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

Jacek Kursa, Jacek Lembus, and Marek Skomorowski

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

\[
\begin{align*}
\frac{dx}{dt} &= v \\
\frac{dv}{dt} &= -\frac{K}{M}x - \frac{D}{M}v + \frac{F}{M}
\end{align*}
\]  

(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).

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.

Now for some explanation about program itself. Elements shown in upper case can be written in lower case and vice versa. The statement \texttt{MODEL} identifies the model. The statement \texttt{PREPARE} specifies which variables are to be collected for later printing or plotting. The statement \texttt{DYNAMIC} identifies the beginning of the dynamic section. The dynamic section comprises a set of equations defining the model. The statement \texttt{DYNAMIC} must be accompanied by the matching statement \texttt{END}. In the example used here the dynamic section contains standard functions \texttt{INTEG} and \texttt{STEP}.

The statement \texttt{LOAD} identifies the beginning of runtime commands. The statement \texttt{LOAD} must be accompanied by the matching statement \texttt{FINISH}. The statement \texttt{DATA} is used to assign values to parameters of the model. The statement \texttt{EXECUTE} initiates the run. The parameter \( DT \) is the integration step size. The parameter \texttt{METHOD} specifies the integration algorithm. The parameter \texttt{TMAX} specifies the end of the simulation. The parameter \texttt{COMDEL} is the interval size for printing and plotting simulation results.

The statement \texttt{PLOTXY}(t, F) creates a plot of the force \( F \) versus time \( t \). Similarly, the statement \texttt{PLOTXY}(t, X) creates a plot of the position \( x \) of the mass \( M \) versus time \( t \). The outputs of the \texttt{PLOTXY}(t, F) and \texttt{PLOTXY}(t, X) statements are shown in Fig. 4 and in Fig. 5 respectively.

For the comparison of 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 the following state-space differential equations:

\[
\begin{align*}
\frac{dq}{dt} &= i \\
\frac{di}{dt} &= \frac{R}{L} i - \frac{1}{C} q + \frac{E}{L}
\end{align*}
\]

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
PARAM T0, R, L, C
DYNAMIC
E = STEP(t - T0)
q = INTEG(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 $t$.

![Diagram](image1)

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

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).

![Diagram](image2)

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

![Diagram](image3)

Fig. 14 The plot of $i$ versus time $t$ generated by Simulink®
Further let us assume that the parameter $k \in (5, 25)$ has to be determined such that the following cost function:

$$F = \int_0^T e^2 dt$$

is minimal.

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

```plaintext
MODEL CONTROL
PREPARE x, y, e, F
PARAM k, T
DYNAMIC
x = STEP(t - T)
\theta = x - y
u = k * REALPL(e, 8, 0)
y = REALPL(0.01*INTEG(u, 0), 2, 0)
f = INTEG(e^2; 0)
END
LOAD CONTROL
DATA @o = 20
EXECUTE(\text{max} = 175, \text{dt} = 0.1, \text{cosmol} = 0.1, @
\text{opt} = (F(x = (5, 25)), \text{alg} = \text{ns}, \text{lim} = 10))
PLOTXY(t, x)
PLOTXY(t, y)
PLOTXY(t, e)
PLOTXY(t, F)
FINISH
```

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

$$y = \text{REALPL}(x; a, y0)$$

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.

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.
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.

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 have been provided both in GODYS-PC and Simulink®.

REFERENCES


