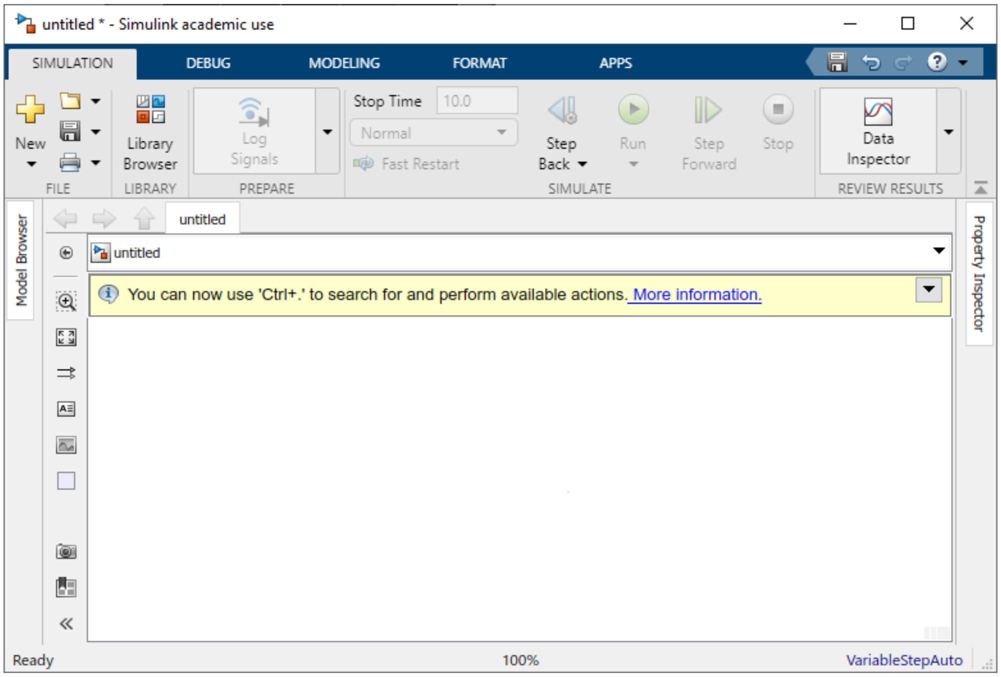
AE4610 Dynamics & Controls Lab
Simulink Tutorial

Last updated: Fall 2021

1 Introduction

- To begin, type `simulink` in the MATLAB command window or click **Simulink** icon in the **Home** tab:
- Select → **New** → **Blank Model**. The blank model window is shown in Fig. 1.



You can now use 'Ctrl+.' to search for and perform available actions. More information.

Figure 1: Simulink blank model window.

- Open the Simulink Library of operations by clicking the **Library Browser** icon:

  Please take a note of the **Simulation** as you may be required to change simulation parameters for some labs.

  This is a drag and select library (Fig. 2). Once this is done, you are ready to begin using Simulink.

  We mainly use blocks from **continuous**, **discrete**, **math** operations, **signal routing**, **sinks** and **sources**. The following tutorial will outline the main functions we use from each of these categories.

- *Note: You can access any block parameters window by double-clicking on the block.*

