Lab 1 — Introduction to numerical simulation using Simulink and MATLAB

Agenda

Time	*Item*
10 min	Review agenda Overview of numerical simulation and simulation diagrams
10 min	Drawing a simulation diagram for an equation of motion
25 min	Using Simulink and MATLAB to solve an equation of motion
60 min	Lab activity
	I'll try to give you a 5-minute warning before the end of the lab period to put all the chairs back where they belong and to pack up your laptops.

Learning objectives:

By the end of the lab period, students should be able to:

- On paper, draw a simulation diagram for a differential equation ("equation of motion").
- State the operations represented by commonly encountered elements in a simulation diagram.
- Implement a simulation diagram in Simulink.
- Create a MATLAB m-file to run a numerical simulation created in Simulink.
- Set numerical simulation parameters in Simulink and MATLAB to improve the credibility of results.
- Plot the results of a numerical simulation using MATLAB.

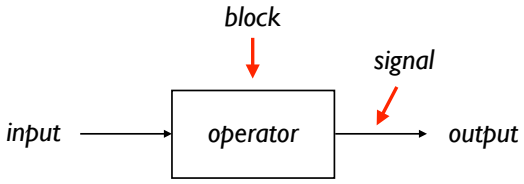
What exactly do you mean by "numerical simulation?"

Numerical simulation means we're solving a differential equation that describes how the behavior of a system changes over time using a numerical integration scheme. For a mechanical system, this differential equation is obtained by applying the rate form of linear or angular momentum conservation. This differential equation is often called an "equation of motion."

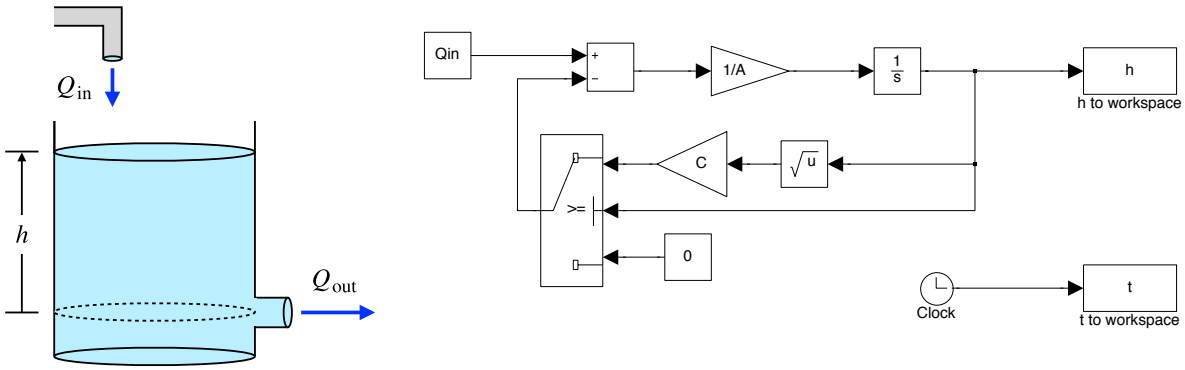
How do you create a numerical simulation?

There are many ways to create a numerical simulation. One approach is to write the equation of motion you want to solve in the form of a *simulation diagram*.

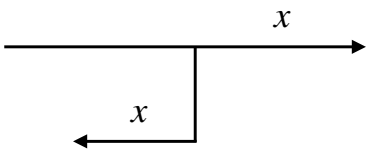
Simulation diagram: A graphical representation of a differential equation using arrows to denote *signals* and *blocks* for mathematical operators.



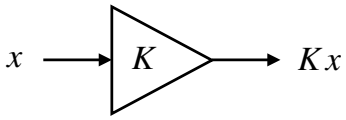
An example of a simulation diagram is shown below. This simulation diagram represents how the height h of water in a tank changes over time as water flows in (Q_{in}) and as the tank is allowed to drain (Q_{out}).



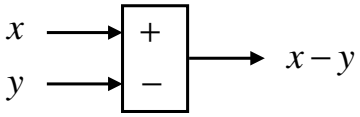
Some of the common elements used in a simulation diagram are illustrated below.



