International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:8, 2007

GODYS-PC: a Software Package for Modeling, Simulating and Analyzing Dynamic Systems

Jacek Kuraś, Jacek Lembas, and Marek Skomorowski

***Abstract*—**In this paper, we introduce GODYS-PC software package for modeling, simulating and analyzing dynamic systems. To illustrate the use of GODYS-PC we present a few examples which concern modeling and simulating of engineering systems. In order to compare GODYS-PC with widely used in academia and industry Simulink®, the same examples are provided both in GODYS-PC and Simulink®.

***Keywords*—**Modeling, simulating and analyzing dynamic systems.

I. Introduction

THE purpose of this paper is to present GODYS-PC software package for modeling, simulating and analyzing dynamic systems. Typical areas in which GODYS-PC may be successful applied come from a wide diversity of realistic situations in engineering, control system design, economics, biology and so on.

There are a number of continuous simulation languages currently in use in applied science and engineering, for example Simulink®, which is widely used in academia and industry. Simulink® provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations ([1]). This is far from describing models by mathematical expressions like in GODYS-PC. A description and comparison of some of the most popular continuous simulation languages has been presented, for example, in [2]. A discussion on evolution of continuous modeling and simulation has been presented, for example, in [3].

II. Simulation of Dynamic Systems using GODYS-PC

The basic structure of GODYS-PC follows the CSSL standard presented in [4]. GODYS-PC (abbreviated from Graph Oriented Dynamic System Simulation for Personal Computers) is a new version of GODYS continuous simulation language. GODYS was originally developed at the Institute of Computer Science, Jagiellonian University, Kraków in the mid-1970s and has been implemented on Honeywell, ICL 1900 and IBM 360 computers respectively ([5], [6], [7], [8], [9]). It has been developed in recent years and has been implemented on Personal Computers.

Authors are with the Institute of Computer Science, Jagiellonian University, Kraków, Poland.

Using the same basic approach as GODYS, GODYS-PC provides an interactive simulation facility and a number of advanced features. One of these features is 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 ([10]). GODYS-PC provides four algorithms for parameter optimization in a cost functions. At most eight parameters may be optimized in a cost function in GODYS-PC.

GODYS-PC provides a set of standard functions like, for example, INTEG (integration function), DERIVT (derivation function), STEP (step input function) and so on. GODYS-PC 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 source model or runtime commands may be continued onto another line by ending the first line with the symbol @. All text after the symbol # to the end of the line is considered as a comment. Equations describing a model may be written in any order. This is because GODYS-PC provides an algorithm for automatic sorting of equations of a model. The sorting algorithm is based on the theory of functional graphs ([7], [8]).

After describing a model one can simulate it, using a choice of integration method. GODYS-PC provides five fixed step integration methods and a one variable step integration method. GODYS-PC provides a set of statements for generating a graphic representation of calculated results.

A model described in GODYS-PC consists of the following two parts: a model description and a runtime commands. The model description defines a model of a system being modeled. 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 may 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.

GODYS-PC is written in FORTRAN and consists of two modules. One of them is a syntax-directed translator, which generates an object program in a language of an abstract machine. This machine is implemented by an effective interpreter, which is the main part of the second module ([9]).

In order to illustrate the use of GODYS-PC let us consider the following classical model of the spring-mass-damper system shown in Fig. 1, where M is the mass, F is the force, K is the spring constant and D is the damping constant.

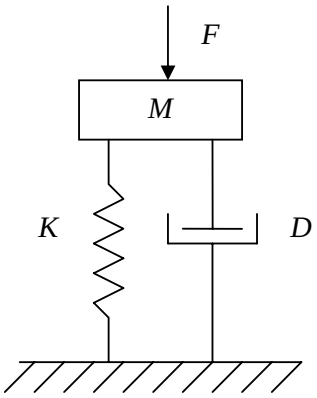

Fig. 1 The classical spring-mass-damper system

The mathematical model of the classical spring-mass-damper system, shown in Fig. 1, may be written by the following state-space differential equations:

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = -\frac{K}{M}\cdot x - \frac{D}{M}\cdot v + \frac{F}{M} \qquad (1)$$

where x is the position of the mass M and v is the velocity of the mass. Further, let us assume that the initial conditions are the following:

$$x(0) = 0 \ , \quad v(0) = 0$$

Further, let us assume that force F is the step input shown in Fig. 2 (step time = 10, initial value = 0, final value = 1).

