Lab 4 Prelab — Dot operators and importing arrays into Simulink

Introduction

In Lab 4, you'll be creating a numerical simulation of an overhead crane moving a suspended payload to explore the system's behavior for two different motions of the crane trolley. The trolley's motion is prescribed by specifying its acceleration over time in an array that is defined in your MATLAB m-file and then accessed by Simulink during the simulation. To aid in setting up your m-file and Simulink model during the lab, the following prelab exercises are designed to help you become familiar with creating and manipulating arrays in MATLAB and be able to have Simulink take in an array from the MATLAB workspace during simulation.

Arrays and dot operators

Creating arrays in MATLAB

Let's say we want to specify the time function $x(t) = \sin 2t$ in MATLAB, where time t ranges from 0 to 4 s. The first thing we need to do is define a time vector, say, `T`, and we'll set `T` to range from 0 to 4 s in increments of 0.01 s. To do this, type the following line of code in the command window:

```
T = [0 : 0.01 : 4]';
```

The array `T` should appear in the workspace after pressing the ↵ **Enter** key. Notice that `T` has 401 rows and 1 column, i.e., `T` is a column vector. If we instead typed `T = [0 : 0.01 : 4]` in the command window, MATLAB would have returned a row vector (i.e., `T` would have 1 row and 401 columns), but the operator `'` transposed the array into a column vector.

Now we can create a column vector `x` representing $x(t) = \sin 2t$ by typing

```
x = sin(2*T);
```

Congratulations! You've just created arrays in MATLAB without using `for` loops! Plot `x` against `T` to graphically verify that you've defined a function $x(t)$ that varies sinusoidally over time.

Dot operators

Define the following column vectors `a` and `b` in the command window:

```
a = [1, 2, 3]';
b = [4, 5, 6]';
```

Let's perform some elementary operations on these arrays. Add and subtract the column vectors:

```
op1 = a + b
op2 = a - b
```

Are the results for `op1` and `op2` what you expected?

Now, let's multiply `a` and `b`. When we say "multiply," what we mean in this case is that we want the first element of `a` (i.e., 1) to be multiplied with the first element of `b` (i.e., 4), and so on. This is referred to as *element-by-element multiplication*.

Since a `*` is a multiplication operator, type

```
op3 = a*b
```

What happened? You should have received the following error message:

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

So what went wrong? Notice that both `a` and `b` have 3 rows and 1 column. MATLAB interprets the assignment `op3 = a*b` as the matrix operation

$$\texttt{op3} = \begin{bmatrix}1\\2\\3\end{bmatrix}\begin{bmatrix}4\\5\\6\end{bmatrix}$$

which makes no sense because the inner matrix dimensions don't match (and hence the error message). To execute element-by-element multiplication of `a` and `b`, we need to use *dot operator* syntax:

```
op3 = a.*b
```

yields the desired result

```
op3 =

     4
    10
    18
```

The dot operator syntax also holds for element-by-element division (`./`) and exponentiation (`.^`). Try the following operations:

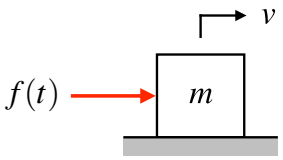
```
op4 = a./b
op5 = a.^2
```

Are the results for `op4` and `op5` what you expected for the element-by-element operations?

Importing an array into Simulink from the workspace

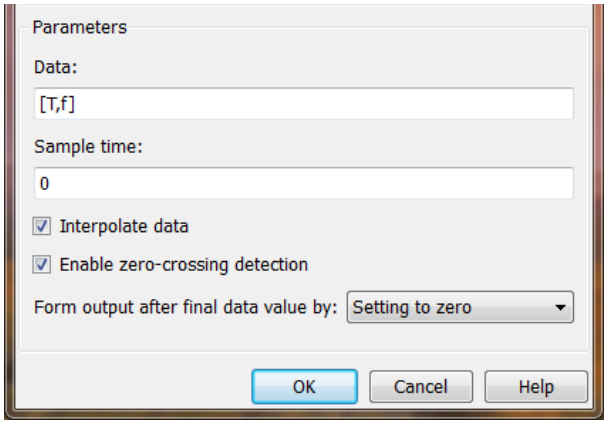
Consider a block with mass m that is pushed over a smooth horizontal surface by a force $f(t)$, as illustrated below. Create a simulation diagram in Simulink and an accompanying MATLAB m-file that solves the system's equation of motion for the block's speed v over time t. Take the mass $m = 2$ kg, and have your simulation run for 4 s with the block initially at rest (i.e., $v(0) = 0$). Remember to set the maximum step size and error tolerance.

