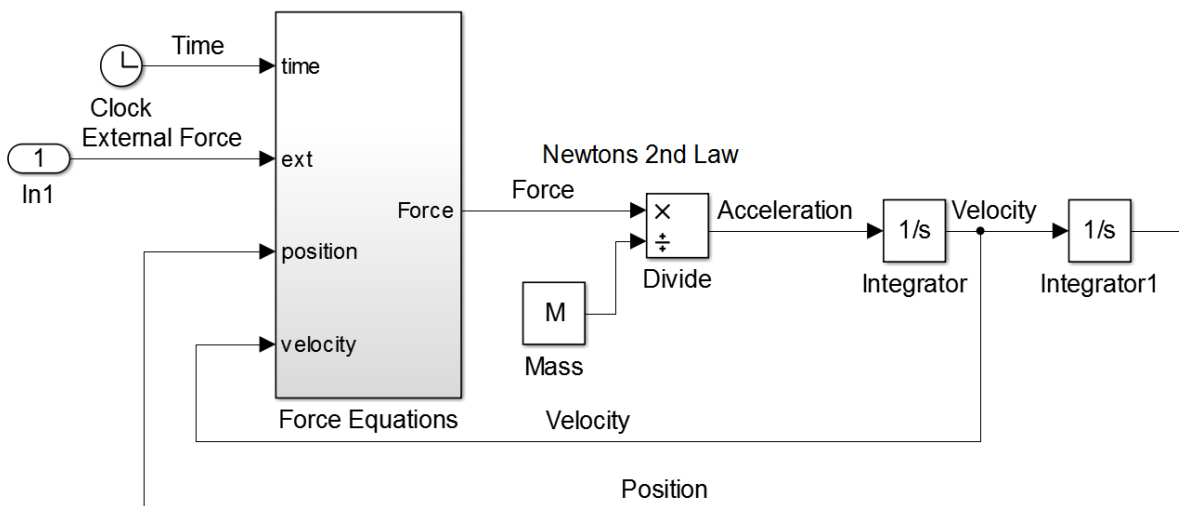


An Introduction to Using Simulink



Exercises

Eric Peasley, Department of Engineering Science, University of Oxford

Adapted and updated by Dr I. F. Mear using MATLAB 2017b and MATLAB 2018b version 6, 2018

Contents

Exercise 1: Simulating flight of cannon ball

- 1. a) Creating a Simple Simulink Cannon Model page 2
- b) Introduction to Subsystems by Modelling Air Resistance page 5
- c) Adjusting Model Parameters page 7
- d) Further Plotting page 9
- e) Using MATLAB Expressions page 10
- f) Combining MATLAB and Simulink page 11

Exercise 2: Ordinary Differential Equations

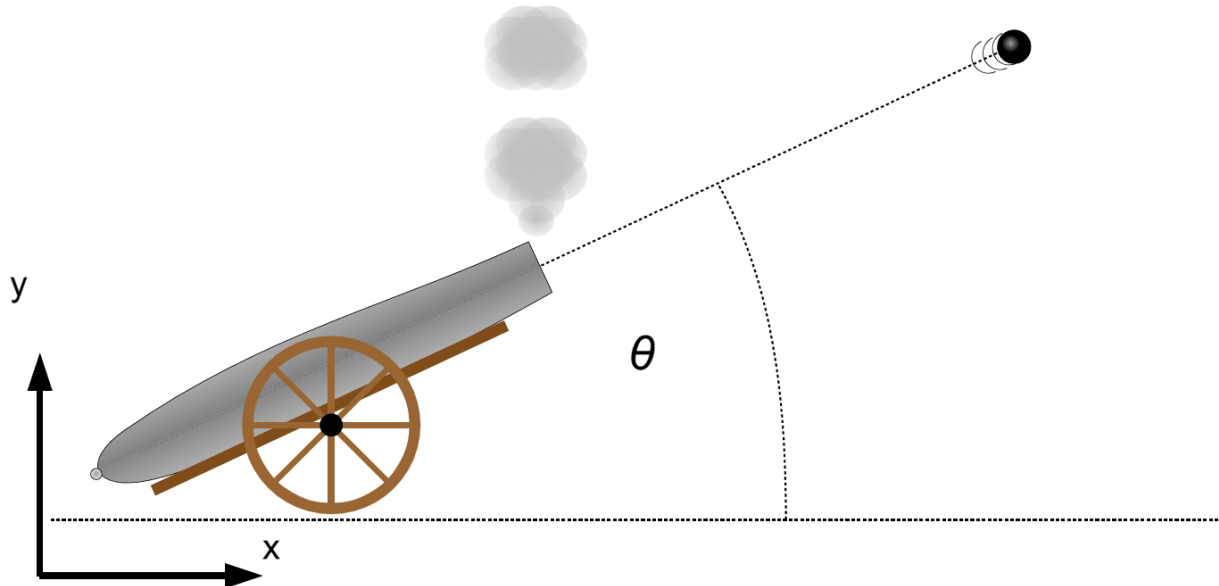
- 2. a) System A page 12
- b) System B page 14

Exercise 3: Bouncing Ball page 16

Where Next? Further Practice page 18

Exercise 1: Cannon Ball

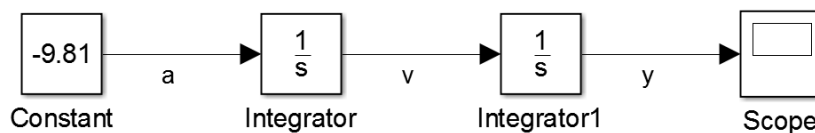
This exercise is designed to introduce you to modelling in Simulink. It will show you how to use Simulink to model and simulate a dynamic system. The problem is to simulate the flight of a cannon ball after it has been shot out of a cannon.



We will start by considering only the vertical motion of the cannon ball under the influence of gravity. What happens when you drop a cannon ball? This can be described by the equation below.

$$\frac{d^2y}{dt^2} = -g \quad (1)$$

To solve this equation analytically you would integrate the right hand side twice. In the Simulink model below, we do the same.



1. a) Creating a Simple Simulink Cannon Model

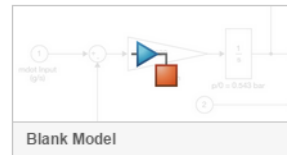
Creating a New Model



Start Simulink by clicking on the **Simulink** icon, under the **HOME** tab, on the MATLAB toolbar. This will open the **Simulink Start Page**.

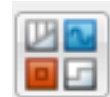
You can also type `simulink` in the MATLAB command line.

