

EENG307: Mechanical Impedance and Introduction to Modeling with Simulink*

Lecture 6

Elenya Grant, Kathryn Johnson, and Hisham Sager[†]

Fall 2022

Contents

1	Pre-requisite Material	1
2	Mechanical Impedance	1
2.1	Example	3
3	Modeling Systems in Simulink	4
4	Lecture Highlights	11
5	Quiz Yourself	11
5.1	Questions	11
5.2	Solutions	12

1 Pre-requisite Material

This lecture assumes that the reader is familiar with the following material:

- Lecture 2: Modeling Mechanical Systems
- Lecture 5: Impedance and Transfer Functions

2 Mechanical Impedance

We have seen how impedance is used to find transfer functions for circuits in Lecture 5. Impedance can also be used in other modeling domains. While some textbooks define mechanical impedance as the ratio of the Laplace transform of force to position, we will *not* use this definition, as it is not compatible with electrical systems. Instead, we will make the following general definition of impedance.

Definition 1. The *impedance* of an element is the ratio of the Laplace Transform of the across variable over the Laplace Transform of the through variable

^{*} This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

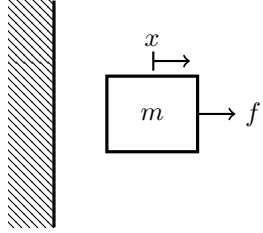
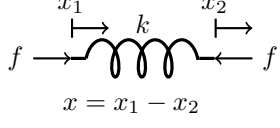
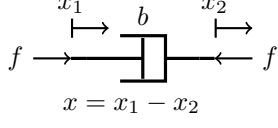
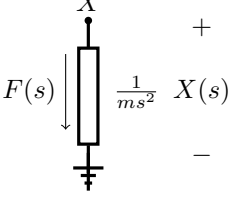
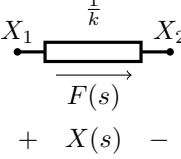
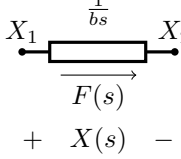
[†] Developed and edited by Tyrone Vincent and Kathryn Johnson, Colorado School of Mines, with contributions from Salman Mohagheghi, Chris Coulston, Kevin Moore, CSM and Matt Kupilik, University of Alaska, Anchorage

Across and Through Variables with Mechanical Emphasis

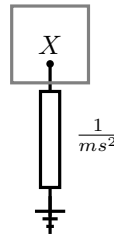
Domain	Across Variable	Through Variable
Electrical	Voltage	Current
Translational Mechanical	Position	Force
Fluid	Pressure	Flow
Rotational Mechanical	Angular Position	Torque
Thermal	Temperature	Heat Flow

As shown in the table above, in mechanical systems the across variable is position, while the through variable is force, so we have the impedances in the following table.

Mechanical Impedance

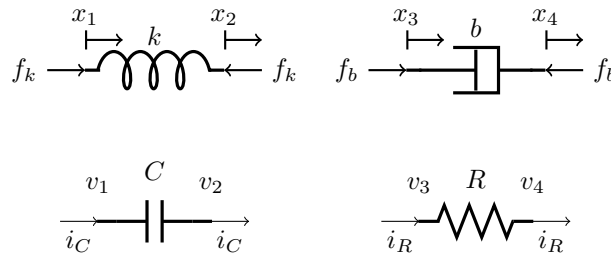
	mass	spring	damper
Component			
Component law	$m\ddot{x} = f$	$f = kx$	$f = b\dot{x}$
Laplace Transform	$X(s) = \frac{1}{ms^2}F(s)$	$X(s) = \frac{1}{k}F(s)$	$X(s) = \frac{1}{bs}F(s)$
Impedance Component (positive f direction agrees with positive x direction)			

Note that since the position of the mass is relative to a fixed measurement frame – i.e., one with position, velocity, and acceleration equal to zero – one end of the mass impedance element is *always* connected to ground. When translating between mechanical drawings and impedance networks, it is helpful to visualize the mass sitting on top of the mass impedance, like this:



Remark 2. One detail that may seem odd when considering force as a flow variable is the fact that force is defined as acting in opposite directions on the ends of each element. However, despite this, the interconnection rules are exactly the same. That is, for the example below, if the spring and damper are interconnected, we have $f_k = f_b$, while if the capacitor and resistor are connected, $i_C = i_R$. So in both cases, the flow variable on the left hand side of one element “flows” to the left hand side of the other element.

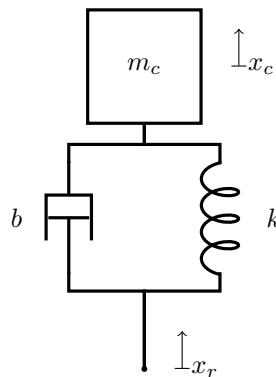
Force Flow vs Current Flow



It's easiest to explain modeling with impedances using an example that lays out the necessary steps.