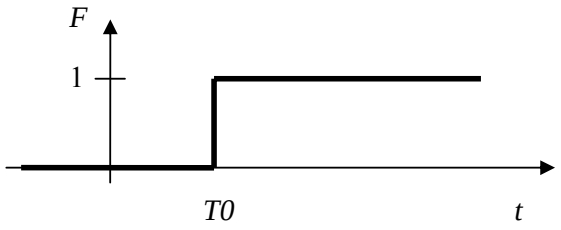

Fig. 2 The step input F

The program for the classical spring-mass-damper system (Fig. 1) written in GODYS-PC based on the state-space differential equations (1) is shown in Fig. 3.

```
MODEL SPRING
PREPARE F, x
PARAMT K, D, M, t0, A
DYNAMIC
x = INTEG(v; 0)
v = INTEG(– (K/M) * x – (D/M) * v + (F/M); 0)
F = A * STEP(t – t0)
END

LOAD SPRING
DATA K = 1, D = 0.3, M = 1, A = -1, t0 = 10
EXECUTE(DT = 0.1, METHOD = TRAPEZ, @
        TMAX = 50, COMDEL = 0.2)
PLOTXY(t, F)
PLOTXY(t, x)
FINISH
```

Fig. 3 The program for the classical spring-mass-damper system (Fig. 1) in GODYS-PC

Now for some explanation 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 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 METHOD specifies the name for the integration algorithm The parameter TMAX specifies the end of the simulation. The parameter COMDEL is the interval size for printing and plotting simulation results. The statement PLOTXY(t, F) creates a plot of the force F versus time t. Similarly, the statement PLOTXY(t, x) creates a plot of the position x of the mass M, versus time t. The outputs of the PLOTXY(t, F) and PLOTXY(t, x) statements are shown in Fig. 4 and in Fig. 5 respectively.

For comparison the use of GODYS-PC with the use of Simulink®, a model for the same classical spring-mass-damper system (Fig. 1) described in Simulink® is shown in Fig. 6. Fig. 7 shows a plot of the force F versus time t generated by Simulink® for the model shown in Fig. 6. Similarly, Fig. 8 shows a plot of the position x of the mass M, versus time t generated by Simulink® for the model shown in Fig. 6.