On the **Simulink Start Page**, select **Blank Model**:



Adding the Blocks to the Model

Open the **Simulink Library Browser** by clicking on the icon on the toolbar:



In the Simulink Library Browser, click on **Commonly Used Blocks**.

This contains all the blocks we will need for the first version of our model. Have a look at the blocks in this library. Notice that they are in alphabetical order.



Constant

Constant value: |

- **Drag a Constant Block** onto the canvas of your model.

When you place the block down, a menu will pop up asking you to enter the **constant value** of the block. **Type -9.81**. If you miss this pop up you can change the value in the Block Parameter menu as explained on the next page.



Integrator2

Initial condition: |

- **Drag an Integrator Block** onto the canvas of your model.

The pop up may ask you to enter the **initial condition** for the block. **Type 0**. If you miss this pop up you can change the value in the Block Parameter menu later.



Scope1

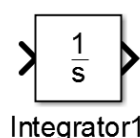
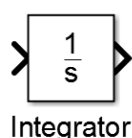
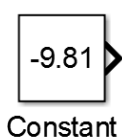
Number of input ports: |

- **Drag a Scope Block** onto the canvas of your model.

The pop up may ask you to enter the **Number of input ports** for the block. **Type 1** or click anywhere else to close.

You can **move** the blocks around the model by dragging them around using the left hand mouse button. You can **duplicate** a block by dragging it using the right mouse button.

- **Arrange the blocks** so that your model looks something like the figure below:



(The duplicate Integrator block is called 'Integrator1' as all block names must be unique).

Block Parameters

Most blocks need to be configured for use in a model. You may have configured some values using the quick configuration pop up menu that appears when you first drag a block on the canvas, but it is easy to miss these.

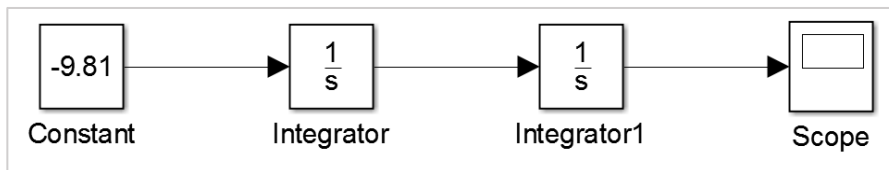
Double Click on any block to **open the Block Parameters**.

The value of the constant in the Constant Block needs to be set to **g**, the acceleration due to gravity. **Double click on the constant block** to display the block parameters. If needed, change the **constant value to -9.81**. Then select **OK**.

We are also going to rename the constant block. **Click on the word 'constant'** under the block. Delete the word constant and **replace with the letter g** to represent gravity.

Wiring the Model

The next task is to wire the blocks together as in the figure below.



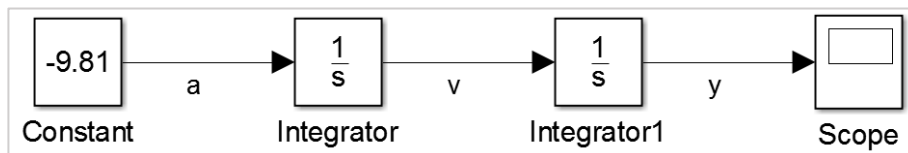
This can be done in two ways:

- Drag with the left mouse button from output to input or input to output.
- Select the source block. Hold down Ctrl and click on destination block.

If the wires are not straight, you can straighten them by using the cursor keys to move selected blocks. To remove a wire, select it, then press the delete key.

Labelling the wires

Double click on the wire connecting the Integrator and the Constant block. Then enter **a** (for acceleration). In a similar manner, label the wire between the two Integrators **v** (for velocity) and the input to the Scope **y**. Now drag the labels around so that the model looks like this.



Saving the Model

The completed model should now look like the image above.

Save the model as cannon1.slx by using **File > Save As...**

Running the Simulation

Click on the **Start Simulation Icon** on the icon bar of the model. This is the triangle pointing to the right, in a green circle.

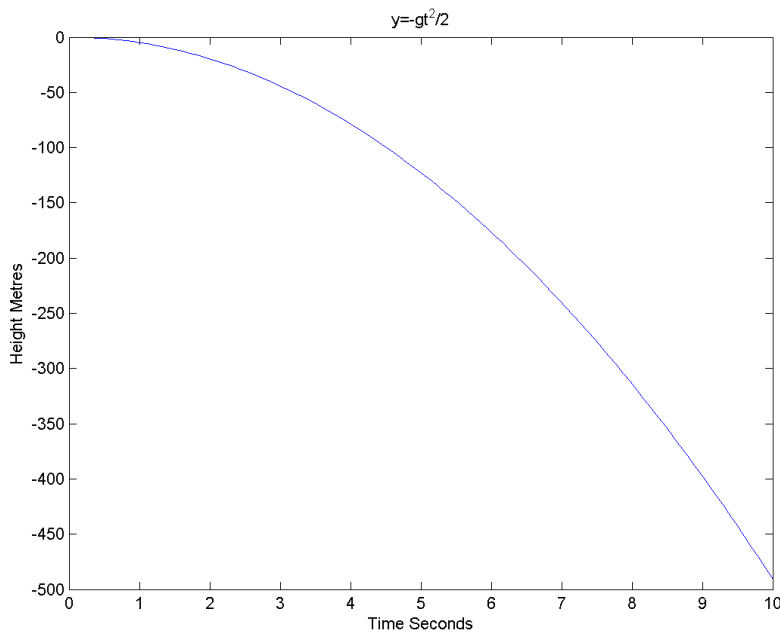



Double click on the **Scope Block** so that you can see the scope display.

If you solve equation (1) analytically with initial position and velocity both zero, then you get the following equation

$$y = \frac{-9.81t^2}{2} \quad (2)$$

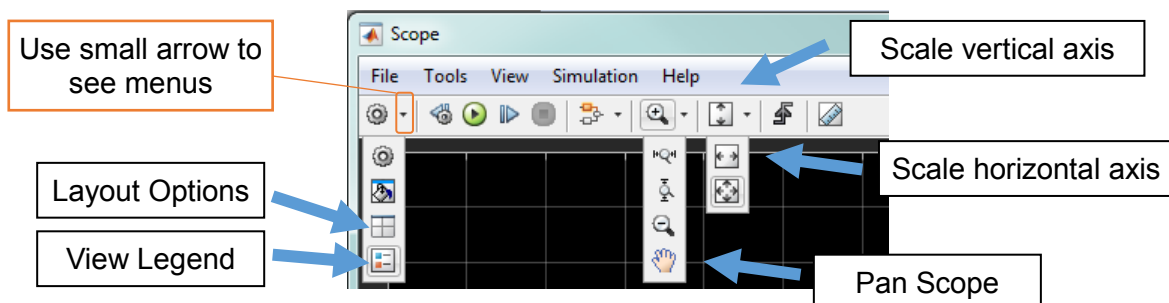
The analytical results are plotted below, this should match the result on your scope.



