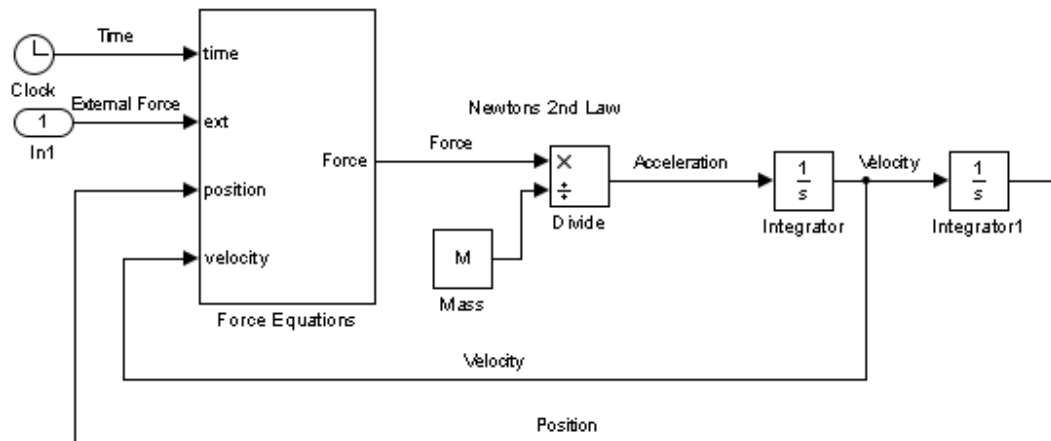


## Examples Models

The following examples will hopefully give ideas for the best way to lay out models. While there is no right or wrong way, always prioritise readability.

### Example 1: Dynamic Systems



The above shows the general approach to modelling dynamic systems. You calculate the force, use Newton's 2<sup>nd</sup> law to calculate the acceleration, integrate to get the velocity and then integrate again to obtain the position.

The above model is a general guide, it can get a bit more complicated. For example, the mass is required to calculate gravitational forces. It is also possible that the mass will be a function. For example, a rocket loses most of its mass as the fuel is burnt off.

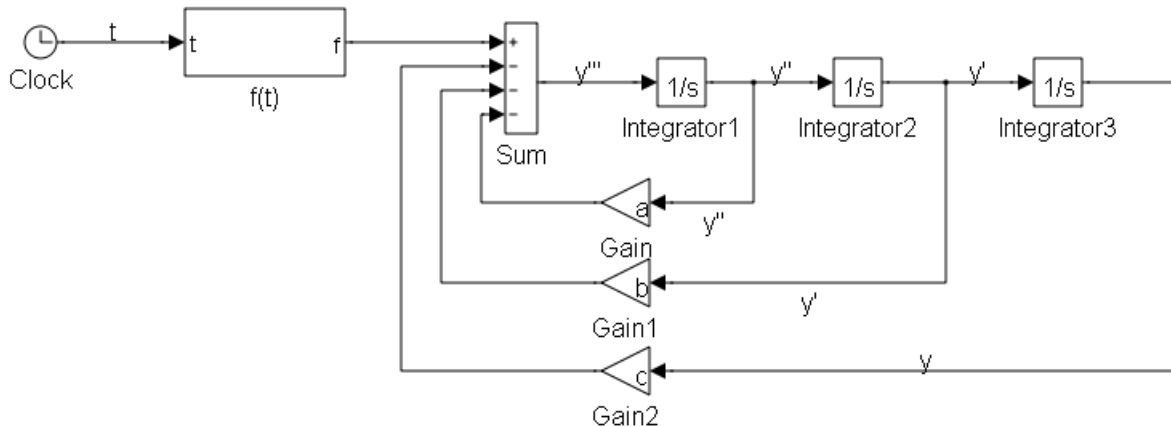
## Example 2: Ordinary Differential Equations (ODE)

The general rule for solving differential equations is to write the equation in terms of the highest differential. For example, consider the general second order equation below.

$$\ddot{y} + a\dot{y} + by + cy = f(t) \quad (1)$$

$$\ddot{y} = f(t) - a\dot{y} - by - cy$$

You then use integrators to obtain lower terms:

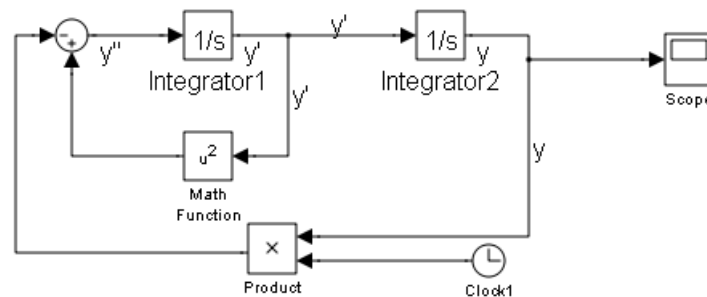


The right hand side of the equations is formed by feeding back these terms to form the expression required.