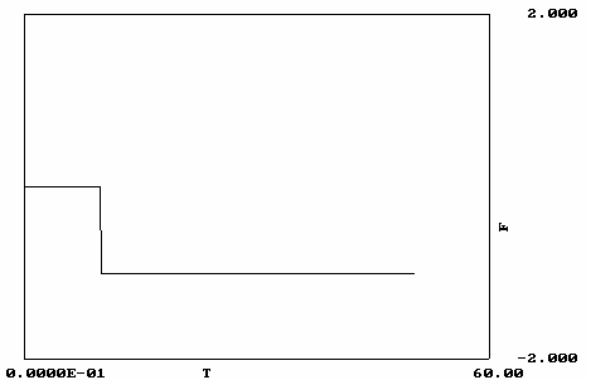

Fig. 4 The output of the PLOTXY(t, F) statement

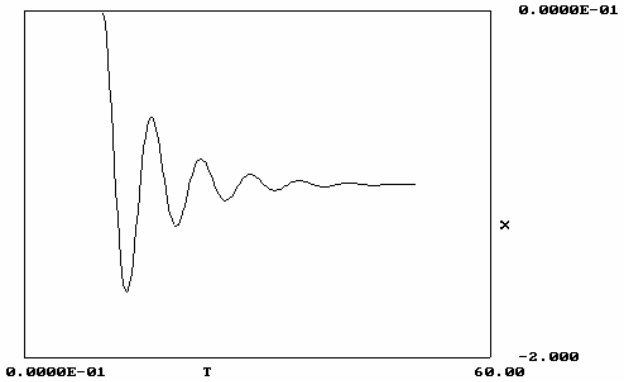

Fig. 5 The output of the PLOTXY(t, x) statement

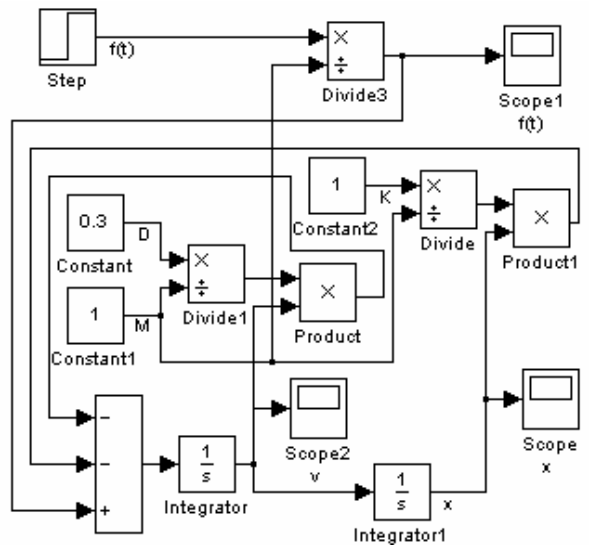

Fig. 6 A model for the classical spring-mass-damper system (Fig. 1) described in Simulink®

Now, let us consider the following classical electrical circuit *RLC* shown in Fig. 9, where *R* is the resistance, *L* is the inductance, *C* is the capacitance and *E* is the voltage. Further, let us assume that the voltage *E* is the step input.

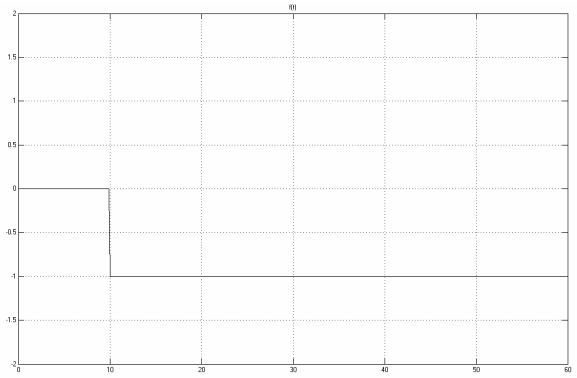

Fig. 7 The plot of F versus time t generated by Simulink®

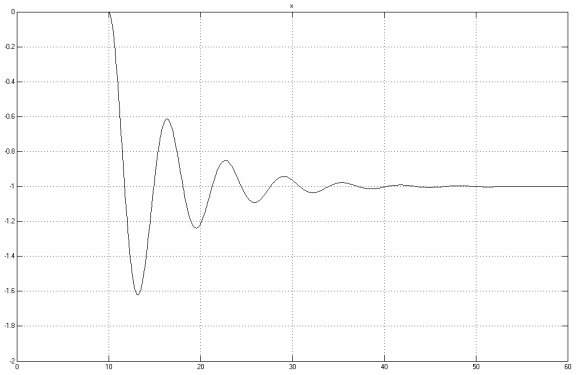

Fig. 8 The plot of x versus time t generated by Simulink®

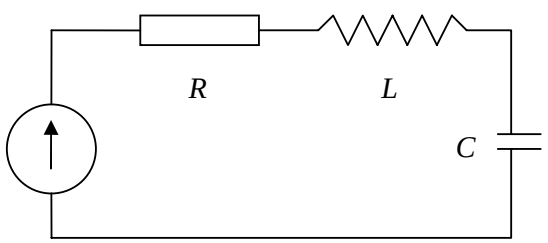

Fig. 9 The classical electrical circuit *RLC*

The mathematical model of the classical electrical *RLC* circuit shown in Fig. 9 may be written by following state-space differential equations :