At $t = 10$, $y = 490.5 \text{ m}$. Use the Zoom icon  on the scope to take a close look at the region near 10 seconds, you will see that they are both in agreement.

Double click on the graph to return to normal magnification.

Scope shortcuts are similar to a MATLAB figure. The most common are highlighted below:



There are also many useful shortcuts for Simulink, such as **spacebar** to scale the model to the size of your screen. For more, see 'Shortcuts' section of the Simulink Quick Reference.

1. b) Introduction to Subsystems by Modelling Air Resistance

We are now going to add a drag factor to the model. We are going to use a quadratic function to represent the **drag**, F_d :

$$F_d = -0.02v|v| \quad (3)$$

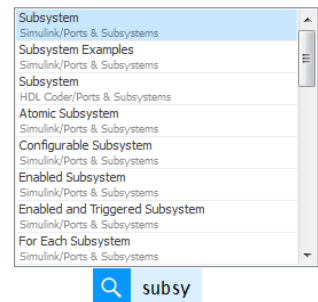
We use the absolute value so that the drag always opposes the velocity, even if the velocity is negative. Using *Total Force* $F = ma$ we can rearrange for acceleration:

$$F_d - mg = -0.02v|v| - mg = ma$$

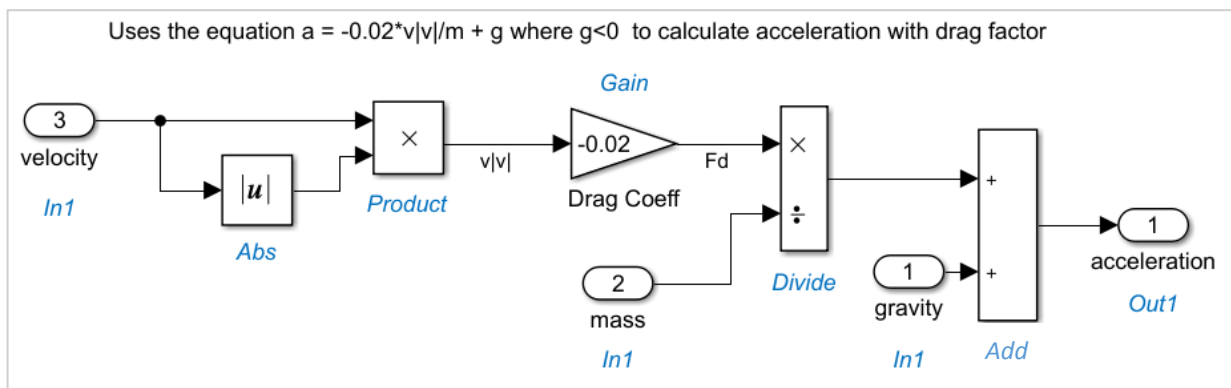
$$a = \frac{1}{m}(-0.02v|v| - mg) \quad (4)$$

The acceleration (4) is calculated from **mass**, **velocity** and **gravity**. We will make a **subsystem** with those three inputs, which outputs acceleration as per equation (4).

- Single left click on the background of your model and type **subsystem**. This uses the **quick search to find a subsystem Block** – click on the top entry to add it to your model. This is faster than via the Library Browser. Double click on the subsystem to open it.



- Delete any default starting blocks present and modify the model so it looks as below. The block names are given in italics.



- You will find the **Sum** and **Gain** blocks in the **Commonly Used Blocks** library.
 - The **Abs**, **Product** and **Divide** blocks are in the **Math Operations** library.
 - The **In1** and **Out1** blocks are in the **Sources** library.
 - To make branches from wires, hold down **Ctrl** while left clicking and dragging.
 - You can copy and paste blocks.
- Double left click on the background and select **Create Annotation**. This will allow you to add text to your model, as shown above.

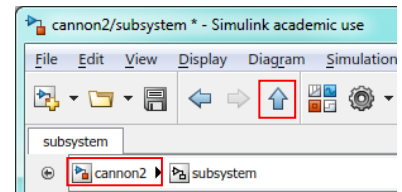
As you work, wire the model as shown above. Drag from a port to signal, or hold down **Ctrl** to **drag a branch** from a signal. Save your model as **cannon2.slx** as you are working.

Once your subsystem model is complete, return to the top page by

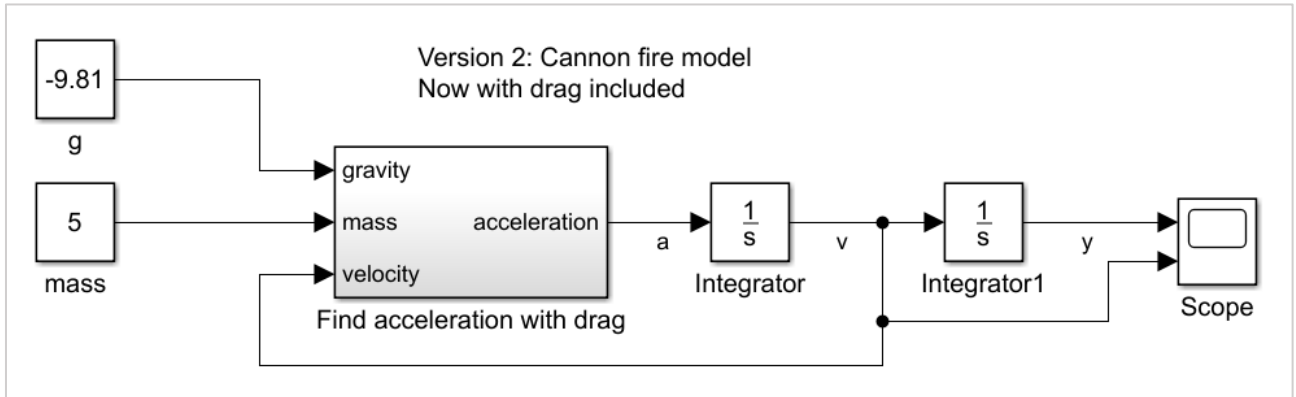
- Selecting the **blue 'Up to Parent' arrow**

OR

- Selecting the home tab, named 'cannon2'



You will now need to connect the subsystem, as shown below:

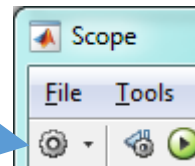


Add Extra Scope Input

It is useful to observe what is happening to the velocity signal. You could add another scope block into the model. However, let us instead add another axis to the scope.