Notice that the model contains no differentiators, even though we are modelling a differential equation. Models with differentiators tend to produce a lot of noise, so are avoided if possible.

The next example is not linear or time invariant:

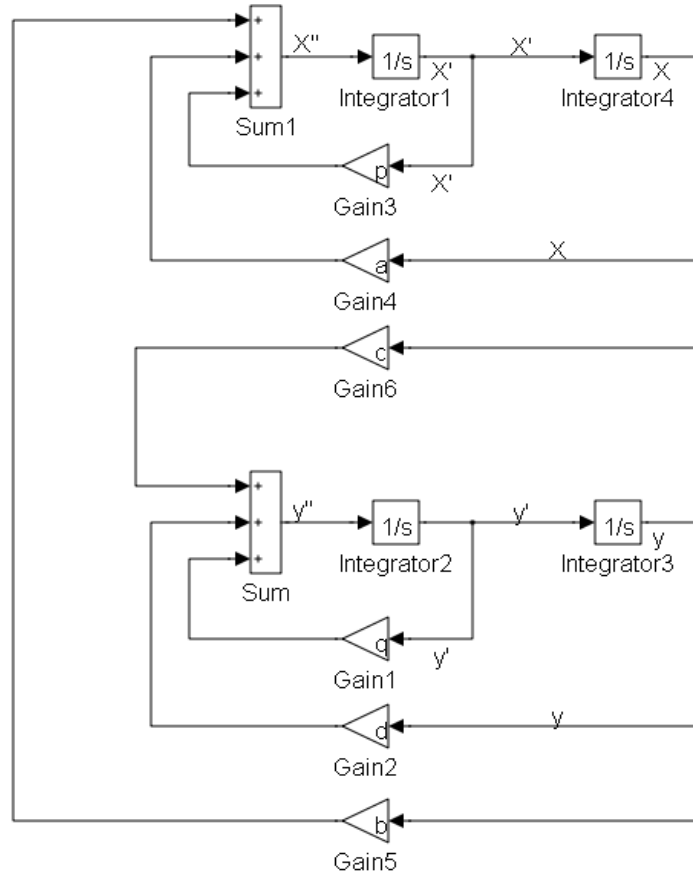
$$\ddot{y} = \dot{y}^2 - ty \quad (2)$$



### Example 3: Simultaneous Ordinary Differential Equations

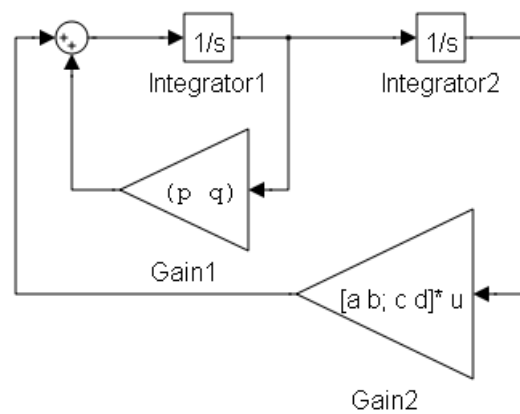
Of course Simulink is not limited to equations in one variable.  
Consider the simultaneous equation below:

$$\begin{aligned}\ddot{x} &= p\dot{x} + ax + by \\ \ddot{y} &= q\dot{y} + cx + dy\end{aligned}\quad (3)$$



This can be simplified by using matrices:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} p & q \\ a & b \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}\quad (4)$$



### Example 4: Linear Systems

Many systems can be modelled by linear, time invariant (LTI) differential equations, such as equation 5 below.

$$a_3\ddot{y} + a_2\dot{y} + a_1\dot{y} + a_0y = b_2\ddot{x} + b_1\dot{x} + b_0x \quad (5)$$

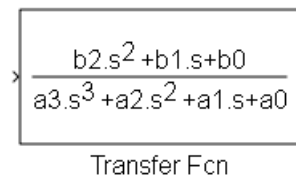
where  $y$  is the output and  $x$  the input.