$$\frac{dq}{dt} = i$$
$$\frac{di}{dt} = -\frac{R}{L}\cdot i - \frac{1}{C}\cdot q + \frac{E}{L} \qquad (2)$$

where i is the current. Further, let us assume that the initial conditions are the following:

$$q(0) = 0 \ , \quad i(0) = 0$$

The program for the classical electrical circuit *RLC* system (Fig. 7) written in GODYS-PC based on the state-space differential equations (2) is shown in Fig. 10.

```
MODEL RLC
PREPARE E, i
PARAMT t0, R, i, C
DYNAMIC
E = STEP(t - t0)
q = INTEG(i; 0)
i = INTEG((E/L) - (R/L)*i - (1/(L*C)*q); 0)
END

LOAD RLC
DATA t0 = 20, R = 1, L = 1, C = 1
EXECUTE(DT = 0.1, TMAX = 50, COMDEL = 0.1)
PLOTXY(t = (0, 50), E = (0, 2))
PLOTXY(t, i)
FINISH
```

Fig. 10 The program for the classical electrical circuit *RLC* (Fig. 7) in GODYS-PC

The output of the PLOTXY(t = (0, 50), E = (0, 2)) shown in Fig. 11 creates a plot of the voltage *E* versus time *t*. Similarly, the output of the PLOTXY(t, i) shown in Fig. 12 creates a plot of the current *i* versus time *t*.

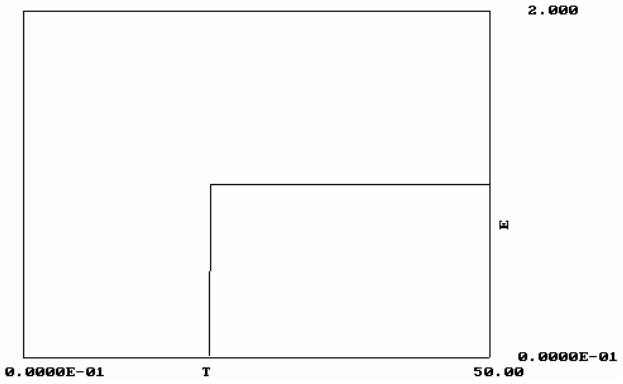

Fig. 11 The output of the PLOTXY(t = (0, 50), E = (0, 2)) statement

For comparison the use of GODYS-PC with the use of Simulink®, a model for the same classical electrical circuit *RLC* (Fig. 9) described in Simulink® is shown in Fig. 13. Fig. 14 shows a plot of the current *i* versus time *t* generated by Simulink® for the model shown in Fig. 13.

Now, let us consider a control loop block diagram shown in Fig. 15 ([11]).