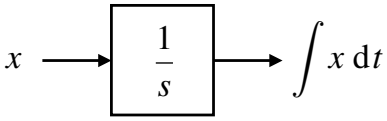
Branch point



Gain



Summing junction



Integrator

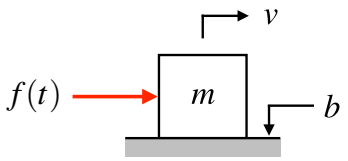
Given an equation of motion, how do you draw its simulation diagram?

Let's work through a simple example to illustrate the process for taking an equation of motion and expressing it in the form of a simulation diagram.

Example:

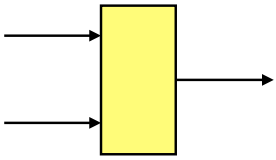
A block with mass m is pushed over a horizontal surface by a force $f(t)$. The friction between the block and surface is viscous (proportional to the speed v) with damping coefficient b.

Equation of motion: $m\dot{v} + bv = f(t)$

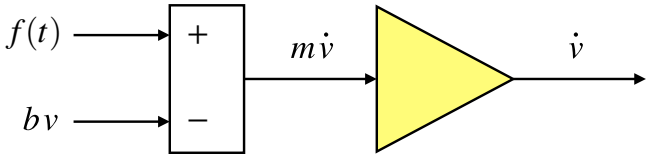


Step 1: Solve for the term with the highest derivative.

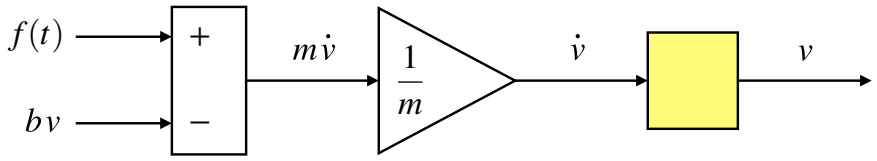
Step 2: If the right-hand side has several terms, make them inputs to a summing junction and make the highest derivative term the output.



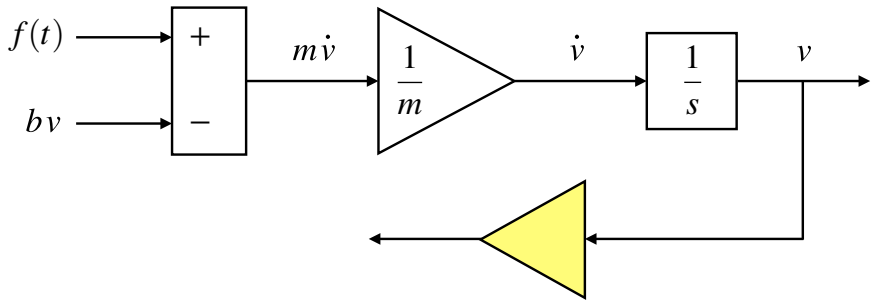
Step 3: Isolate the highest derivative. If needed, use a gain.



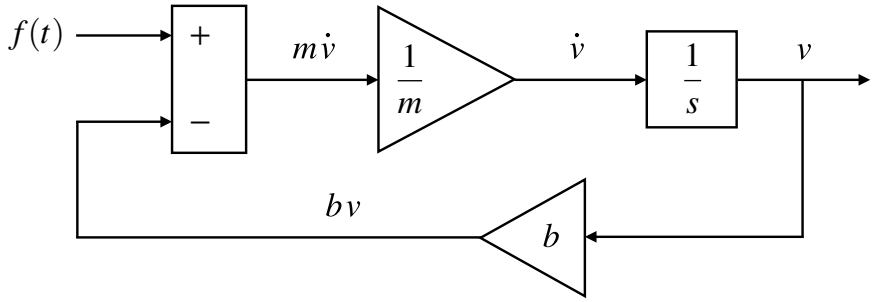
Step 4: Insert integrators to solve for the lower derivatives.



Step 5: Create the signals that feed back to the summing junction, using gains as needed.



Step 6: Complete the simulation diagram by connecting the feedback signals to the summing junction.

