Using the SIMULINK as a Teaching Tool

[1]Asad Yousuf, [2]Jiecai Luo, [3]Chun Ling Huang

[1]Engineering Technology Department, Savannah State University, Savannah, GA 31404/ [2]Electrical Engineering Department, Southern University, Baton Rouge, LA 70813/ [3]Mechanical Engineering Department/ Southern University, Baton Rouge, LA 70813

Abstract

SIMULINK is a tool for modeling, analyzing, and simulating physical and mathematical systems, including those with nonlinear elements and those that make use of continuous and discrete time. As an extension of MATLAB®, SIMULINK adds many features specific to dynamic systems for powerful and intuitive modeling. This paper introduces the basic elements of the SIMULINK in MATLAB® and describes how to use SIMULINK as a teaching tool to determine a target value for a design variable. This technique is applicable for use in teaching a design course or in a numerical methods course. We have solved two popular single degree of freedom dynamic problems to validate the proposed methodology. Several electrical or mechanical engineering courses such as ELEN 390 Linear Systems, ELEN 431 Control Systems Analysis, and MEEN 456 Engineering Modeling, Analysis and Control can be effectively use this technique to build the subject concepts and solve numerical problems in class. The present paper discusses and describes possible ways of using SIMULINK in ELEN 390 Linear Systems course. Possibilities of developing a control system for interactive problem solving are also discussed. Some unique advantages of SIMULINK include visualization of problem, extensive monitoring and evaluation of students' performance at each problem solving step, instant feedback and fair evaluation. Extensive data gathering on students' performance leading to better identification of each student's weak points will help in deciding the future orientation of the course at each step.

Introduction

Simulink[1] is a software package for modeling, simulating, and analyzing dynamic systems (whose outputs change over time). It can be used to explore the behavior of a wide range of real-world dynamic systems, including electrical circuits, shock absorbers, braking systems, and many other electrical, mechanical, and thermodynamic systems. It also supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multi-rate, i.e., have different parts that are sampled or updated at different rates. For modeling, Simulink provides a graphical user

interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, users can draw the models just as they would with pencil and paper. Simulink provides a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors.

Models are hierarchical, so users can build models using both top-down and bottom-up approaches. They can view the system at a high level, and then double-click on blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact. After they define a model, they can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running simulations. Using scopes and other display blocks, users can visualize the simulation results while the simulation is running. In addition, they can change parameters and immediately see what happens. The simulation results can be put in the MATLAB[2,3] workspace for post processing and visualization.

Model analysis tools include linearization[4] and trimming tools, which can be accessed from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes. And because MATLAB and Simulink are integrated, users can simulate, analyze, and revise their models in either environment at any point.

The rest of this paper is organized as follows: Section 2 describes the Simulink model selection. Section 3 sets up Simulink model. Simulink results will be provided in section 4. Some discussions will be given in the section 5.

Simulink Model Selection

There are many systems, which their dynamical behaviors can be described approximately by linear second differential equations, four typical examples are shown below.

(a) Electrical Systems[5]: Consider an inductance L, a resistance R and a capacitance C serial connection circuit. If the input voltage source to the network is $e_i(t)$, then the output voltage across the capacitance C, $e_o(t)$ is satisfied the equation:

$$\text{In frequency domain: } \frac{E_o(s)}{E_i(s)} = \frac{1}{LCs^2 + RCs + 1} \qquad (1)$$

$$\text{In time domain: } LC\frac{d^2 e_o(t)}{dt^2} + RC\frac{de_o(t)}{dt} + e_o(t) = e_i(t) \qquad (2)$$

(b) Mechanical Systems[6]: Consider a spring-mass damper system with an extra force *r(t),* and corresponding friction force and viscous force produced. The moving position *y(t)* satisfies in time domain:

$$M\frac{d^2 y(t)}{dt^2} + b\frac{dy(t)}{dt} + ky(t) = r(t) \qquad (3)$$

(c) DC motor with field controlled[6]: The input armature voltage *v(t)* to the rotational velocity $\omega(t)$ satisfies:

$$\text{In frequency domain: } \frac{\omega(s)}{V(s)} = \frac{K_m}{(Js+b)(L_f s + R_f)} \qquad (4)$$