2.1 Example

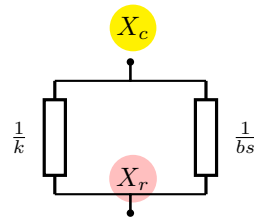
Let's model the automobile suspension system first introduced in Lecture 2: Modeling Mechanical Systems using impedance concepts. The system description using ideal elements is the following, where x_r is the road position (input signal) and x_c is the car position (output signal)



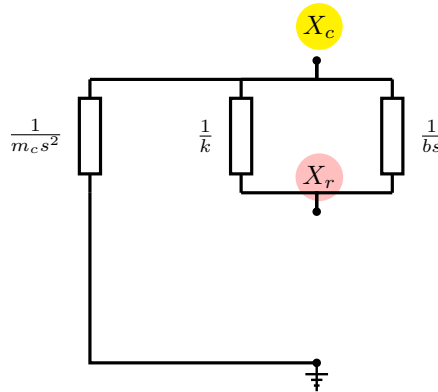
- Step 1 Identify all node variables.** For mechanical systems, positions are the across variables, so positions are the nodes. For this system, we see two independent positions: x_c and x_r . **Note:** In impedance diagrams, we don't have the flexibility to define a different direction as positive at each node. When drawing the diagrams all the nodes refer to positions that are defined with the same direction as positive. For example, in the above diagram, both x_c and x_r are defined with positive being upward.
- Step 2 Identify one node as ground, or add a ground node.** Since both x_c and x_r will change, neither can be considered a ground. We must add a ground node. The system description thus far is the following:



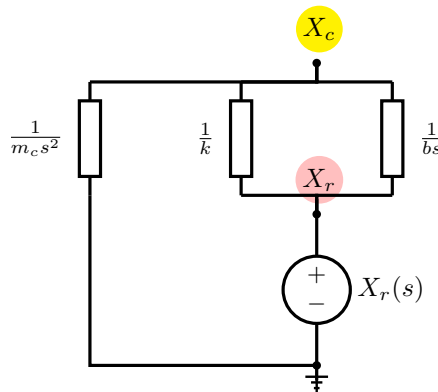
- Step 3 Connect components between nodes.** There is a spring and a damper between x_c and x_r .



There is a mass at node x_c . For a mass, the other end of the impedance *must* be connected to ground



Finally, we apply the boundary condition x_r using a voltage source



Once we have the equivalent impedance network sketched, we can use the impedance rules for parallel, series, voltage divider, and current divider elements from Lecture 5: Impedance and Transfer Functions to find the transfer function from the input signal to the output signal. For practice, verify for yourself that the transfer function from road position X_r to car position X_c is given by

$$\frac{X_c(s)}{X_r(s)} = \frac{bs + k}{ms^2 + bs + k}$$

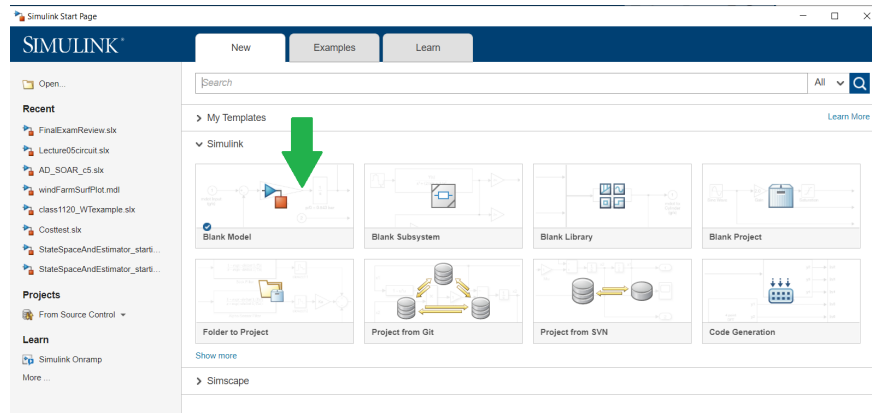
3 Modeling Systems in Simulink

Once we have a transfer function, Simulink makes it very easy to predict the behavior of the output signal for a wide variety of input signals. Let's take the automobile suspension system as an example, building first a model of the plant

system (open loop) and then looking at what we could do with a control system. *Note: the images in this section are produced using Matlab R2022a. If you are using a different version of Matlab, you may see slightly different images.*¹