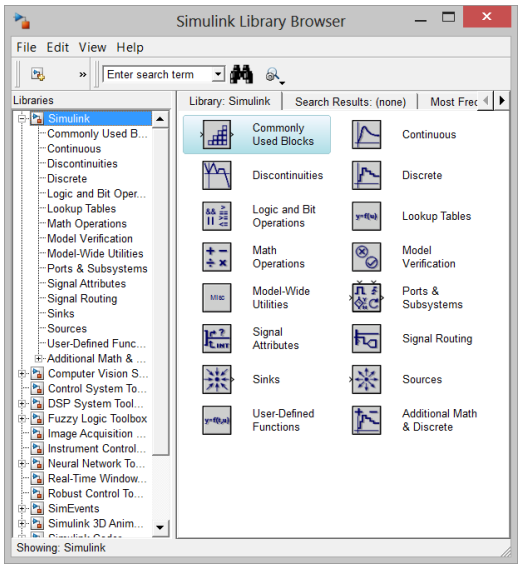


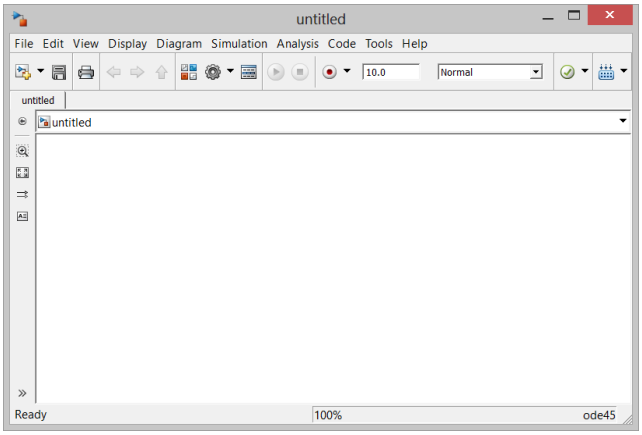
How do you use the simulation diagram to solve the equation of motion?

Actually solving the equation of motion represented by your simulation diagram requires that we use some computational software. In this class and the next class in the Sophomore Curriculum series, ES 205 (Analysis and Design of Engineering Systems), we'll be using Simulink and MATLAB to run our simulations. To do so, we must first draw the simulation diagram in Simulink. Next, we'll need to set simulation parameters for numerical integration, like the simulation time, initial conditions, maximum step size, and error tolerances. Finally, we'll use MATLAB to run the simulation in Simulink and analyze the results.

Drawing the simulation diagram in Simulink

Open MATLAB and type `simulink` in the command window to launch Simulink. Eventually, the Simulink Library Browser, shown below on the left, will open. To draw your simulation diagram, go to **File > New > Model** to create a new Simulink model. A blank window like the one shown below on the right should pop up.





The first step in creating your simulation diagram is to find the various elements in the Library Browser, drag them to the Simulink window, and arrange them to match the layout of your simulation diagram drawn on paper:

- Explore the various categories in the Library Browser to find the blocks for the gains, integrator, and summing junction, and then arrange them appropriately.

- Notice that you'll need to flip the direction of one of the gain blocks. To do so, right-click on the block, go to "Rotate & Flip," and then select "Flip Block."

- Let's take the applied force $f(t)$ to be a constant, so use the "Constant" block under "Sources."

- You can modify the properties of each block (like the signs in the summing junction, the gain block value, etc.) by double-clicking the block and making the changes in the window that pops up. Use variable names like `m`, `b`, and `f` in your blocks to easily relate your simulation diagram in Simulink to the one drawn on paper. Don't worry about the actual values for `m`, `b`, and `f` just yet – we'll define these later in MATLAB.

- To keep track of time during the simulation, we need to place a "Clock" (under "Sources") in the model.

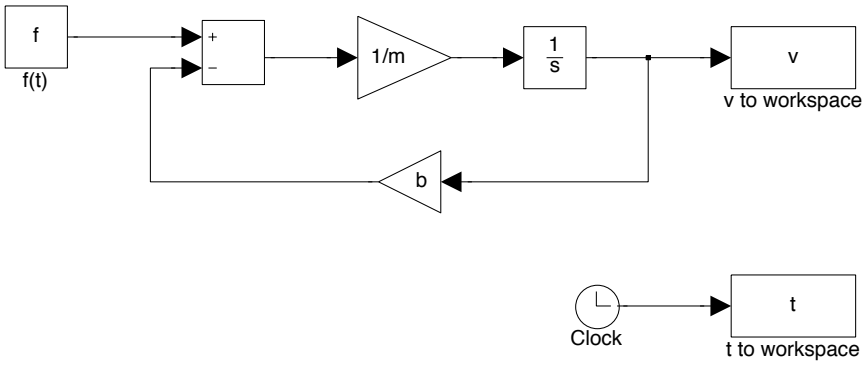
- We'll want to export the simulation results to MATLAB for further processing, so include a "To Workspace" block (under "Sinks") for both the speed and time. Use a more appropriate variable name like `v` for speed and `t` for time instead of the default `simout`.