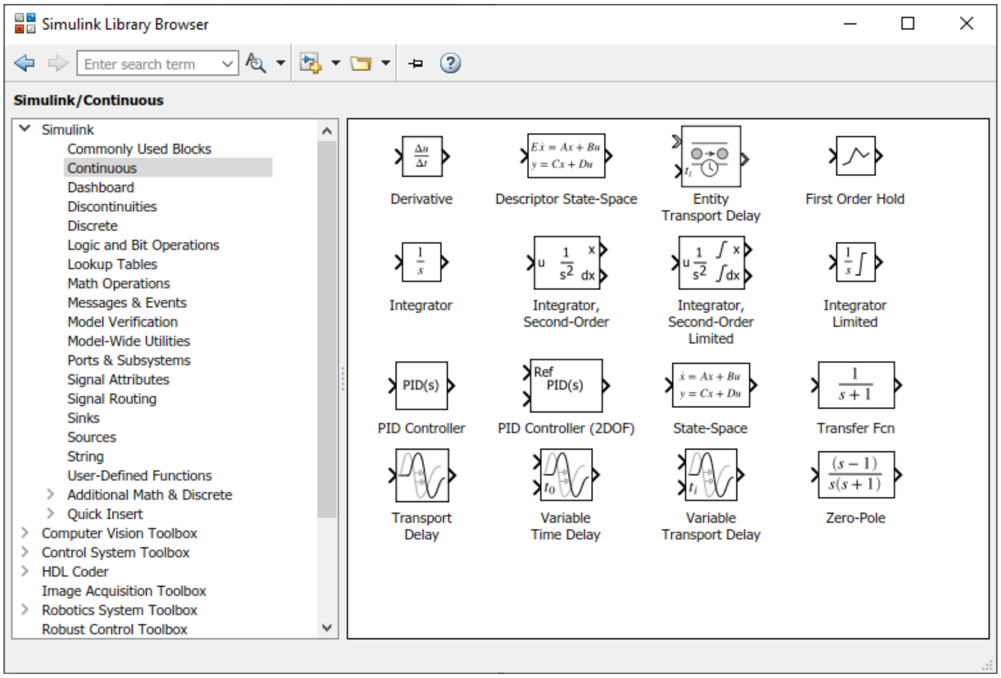
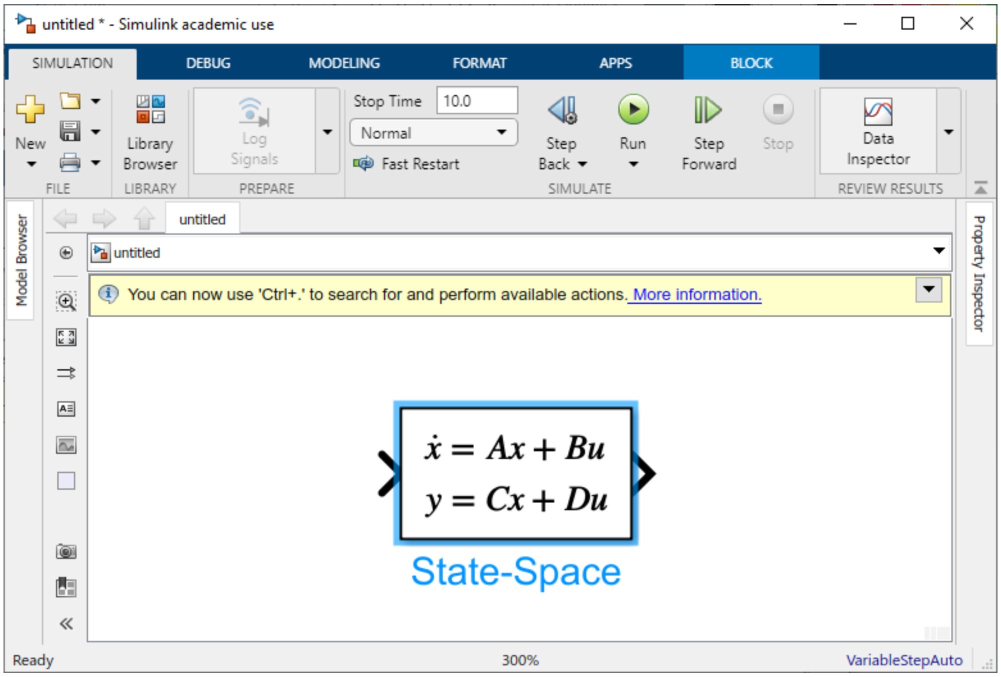


Figure 2: Simulink library window.

- *Note: You can select a block without accessing the Simulink library by double-clicking blank space in the model window and typing the name of the desired block.*
- *Note: You can rotate block by selecting it and pressing Ctrl+R.*

2 Continuous Tab

The state space block shown in Fig. 3 is one of the most important blocks you will need in Simulink. It is located in the **continuous** category of in the Simulink library.



You can now use 'Ctrl+.' to search for and perform available actions. More information.

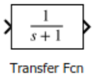
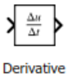
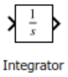
Figure 3: State-space block.

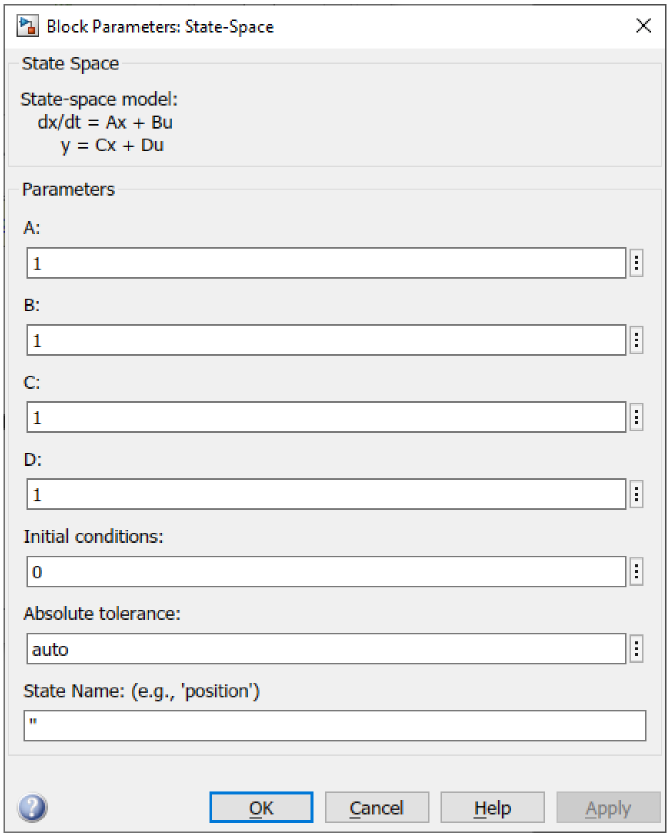
To change the input and output matrices, open the block parameters by double clicking on it (Fig. 4).