LTI systems can be represented by a transfer function:

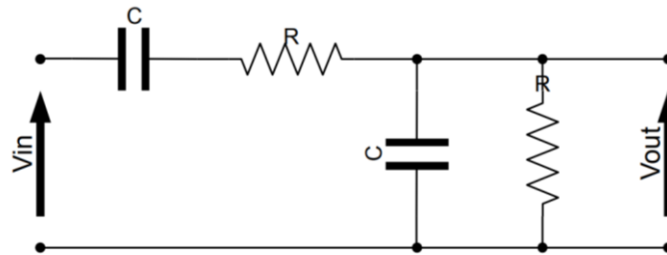
$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_2s^2 + b_1s + b_0}{a_3s^3 + a_2s^2 + a_1s + a_0}$$

$$H(j\omega) = \frac{b_2(j\omega)^2 + j\omega b_1 + b_0}{a_3(j\omega)^3 + a_2(j\omega)^2 + j\omega a_1 + a_0}$$

It is the convention in MATLAB to represent polynomial expressions with row vectors of the coefficients. So the numerator of the above transfer function is represented by  $[b_2 \ b_1 \ b_0]$  and the denominator by  $[a_3 \ a_2 \ a_1 \ a_0]$ . Entering these two vectors to the appropriate block parameters of a transfer function block will produce the following block:

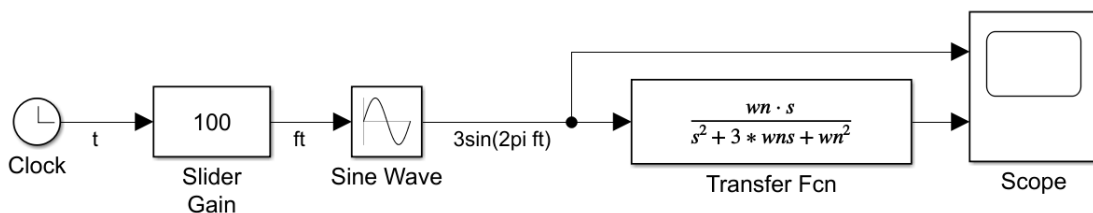


LTI transfer functions are used extensively in electronics to represent idealized electronic circuits. Take for example this circuit and its transfer function representation below:



$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{s\omega_n}{s^2 + 3s\omega_n + \omega_n^2} \quad (6)$$

The following model can be used to observe the behaviour in Simulink:



In the Transfer Function block parameters values are set with  $wn$  being a predefined variable in the MATLAB workspace: **numerator** =  $[wn \ 0]$  and **denominator** =  $[1 \ 3*wn \ wn^2]$

## Poles and Zeros

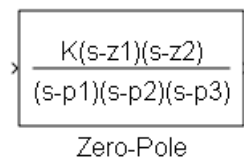
An alternative way of representing a transfer functions is to use the pole-zero description. If you solve the numerator polynomial you get the zeros. So called because the transfer function is zero at that value. If you solve the denominator polynomial, you get the poles. They are called poles because if you plot the absolute value of a transfer function, it looks a bit like a tent, with the poles being the location of the tent poles.

The transfer function from the circuit example can then be represented in its pole zero form:

$$H(s) = K \frac{(s - z_1)(s - z_2)}{(s - p_1)(s - p_2)(s - p_3)} \quad (7)$$

Where  $p_i$  is a pole and  $z_i$  is a zero and  $K$  is a constant.

You can model the transfer function in this form using a zero-pole block:



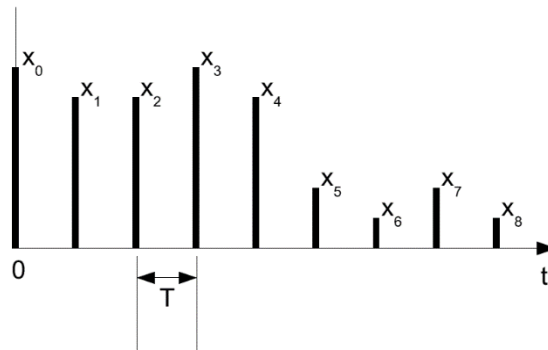
To configure this block you provide a vector for the numerator and the denominator. In this case the numerator is [**z1 z2**] and the denominator is [**p1 p2 p3**] and the gain is **K**.

### Useful MATLAB functions:

- The MATLAB function **roots** will solve a polynomial, given the coefficients of the polynomial. The function **poly** does the opposite. Given the roots of a polynomial, it will return the coefficients of the polynomial.
- The Signal Processing toolbox provides a number of functions to provide the coefficients required to implement various filters. See help for **butter**, **cheby1**, **cheby2** and **besself**.
- The function **freqs(B,A)** will plot the frequency response of a system, where **B** is a vector of the numerator coefficients and **A** is a vector of the denominator coefficients.

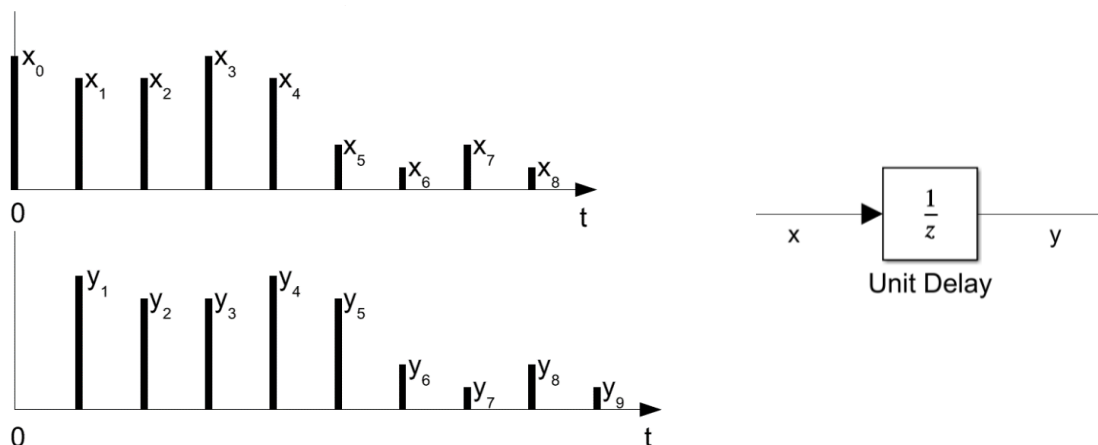
## Example 5: Modelling Discrete Systems

A discrete signal has values only at discrete points in time. A sampled signal is always discrete. The sample period,  $T$ , is the time between two successive samples and sample frequency,  $f_s$ , is  $1/T$ .



You will need to set the Sample Time in Block Parameters for many of the blocks in the discrete library. Most blocks use '-1' which is simply the inherited value from the model. Unfortunately, inherited sample time does not work for discrete models. If you find that your model is sampling at one second, regardless of the solver settings, then check the sample time of your blocks.

The fundamental component of a discrete system is a Unit delay. This delays the signal by one time period. In general,  $y_n = x_{n-1}$ , as seen in the example below.

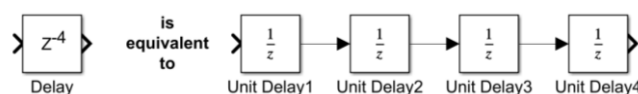


The Z transform replaces each delay by one sample with a multiplication by  $z^{-1}$ :

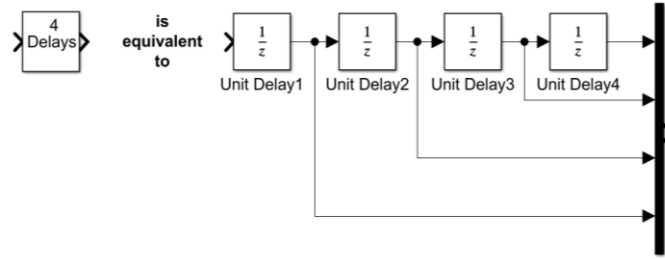
$$Y(z) = z^{-1}X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{z}$$

There are other blocks in the discrete library that contain combinations of unit delays.



You can also use the Tapped Delay block:



## Discrete Transfer Functions

Continuous systems are described by differential equations, discrete systems are described by recurrence equation. Equation 8 below is a typical recurrence equation:

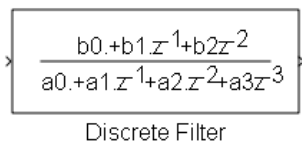
$$a_0 y_n + a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} \quad (8)$$

where  $x$  is the input and  $y$  the output. Each unit delay is replaced by  $z^{-1}$  in the Z transform.

$$(a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3})Y(z) = (b_0 + b_1 z^{-1} + b_2 z^{-2})X(z)$$

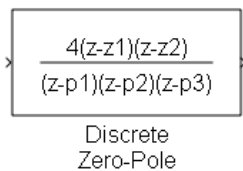
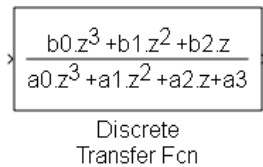
$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} \quad (9)$$

which is the transfer function of a digital filter and is **defined in terms of  $z^{-1}$** . This can be represented in a Simulink model by the **discrete filter block**.



This block was produced by setting:  
**numerator** = [b0 b1 b2] and **denominator** = [a0 a1 a2 a3]  
 in the block parameters.  
 Do not forget to set the sample time too.

An alternative form is to write the **transfer function in terms of  $z$** . If we multiply top and bottom of equation 9 by  $z^3$  we get equation (10). You can represent this with the Transfer Function block or if you have the poles and zeros, the Zero-Pole block.

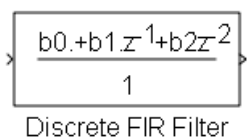


$$H(z) = \frac{b_0 z^3 + b_1 z^2 + b_2 z}{a_0 z^3 + a_1 z^2 + a_2 z + a_3} \quad (10)$$

Finite Impulse Response (FIR) digital filters do not have any poles. The recurrence equation:

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} \quad (11)$$

gives the following Z transform and can be represented by the block especially for FIR filters:



$$Y(z) = (b_0 + b_1 z^{-1} + b_2 z^{-2})X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + b_2 z^{-2} \quad (12)$$

## Simulink Shortcuts

This section contains the short cut keys that can be used to build and edit your model.  
For more details select from the model menu bar **Help ► Keyboard Shortcuts**


OBJECT SELECTION SHORTCUTS	
Select an object	Click
Select more objects	<b>Shift</b> +click
Select all objects	Ctrl+A
Copy object	Drag with right mouse button <b>Ctrl</b> +drag
Delete selected object	Delete <i>or</i> Backspace
Cut	Ctrl+X
Paste	Ctrl+V
Undo	Ctrl+Z
Redo	Ctrl+Y

BLOCK SHORTCUTS	
Search for blocks	Click and type
Add text to model	Double click and type
Move block	Drag <i>or</i> Arrow keys
Resize block	Drag handles in corners
Resize block, keeping same ratio of width and height	<b>Shift</b> + drag handle
Resize block from the center	<b>Ctrl</b> + drag handle
Rotate block counterclockwise	Ctrl + Shift + R
Flip block	Ctrl+I
Rotate block clockwise	Ctrl+R
Rotate block counterclockwise	Ctrl+Shift+R
Connect blocks	Drag from port to port
	Select first block, <b>Ctrl</b> +click second block
Draw branch line	<b>Ctrl</b> +drag line
	Right-mouse button+drag
Create subsystem from selected blocks	Ctrl+G
Open selected subsystem	<b>Enter</b> <i>or</i> Double click
Go to parent of selected subsystem	Esc

SIMULATION SHORTCUTS	
Open Configuration Parameters dialog box	Ctrl+E
Update diagram	Ctrl+D
Start simulation	Ctrl+T
Stop simulation	Ctrl+Shift+T
Build model (for code generation)	Ctrl+B



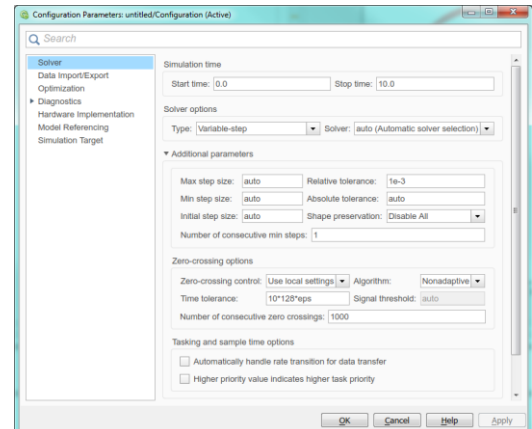
SIGNALS SHORTCUTS	
Name a signal line	Double-click signal and type name
Delete signal label and name	Delete characters in label or delete name in Signal Properties dialog box.
Delete signal label only	Right-click label and select <b>Delete Label</b> .
Open signal label text box for edit	Double-click signal line Click label
Move signal label	Drag label to a new location on same signal line
Copy signal label	<b>Ctrl</b> +drag signal label
Change the label font	Select the signal line (not the label) and use <b>Diagram &gt; Format &gt; Font Style</b>

ZOOMING SHORTCUTS	
Zoom in	Ctrl + +
Zoom out	Ctrl + -
Zoom to normal (100%)	Ctrl + 0 or Alt + 1
Zoom with mouse	<b>Ctrl</b> + scroll wheel
Zoom in on object	Drag the <b>Zoom</b> button  from the palette to the object
Fit diagram to screen	Spacebar
Scroll view	Arrow keys <i>or</i> <b>Shift</b> + arrow for larger pans
Scroll with mouse	Spacebar + drag Hold the scroll wheel down and drag the mouse

## The Solver: Zero-Crossing Options

A variable-step solver dynamically adjusts the time step size, causing it to increase when a variable is changing slowly and to decrease when the variable changes rapidly. This behaviour causes the solver to take many small steps in near a discontinuity because the variable is rapidly changing in this region. This improves accuracy but can lead to excessive simulation times.

Simulink uses a technique known as zero-crossing detection to accurately locate a discontinuity without resorting to tiny time steps. Usually this technique improves simulation run time, but it can cause some simulations to halt before the intended completion time. Understanding how Simulink's zero-crossing detection algorithms, adaptive and non-adaptive, work is beyond the scope of the course.



The table below should help you overcome some errors associated with zero-crossing, particularly a halting model. Implementing most of the changes, involves using the **Model Configuration Parameters dialog (MCP) box**, accessed via the Cog symbol.

Possible Change...	How to make this change...	Rationale for making this change...
Increase the number of allowed zero crossings	Increase the <b>Number of consecutive zero crossings</b> on the <b>Solver</b> pane in the MCP box.	This may give your model enough time to resolve the zero crossing.
Disable zero-crossing detection for a specific block	First, clear the <b>Enable zero-crossing detection</b> check box on the block's parameter dialog box. Then, select Use local settings from the <b>Zero-crossing control</b> pull down on the <b>Solver</b> pane of the MCP box.	Locally disabling zero-crossing detection prevents a specific block from stopping the simulation because of excessive consecutive zero crossings. All other blocks continue to benefit from the increased accuracy that zero-crossing detection provides.
Disable zero-crossing detection for the entire model	Select Disable all from the <b>Zero-crossing control</b> pull down on the <b>Solver</b> pane of the MCP box.	This prevents zero crossings from being detected anywhere in your model.
Reduce the maximum step size	Enter a value for the Max step size option on the <b>Solver</b> pane of the MCP box.	This can insure the solver takes steps small enough to resolve the zero crossing. However, reducing the step size can increase simulation time, and is seldom necessary when using the Adaptive algorithm.
Use the Adaptive Algorithm	Select <b>Adaptive</b> from the <b>Algorithm</b> pull down on the <b>Solver</b> pane in the MCP box.	This algorithm dynamically adjusts the zero-crossing threshold, which improves accuracy and reduces the number of consecutive zero crossings detected. You can now specify <b>Time tolerance</b> and <b>Signal threshold</b> .
Relax the Signal threshold	Select <b>Adaptive</b> from the <b>Algorithm</b> pull down and increase the value of the <b>Signal threshold</b> option on the <b>Solver</b> pane in the MCP box.	The solver requires less time to precisely locate the zero crossing. This can reduce simulation time and eliminate an excessive number of consecutive zero-crossing errors. However, relaxing the <b>Signal threshold</b> may reduce accuracy.