To do this **double click on the scope block.**

Then **click on the Scope Parameters icon.**



Set the **Number of axes** to **2**. Using the **Display** Tab, you can also tick a Check Box to **Show Legend** to see the signal labels.

An addition input will appear on the scope block.

Changing the Stop Time

To get close to the terminal velocity, the model needs to run for longer. The stop time is in the box to the right of the start simulation icon on the banner of the model.



At the moment, it is set to the default value of 10 seconds. **Set the stop time to 20 seconds.**

Run the Simulation

Run the simulation and observe the scope output.

You should notice that the terminal velocity is a bit less than 50 m/s.

1. c) Adjusting Model Parameters

Initial Conditions

So far we have been using the default initial conditions. The position and velocity are both zero at the start of the simulation. We are now going to change the initial conditions so that the cannon ball is fired directly upwards at a speed of 100 m/s.

- **Double click the integrator** with **v** (velocity) on the output.
Change the initial condition to 100.
- **Save the model.**
- **Run the simulation.**



The cannon ball should hit the ground about 13 seconds after being fired. You may need to use the **autoscale shortcut** on the scope to see all the data.

Two Dimensions

Although we know how high the cannon ball has travelled (by the y direction values), we do not know how far it has travelled in the x direction. There are a number of ways of adding in the calculations for the x direction. The most obvious way is to duplicate what you have already done in the y direction. However, it is far easier to use vectors. Simulink is built on MATLAB, hence is excellent at handling vectors and matrices. We can modify the model so that we can use vectors of the form $r = [x \ y]$.

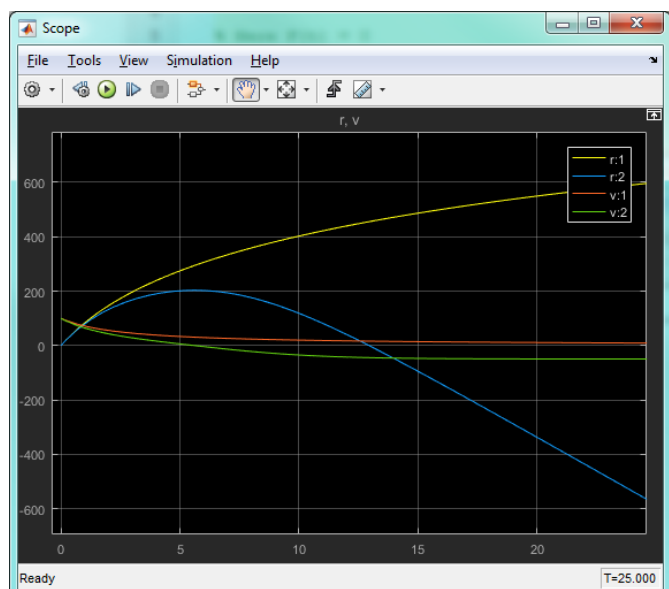
- Change the constant **g** to **[0 -9.81]**. This gives 0 in the x direction, -9.81 in the y direction.
- Change the initial conditions of the integrator with **v** output to **[100 100]**.
- Change the signal label from **y** to **r**.
- Change the initial conditions of the integrator with **r** output to **[0 0]**.

Save as cannon3.slx

Run the simulation

Scope key:

Line	Description	Array
Yellow	distance travelled in x	$r(1)$
Blue	distance travelled in y	$r(2)$
Red	velocity in x direction	$v(1)$
Green	velocity in y direction	$v(2)$



Hidden Errors with Vectors

There is an error with this model. To see what it is, do the following:

Display ► Signals & Ports ► Wide Nonscalar Lines

This makes all signals with vectors wider than normal.

Look at the output of the Abs block (inside the subsystem). The thick line indicates the output is a vector. This is supposed to be $|\mathbf{v}|$, that is $\sqrt{v_x^2 + v_y^2}$. What we are actually getting is $[|v_x| |v_y|]$, where v_x and v_y are the x and y component of \mathbf{v} . This has happened because we assumed the Abs block was working in one way, but it was in fact working in another. We should always inspect the output of blocks. Just because a model runs, doesn't mean it will always be correct! To correct this here, we are going to create a subsystem that will perform the correct calculation.

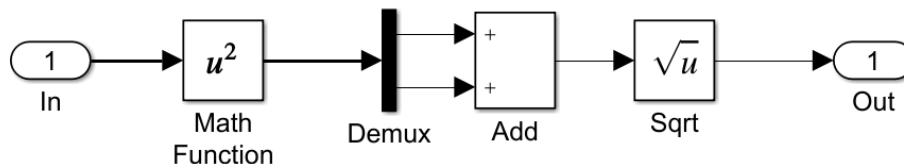
Select the Abs block and Right Click.

From the menu select **Create Subsystem from Selection**.

The subsystem is a larger than the Abs block, so you may have to move things around a bit for it to fit in. Click on the name under the subsystem and **rename it 'Abs of Vector'**.

Double click on the subsystem to open it.

Remove the Abs block and replace it with the following:

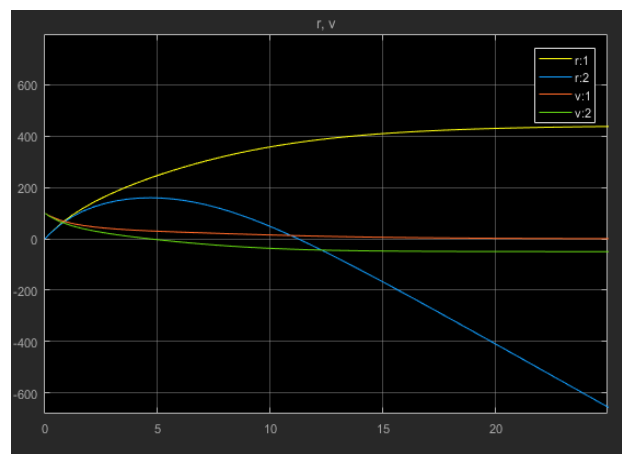


You will find the **Sqrt**, **Add** and **Math Function** Blocks in the **Math Operations** Library.

The actual function implement by the Math Function Block is selectable as a Block Parameter (double click on the block and select the desired function). You will also need a **Demux** from the **Signal Routing** Library. This is used to split the vector into its separate components (notice the input line is bold and the output lines are thin, this is to distinguish between arrays and single values).