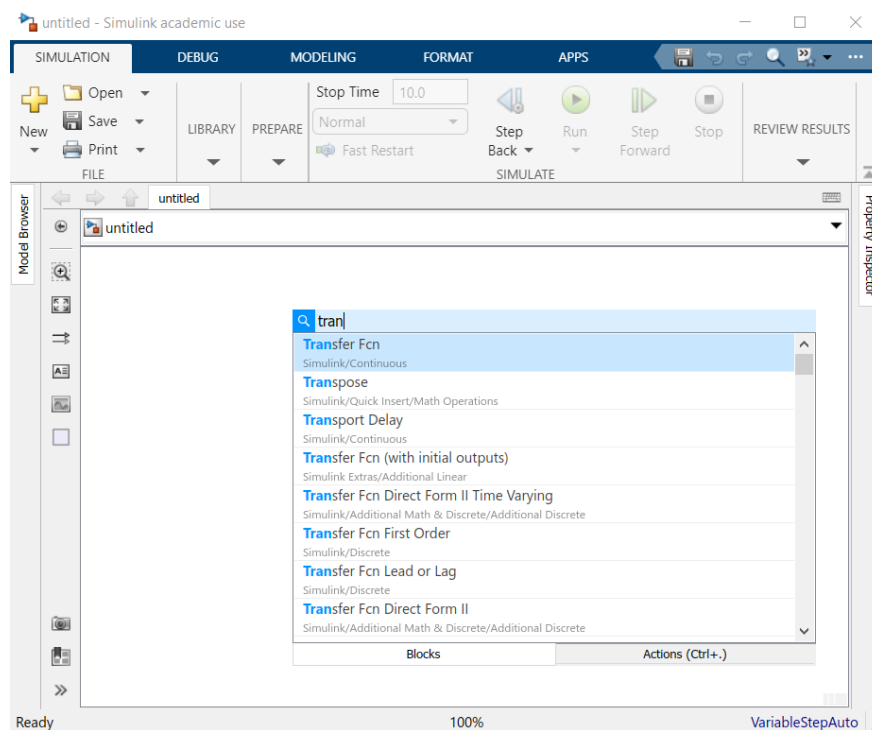
From within Matlab, either type `simulink` at the command prompt or select the Simulink button from the top menu bar. You'll see a window that looks like this:

Initial Simulink Menu



As indicated by the green arrow, select “Blank Model,” then click anywhere in the blank menu and start typing the word “transfer”. After a few letters you’ll be able to select the “Transfer Function” option from the menu as shown below:

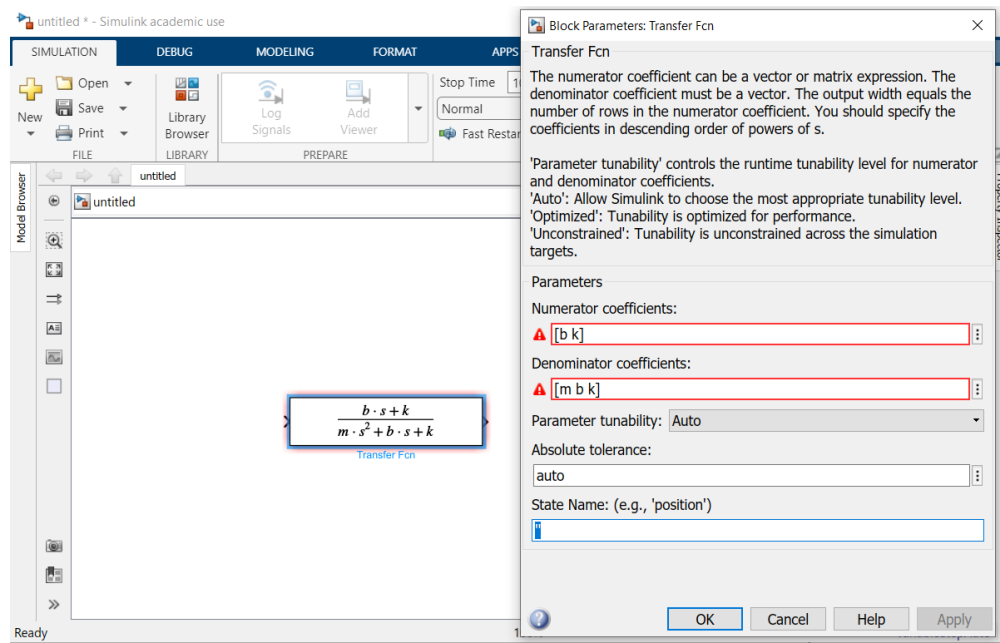
Adding a Transfer Function Block



¹For additional support, either refer to the solar panel demo video linked in Lecture 1: Course Introduction or see one of the many tutorials available at <https://www.mathworks.com/support/learn-with-matlab-tutorials.html>.

Once you have your transfer function block, double click it and you'll have the opportunity to enter the coefficients of s in the numerator and the denominator of the transfer function. To make it easier to update later, go ahead and use the letters m , b , and k to represent the mass m , damping coefficient b , and spring constant k from the equation at the end of Section 2. Matlab will tell you there's a problem because you haven't yet defined these values, but that's OK.