Once you have the various blocks in place, you need to connect them as shown in your simulation diagram on paper:

- To connect two blocks by a signal, click the output terminal of one block, then hold and drag to draw a signal until you reach the input terminal of another block.

- Notice that you'll need to create a branch point to send the speed signal to the gain block for the damping coefficient b. To do so, first connect the integrator to the "To Workspace" block used for exporting the speed data. Next, click on the input terminal of the gain block for b, and then hold and drag until you reach a point along the existing speed signal.

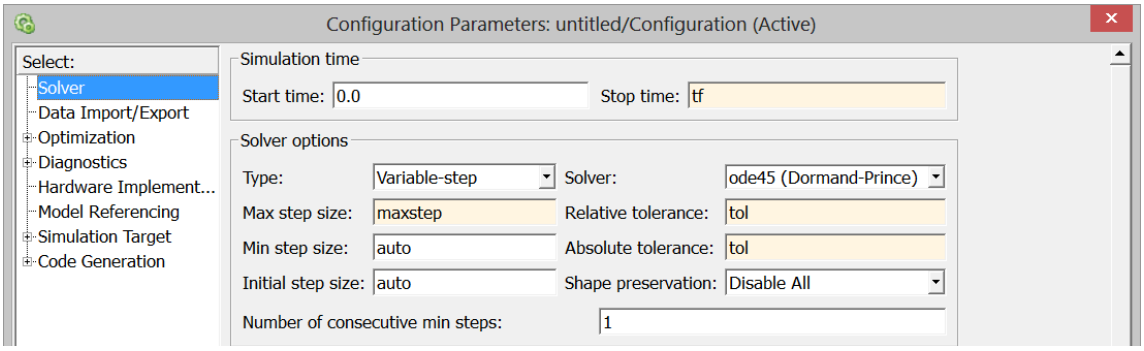
When finished, your simulation diagram should look similar to the diagram below.



Setting simulation parameters in Simulink

Now that the simulation diagram is drawn, we need to set a few parameters before running our simulation. First, we want to make sure that Simulink exports the speed and time data as arrays (vectors) to MATLAB. To set this, double-click on each "To Workspace" block and change the "Save format" option to "Array," as shown below.

Next, let's change a few of the numerical simulation parameters by going to **Simulation > Model Configuration Parameters…** In the window that appears, change the default "Stop time" to the variable `tf` so we can define our own stop time later. Also, replace `auto` in "Max step size" to `maxstep`, and set "Relative tolerance" and "Absolute tolerance" to `tol`. We'll set the maximum step size and tolerance to be small numbers to get a smoother and more accurate solution when we run the simulation. Your Configuration Parameters window should like the window below when you've made the changes.



Lastly, we need to specify the initial condition for our equation of motion, which in this case is the block's initial speed. We can do this by double-clicking on the "Integrator" block and changing the default "Initial condition" to the variable name `v0` (we'll define this value later in MATLAB), as shown below. If you have more than one "Integrator" block, you need to set the initial conditions in all blocks.

If you haven't saved your Simulink model yet, now would be a good time to do so.

Running the simulation with MATLAB

Lastly, let's create a MATLAB m-file to run the simulation instead of using the controls available in Simulink itself. The big advantage to using an m-file to run the simulation is that it's very easy to change system and simulation parameters (like the mass `m` and the stop time `tf`) if we need to without having to modify the simulation diagram itself.