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



This system is similar to one we worked through during Lab 1 when you were first introduced to numerical simulations. For that system, we set the applied force $f(t)$ to be a constant. This time, let's keep things general and allow $f(t)$ to be an arbitrary time function. Specifically, you'll define an array for $f(t)$ called `f` in your MATLAB m-file using a specified time vector `T` (and these arrays then show up in the workspace) and then have your Simulink model access these arrays during simulation.

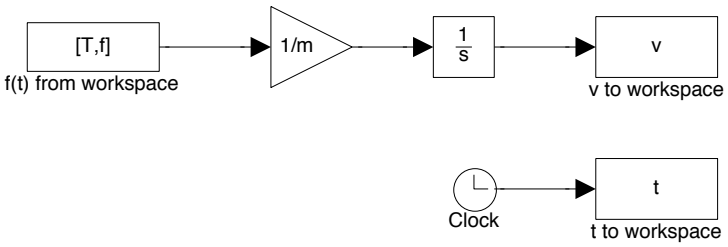
To have Simulink take in the array `f` for the applied force $f(t)$, you'll need to represent the force input with a "From Workspace" block (under "Sources"). Appropriately connect the "From Workspace" block to your simulation diagram, and then double-click on the block to open its properties window. Set up the properties window as shown in the figure below.



The array `[T,f]` under "Data" instructs Simulink to read from the force array `f`, which has force values defined at the time values in the time array `T`. Since Simulink doesn't use a fixed time step for integration by default, the time vector `t` it uses is different than `T`, and so we need

to have the "Interpolate data" option checked to make sure Simulink interpolates the `f` and `T` arrays to calculate force values that correspond to the times in `t`. Lastly, when the "Form output after final data value by" option is set to "Setting to zero," the applied force $f(t)$ becomes zero at the end of `T`, even if the simulation isn't finished running. For example, if $f(t)$ acts for 4 s but the simulation runs for 5 s, then $f(t) = 0$ between 4 s and 5 s if we've selected the "Setting to zero" option. To avoid confusion, it's best to have the simulation stop at the end of `T`, if possible.

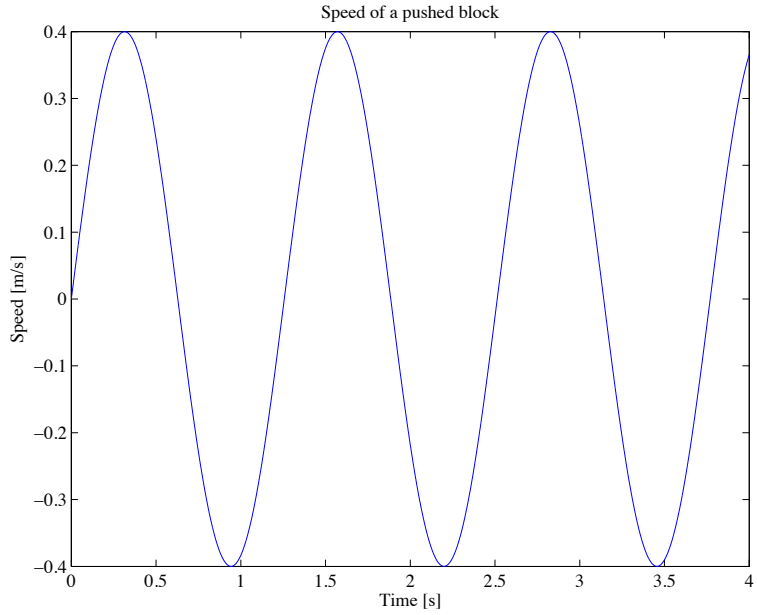
When finished, your simulation diagram should look similar to the diagram below.



Now, in your MATLAB m-file, define the force array `f` and time vector `T` such that $f(t) = 4\cos 5t$ N from 0 to the simulation stop time `tf` in increments of 0.01 s:

```
T = [0 : 0.01 : tf]';    %  s
f = 4*cos(5*T);          %  N
```

Run your simulation, plot the block's speed over time, and confirm that you obtain a plot similar to the following:



Prelab activity

Modify your MATLAB m-file so that the applied force $f(t) = 10\,e^{-0.6t}\sin 8t$ N. Run your simulation and generate a figure of the block's speed v over time t. Be sure to label your axes and include units. Also, include your initials and the date in the title of the figure, and remove the gray border around the figure. Copy your plot into a Word file by selecting **Edit → Copy Figure** in the figure window and pasting the plot into your Word file.

Due at the beginning of Lab 4:

Attach the following items to the Lab 4 prelab rubric posted on the course website:

1. A printout of your plot illustrating how the block's speed changes over time. Don't forget to format your figure as instructed above.

2. A snapshot of your Simulink model used for simulation.

3. A printout of your MATLAB m-file used to run your simulation and plot the results.