Resave as cannon3.slx

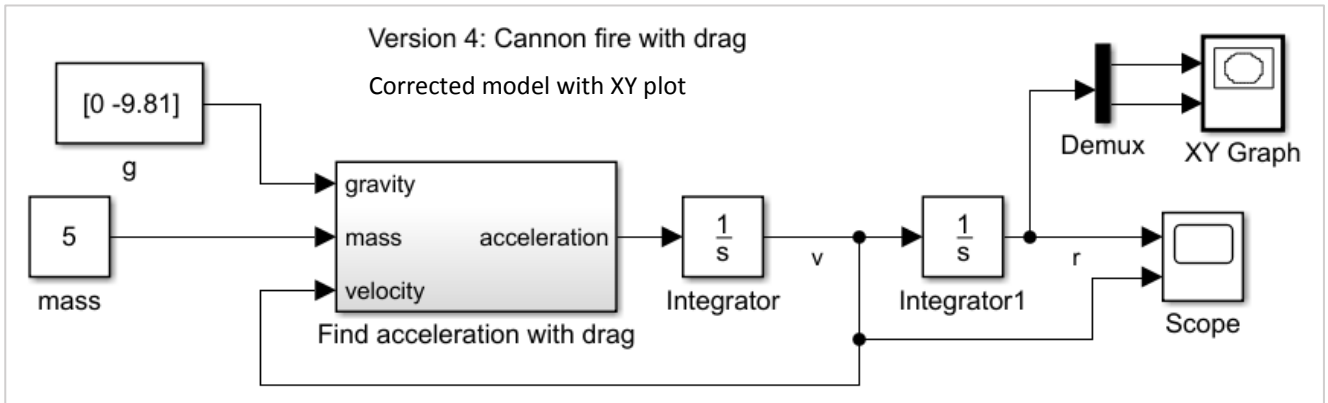
The results should be as shown on the right:



1. d) Further Plotting

XY Plot

Add an XY Graph Block from the Sinks Library. Use a Demux Block to split r into the x and y components and feed them into the x y graph. The model should now look like this.

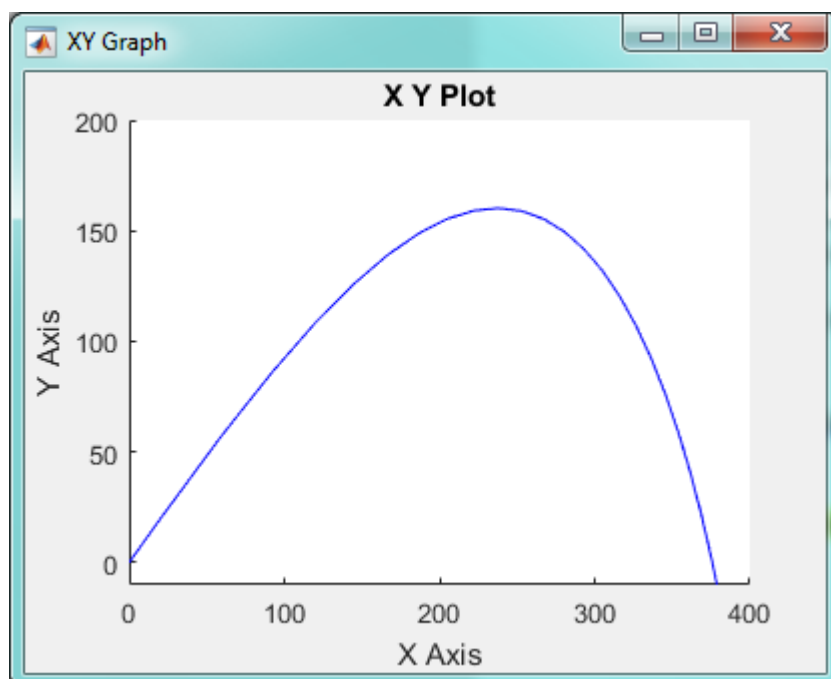


Change the Block Parameters of the XY Graph so that

- x min = 0**
- x max = 400**
- y min = -10**
- y max = 200**

Save as cannon4.slx and run.

The XY Graph should look as shown below:



1. e) Using MATLAB Expressions

It would be nice to change the model so that the **elevation** of the cannon can be selected. This is easily done by using a MATLAB expression to set the initial velocity.

- Change the initial condition of the velocity integrator to:

$$100 * [\text{cosd}(\text{theta}) \text{ sind}(\text{theta})]$$

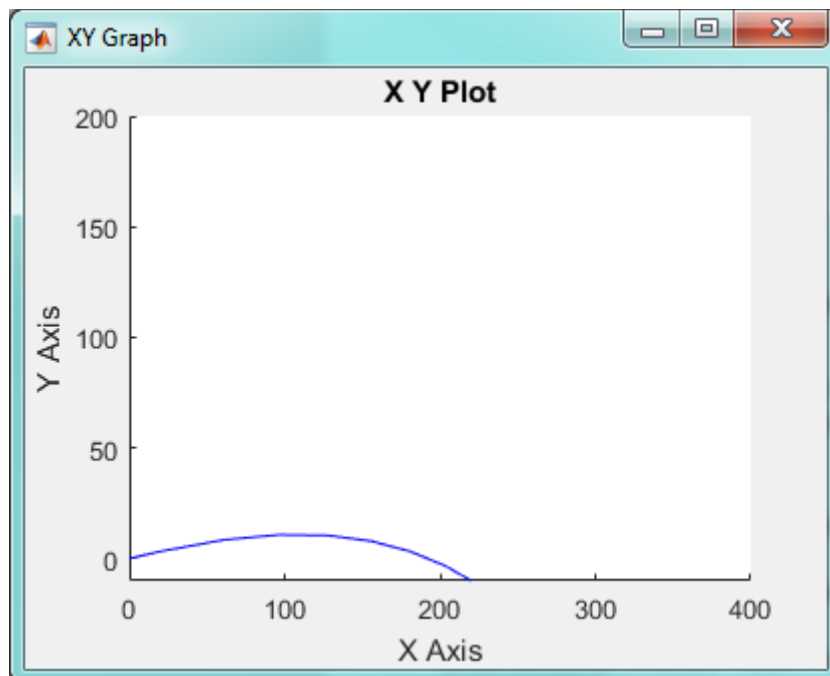
theta is the elevation of the cannon in degrees.

- In the MATLAB command window enter **theta = 10;**
This will set the elevation of the cannon to 10 degrees.

This works as Simulink models have access to variables in the MATLAB workspace.

