



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA  
CAMPUS DI FORLÌ

# Introductory Guide to Simulink

**Master's Degree in Mechanical Engineering for Sustainability**

[didatticaforli.ingstudenti@unibo.it](mailto:didatticaforli.ingstudenti@unibo.it)

## Table of Contents

Table of Contents .....	2
1. Introduction .....	3
2. Installation Guide.....	3
3. Working with Simulink.....	3
3.1. Creating a Simple Model .....	5
3.2. Modelling General Transfer Functions .....	10
3.3. Discrete and Continuous Variables.....	10
3.4. Model and Solver Settings .....	11
3.5. Modelling Differential Equations.....	12
4. MATLAB-Simulink Interactions .....	16
5. Model Examples .....	16
6. Where to go from here... ..	16
7. Additional Resources.....	17

# 1. Introduction

Simulink is a block diagram environment for multidomain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink offers a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling users to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.<sup>1</sup> In short, Simulink is an extension of MATLAB, allowing users to utilize its functions and graphical programming to export data from MATLAB, process them, and import the result of their analysis back to the MATLAB environment.

## 2. Installation Guide

Simulink can be set up during the installation of MATLAB. Alternatively, it is possible to install it later by selecting it in the Add-Ons selection. This can be found in the HOME toolbar in the [MATLAB interface](https://www.mathworks.com/help/matlab/matlab_interface.html)<sup>2</sup>. In the Add-Ons Explorer, users can utilize the search bar to find "Simulink". Once found, it can be selected, and by clicking the "Add" button the installation process can be started.

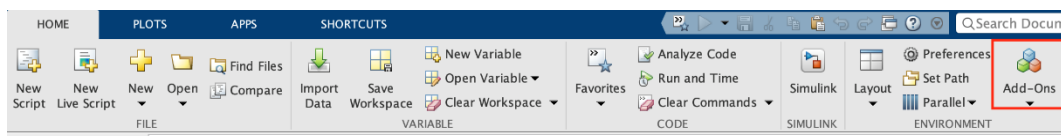


Figure 1: Add-Ons Selection in the MATLAB HOME toolbar.

## 3. Working with Simulink

As shown in Fig. 2, Simulink can be launched by selecting it in the HOME tab of the MATLAB interface. Alternatively, the "simulink" command in the command window will open up the interface.

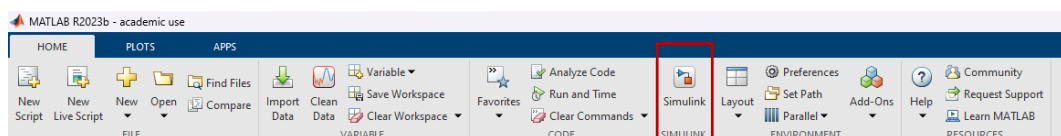


Figure 2: Opening Simulink from MATLAB.

<sup>1</sup> <https://www.mathworks.com/help/simulink/getting-started-with-simulink.html>

