



## **ET1222 – Modeling and Verification**

### **Hands-on Lab**

## **Introduction to Simulink and Systems Simulation**

*Anders Hultgren & Shayan Haider*

### **Objective**

The objective of this lab is to get familiarized with Simulink, which is an extension of MATLAB, learn how to implement systems with given mathematical description and analyze them. You will use what you have learned in this lab in upcoming projects where various systems will have to be simulated in Simulink or implemented in dSPACE DSP board.

# Simulink & Matlab – Simulating Dynamic Systems

This lab introduces Simulink concepts necessary to model dynamic systems. Simulink is an extension of Matlab that provides a graphical environment to model, simulate and analyze dynamic systems. Simulink contains a number of toolbox libraries and each toolbox in Matlab has its own specific function blocks, which can be combined to give great flexibility in simulation and modeling.

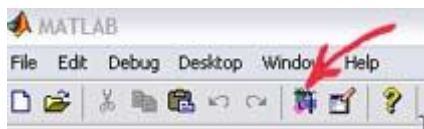
## Concept 1: Getting to Know Simulink

To start a Simulink session, you'd need to bring up Matlab program first.

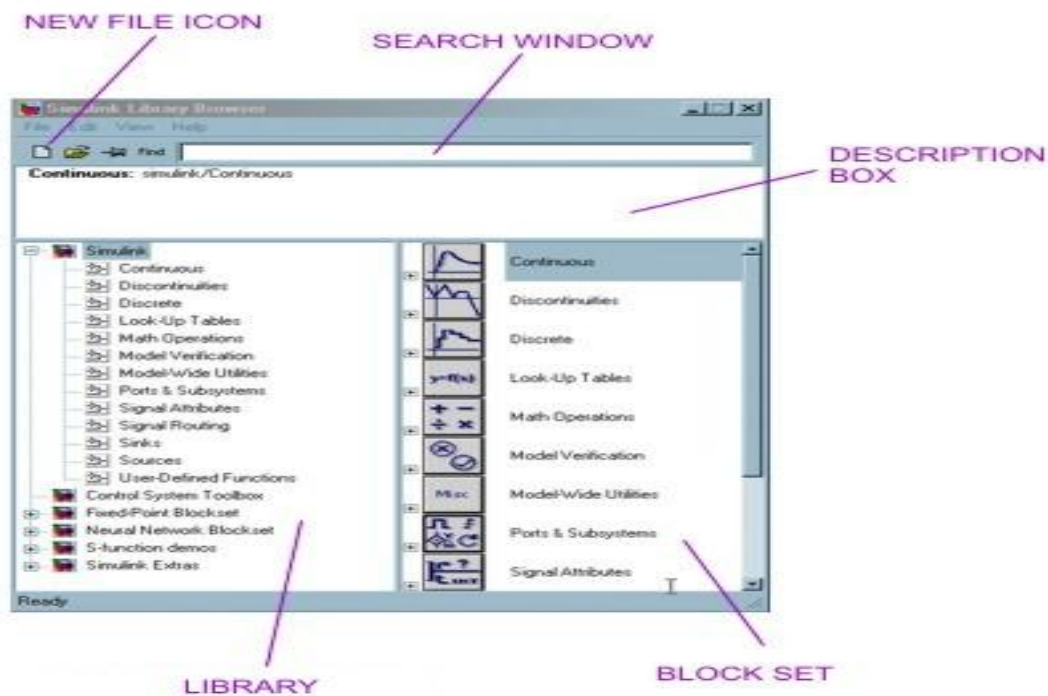
From Matlab command window, enter:

```
>> simulink
```

Alternately, you may click on the Simulink icon located on the toolbar as shown:



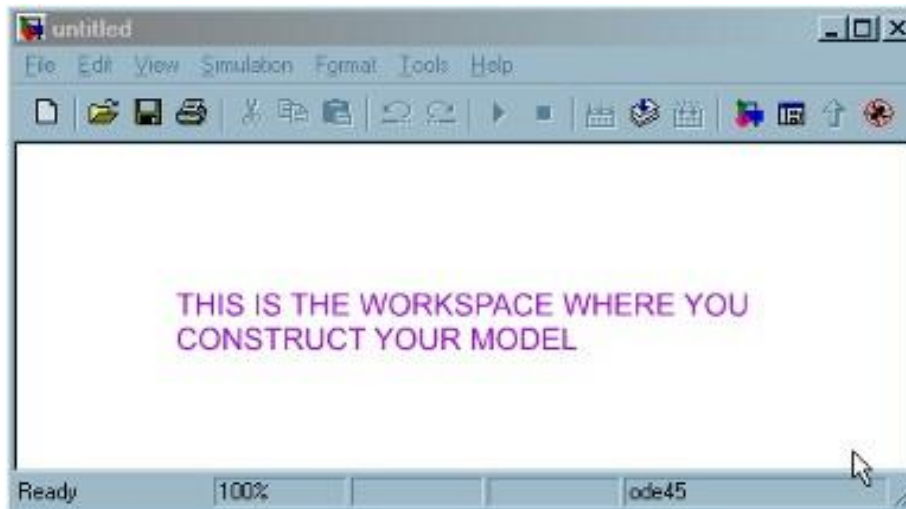
Simulink's library browser window like one shown below will pop up presenting the block set for model construction.



To see the content of the blockset, click on the "+" sign at the beginning of each toolbox.

To start a model click on the NEW FILE ICON as shown in the screenshot above. Alternately, you may use keystrokes **CTRL+N**.

A new window will appear on the screen. You will be constructing your model in this window. Also in this window the constructed model is simulated. A screenshot of a typical working (model) window is shown below:



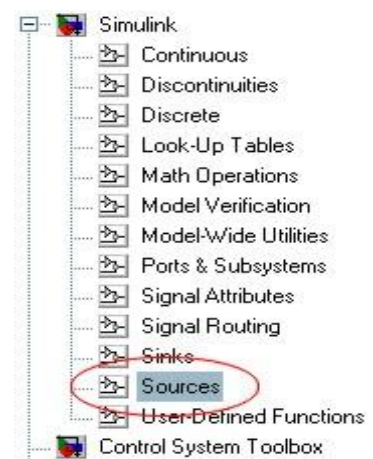
To become familiarized with the structure and the environment of Simulink, you are encouraged to explore the toolboxes and scan their contents. You may not know what they are all about at first, but perhaps you could catch on the organization of these toolboxes according to their categories. For instance, you may see that the Control System toolbox consists of the Linear Time Invariant (LTI) system library and the Matlab functions can be found under Function and Tables of the Simulink main toolbox. A good way to learn Simulink (or any computer program in general) is to practice and explore. Making mistakes is part of the learning curve. So, fear not you should be!

A simple model is used here to introduce some basic features of Simulink. Please follow the steps below to construct a simple model.

### *STEP 1: CREATING BLOCKS.*

From **BLOCK SET CATEGORIES** section of the **SIMULINK LIBRARY BROWSER** window, click on the "+" sign next to the **Simulink** group to expand the tree and select (click on) **Sources**.


A set of blocks will appear in the **BLOCKSET** group. Click on the **Sine Wave** block and drag it to the workspace window (also known as model window).





Now you have established a source of your model.

NOTE: It is advisable that you save your model at some point early on so that if your PC crashes you wouldn't lose too much time reconstructing your model.

Save this model under the filename: "simexample1". To save a model, you may click on the floppy diskette icon  or from FILE menu, select **Save** or using keystrokes CTRL+S. All Simulink model file will have an extension ".mdl". Simulink recognizes file with .mdl extension as a simulation model (similar to how MATLAB recognizes files with the extension .m as an MFile).

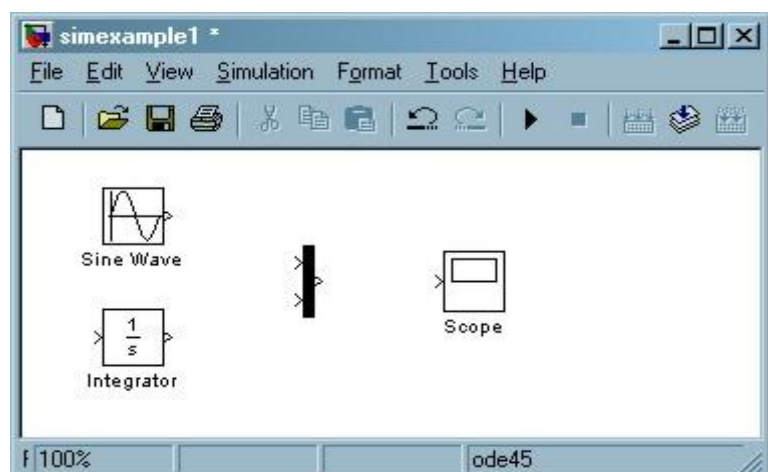
Continue to build your model by adding more components (or blocks) to your model window. We'll continue to add a **Scope** from **Sinks** library, an **Integrator** block from **Continuous** library, and a **Mux** block from **Signal Routing** library.