(d) Car cruise-control model: The position *x(t)* of motion of a car with assuming that the engine imparts a force *u(t)* satisfies the equation[5] :

$$M\frac{d^2x(t)}{dt^2} + b\frac{dx(t)}{dt} = u(t) \qquad (5)$$

In summary, the linear second order differential models are a kind of typical models. For illustrative purpose and simplicity, here a linear second order differential model is selected to show how to use Simulink for simulation results. The model maybe is an electrical system, a mechanical system, a mechatronic system or some other. For the higher order systems, once our students get familiar with Simulink on second order system, they will be easily to simulate higher order systems with Simulink.

The standard linear second order differential model adopted here is

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) = \omega_n^2 u(t) \qquad (6)$$

From the system point of view, where *y(t)* is the system output and *u(t)* is the system input function. ω_n is the system's natural frequency, and ζ is the system's damping ratio.

In state space, by setting $x_1(t) = y(t), x_2(t) = \frac{dx_1(t)}{dt} (= \dot{x}_1)$, the equation (6) can be represented as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\omega_n^2 x_1 - 2\zeta\omega_n x_2 + \omega_n^2 u \end{aligned} \qquad (7)$$

According to the equation (7), a Simulink block model is set up in the next section.

Simulink Model Set-up

The Simulink block is built according to the equation (7), the detailed see figure 1. Here we leave the following parameters to be variables: $x_1(0), x_2(0), \zeta, \omega_n$ and simulation duration T. At the same time, three different input functions: step, ramp and sinewave are provided. Once an initial condition Matlab code has been run, all the data can be inputted from the keyboard. The Simulink model can be run either from the Matlab code or directly from the Simulink bar after the initial condition Matlab code is ran is ran.
The initial condition Matlab code is as follow:

```
clear;
disp('This code is to set all initial conditions for ');
disp('the simulation system');
```

```
disp('The initial parameters x_0 and dx_0/dt');
x10=input('please input x_0=');
x20=input('please input dx_0/dt=');
disp('The simulation duration T');
Tf=input('please input T=');
disp('natural frequency Omega_n');
omega=input('please input Omega_n=');
disp('damping ratio zeta');
zeta=input('please input zeta =');
K=menu('setting input functions:', 'step', 'ramp','sinewave', 'or no
input');
if K==1;
   a=0;b=1;c=0;
elseif K==2;
   a=1;b=0;c=0;
elseif K==3;
   a=0;b=0;c=1;
else
   a=0;b=0;c=0;
end;
Kk=menu('Choosing simulation path','from simulink block','from this
program');
if Kk==1;
   HL_asee_2003;
else
   [t,Y]=sim('asee_2003_simulink',[0,Tf]);
x=Y(:,2);y=Y(:,1);

figure(1)
plot(t,x);
xlabel('time t')
ylabel('x(t)')
title('time vs x(t)')

figure(2)
plot(t,y);
xlabel('time t')
ylabel('dx(t)/dt')
title('time vs dx(t)/dt')

figure(3)
plot(t(1:length(y2)),y2);
xlabel('time t')
ylabel('d^2x(t)/dt^2')
title('time vs d^2x(t)/dt^2')

figure(4)
plot(x,y);
xlabel('x(t)')
ylabel('dx(t)/dt')
title('x(t) vs dx(t)/dt')

figure(5)
plot(x(1:length(y2)),y2);
xlabel('x(t)')
ylabel('d^2x(t)/dt^2')
title('x(t) vs d^2x(t)/dt^2')
```

```
end;
```