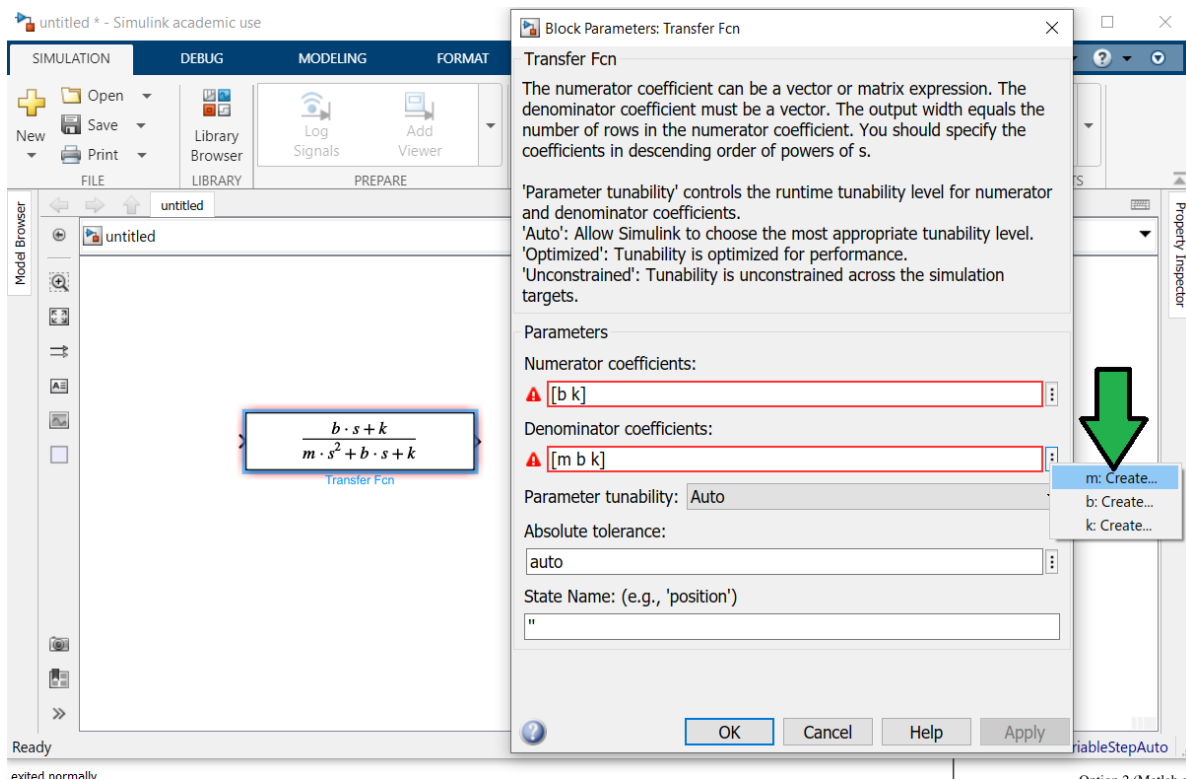
Defining the Transfer Function



You can define your variables in several different ways. One way is to click the three vertical dots at the right side of the "Numerator coefficients:" or "Denominator coefficients:" box (see green arrow in the next image), and another is to define them in the command line. Note also that the size of the Transfer Fcn block has been increased so that we can see the transfer function.

Defining Your Variables

Option 1:



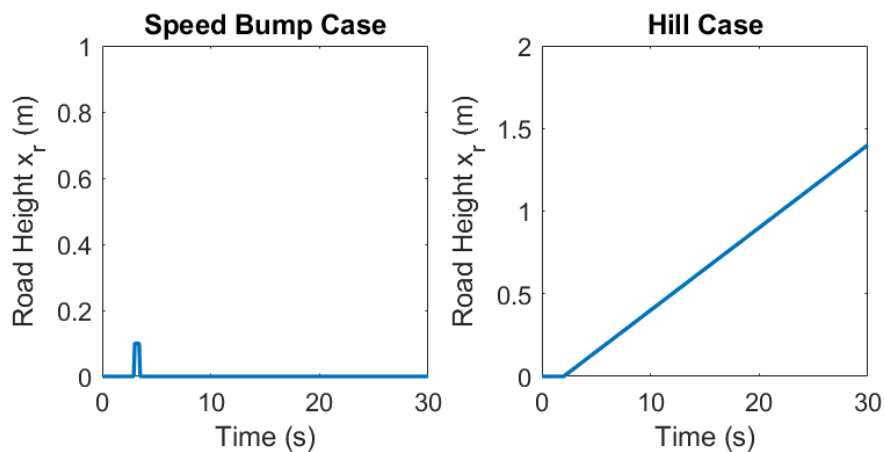
Option 2 (Matlab command line):

```
>> m = 1000; b = 600; k = 800;
```

Input Signals for the Automobile Example

Let's consider what kinds of input signals - road surface heights x_r - a car might encounter. Two possibilities might be a speed bump and a hill. Let's define two possible input signals accordingly.