This allows you to change the matrices for the state space. Make sure to keep track of the dimensions of the matrices. You might be unable to close the window due to the dimensions of A&B, A&C, C&D, or B&D mismatch. The number of states must be equal for A and C, the number of inputs for B and D, the number of states for A and B and the number of outputs for C and D.

Note: you can use variable names from MATLAB workspace instead of typing matrices manually.

You can also find tab are **Transfer Fcn**, **Integrator** and **Derivative** blocks in the **continuous** section of the library:

- The **transfer function**  will be used in several labs in lieu of a state space model, together with integrators and derivatives. Parameters of the transfer function can be changed in the block parameters window.
- The **derivative** block , although useful for some of our labs and required for at least one, would almost never be used in a real life situation since the error that occurs from taking the derivative of data can be hurtful. There is nothing to change for this block.
- The **integrator** block  changes the signal from, for example, acceleration → velocity or velocity → position. You can set up initial conditions for the integrator at $t = 0$ by double clicking it.



Block Parameters: State-Space

State Space

State-space model:
dx/dt = Ax + Bu
y = Cx + Du

Parameters

A: 1

B: 1

C: 1

D: 1

Initial conditions: 0

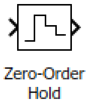
Absolute tolerance: auto

State Name: (e.g., 'position') ''

Figure 4: State-space block parameters.

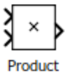
3 Discrete Tab

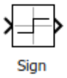
Discrete tab allows us to simulate data as we would collect it in the lab, with gaps between data points. We will only use **zero-order hold** from this section.

- The **zero-order hold**  is the main function we use from this group. This basically allows us to set a sample time for our "observer" to collect data. This sample time can be set by double clicking on the zero order hold and changing the time there.

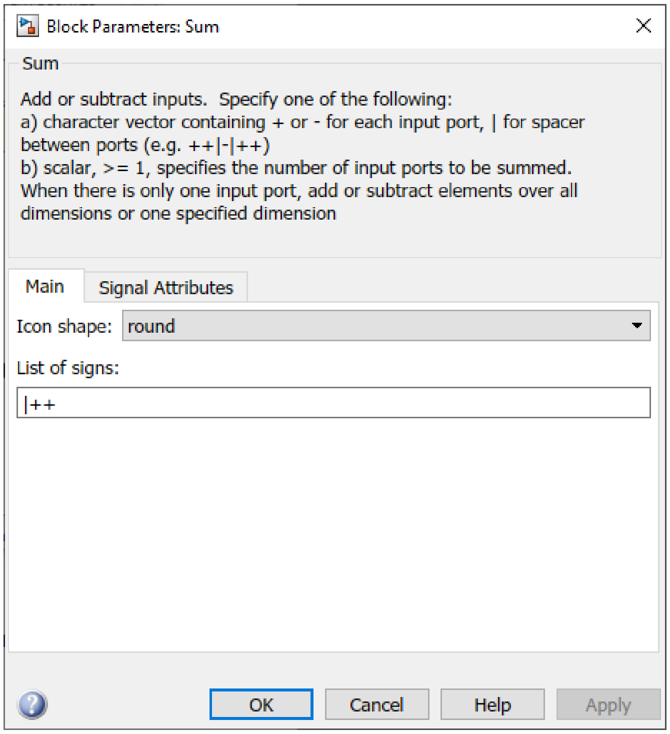
4 Math Operations Tab

- The **gain** block is the most common block besides the sum block we will be using in this category. The gain block multiplies the signal by the number inside the block, which can be changed by double clicking the block. The gain shown takes an input from the left and outputs on the right, however, by right clicking on the block, going down to "format" and selecting "flip block" you can change the direction of input for the block.

- The **product** block  allows multiple signals to be multiplied together into one signal. Now, by double clicking on the product block, you can change the number of inputs from 2 to whatever is required, and you can change the multiplication. You can either have the block do scalar or matrix multiplication, however, for our labs, we will only need scalar multiplication.

- The **sign** block  will output 1 for a positive input, -1 for a negative input, and 0 for 0 input.

- The **sum** block is as popular as the gain block because it allows us to add two signals together into one. This DOES NOT allow us to create a vector of the signals, that is done by another block known as a **mux** which will be covered below in **signal routing**. The sum block also allows us to subtract values from each other. By double clicking, you will get the block parameter screen shown in Fig. 5.



Block Parameters: Sum

Sum

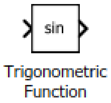
Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Icon shape: round

List of signs: |++

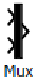
Figure 5: Sum parameters windows

By clicking on this, you can change the shape of the block. Now, under **List of signs**, by changing one of those signs from a + to a − will change it from addition to subtraction. To add additional signs, just add the appropriate sign to the list.