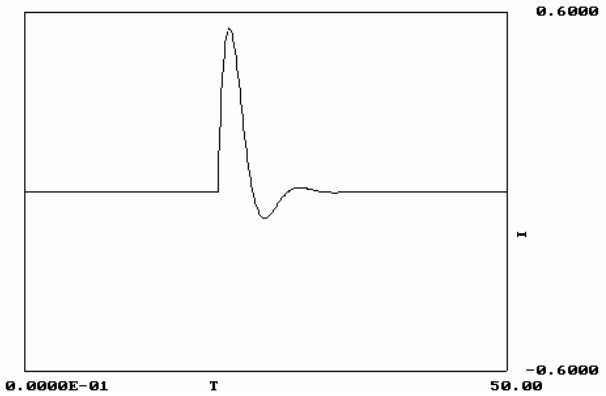

Fig. 12 The output of the PLOTXY(t, i) statement

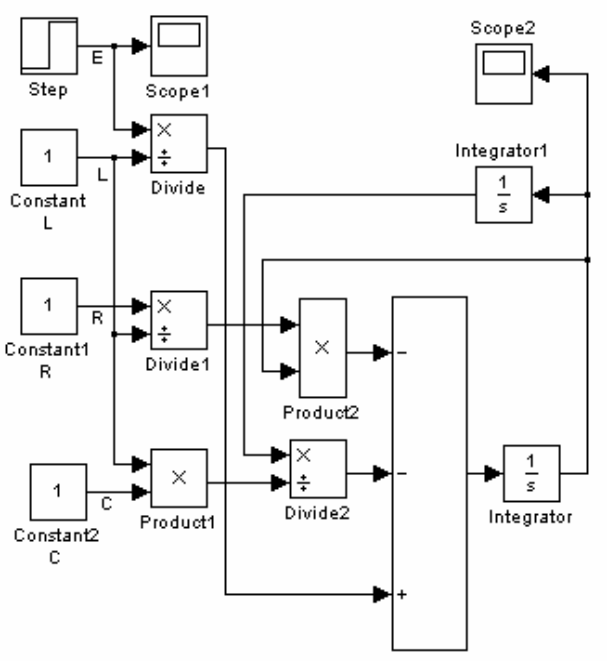

Fig. 13 A model for the classic electrical circuit *RLC* (Fig. 9) described in Simulink®

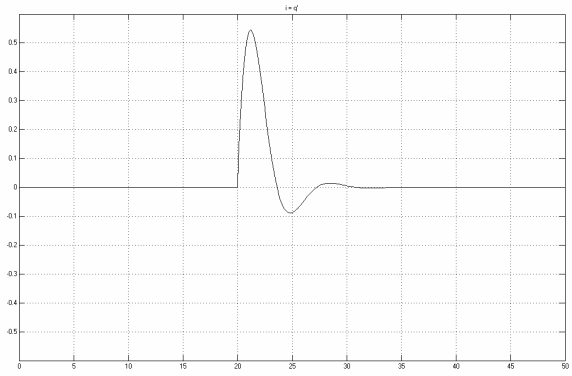

Fig. 14 The plot of *i* versus time *t* generated by Simulink®

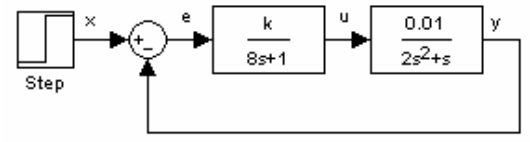

Fig. 15 A control loop block diagram

Further let us assume that the parameter k ∈ (5, 25) has to be determined such that the following cost function:

$$F = \int_0^{\infty} e^2 dt$$

is minimal.