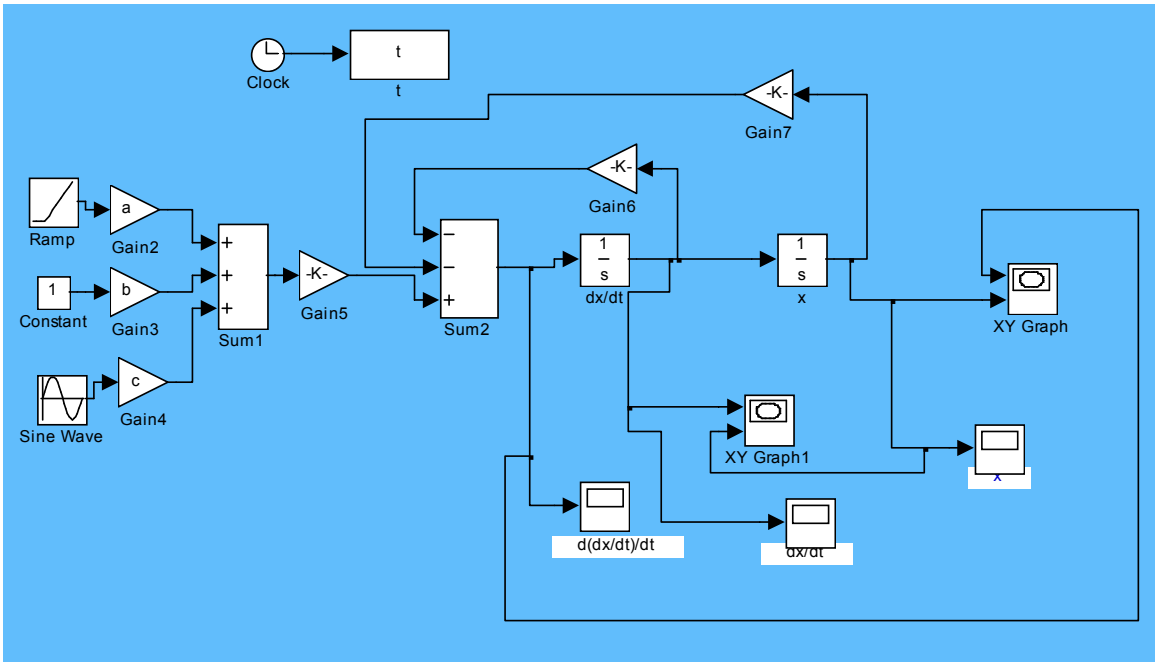

Figure 1: Simulink block model for equation (7).

Some simulation results by running the simulink block model above are given in the next section.

Simulink Model Running Results

Here we will use two examples to show how efficient to get simulation results.

Example One: By running Matlab initial setting code, we set $x_1(0) = 0, x_2(0) = 0, \zeta = 1, \omega_n = 1$, input is set to be unit step function, the simulation duration T=30 seconds. The first example is run from Matlab initial code. The simulation results are as follow:

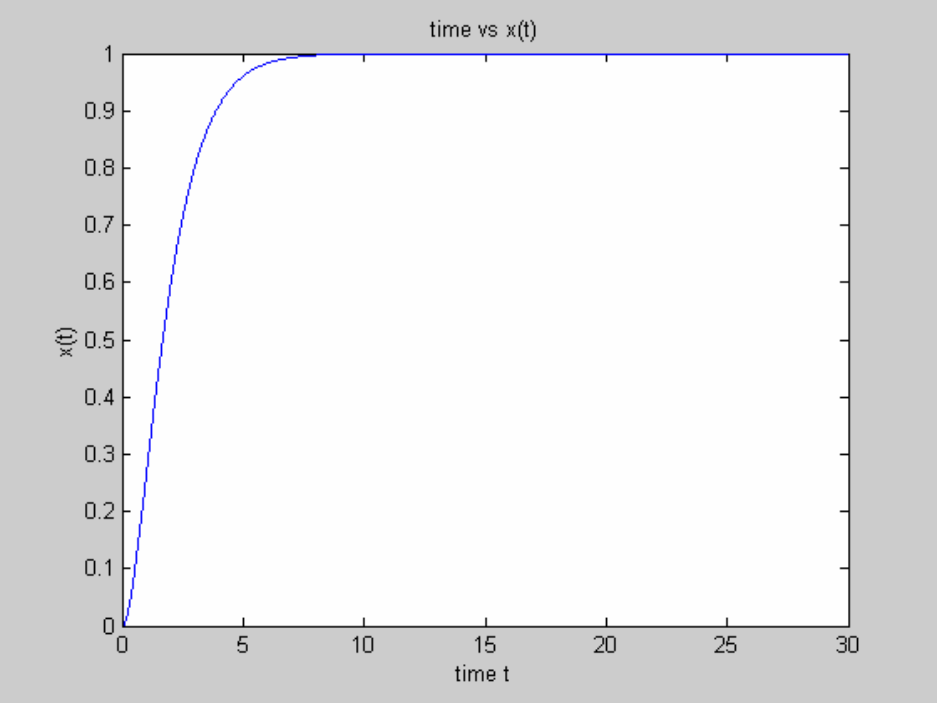

Figure 2: $x_1(t)$ (position) vs time

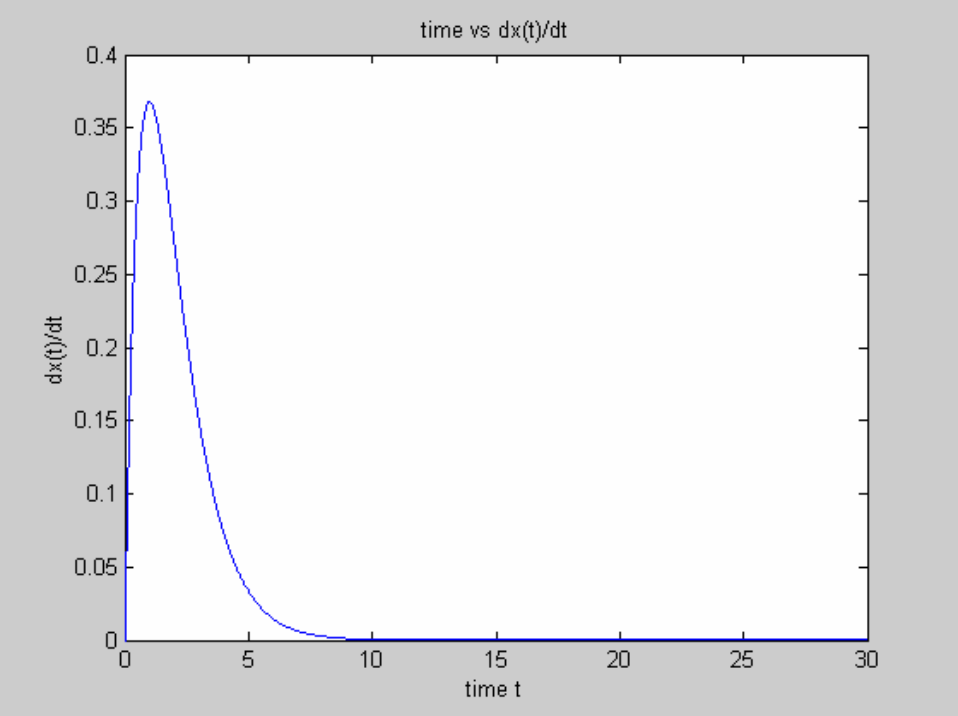

Figure 3: $x_2(t)$(speed) vs time

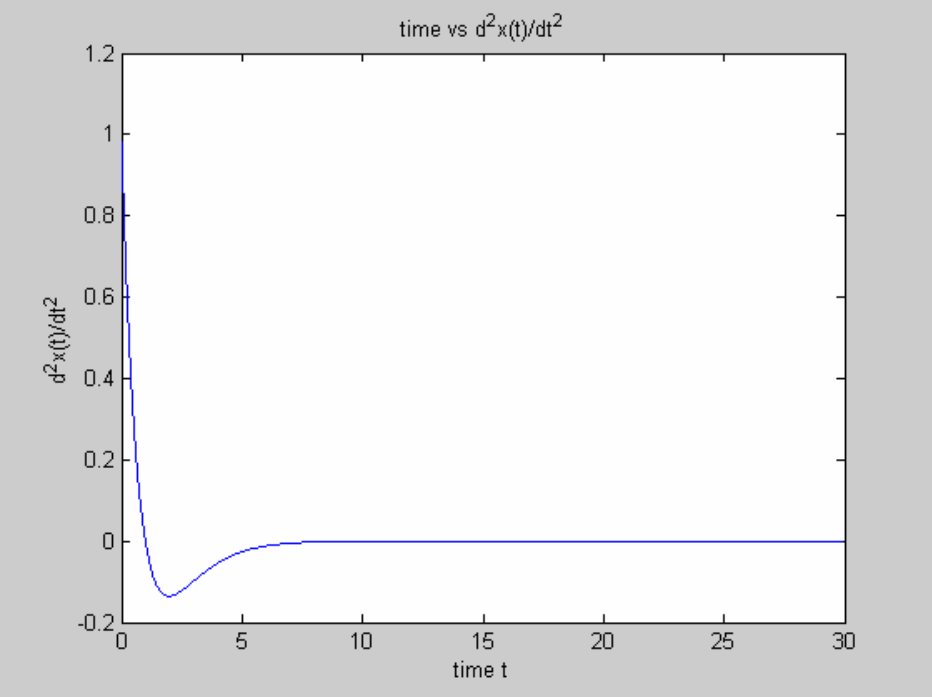

Figure 4: $dx_2(t)/dt$ (acceleration) vs time

Example Two: By running Matlab initial setting code, we set $x_1(0) = 1, x_2(0) = 1, \zeta = 0.3, \omega_n = 1$, input is set to be zero, the simulation duration T=30 seconds. This example is run from Simulink block board. The simulation results are as follow

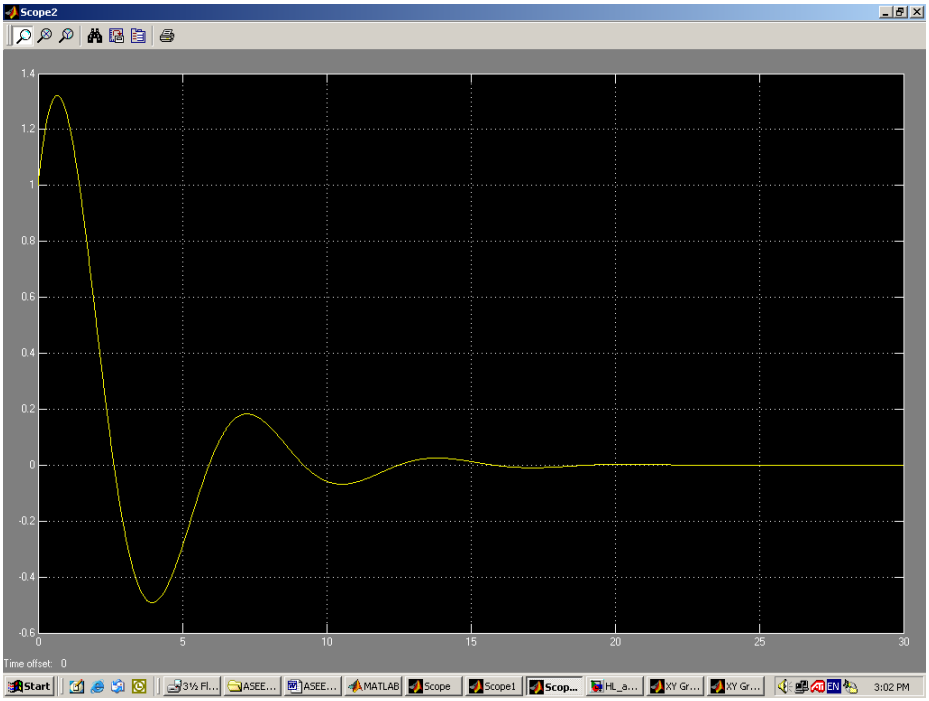

Figure 5: $x_1(t)$ (position) vs time

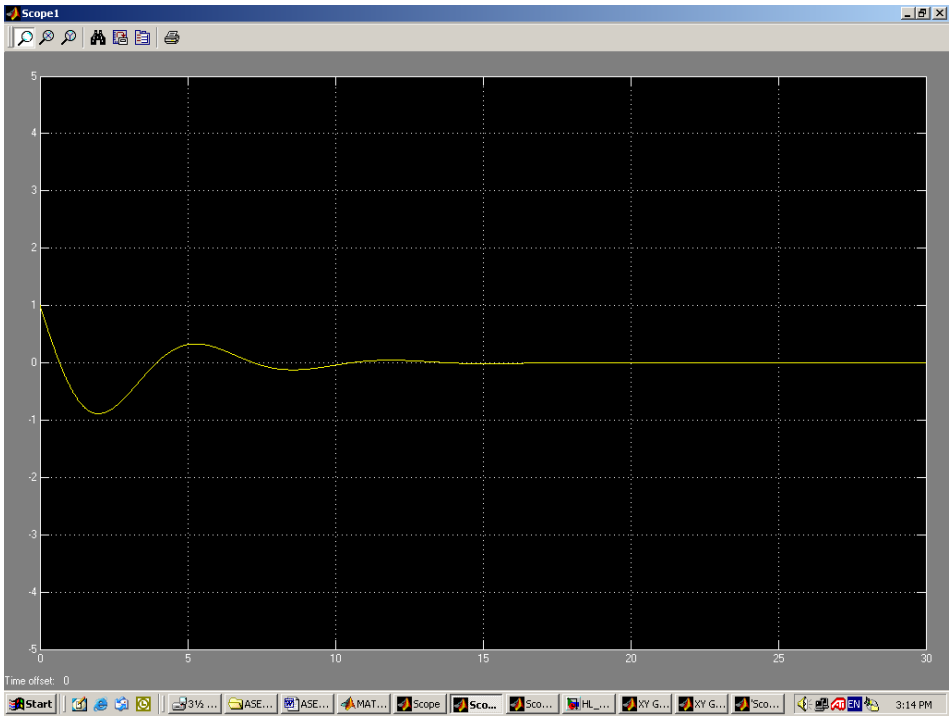

Figure 6: $x_2(t)$(speed) vs time

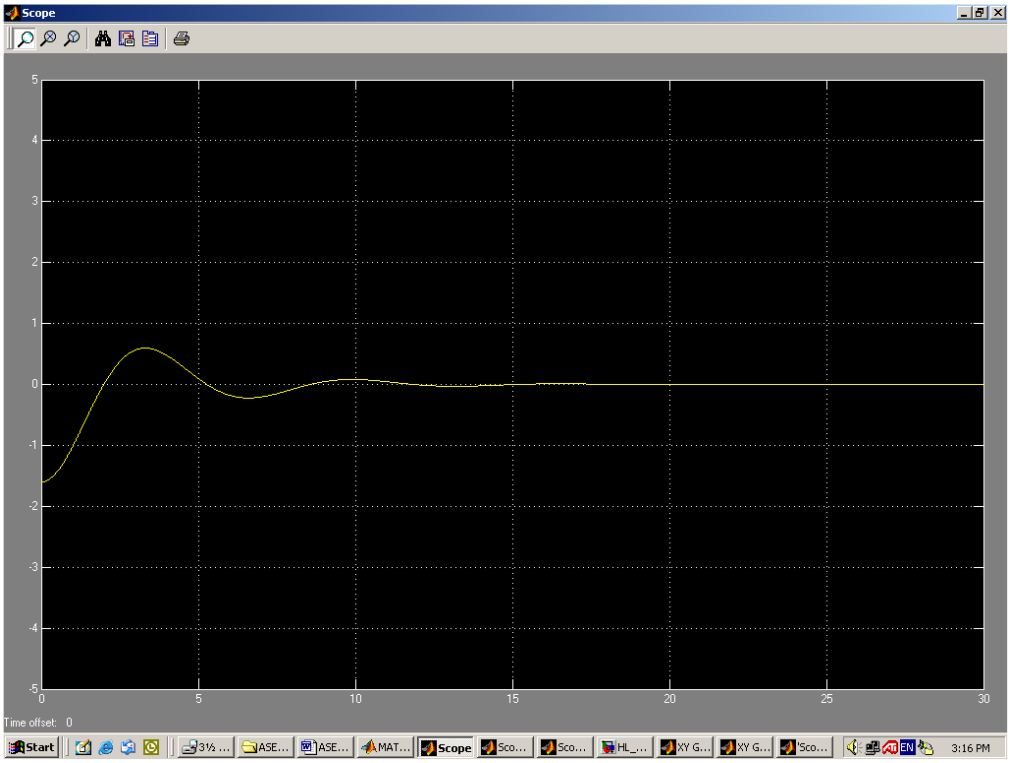

Figure 7: $dx_2(t)/dt$ (acceleration) vs time

The two example simulations give results which allow students to predict system performance based on the values of various operating parameters. Even though the

Simulink examples are linear second order systems, the principles can be used to model many other dynamic systems.

Several electrical or mechanical engineering courses such as ELEN 390 Linear Systems, ELEN 431 Control Systems Analysis, and MEEN 456 Engineering Modeling, Analysis and Control can be effectively use this technique to build the subject concepts and solve numerical problems in class. The present paper discusses and describes possible ways of using SIMULINK in ELEN 390 Linear Systems course. Possibilities of developing a control system for interactive problem solving are also discussed. Some unique advantages of SIMULINK include visualization of problem, extensive monitoring and evaluation of students' performance at each problem solving step, instant feedback and fair evaluation. Extensive data gathering on students' performance leading to better identification of each student's weak points will help in deciding the future orientation of the course at each step.

Currently, in new engineering building for College of Engineering, we have several multi-media lecture rooms for all the courses mentioned above. We may demonstrate the mathematical simulation results by MATLAB and SIMULINK software showing the parameters changing.

Conclusions

In this paper, we have seen how to use Simulink as a teaching tool for design. We have solved these single degree of freedom control problems[7]. A more interesting problem is to apply this method to more complex problems such as multi-degree of freedom problems with variable coefficients. This is beyond the scope of this paper and will be addressed in a future paper.