Start a new m-file in the same folder where your Simulink model is saved. It's good practice to start with a clean slate by clearing all variables, closing all open figure windows, and clearing the command window, which you can do with the following code:

```
clear all
close all
clc
```

Let's next assign values for the system parameters. Remember that we used the variable names `m`, `b`, and `f` in the Simulink model, so we need to specify what their values are before running the simulation. Let's set the mass $m = 1$ kg, the damping coefficient $b = 1$ N-s/m, and the constant applied force $f(t) = 10$ N:

```
m = 1;          %  kg
b = 1;          %  N-s/m
f = 10;         %  N
```

We also need to set the various simulation parameters. Let's take the stop time (`tf`) to be 6 s, and we'll set the maximum step size (`maxstep`) and error tolerance (`tol`) to be 0.01 s and 1×10^{-6}, respectively. Specify the initial condition (`v0`) as 1 m/s.

```
tf = 6;         %  s
maxstep = 0.01; %  s
tol = 1e-6;
v0 = 1;         %  m/s
```

We're now ready to tell MATLAB to run the simulation. To do this, use the `sim` command:

```
sim('YourModelName')
```

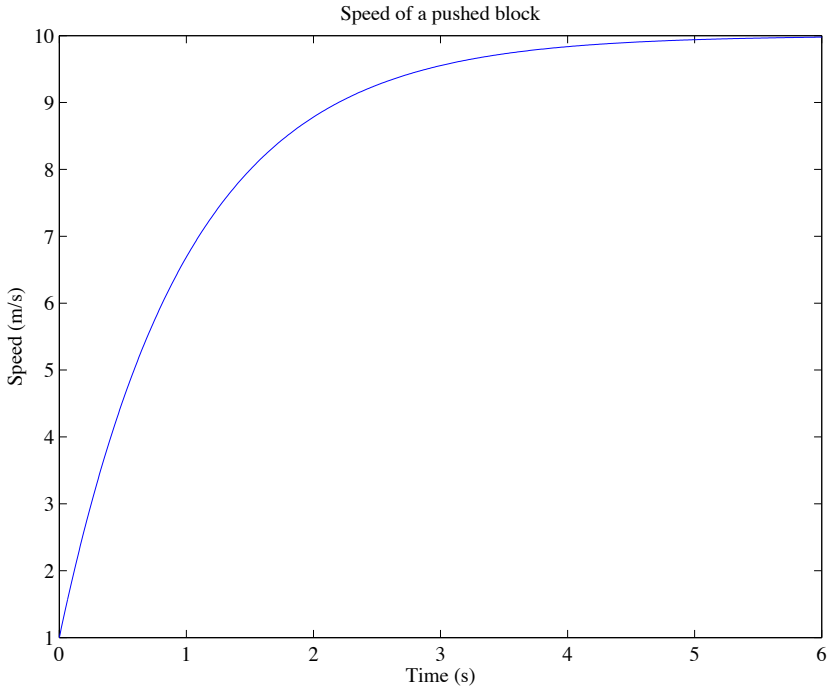
Save your m-file if you haven't done so yet. A word of caution – don't give your m-file the same filename as your Simulink model, otherwise MATLAB will get confused!

Now, run your m-file. When MATLAB is finished, you should see your simulation results in the workspace. Since we named the time and speed vectors `t` and `v`, respectively, in Simulink, these should appear in your workspace.

Lastly, let's plot the simulation results to get a better understanding of the block's response over time. Append the following lines of code to your m-file:

```
figure(1)
plot(t, v)
xlabel('Time (s)')
ylabel('Speed (m/s)')
title('Speed of a pushed block')
```

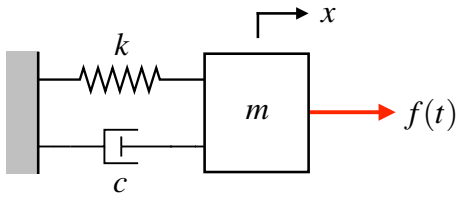
When you run your m-file again, you should get a plot that looks like the following:



Lab activity

Together with your lab team, build a simulation that solves the equation of motion for the mass-spring-damper system shown below.

Equation of motion: $m\ddot{x} + c\dot{x} + kx = f(t)$



- Take the mass $m = 2$ kg, damping coefficient $c = 1$ N-s/m, and spring stiffness $k = 20$ N/m.
- Set the applied force $f(t)$ to be a constant value of 6 N.
- Run your simulation for 10 s with the initial conditions $x(0) = 0.2$ m and $\dot{x}(0) = -1$ m/s.
- Produce two figures, one of the displacement x over time t, and the other of the velocity $\dot{x}$ over time.
- Be sure to label your axes and include units. Include your initials and the date in the title of each figure.

Deliverables:

1. Printouts of your plots illustrating how the system's displacement and velocity change over time.
2. A printout of your Simulink model used for simulation.
3. A printout of your MATLAB m-file used to run your simulation and plot the results.