Save the model as cannon5.slx

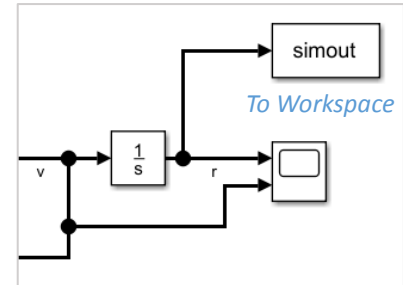
Try running the simulation with different angles of elevation.



1. f) Combining MATLAB and Simulink

MATLAB and Simulink are both valuable and have their different strengths. MATLAB is particularly useful for analysis. If you have experience with MATLAB, you may want to work with your simulation results within MATLAB. You can even run the simulation programmatically within MATLAB, and collate the results of many test cases.

- Replace the Demux & XY Graph block with a **To Workspace Block**.
In Block Parameters, change **Save Format** to **Array**.
- Save this new version as **cannon6.slx**
Now the $r = [x \ y]$ signal will be available in MATLAB.



- Type `edit fireCannon` in the command window to **start a new script**.
If you have not used MATLAB scripts, ask a demonstrator for help here.

- Use the `sim` command to run the simulation from MATLAB.

Then access `r` using the `get` function, and assign vectors `x` and `y`:

```
s = sim('cannon6', 'StopTime', '25', 'MaxStep', '0.01');  
r = s.get('simout');  
x = r(:,1);  
y = r(:,2);  
plot(x,y)
```

From here you can easily analyse the results of the model. You could use a **for** loop to plot trajectories for a range of different masses, or starting elevation angles. An example of a more detailed script can be seen in `exampleCannon.m` which is on the lab computers.

```
% exampleCannon.m Script to run Simulink Model of Firing a Cannon  
clear variables  
  
m = 5;           % mass of canon ball  
D = -0.02;      % Drag coeff  
theta = 20;     % Launch angle in degrees  
y0 = 0;        % Launch Height (m)  
v0 = 100;      % Initial velocity (m/s)  
  
s = sim('cannon6', 'StopTime', '25', 'MaxStep', '0.01');  
r = s.get('simout');  
x = r(:,1);  
y = r(:,2);  
  
%% Logical to find where positive  
L = find(y>=0);  
plot(x(L),y(L))  
xlabel('Distance (m)'); ylabel('Height (m)')  
axis([0 max(x(L))*1.1 0 max(y)*1.1])  
subtitle = sprintf('Launched from %0.1f m at %0.1f degrees and %0.1f  
m/s',y0,theta,v0);  
title({'Trajectory of Cannon Ball';subtitle})  
txt = sprintf('Ball stopped at x=%0.1f m',x(length(L)));  
text(max(x)/3,max(y)/5,txt)
```

Exercise 2: Ordinary Differential Equations

System One

Model the following differential equation:

$$\frac{d^2y}{dt^2} + 2\frac{dy}{dt} + 5y = 1 \quad (1)$$

$$\dot{y}(0) = y(0) = 0$$

First, rearrange the equation so that you have an expression for the highest derivative.

$$\frac{d^2y}{dt^2} = 1 - 2\frac{dy}{dt} - 5y \quad (2)$$

While constructing the model, you will find that two of the blocks need to be flipped, so that the input is on the right and the output on the left. Two of the blocks in this model need to go in the opposite direction, so need to be reversed.

Select the blocks to be reversed. You can select all at once by dragging a box around them or hold down the shift key while selecting. If you prefer, reverse each block individually.

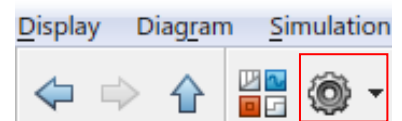
Right Click on the selected block(s) and select:

Rotate & Flip ► Flip Block

When you have constructed the model, check it with the diagram on the next page.

Set **max step size 0.01**

(in **Model Configuration Parameters**, accessed by clicking the **Cog** symbol and under drop down **Additional Parameters**).

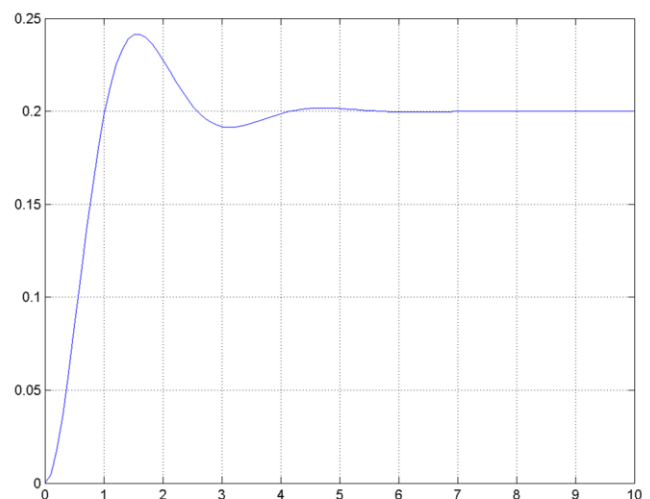


Save as **ODE1.slx**

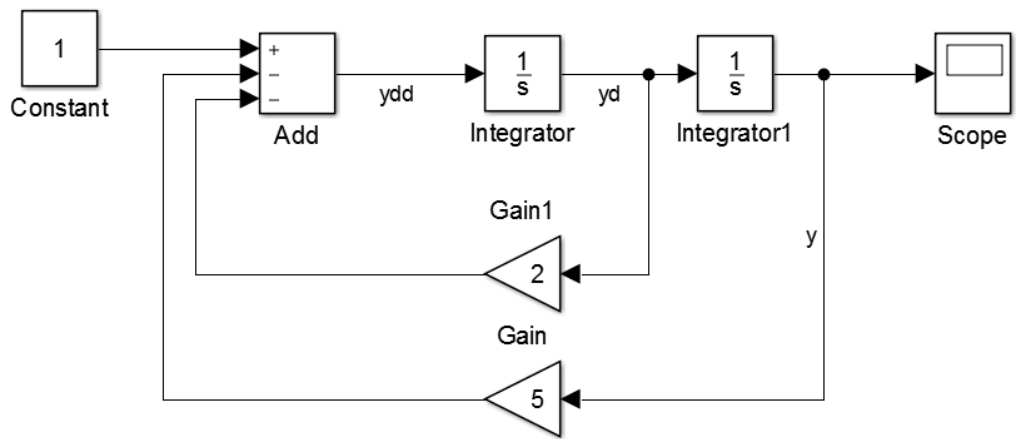
Run the model

The analytic solution to this problem is

$$y = \frac{2 - e^{-t}(2\cos(2t) + \sin(2t))}{10} \quad (3)$$



The model that you constructed should look something like this.