<sup>2</sup> <https://www.mathworks.com/videos/add-on-explorer-106745.html>

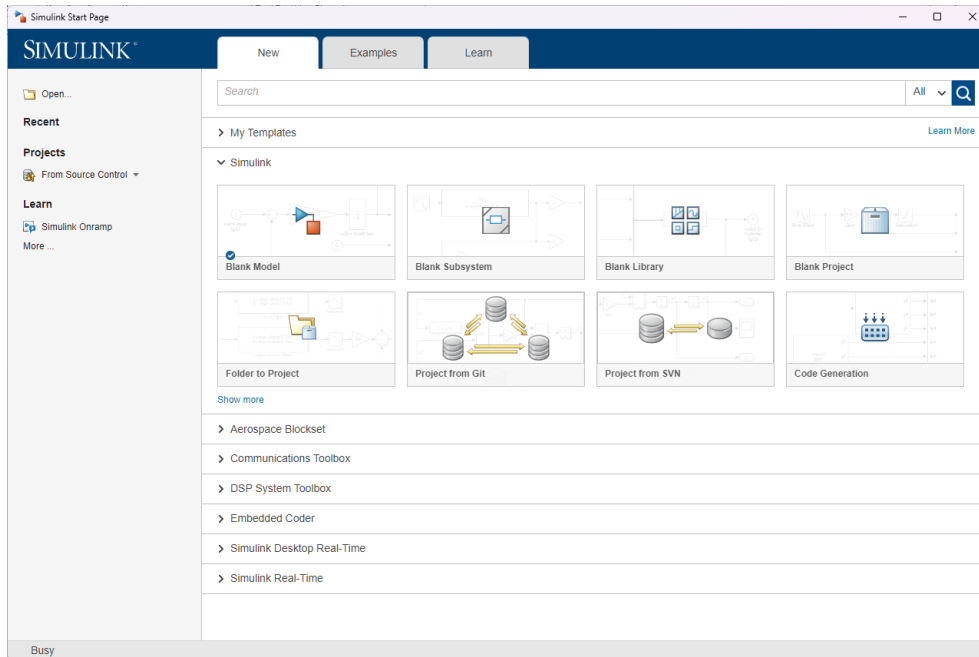


Figure 3: Simulink Start Page.

After selecting the correct icon, the Simulink Start Page will open as a new window. Once Simulink has been opened, new, recent projects, and examples can be opened. From this window, the [Simulink Onramp self-paced course](https://matlabacademy.mathworks.com/it/details/simulink-onramp/simulink)<sup>3</sup> can be launched. New Simulink users are strongly suggested to take this to get up to speed with the software features. Additionally, some Add-On packages provide ready-to-launch models (e.g. Aerospace Blockset, Communications Toolbox, etc.).

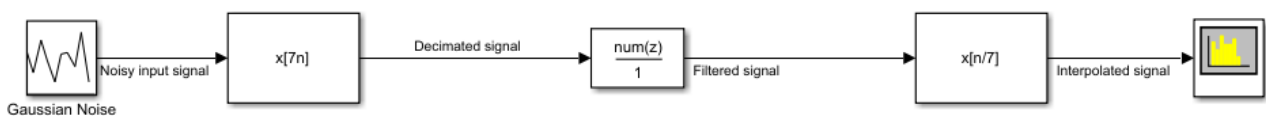


Figure 4: Simulink model example.

Simulink is based on blocks and the interconnection between them. Fig. 4 shows an example of Simulink model, which consists in blocks, signals and annotations. In general:

- **Blocks** are like MATLAB functions, they wrap and process a variable number of inputs and can have multiple outputs.

<sup>3</sup> <https://matlabacademy.mathworks.com/it/details/simulink-onramp/simulink>

- **Signals** consist in the lines connecting the blocks. These pass information along the model.
- **Annotations** are user-friendly comments or notes describing the model. These are ignored during computations but help users understand the processing steps.

### 3.1. Creating a Simple Model

The Simulink Onramp self-paced course provides a general introduction to most Simulink functionalities, and in this section a simple example on how to setup and run a model will be provided.

#### 1. From the start page, create a Blank Model.

Observe how the Simulink main interface opens up. From here, it is possible to insert blocks in the canvas and perform simulations. Take note of the Simulation Toolbar above the canvas (Fig. 5), which contains the commands required to start-up and run simulations.

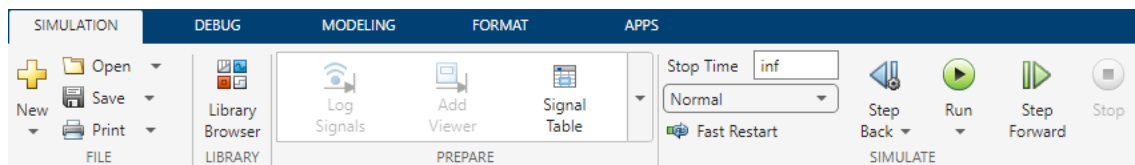


Figure 5: Simulation Toolbar.

#### 2. Create a Sine Wave block.

In this example we're going to simulate a signal directly in Simulink. We need to add the **Sine Wave** block from the **Simulink/Source** group. To add a block, users can click on the **Library Browser** in the Simulation Toolbar, or simply click on the canvas and start typing the name of the block to insert to open up the Quick Search tool. Once the block has been implemented in the canvas, it can be resized and moved at will.

#### 3. Inspect the Block Parameters

Double clicking on a block opens up its parameters. From here, it is possible to customize the behaviour of blocks. Fig. 6 shows the Block Parameters of the Sine Wave source block.