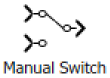
The **trigonometric function** block  allows us to take the sin, cos or any other trigonometric function of the signal.

5 Signal Routing

Signal routing is where we can change vector outputs from our state space and other functions into individual signals. It also allows us to change individual signals into vector signals. There are only two blocks we will be using from this, the **mux** and the **demux**.

- The **mux**  allows us to change individual signals into one signal roughly equivalent to a vector or length n, where n is the number of signals entering the mux. The number of signals entering the mux can be changed by double clicking the mux.

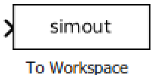
- The **de-mux** does the exact opposite of the mux, taking one signal and breaking it into several components. Now, for the de-mux, the output must be equal to the length of the vector in the signal going into the de-mux.

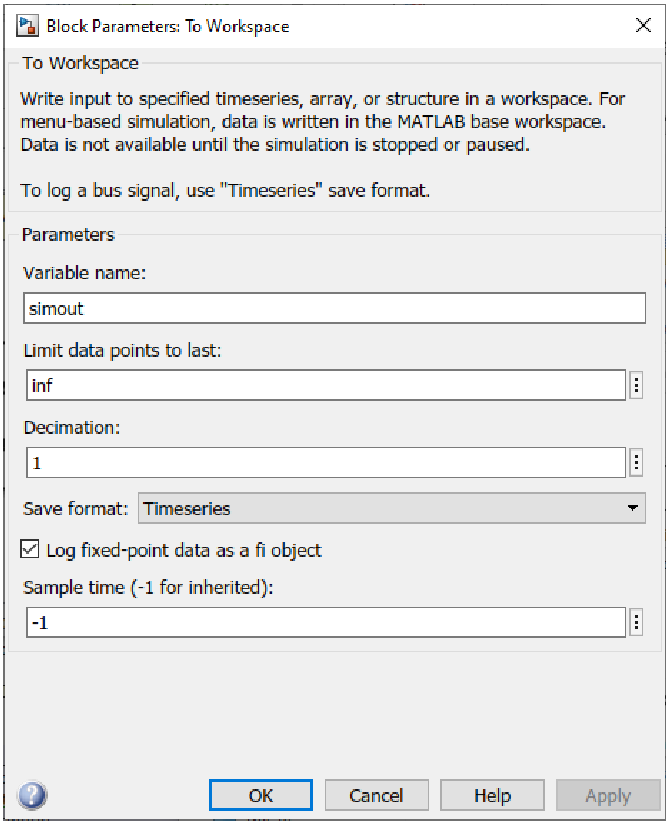
- Although you will not be required to use this in class, **manual switch**  is a useful block to know and one you will see a few times in lab. This allows us to switch from one input/signal to another by double clicking on the block. This is useful for sources, so we can switch from a step input to a ramp input, for example.

6 Sinks

Sinks are basically blocks where data can be dumped/stored from our diagram. There are two main ones we will be using throughout the semester.

- The **scope** is a quick way to see how your data looks. By inputting this into your diagram, you can get immediate results as to how the signal looks after running it, but only over time. It is supposed to be a basic diagram, and these will NOT be accepted as graphs in the lab reports.

- **To workspace**  allows us to export data to the workspace as an array or structure, however, we will only need to work with arrays. The default output format is structure and to change the format or variable name, double click on the block to access the block parameters (Fig. 6).



Block Parameters: To Workspace

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: simout

Limit data points to last: inf

Decimation: 1

Save format: Timeseries

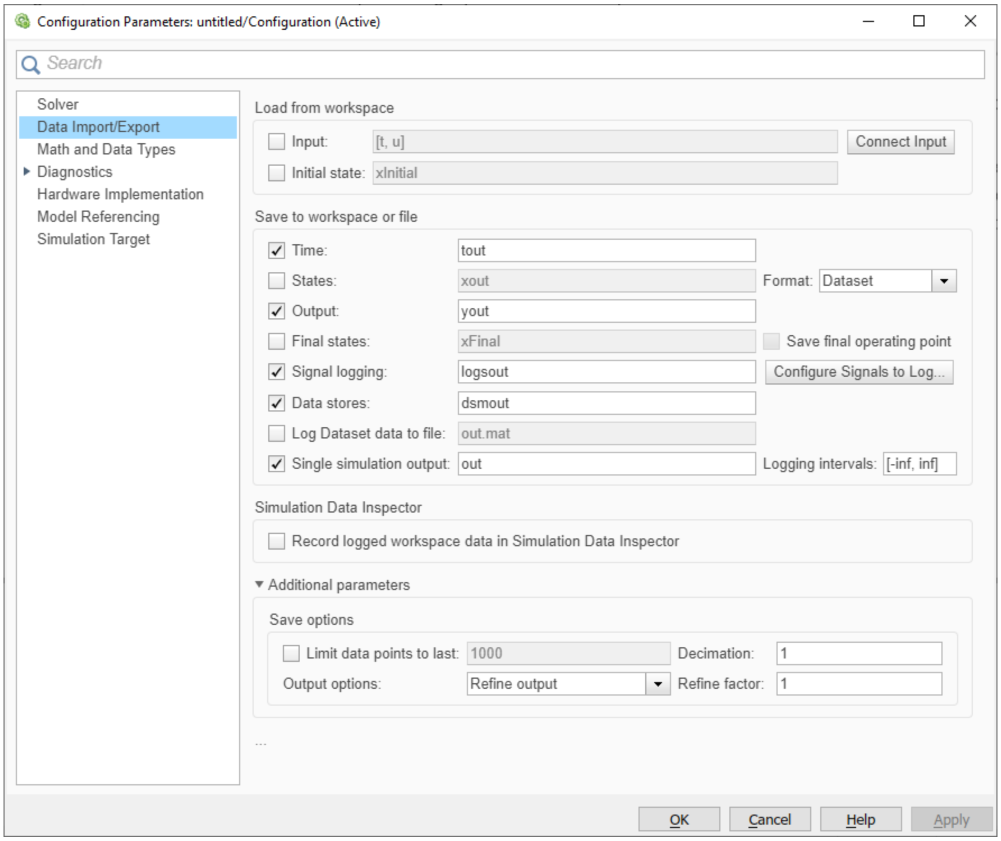
☑ Log fixed-point data as a fi object

Sample time (-1 for inherited): -1

Figure 6: To Workspace block parameters.

Note: By default, the program might record only 1000 data points, especially if using a fixed step integrator.

To change the number of data points, go to **Modeling** tab, click **Model Settings** icon, select **Data Import / Export** → **Additional parameters** and un-check the **Limit data points to last** checkbox (Fig. 7). You can also change the `tout` if you so desire to another variable name here.



Configuration Parameters: untitled/Configuration (Active)

Load from workspace

- □ Input: [t, u] — Connect Input
- □ Initial state: xInitial

Save to workspace or file

- ☑ Time: tout
- □ States: xout — Format: Dataset
- ☑ Output: yout
- □ Final states: xFinal — □ Save final operating point
- ☑ Signal logging: logsout — Configure Signals to Log...
- ☑ Data stores: dsmout
- □ Log Dataset data to file: out.mat
- ☑ Single simulation output: out — Logging intervals: [-inf, inf]

Simulation Data Inspector

- □ Record logged workspace data in Simulation Data Inspector

Additional parameters

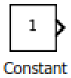
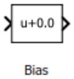
Save options

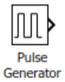
- □ Limit data points to last: 1000 — Decimation: 1
- Output options: Refine output — Refine factor: 1

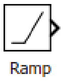
Figure 7: Confuguration parameters window.

7 Sources

Sources are the blocks used to generate signals, from step inputs to ramps. The following are a few we will be using in lab and a useful combination of signals.

- The **constant**  block adds a constant to the signal using a sum block. In newer versions, there is a **bias**  block that also does the same thing but without the requirement of a sum block.

- The **pulse generator**  block is included in the tutorial because you might see it in a few labs, however you shouldn't be using this in your Simulink diagrams. It generates a pulsed signal of a desired amplitude and pulse length, both of which can be changed in the block parameters window accessed by double clicking the block.

- A **ramp**  input does exactly what it shows; it is an input that increases at a constant rate. In the block parameters window, you can change the initial time ramp begins, slope, and the initial value of the ramp input.

- The **sine wave** input generates a sine wave. You can change the amplitude and the frequency as well as the phase by double clicking on the block.

- The **step** is the most important and most frequently used input. Basically it creates an input at a given time to a specific value. This is important because it can have an initial value, say 1, which it will output from t=0 until the step time, where it will change its output to the final value.

Note: For most of the labs, you will be required to generate data for a single pulse. Instead of using a pulse generator, a pulse can be created by combining two step inputs as shown in Fig. 8.

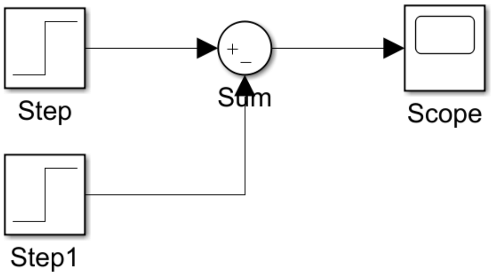
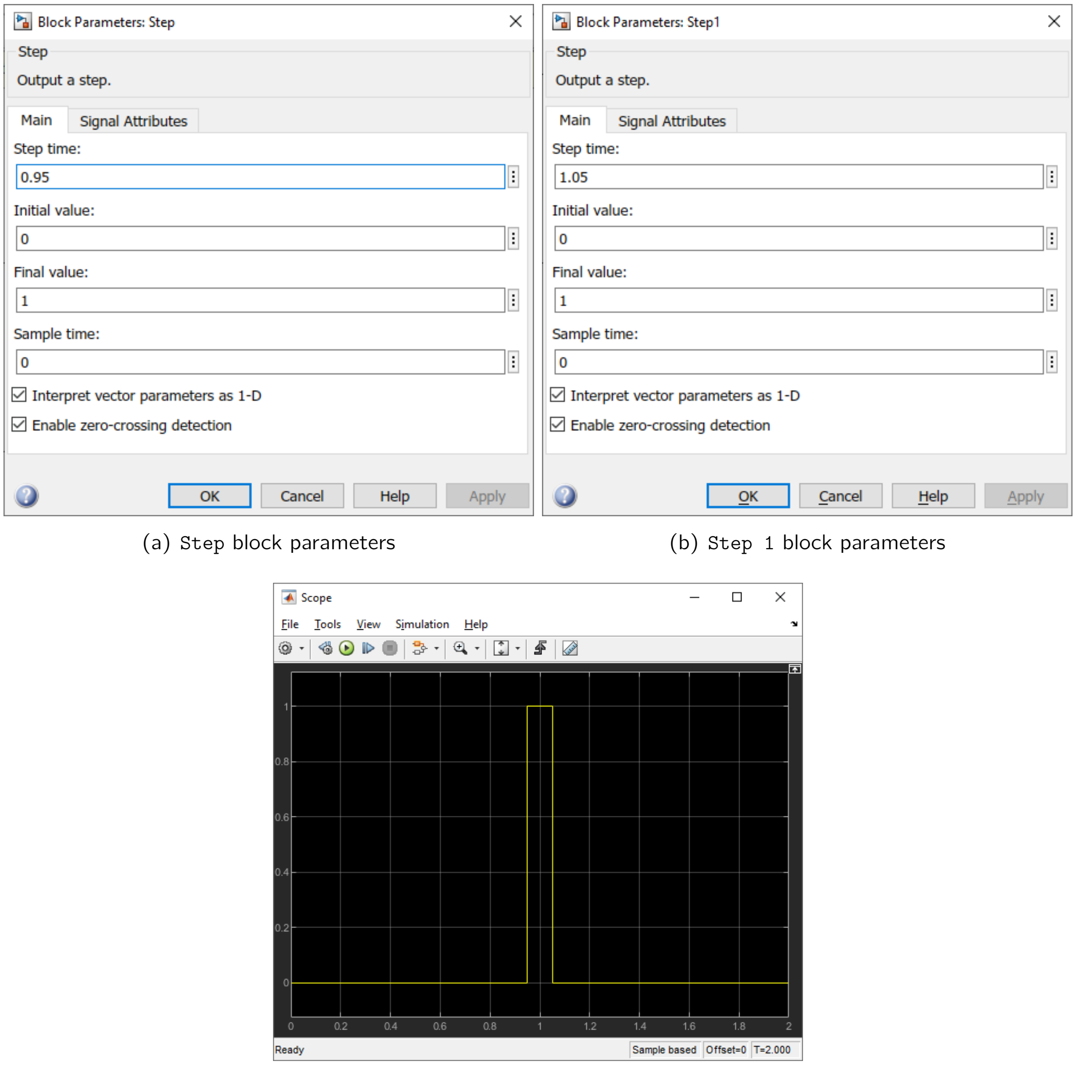


Figure 8: Single pulse from two step inputs.

Both steps have to be of the same amplitude. First step (`Step`) can start at any time while second step (`Step1`) start some time after the first one, canceling each other. Figures 9, 10, 11 show examples how combination of step parameters can be used to achieve various pulse inputs and will be useful in the future labs.



Block Parameters: Step

Step

Output a step.

Main | Signal Attributes

Step time: 0.95

Initial value: 0

Final value: 1

Sample time: 0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

OK | Cancel | Help | Apply

Block Parameters: Step1

Step

Output a step.

Main | Signal Attributes

Step time: 1.05

Initial value: 0

Final value: 1

Sample time: 0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

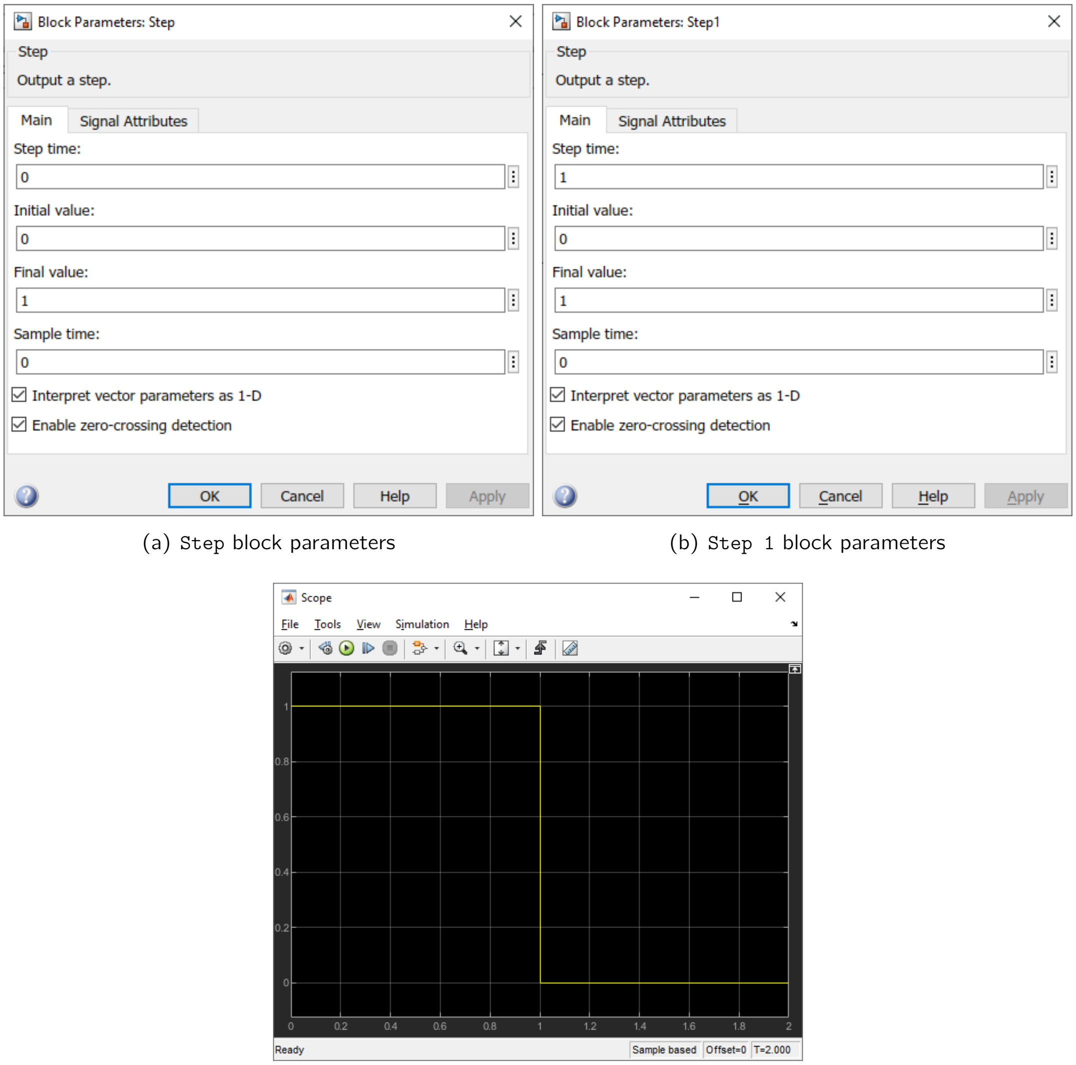
OK | Cancel | Help | Apply

(a) `Step` block parameters

(b) `Step 1` block parameters

(c) `Step 1` block parameters

Figure 9: Example 1: Single pulse.



Block Parameters: Step

Step

Output a step.

Main | Signal Attributes

Step time: 0

Initial value: 0

Final value: 1

Sample time: 0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

(a) `Step` block parameters

Block Parameters: Step1

Step

Output a step.

Main | Signal Attributes

Step time: 1

Initial value: 0

Final value: 1

Sample time: 0

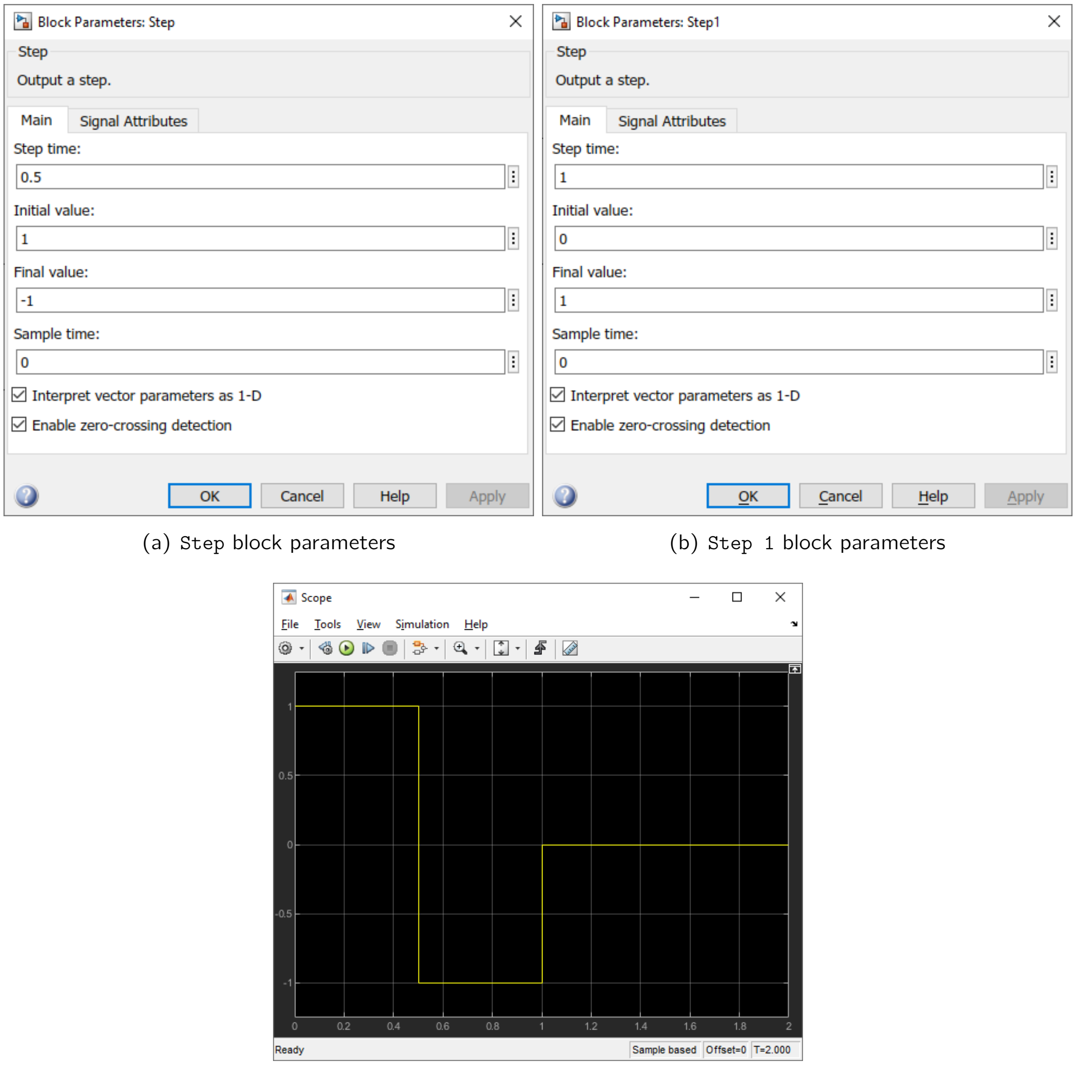
☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

(b) `Step 1` block parameters

(c) `Step 1` block parameters

Figure 10: Example 2: a combination of two steps is used to cancel step input after one second into the simulation.



Block Parameters: Step

Step

Output a step.

Step time: 0.5

Initial value: 1

Final value: -1

Sample time: 0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Block Parameters: Step1

Step

Output a step.

Step time: 1

Initial value: 0

Final value: 1

Sample time: 0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Ready | Sample based | Offset=0 | T=2.000

(a) `Step` block parameters

(b) `Step 1` block parameters

(c) `Step 1` block parameters

Figure 11: Example 3: Positive → negative → zero pulse. *Note: sum block was changed to ++.*

8 Helpful Tips

- To connect two blocks in simulink, click on one, hold Ctrl and then click the one you wish a signal to go to. This will automatically link the two together.
- If you wish to split a signal into two to go two to different sources, first attach the signal to one of the sources. Then, hold Ctrl, click on the signal and you can drag a new link to the second source.
- To show the automatic block names in the new versions of MATLAB, which are hidden by default, in the **Debug** tab, select **Information Overlays** icon and un-check Hide Automatic Block Names (Fig. 12).

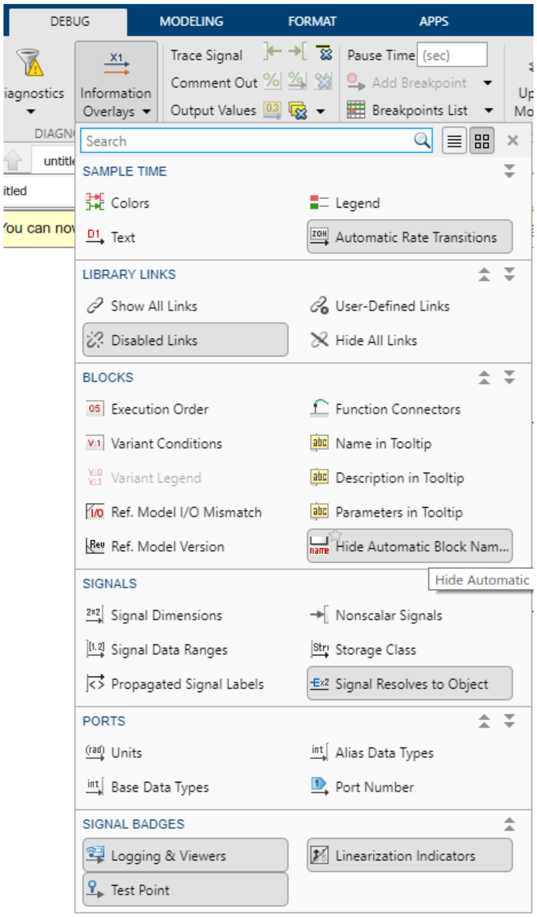


Figure 12: Show block names.