Road Height Cases: Speed Bump and Hill



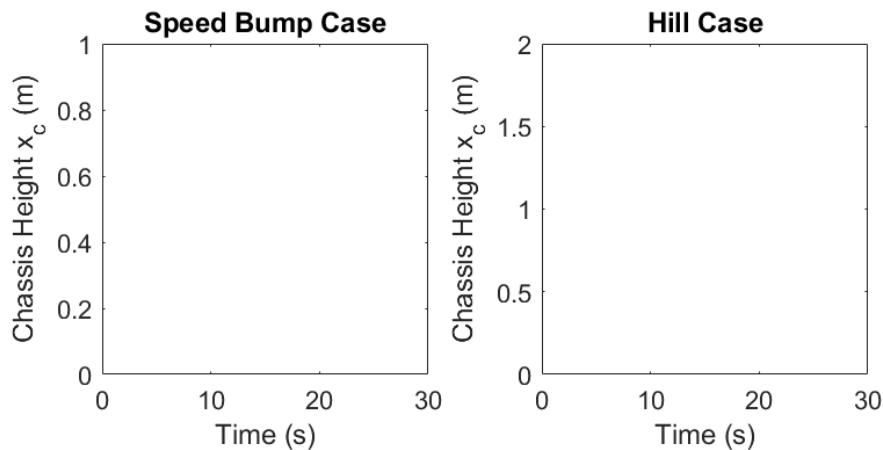
These plots were created using the following Matlab code in the command line:

```
clear;clc
t = [0:0.1:30];
xsb = zeros(size(t)); % Set up vector for speed bump
xsb(31:35) = 0.1; % Define 10 cm tall speed bump that wheel hits for 0.5 s
xhi=zeros(size(t)); % Set up vector for hill
xhi(21:end)=0.05*(t(21:end)-2); % Define 5% grade hill that car hits starting at
2 s
subplot(1,2,1); plot(t,xsb); title('Speed Bump Case'); xlabel('Time (s)'); ylabel('Road
Height x_r (m)'); ylim([0 1])
subplot(1,2,2); plot(t,xhi); title('Hill Case'); xlabel('Time (s)'); ylabel('Road
Height x_r (m)'); ylim([0 2])
```

Before we see what Matlab gives us, let's take a few minutes to predict what will happen to the height of the car chassis x_c in each case. You don't have to get the exact details right, but approximately what do you expect the output signal x_c to look like in each case?

Car Chassis Height Predictions: Speed Bump and Hill

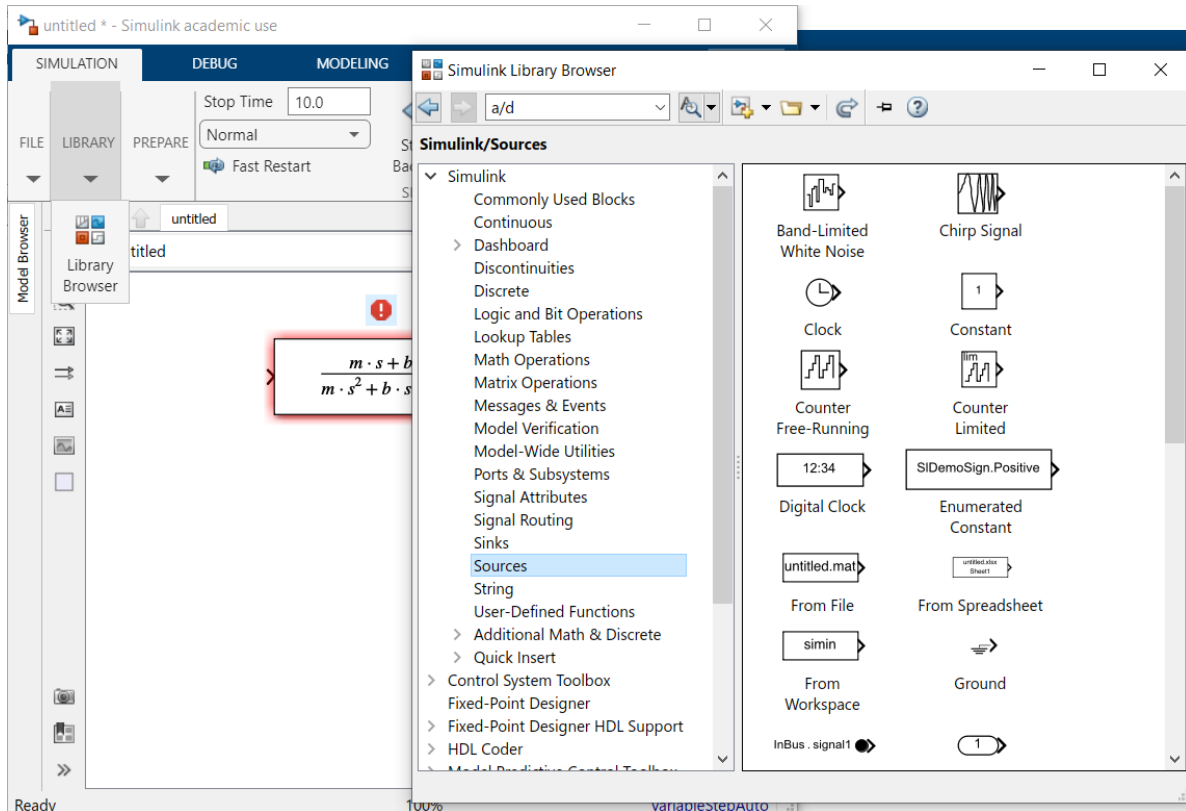
In the two blank plots below, sketch what you predict will happen to the height of the car chassis for each of the two inputs



Congratulations! This prediction of behavior is the essence of modeling, which you've now done using your intuition and experience of how a car works. Now, let's add the input signals and a way to view them in Simulink so we can get a more accurate prediction.