Bibliography

1. James B. Dabney and Thomas L. Harman, *Mastering Simulink 4*, *ISBN 0-13-017085-2*, Prentice Hall, Inc. (2001).
2. Duane Hanselman and Bruce Littlefield, *Mastering MATLAB® A Comprehensive Tutorial and Reference*, Prentice Hall, Inc. (2001).
3. http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/simulink.shtml
4. Steven C. Chapre and Raymond P. Canale, *Numerical Method for Engineers*, 4th Ed., McGraw Hill, Inc. (2002).
5. Katsuhiko Ogata, *Modern Control Engineering*, Third Edition, Prentice Hall, Inc.1997
6. Richard C. Dorf, *Modern Control Systems*, Ninth Edition, Prentice Hall, Inc. 2000.
7. Gene F. Franklin, J. David Powell and Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, Third Edition, Addison Wesley Publishing Company inc. 1994.

ASAD YOUSUF
Asad Yousuf is a Professor of Electronics Engineering Technology at Savannah State University. He received his BS in Electrical Engineering from the NED Engineering University, Karachi, Pakistan in 1980 and MS in Electrical Engineering from the University of Cincinnati in 1982 and an Ed.D. from the University of Georgia in 1999. Asad is a registered Professional Engineer in Georgia He is also a Microsoft Certified System Engineer (MCSE).

JEICAI LUO
Jiecai Luo earned B. S. and M. S. in Electrical Engineering from Tongji University at Shanghai and HUST at Wuhan in China respectively, and PhD from University of Minnesota. He currently serves as an Assistant Professor of Electrical Engineering at Southern University, Baton Rouge. His research interests include control systems, optimal control and material science. He is a member of IEEE.

CHUN LING HUANG .
Chun Ling Huang earned B.S. and M.S. degrees in Mechanical Engineering from Chung Yuan Christian University (CYCU) in Taiwan, and a Ph.D. degree in Mechanical Engineering from the University of Alabama (UA) at Tuscaloosa. Currently, Huang is an Associate Professor of Mechanical Engineering at Southern University, Baton Rouge, Louisiana. He is a member of ASME and ASEE.