System Two

Model the following set of differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & -5 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (4)$$

$$x(0) = y(0) = 0$$

Hint

It may help to first model the following equation:

$$\dot{U} = C + AU \quad (5)$$

where A and C are constants.

Then you set

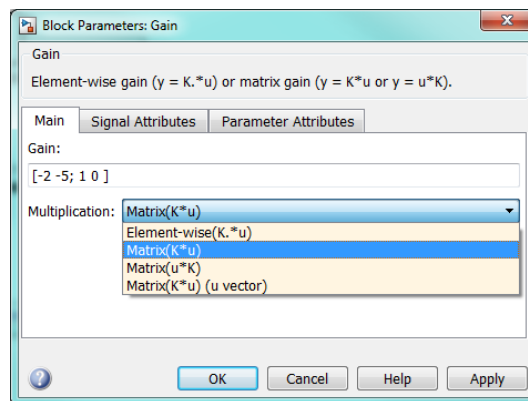
$$C = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} -2 & -5 \\ 1 & 0 \end{bmatrix}$$

and configure the multiplication to matrix multiplication. This means that in the **Gain Block Parameters** you will need to choose **Matrix** rather than **Element-wise multiplication**.

Set max step size to 0.01

Save as ODE2.slx

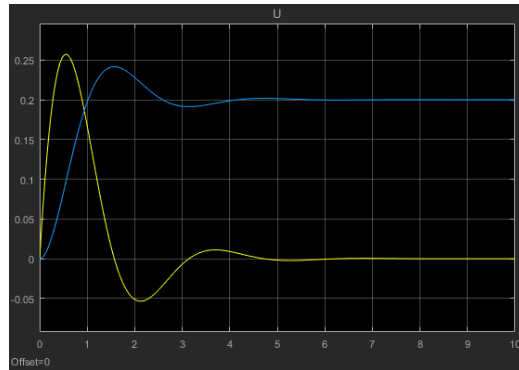
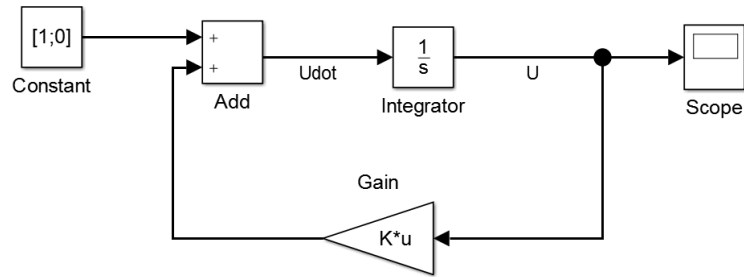
Run the model



The output y should be the same as y in equation (1).

See the next page for an explanation of why this is the case.

The model that you constructed should look something like this:



Why are the plots the same?

System Two is just a reformulated version of the equation in System One. Here is why:

Take the equation from part A:

$$\frac{d^2y}{dt^2} = 1 - 2\frac{dy}{dt} - 5y \quad (2)$$

$$\dot{y} = 1 - 2\dot{y} - 5y = 1$$

Let

$$x = \dot{y} \quad (6)$$

therefore

$$\dot{x} = \ddot{y} \quad (7)$$

Substituting (6) and (7) into equation (2).

$$\dot{x} = 1 - 2x - 5y \quad (8)$$

Equations (6) and (8) form a set of differential equations

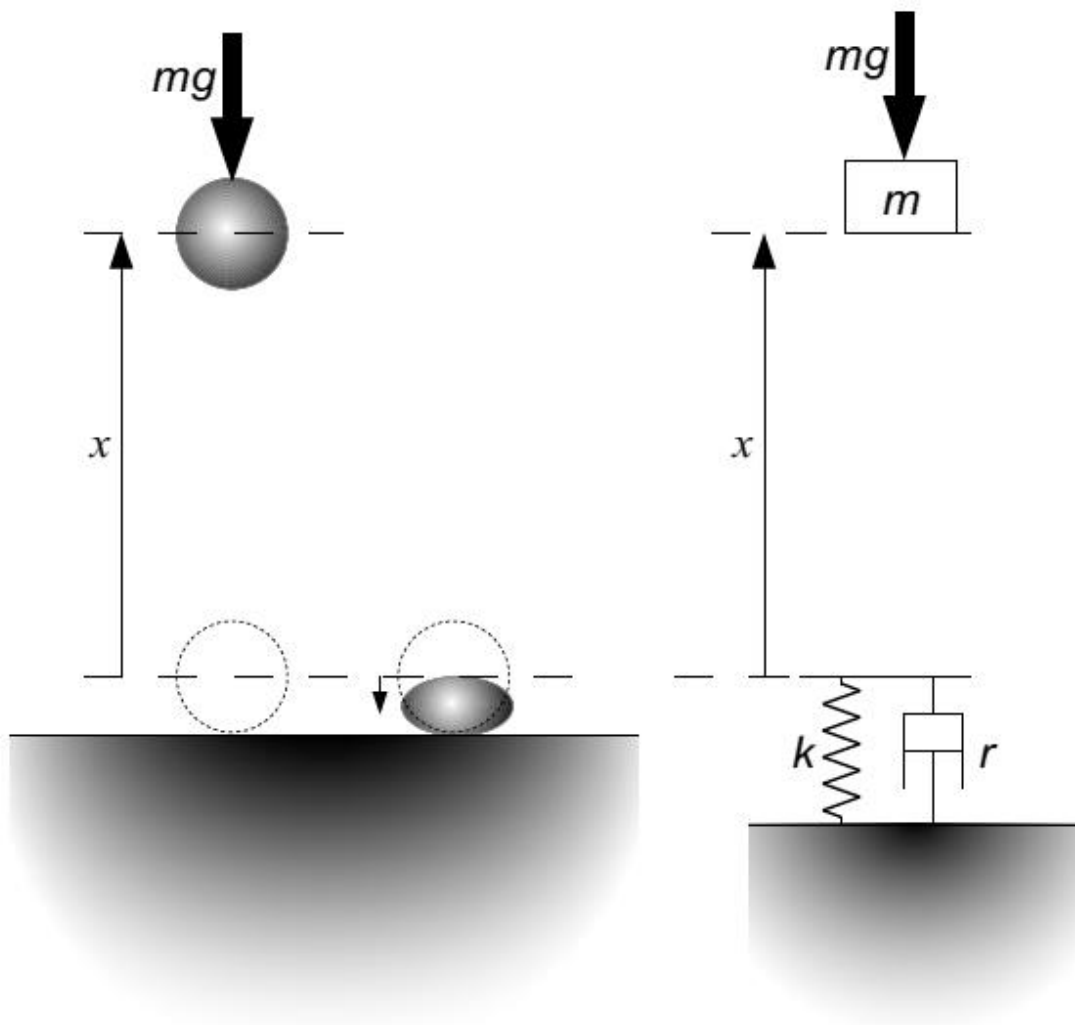
$$\dot{x} = 1 - 2x - 5y \quad (9)$$

$$\dot{y} = x$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & -5 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (10)$$

Exercise 3: Bouncing Ball

(a) Build a Simulink model of a bouncing ball. Model the ball as a mass being dropped onto a spring damper system.