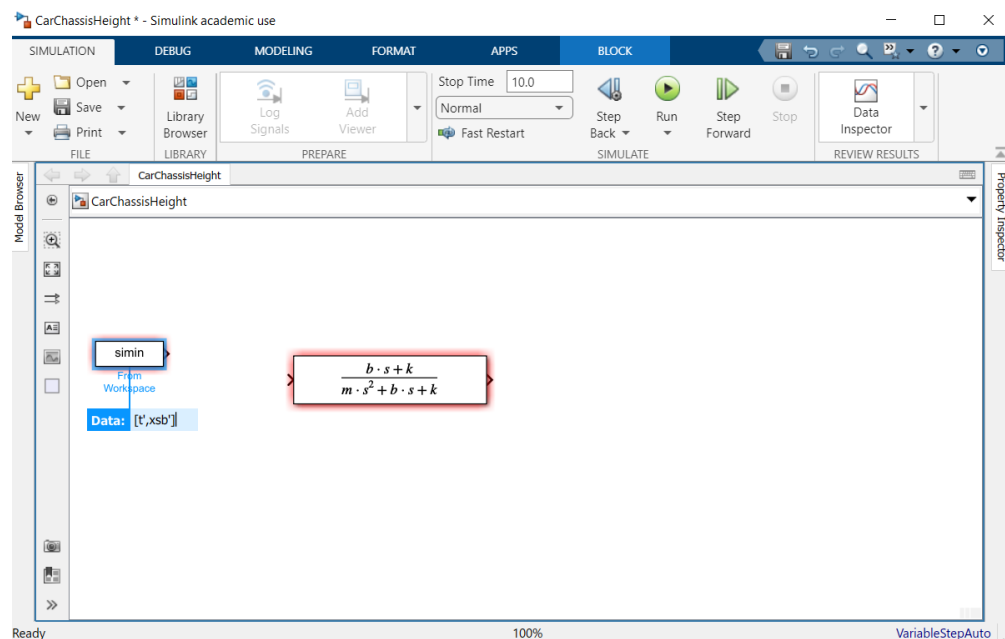
Simulink has a lot of options for adding input signals by selecting "Library" → "Library Browser" → "Sources". For this example, I could use a "Ramp" source for the hill and a sum of two steps - one up starting at 3 s and one down starting at 3.5 s - for the speed bump case.

Adding an Input Signal (Source)



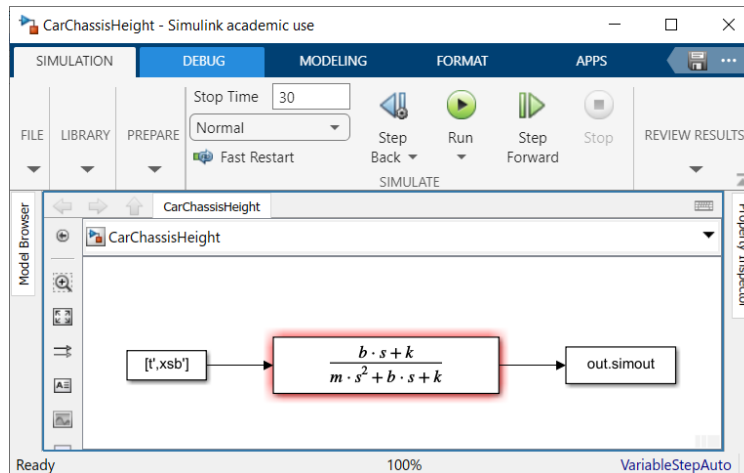
However, since we already generated these signals in the command line, let's go ahead and use a "From Workspace" block, setting it up to run with my speed bump case `xs_b` first. Note: there are lots of ways for these signals to be defined depending on which options are chosen. In the image, the time vector and speed bump height vector appear separately as two columns in a matrix.

Defining the Input Signal



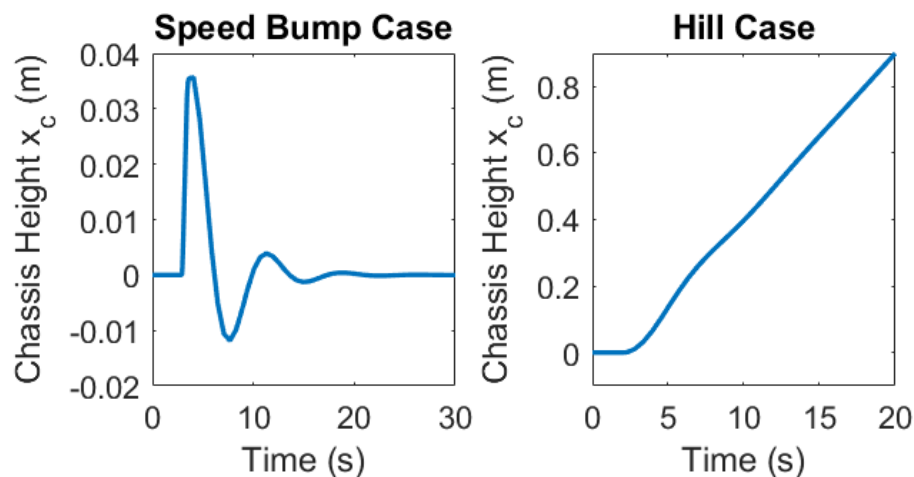
The easiest way to view the output signal (height of the chassis) is to add a scope, but let's make nicer images for a more professional-appearing lecture article using a "To Workspace" block to collect the simulation data. The final Simulink model of the plant system (the transfer function) with its input and output signals looks like this, where you'll note the change in the simulation's "Stop Time" to 30 seconds to correspond with the defined time vector:

Final Open Loop Simulink Model



For the system parameters (m , b , and k) and input signals (x_{sb} , x_{hi}) defined in this lecture, the resulting output signals (chassis heights) are as follows:

Car Chassis Height Results: Speed Bump and Hill



Note the differences in y axis limits. How similar in concept are these output signals to your predictions? Can you explain any differences?

What about the domain? You'll see that we're plotting the input and output signals vs. time, i.e., $x(t)$, but the transfer function is in the Laplace domain (function of s). Matlab takes care of the conversion for us in a similar way that we would solve a differential equation using Laplace transforms and then inverse Laplace transforms.

In the upcoming lectures, we'll first review some differential equation content so that you are better able to assess whether Matlab is giving you reasonable answers (vs. having implemented something incorrectly), then expand on representation of open- and closed-loop systems via block diagrams, and then get into control design. Unlike the example in this lecture, we'll usually be looking at plant system input signals that we can control; the road height is something we usually just have to deal with, and in control systems we will consider it a *disturbance input*, which we'll get to in Lecture 15: Disturbances and Steady State Error.

4 Lecture Highlights

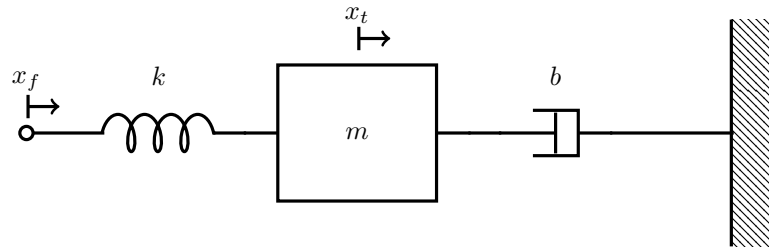
The primary takeaways from this article include

1. Just like electrical systems, mechanical systems can be modeled using impedances, where the impedance is the Laplace transform of the across variable (position) divided by the Laplace transform of the through variable (force).
2. The process for creating an equivalent impedance network for a translational mechanical system is
 - (a) Identify, draw, and label nodes (across variables, i.e., positions)
 - (b) Identify one node as ground (if a fixed node exists) or add a ground node
 - (c) Connect nodes using the component impedances, then connect source(s). between ground and relevant node(s).
3. Matlab's Simulink tool gives us a graphical way to model (predict the behavior of) any system for which we can derive a transfer function.

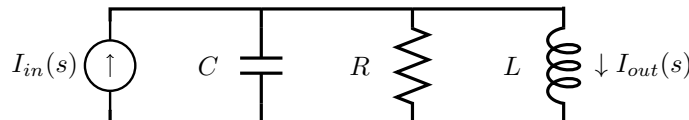
5 Quiz Yourself

5.1 Questions

1. The following is a model for the lateral motion of a building subject to an earthquake. x_f is the position of the foundation of the building relative to a fixed point. During an earthquake, x_f becomes a function of time, and will be considered the input. The horizontal position of the top of the building is x_t . Find the transfer function from x_f to x_t .



2. Consider again the electrical circuit system from the Quiz Yourself problem in Lecture 5: Impedance and Transfer Functions:

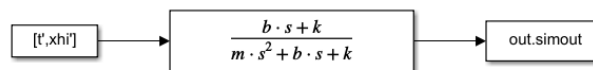


The transfer function from the current source $I_{in}(s)$ to the output current $I_{out}(s)$ is

$$\frac{I_{out}(s)}{I_{in}(s)} = \frac{\frac{R}{sCR+1}}{sL + \frac{R}{sCR+1}} = \frac{R}{s^2 LCR + sL + R}.$$

For parameter values $C = 0.001$ F, $R = 50$ Ω , and $L = 0.1$ H, model this transfer function in Simulink using a sinusoidal “Source” block with $i_{in}(t)$ amplitude 10 A and frequency 60 Hz. Use your knowledge of electrical systems to (briefly) predict the basic parameters (shape, amplitude, frequency) of the output signal $i_{out}(t)$, then run the simulation for 0.5 s in Simulink and comment on whether what you see matches your prediction. Your prediction does not have to be detailed.