The description of a model of the control loop block diagram (Fig. 15) in GODYS-PC is shown in Fig. 16.

```
MODEL CONTROL
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

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

Fig. 16 The program for the control loop block diagram (Fig. 15) in GODYS-PC

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

y = REALPL(x; a, y0)

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

$$\frac{x}{y} = \frac{1}{a \cdot s + 1}$$

where y0 = y(0). The parameter opt in the statement EXECUTE initiates parameter optimization. The parameter alg specifies a method for a cost function minimization algorithm. Each a cost function evaluation involves a simulation run. The parameter lim specifies a number of runs during parameter optimization. The minimal value 9.958 of the cost function *F* has been found for *k* = 17.36. The outputs of the PLOTXY(t, y), PLOTXY(t, e) and PLOTXY(t, F) statements are shown in Fig. 17, Fig. 18 and Fig. 21 respectively.

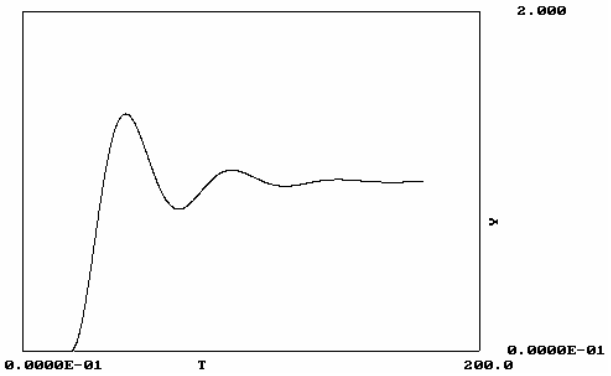

Fig. 17 The output of the PLOTXY(t, y) statement

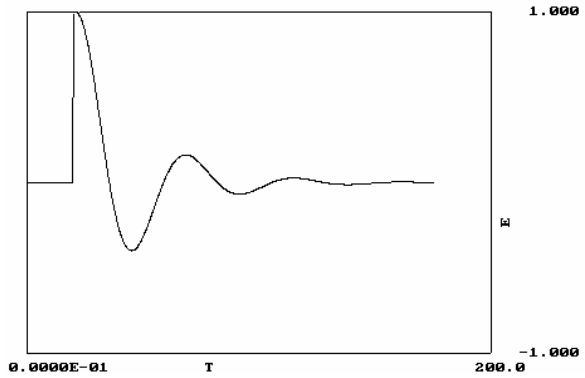

Fig. 18 The output of the PLOTXY(t, e) statement

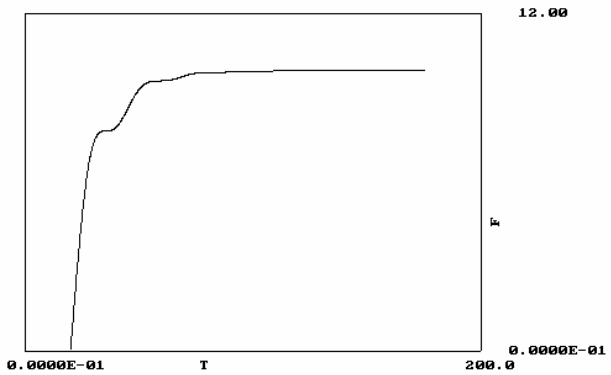

Fig. 19 The output of the PLOTXY(t, F) statement

For comparison the use of GODYS-PC with the use of Simulink®, a model for the same control loop block diagram (Fig. 15) described in Simulink® is shown in Fig. 20.

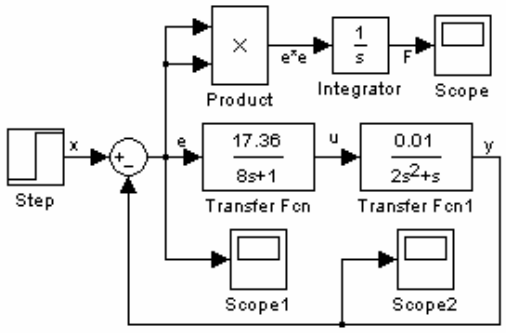

Fig. 20 A model for the control loop block diagram (Fig. 15) described in Simulink®

The plots of y, e and F versus time t generated by Simulink® for the model of the control loop block diagram (Fig. 20) are shown in Fig. 21, Fig. 22 and Fig. 23 respectively.

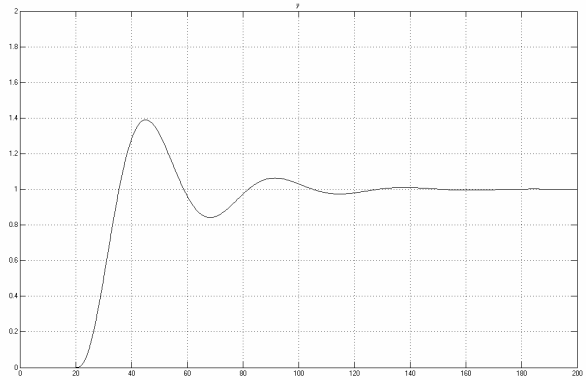

Fig. 21 The plot of y versus time t generated by Simulink®

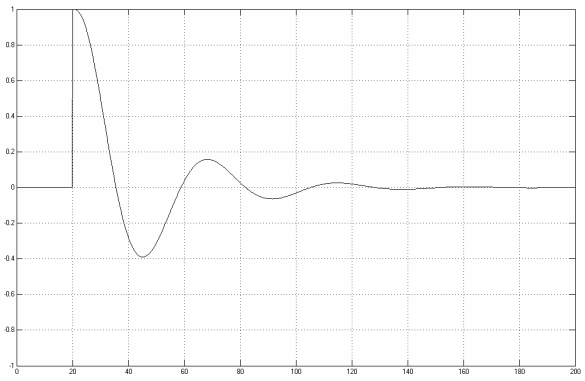

Fig. 22 The plot of e versus time t generated by Simulink®

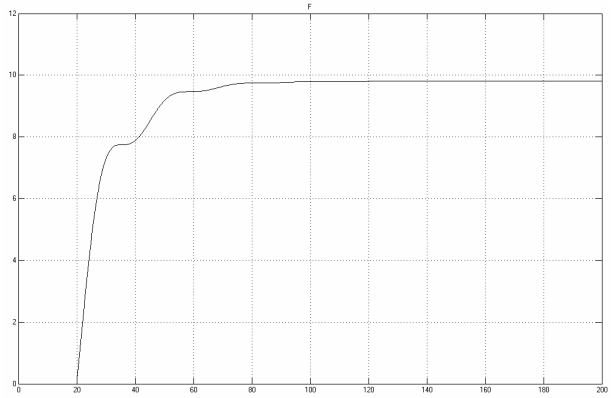

Fig. 23 The plot of F versus time t generated by Simulink®

III. Conclusion