NOTE: If you wish to locate a block knowing its name, you may enter the name in the SEARCH WINDOW (at Find prompt) and Simulink will bring up the specified block.

To move the blocks around, simply click on it and drag it to a desired location.

Once you've dragged over all necessary blocks, the workspace window should consist of the following components:

You may remove (delete) a block by simply clicking on it once to turn on the "select mode" (with four corner boxes) and use the DEL key

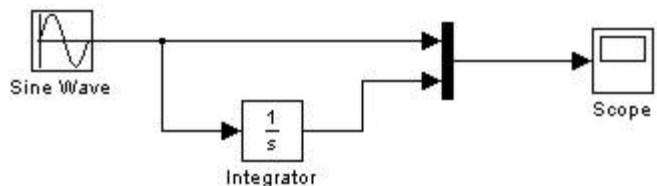


## STEP 2: MAKING CONNECTIONS

To establish connections between the blocks, move the cursor to the output port represented by ">" sign on the block. Once placed at a port, the cursor will turn into a cross "+" enabling you to make connection between blocks.


To make a connection: left-click while holding down the control key (on your keyboard) and drag from source port to a destination port.

The connected model is shown below.

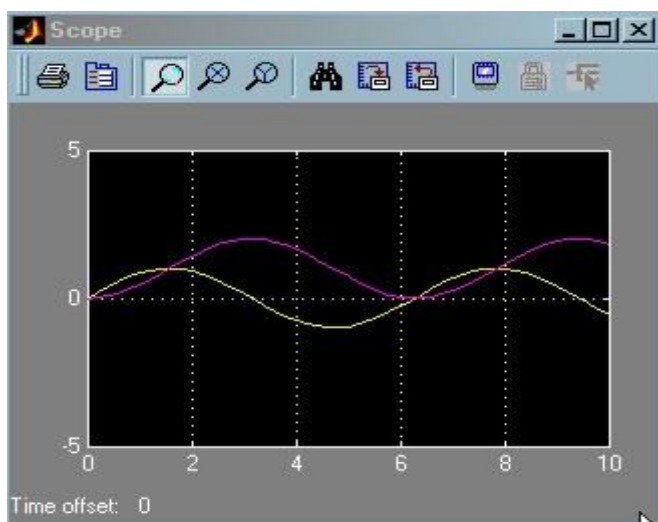


A sine signal is generated by the Sine Wave block (a source) and is displayed by the scope. The integrated sine signal is sent to scope for display along with the original signal from the source via the **Mux**, whose function is to multiplex signals in form of scalar, vector, or matrix into a bus.

## STEP 3: RUNNING SIMULATION

You now may run the simulation of the simple system above by clicking on the play button . Alternately, you may use keystrokes CTRL+T, or choose Start submenu (under Simulation menu).

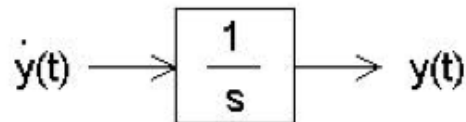
Double click on the Scope block to display of the scope.



To view/edit the parameters, simply double click on the block of interest.

## Concept 2: Modeling Continuous Systems with Differential Equations

The generic building block of a continuous system is the *Integrator* block (found under the *Continuous* tab). The integrator block will, as the name implies, integrate the input signal:



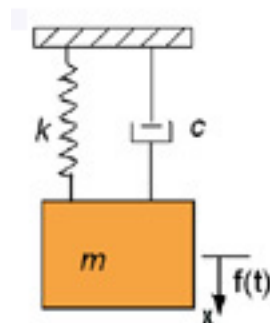
**Note:** Derivative blocks are not commonly used to construct differential equations because they do not allow you to store any initial conditions.

When modeling the dynamics of a system (i.e. a differential equation), the following procedure can be used as a general guideline:

1. Arrange the equation such that the highest order derivative is on the left and all other terms are on the right.
2. Determine the number of *Integrator* blocks required (2<sup>nd</sup> order requires 2) and drag them into the workspace.
3. Connect the Integrator blocks and label the input and output of each block.
4. Construct the equation, one differential equation at a time.

To get some practice modeling a continuous system via differential equation and to further emphasize the guidelines above, let's look at the Mass-Spring-Damper mechanical system.

**Example.** Mass-Spring-Damper System Simulation



Consider a mass-spring-damper system as shown in Figure. The mathematical model for this system is described by

$$m\ddot{x} + c\dot{x} + kx = f(t) \quad (2)$$



where  $m$  is the equivalent mass of the system,  $c$  is the damping ratio,  $k$  is the spring stiffness, and  $f(t)$  is the forcing function in the  $x$ -direction.

This example will illustrate how to use Simulink to simulate the response of this system to unit step input.

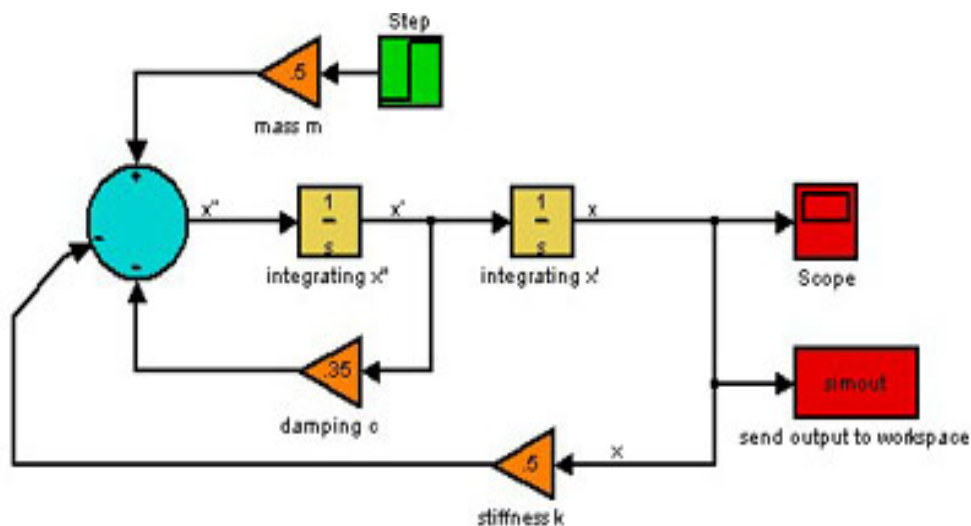
*Step 1.* In Simulink, create a new model window (CTRL+N) and drag the following blocks from the Simulink library window:

Blocks to be dragged to the model window	Where located in Simulink library browser
Step	Sources
Gain	Math Operation
Sum	Math Operation
Integrator	Continuous
Scope	Sinks
To Workspace	Sinks


*Step 2.* By re-arranging Eqn 2 to yield an expression for the acceleration term, Eqn (2) becomes

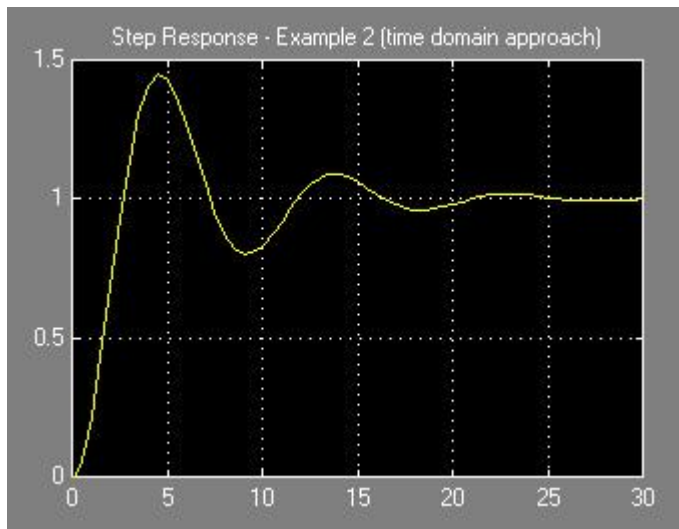
$$\ddot{x} = \frac{1}{m} (f(t) - c\dot{x} - kx) \quad (3)$$

Based on Eqn 3, we connect the blocks in the diagram as shown in Figure Below. Use CTRL+I and CTRL+R to flip and rotate the blocks as necessary (select the block first then execute the key sequence). Note that you can use CTRL+right mouse button to create branches of the connecting lines. Don't worry about the parameter values and the signs for these blocks at this point, as we'll take care of this in STEP 3. Just get them connected first.



*Step 3.* Enter the values of the parameters for each block. In this example, we will set  $m = 2.0$ ;  $c=0.7$ ;  $k=1$ . You are encouraged to try different values and observe the system's response to step input.

Run the simulation by clicking on the  button (alternately you may use keyboard command CTRL+T ). The screenshot of the output from the Scope block is show in Figure below.



That's it! You have successfully modeled and simulated a second-order Under-damped dynamic system. To exam different responses, feel free to change different values for  $m$ ,  $c$ , and  $k$  in the gain blocks.

### Concept 3: Modeling Continuous Systems with State Space Representation

For a continuous time system, state space notation expresses its dynamics in the form of first-order differential equations. The general equation of state space representation is:

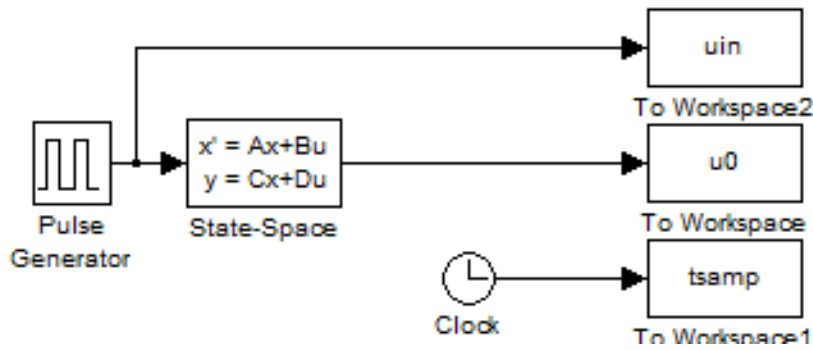
$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Let's model the RLC electrical circuit for which we found the state space parameters earlier.

**Step 1:** Create a new model as shown in the figure below. (*State-Space* block is located in the Continuous library).





- Open the pulse generator block by double clicking on it. Set up the amplitude parameter to  $P_{amp}$  variable, period parameter to  $P_{period}$  variable, pulse width parameter to  $P_{pw}$  variable and phase delay parameter to  $P_{pd}$  variable. Keep other parameters as default.
- Open the State-Space block and set up the state space parameters as  $A, B, C, D$  and initial conditions parameter to  $x_0$ , keep the others as default.
- Open the To Workspace blocks and set up their respective variable names as shown in the figure above, sample time parameter to  $h$  and change save format parameter to *array*.

Blocks of type To Workspace are added such that certain process variables are saved during the simulation. These variables are stored in an internal memory being kept by Matlab.

Save the model as “ElectricCircuitRLC.mdl”.

**Step 2:** Go back to Matlab and create a new MFile. This file will contain definitions of parameters, will run the simulation and eventually will present the result from the simulation.

Copy the text given below in this newly created Mfile.

```

clear all
close all

%% Parameters

%Circuit parameters
CC=1000e-6;
L=470e-6;
RC=0.15;
RL=0.17;

%model parameters

```

```

A=[0    1/CC
   -1/L -(RC+RL)/L];
B=[0
   1/L];
C=[1 0];
D=0;

```

**%Initial conditions**

```
x0=[0;0];
```

**%Pulse Generator Parameters**

```

Pamp=0.5;
Pperiod=0.2;
Ppw=50;
Ppd=0.02;

```

**%Sampling Interval**

```
h=1e-4;
```

**%% Simulation**

```

simtime=0.2;
optionvec=simset('Solver','ode45','AbsTol',1e-9,'RelTol',1e-6);
sim('ElectricCircuitRLC',simtime, optionvec);

```

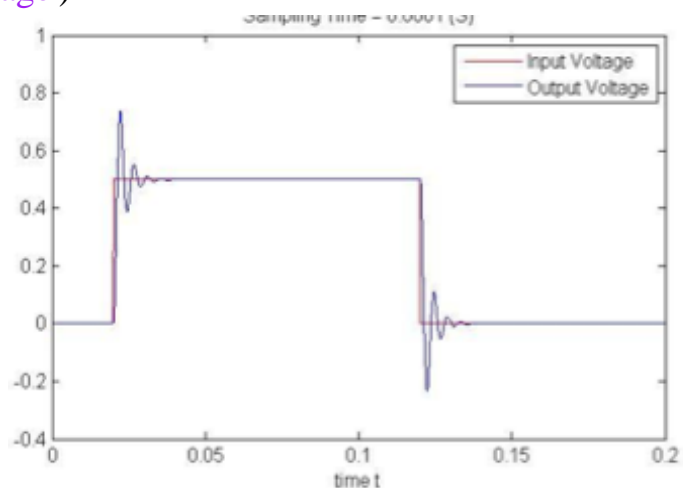
**%% plot**

```

figure(1)
plot(tsamp,uin,'r')
hold on
plot(tsamp,u0)
title('Simulation of electrical circuit with Step Input');
xlabel('time t')
legend('Input Voltage','Output Voltage')