3. Consider again the open loop mechanical impedance model in Simulink derived in this lecture, i.e.,



In terms of how an open loop system with input and output signals is visualized in Simulink vs. our lecture articles, what is the most significant difference?

5.2 Solutions

1.

$$\begin{bmatrix} X_f(s) \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{k} + \frac{1}{ms^2} & -\frac{1}{ms^2} \\ -\frac{1}{ms^2} & \frac{1}{ms^2} + \frac{1}{bs} \end{bmatrix} \begin{bmatrix} I_1(s) \\ I_2(s) \end{bmatrix}$$

$$X_f(s) = \frac{1}{bs} I_2(s)$$

$$\frac{1}{ms^2} I_1(s) = \left(\frac{1}{ms^2} + \frac{1}{bs} \right) I_2(s)$$

$$X_f(s) = \left[\left(\frac{1}{k} + \frac{1}{ms^2} \right) \left(\frac{1}{ms^2} + \frac{1}{bs} \right) ms^2 - \frac{1}{ms^2} \right] I_2(s)$$

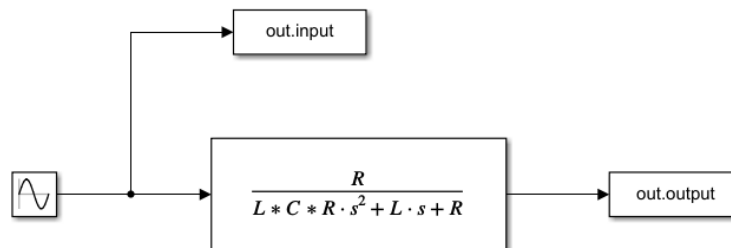
$$= \left(\frac{1}{kms^2} + \frac{1}{ms^2 bs} + \frac{ms^2}{kbs} + \frac{1}{ms^2 bs} \right) ms^2 - \frac{1}{ms^2} \right] I_2(s)$$

$$X_f(s) = \frac{bs^3 + ms^2 + k}{kbs} I_2(s)$$

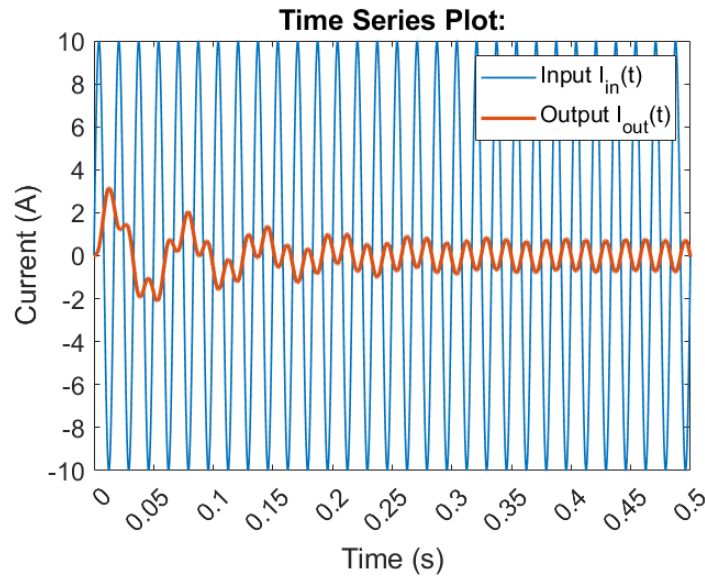
$$X_f(s) = \frac{1}{bs} - \frac{kbs}{ms^2 + bs + k} X_f(s)$$

2. Since the circuit includes both an inductor and capacitor (not just a resistor), we can predict that the output signal might be shifted in phase, as well as amplitude, from the input signal. There might also be some time near the beginning where the output signal is changing in amplitude and mean (this is called the “Transient response,” which we’ll discuss in an upcoming lecture). However, we’d still expect the frequency of the output signal to be 60 Hz.

Your Simulink model should look something like this:



where in this version there are two “To Workspace” blocks so that we can see both the input and output signals together. In terms of settings, since 60 Hz is fast, you’ll want to update the maximum step time in Simulink’s “Configuration Parameters.” This solution uses a max step size of 0.0001 s. (Using such a small value can make simulations take a long time to run, but that’s not a big deal in this case since we’re only running 0.5 s total). Simulink returns the following signals:



Notice that our output signal's frequency is still 60 Hz, the same as the input, as expected. The amplitude of the output is a lot smaller than that of the input, there is some transient response that dies away after about 0.3 s, and the phase does appear to be shifted (the peaks of the output appear to be about 1/2 cycle, or 180 deg., after those of the input). We'll learn an easy way to predict how the amplitude and phase change later this semester, starting with Lecture 24: Sinusoidal Steady State. In the meantime, though, we now know how to find the output for any system for which we can derive a transfer function using the computational tool Simulink.

3. Notice that Simulink uses *blocks* to represent signals - both the "From Workspace" input signal and "To Workspace" output signal look like blocks, not just arrows. This representation is not strictly correct in block diagram nomenclature, which we will detail more fully in Lecture 8: Block Diagrams. A more correct way to draw the block diagram for this open loop system would be:

