of engineering and scientific disciplines (any type of systems that can be described by a set of differential equations or transfer functions). 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. GODYS–PC is easy to learn even for somebody who is not an experienced programmer.

REFERENCES

Amborski, K.; A. Marusak. 1978. *Teoria sterowania w ćwiczeniach.* PWN.

Amyot, J.R. and G. Blokland. 1987. "Parameter optimization with ACSL models." *Simulation* 9, November.

Jakubowski R. 1968. "An algorithm for the simulation of dynamical systems by means of digital computers, based on signal-flow graphs". *Journal of Mathematical Analysis and Applications* 22, no.1 (April).

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

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

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

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

Strauss, J.C., D.C. Augustine, B.B. Johnson, R.N. Linebarger, F.J. Sanson. 1967. "The SCi continuous system simulation language (CSSL)." *Simulation* 9, no.6: 281-303.

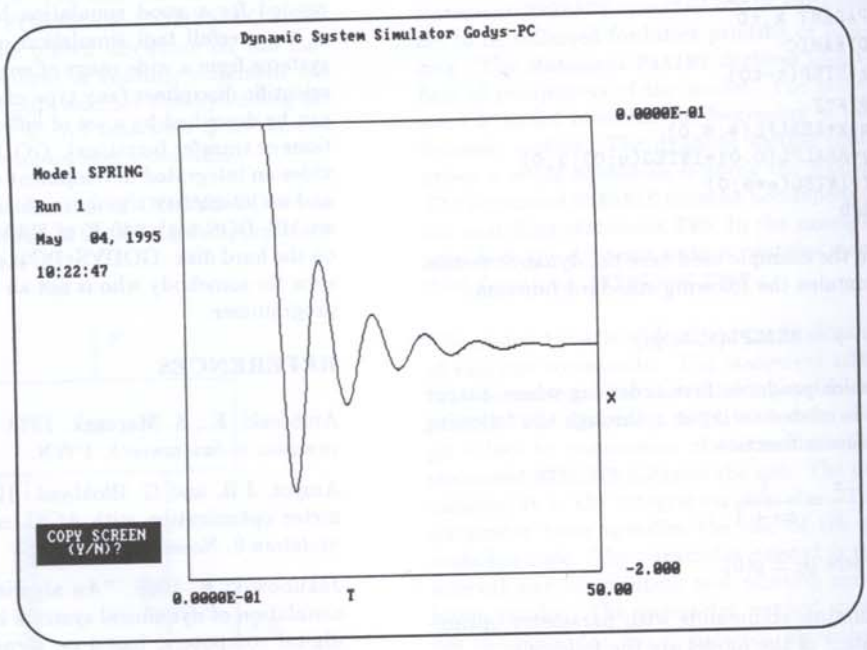

Fig. 2. Output of the `PLOTXY(t=(0,50),x)` statement.

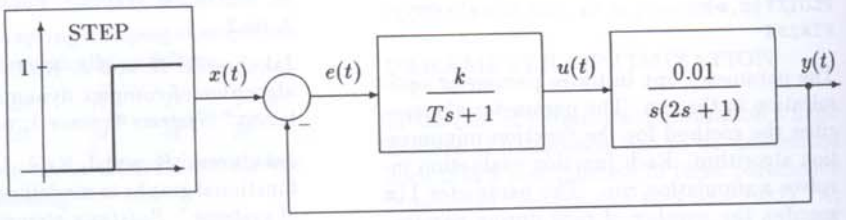

Fig. 3. Control loop block diagram ($T = 8$).

```
>>>>>>  Start of the Optimization Process  <<<<<<

          Algorithm  =  MGS
          Limit      =    10
          RelTol     =  1.0000E-03
          Trace Level=     0

     The quality factor to be minimized: F

     The parameters to be adjusted:

     Ident        Lower Bound     Upper Bound     Initial Value

     K              5.000           25.00           5.000

     The initial value of  F        =   15.48

 *** End of Optimization, Cause: LIMIT

     The minimal value of F        =   9.958     has been found
     for the following values of the parameters:

               K       =   17.36

     The Statistics of the Optimization:

          #Experiments =    11
          #Bases       =     2
          Delta        =   1.000
          Gamma        =  0.0000E-01
          Debasement   =   5.523

          Optimization time:    13 Secs

>>>>>>  End of the Optimization Process  <<<<<<
```

Fig. 4. The results of parameter optimization.