```

**Step 3:** Execute the Mfile by either typing the name of the Mfile created above in the Matlab command window or by opening the m-file and selecting run from the debug menu. You should be able to see the figure as shown.



## Some hints in order to get a more efficient use Simulink

Simulink facilitates simulations of systems, but it is not necessary to use it. We believe that all simulations possible to do with Simulink are also possible to do without. However, Simulink makes it so much easier so we would not even try to describe how to simulate dynamical systems without Simulink.

There is no sharp border between what could be done in Matlab and what could be done in Simulink. For instance values for different blocks and configuration parameters can be given using a window. It is not necessary to do so. In the first preliminary stage of a project you may prefer to play around with system blocks, parameters etc. However, when you start to do work you are going to show other people there is a sharp border between what should be done in Matlab and what should be done in Simulink. Therefore, it is a good practice to define all the model parameters in Mfile rather than setting up the value using block windows as with this approach your Mfile acts as a kind of documentation for another person using your work, you can efficiently change different parameter in one place rather than going to several different configuration windows.

Each command is explained in Matlab in two ways. To see these two explanations you just use the help function. For instance if you want to know about the command *simset* write

*help simset*

in your Command Window. Then you will get an explanation about the command *simset* written in your Command Window.

## Lab Assignment: Modeling an office chair

Figure A is a simple model of an office chair. The model consists of a mass, basically the mass of the seat, an ideal damper and an ideal spring. Our measurements start from the equilibrium that exists before we take a seat on the chair, i.e.  $x(0) = 0$  and  $\dot{x}(0) = 0$ . The action of “taking a seat” is represented by an increase in mass from  $m$  to  $m+M$ , where  $M$  represents the mass of the body of the person placed on the chair (75 Kg). This is illustrated by figure B.

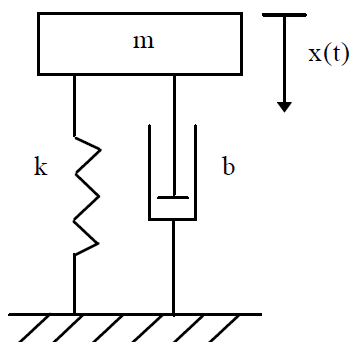


Figure A

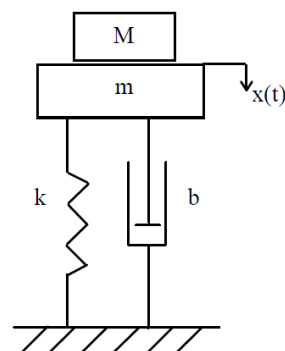


Figure B

*Your assignment is to decide a suitable damping of the chair. Choose a suitable damping by study the movements when you sit down on the chair. The spring is already selected to give a proper static change of level of the seat when a person is sitting on the chair.*

% Gravitation acceleration

g=9.8;

% Mass of office chair seat

m=5;

% Mass of person taking a seat on the office chair

M=75;

% Damping constant, value has to be entered before running simulation

b=?;

% Spring constant

k=25920;

% External force acting on the chair when someone is parked on the seat

Mg=M\*g;

Run four simulations with different values of the damping constant b, 800, 12800, 2880 Ns/m and your own proposal of a suitable value of the damping constant. Change the value of the damping constant in your mfile before every simulation. From the tests on the subject of damping of an office chair, document all experimental results with comments.

### **Bibliography**

- [www.ee.kth.se/control/courses/EL182](http://www.ee.kth.se/control/courses/EL182)
- <http://www.dartmouth.edu/~sullivan/>
- <http://www.pages.drexel.edu/~pyo22/>
- <http://www.me.cmu.edu/ctms/modeling/>
- <http://edu.levitas.net/Tutorials/Matlab/>
- [Introduction to Simulink \(Anders Hultgren and Matz Lennels\)](#)