# Simulink Online Documentation

The full Simulink documentation is available from the help menu. You can obtain this from the MATLAB help, or you can go directly to the Simulink help. From the model menu bar select

**Help ► Simulink ► Simulink Help**

## Block Documentation

The easiest way of obtaining the documentation for a particular block is to hit the help button in the block parameters. An alternative is to select the block and then select

**Help ► Simulink ► Blocks & Blocksets Reference**

from the model menu bar. If no block is selected when you do this, then you will be given a list of all the blocks. You can then select the documentation you want from this list. At the top of the list, on the right hand side you can choose to display by Category or in Alphabetic order.

In the Help Menu you will also find links to Web Resources

## Help ► Web Resources

Particularly useful MATLAB Central which is the hub for the online MATLAB and Simulink community. Here you will find “MATLAB Answers” where people ask for support on MATLAB & Simulink. Once you become confident with MATLAB and Simulink you may wish to explore the File Exchange, where people upload custom files.

## Further Examples

### 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.

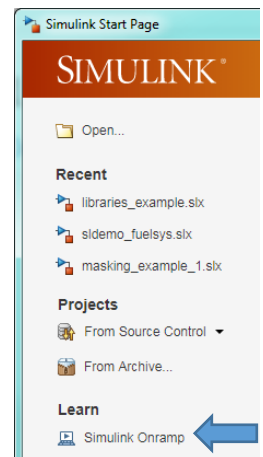
You can access this by installing MATLAB 2018b (see instructions on final pages of notes) and then downloading a toolbox available at

[bit.ly/SimulinkOnRamp](http://bit.ly/SimulinkOnRamp)

OR

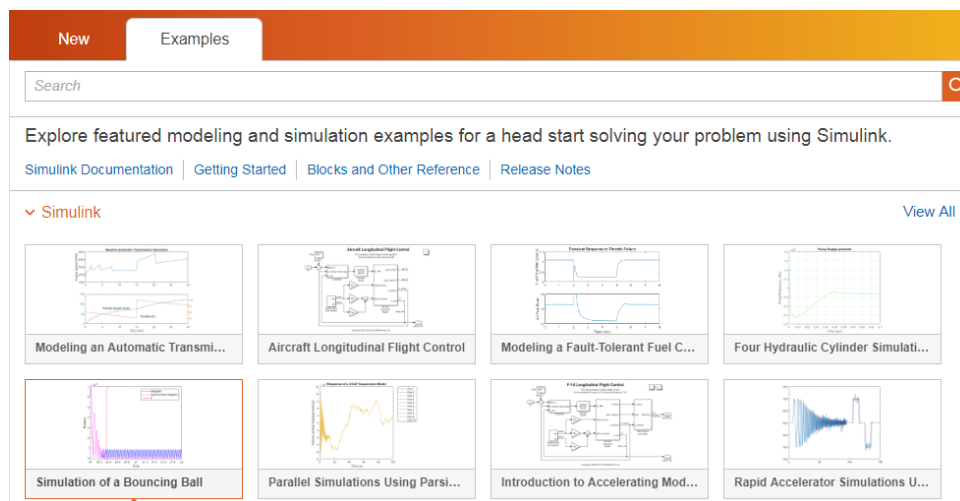
<https://uk.mathworks.com/matlabcentral/fileexchange/69056-simulink-onramp>

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>

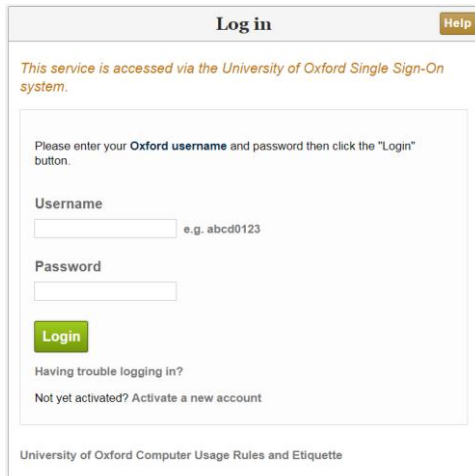
Explore the blocks by double clicking them to get a better understanding of how they work.

Follow the instructions in the section **Tune Parameters During Simulation** to try editing the model.