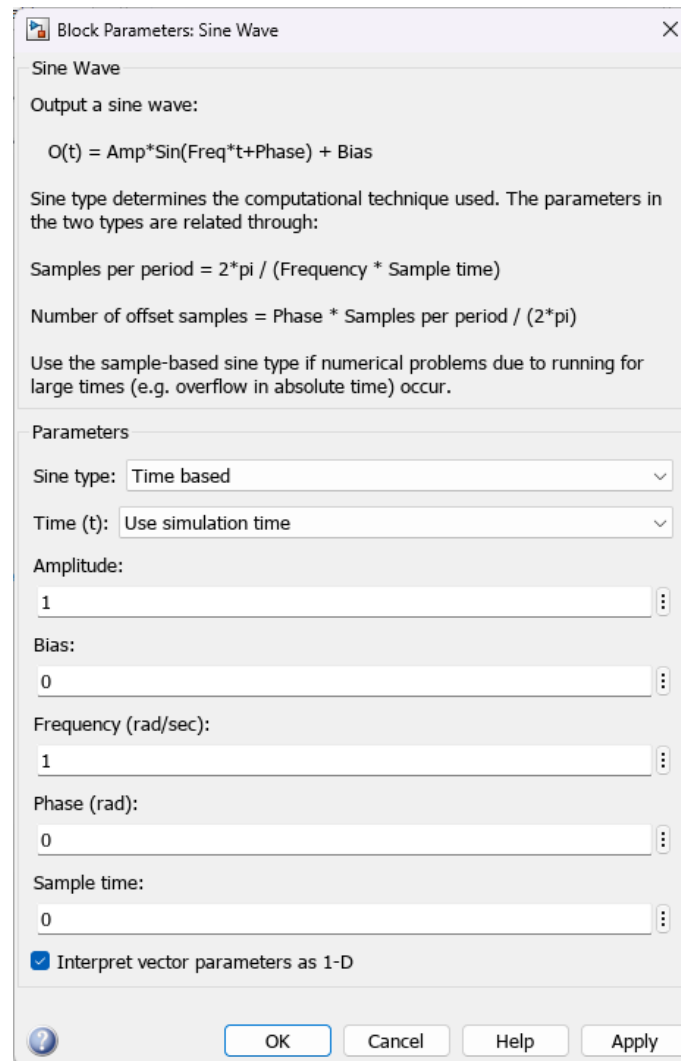


Figure 6: Sine Wave source block parameters.

Right clicking on blocks and selecting **Properties...** allows users to create block annotations. These are helpful in complex models to keep track of important settings during simulations. In this case, it is possible to select the **%<Frequency>** token to dynamically annotate the frequency of the source in the model.

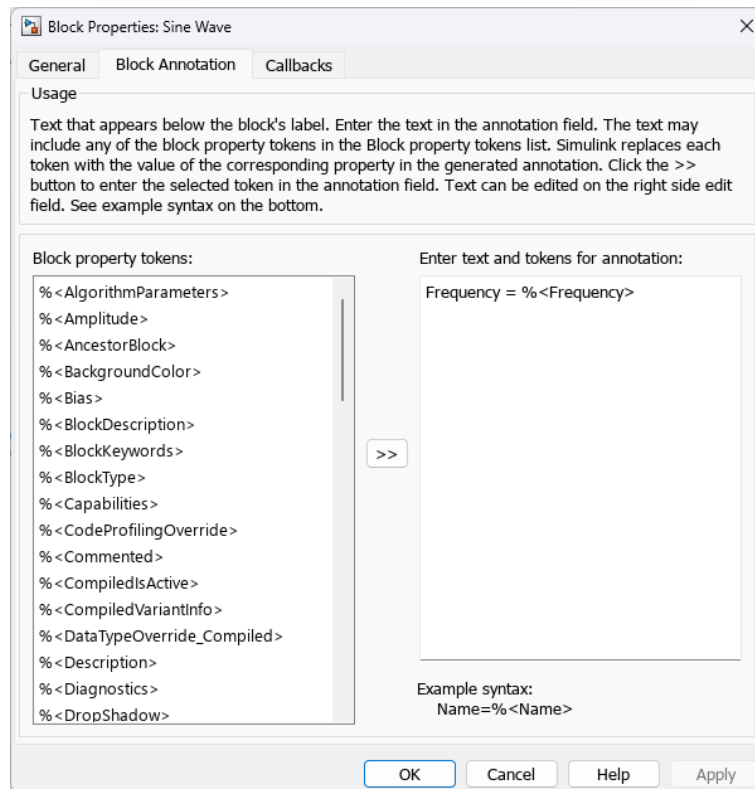


Figure 7: Sine Wave block annotations.

#### 4. Insert an Add Constant Block

The Add block can be selected in the Library Browser, or by typing “Add” in the canvas.