In this paper we have presented GODYS-PC software package for modeling, simulating and analyzing dynamic systems. The basic structure of GODYS-PC follows the CSSL standard. GODYS-PC meets requirements needed for a good continuous simulation language. Typical areas in which GODYS-PC may be applied come from a wide diversity of realistic situations in engineering, control system design, economics, biology and so on. GODYS-PC is easy to learn even for somebody who is not an experienced programmer. In order to compare GODYS-PC with widely used in academia and industry Simulink®, the same examples has been provided both in GODYS-PC and Simulink®.

References

[1] Simulink® Model-Based and System-Based Design, Using Simulink, Version 5, The MathWorks, Inc., 2003.
[2] M. Rimvall, F. Cellier, Evolution and perspectives of simulation languages following the CSSL standard, *Modeling, Identification and Contro*l, 6,, no. 4, 1985.
[3] K. J. Astrom, H. Elmqvist, S. E. Mattsson, Evolution of continuous-time modeling and simulation, Proceedings of the 12th European Simulation Multiconference, Manchester, United Kingdom, June 16-19, 9-18, 1998.
[4] J. C. Strauss, D. C. Augustine, B. B. Johnson, R. N. Linebarger, F. J. Sanson, The SCi continuous system simulation language (CSSL), *Simulation* 9, no. 6, 1967, 281-303.
[5] R. Jakubowski, 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, 1968.
[6] R. Jakubowski, J. Król, Implementation of the simulation language based on functional graphs, *Podstawy sterowania*, 2, no. 2, 1972.
[7] R. Jakubowski, J. Król, General algorithm of complex dynamic systems simulation, *System Science*, 1, no. 1, 1975.
[8] R. Jakubowski, J. Król, Extended functional graphs in modeling and simulation of systems, *Podstawy sterowania*, 9, no. 2, 1979.
[9] J. Król, J. Kuraś, J. Lembas, M. Ślusarek, Implementation of the language for the simulation of continuous systems with discontinuities, *Podstawy sterowania*, 10, no. 1, 1980.
[10] J. R. Amyot, G. Blokland, Parameter optimization with ACSL models, *Simulation*, 9, 1987.
[11] K. Amborski, A. Marusak, Control theory in exercises (in Polish), Państwowe Wydawnictwa Naukowe, 1978.