# Oxford University MATLAB Installation

## 1. VISIT UNIQUE WEB ADDRESS

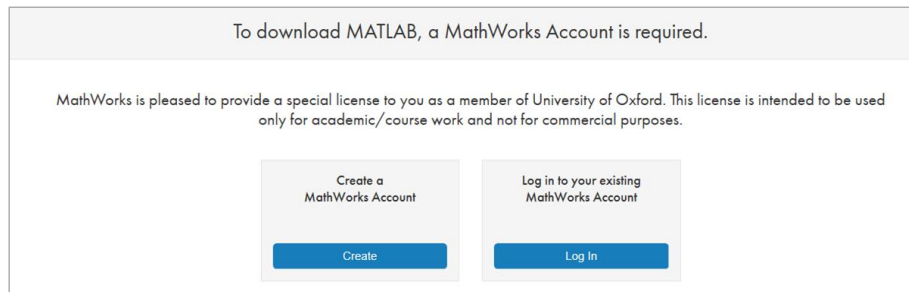
To install MATLAB onto your computer, go to the web page <http://bit.ly/OxUniMatlab> OR <https://www.mathworks.com/login/identity/university?entityId=https://registry.shibboleth.ox.ac.uk/idp>



Use your University of Oxford, single sign-on username and password.

## 2. CREATE UNIVERSITY LINKED MATHWORKS ACCOUNT

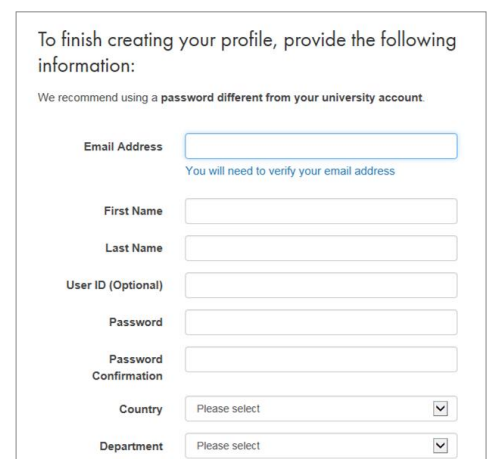
Then click on **Create** to make a MathWorks Account:



To register to use MATLAB, you need an Oxford University e-mail address such as [firstname.lastname@eng.ox.ac.uk](mailto:firstname.lastname@eng.ox.ac.uk).

Fill in the rest of the form. The system will send you an e-mail, with a link you must click to verify.


You can access your e-mail at <https://outlook.office.com/owa/>



**Now you can access many resources:  
e.g. MATLAB Online, Mobile or Academy.**

**You can also download and install MATLAB for your personal computer.  
See the next page for details.**

### 3. DOWNLOAD CHOSEN MATLAB VERSION

After verification you will be taken directly to the MATLAB download page.  
(Also accessible by “My Account” and the Download Icon:  )



*Choose the most recent release (mac users see the table for guidance).*

### 4. SELECT THE CORRECT INSTALLATION METHOD AND LICENSE

When you run the installer, you will be asked to select an **Installation Method**.

*Select **Log in** with a MathWorks Account.*

Later, you will be asked enter an **e-mail address** and **password**.

*Use the **e-mail address** and **password** that you for your MathWorks account*

When asked to **Select a license**, choose the license with the **Individual** Label.

**Toolboxes:** When asked to select the products, there are over 80 toolboxes available to install. If you are using a standard broadband network connection at home, it will take many hours to download all the toolboxes. To save time, select just MATLAB and the toolboxes you need. We suggest MATLAB, Symbolic Math Toolbox and Simulink. You can run the installer again later to add additional toolboxes.

#### Which MATLAB version for mac?

Use the table on the right to choose the correct MATLAB release for your operating system.

To find which version of OSX you are using. On the Mac, Click on the apple in the far top left.

Select **About this MAC**

If you have any problems or queries, have a look at the MATLAB FAQ page:

<http://users.ox.ac.uk/~engs1643/matlab-faq.html>

Mac Operating System		MATLAB
High Sierra	macOS 10.13	R2018a
Sierra	macOS 10.12	R2018a
El Capitan	OS X 10.11	R2018a
Yosemite	OS X 10.10	R2017a
Mavericks	OS X 10.9.5	R2015b
	OS X 10.9	R2014b
Mountain Lion	OS X 10.8	R2014b
Lion	OS X 10.7.4 & above	R2014b
	OS X 10.7	R2012a or b
Snow Leopard	OS X 10.6.4 & above	R2012a or b
	OS X 10.6.x	R2010b
Leopard	OS X 10.5.8 & above	R2010b
	OS X 10.5.5 & above	R2010a
	OS X 10.5.x	R2008b