The equation of the system depends on the position of the ball.

If $x \geq 0$

$$\ddot{x} = -g \quad (1)$$

If $x < 0$

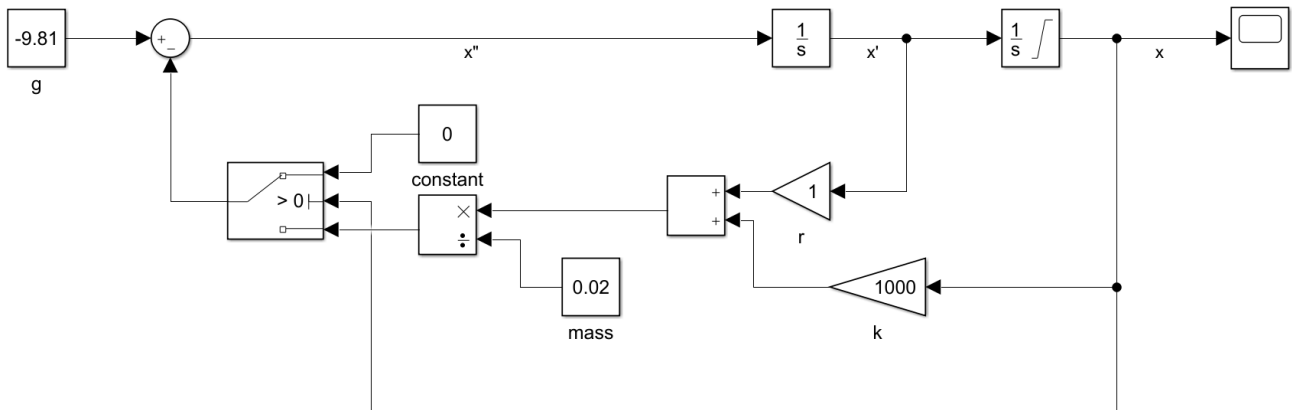
$$\ddot{x} = -\frac{r}{m}\dot{x} - \frac{k}{m}x - g \quad (2)$$

I suggest you start with $m = 0.2$, $k = 1000$ and $r = 1$. Try dropping the ball from a metre up.

(b) At the moment, the ball can go as low as it likes. In practice the ball cannot go lower than its radius. Modify the model so the ball cannot go lower than -0.02 metres.

Hint: Look at the Integrator Limited block.

The model that you constructed should look something like this:



Where next?

These exercises are just the start of learning to use Simulink. Hopefully from the talk and exercises, you now have a taster of what it is capable of. Below are some links to explore.

Simulink Onramp

In MATLAB 2018b, you can complete the **Simulink Onramp** course created by MathWorks. It is around 3 hours of good quality content designed to introduce you to Simulink. It will give you more practice at a similar level to the exercises in this course.

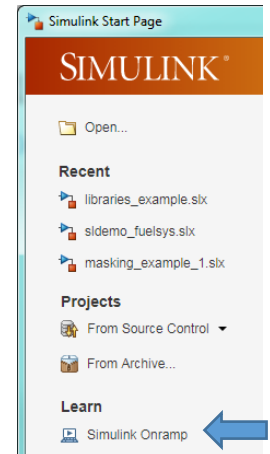
You can access this by installing MATLAB 2018b (see instructions on final pages of notes) and then downloading a toolbox available at

bit.ly/SimulinkOnRamp

OR

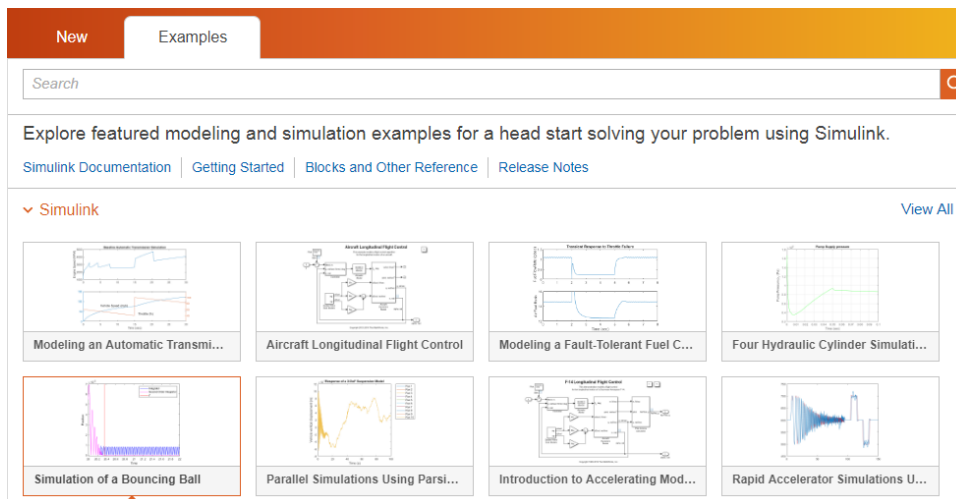
<https://uk.mathworks.com/matlabcentral/fileexchange/69056-simulink-onramp>

Once installed, restart MATLAB, launch Simulink and click the link under **Learn**



Explore Simulink Examples

Use the Examples Tab to explore different Simulink models (File > New > Model...)



There is even a bouncing ball example so you can see a different approach to one you might have taken in the final exercise.

Experiment with Simulink Dashboard Blocks

Open the Fuel System Demo with the command:

```
open_system([matlabroot '\toolbox\simulink\simdemos\automotive\fuelsys\sldemo_fuelsys'])
```

Read through the documentation, so that you understand how the model works:

<https://uk.mathworks.com/help/simulink/ug/tune-and-visualize-your-model-with-dashboard-blocks.html>

Explore the blocks by double clicking them to get a better understanding of how they work. Then follow the instructions under the section **Tune Parameters During Simulation** which will get you to edit the model.