#### 5. Insert a Derivative Block

The Derivative block can be selected in the Library Browser, or by typing “Derivative” in the canvas.

#### 6. Insert a Scope Block

The Scope block can be selected in the Library Browser, or by typing “Scope” in the canvas.

#### 7. Insert a To Workspace Block

The To Workspace block can be selected in the Library Browser, or by typing “To Workspace” in the canvas.

## 8. Connect the blocks

Signals can be passed from block to block by left-clicking the output port and dragging the connection to the input of another block. If multiple inputs can be received (like for the scope block) Simulink will dynamically create a new port in the block if none are empty. In addition, signal paths can be forked by right-clicking and dragging them into different input blocks.

## 9. Inspect the Model

Once all blocks have been added and connected, the model should look like Fig. 8. (with the addition of annotations on the signals by double-clicking on the lines). To connect the blocks, left-click and drag the lines from the source block to the destination. To split a signal into two branches, right-click the line and drag it to its output. Note that for the **To Workspace** block the block parameters have been changed to name the output variable to “x\_dot”.

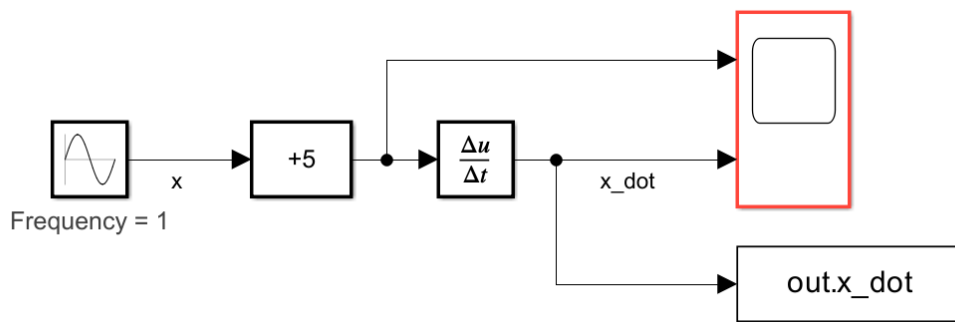


Figure 8: Final version of the example model.

## 10. Run the Model

To run models, users can simply click on Run on the Simulation Toolbar (or press F5). Once the model is running, it is possible to double click on the Scope block to visualize in real (simulation) time the signals going into it. Since the input to this block is a cosine signal in the interval  $\pm 1$  and a sine signal in  $5 \pm 1$ , right-click on the scope window to change the y limits of the axes. Doing so should make the scope visualize something like Fig. 9.



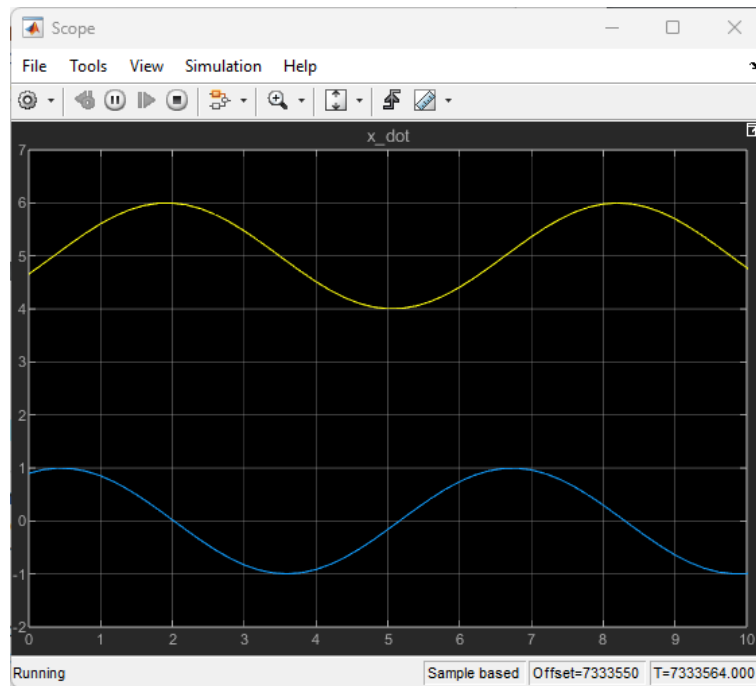


Figure 9: Example model output.

Additionally, it is possible to access `x_dot` directly in the MATLAB environment since it was saved and sent to the Workspace. This is stored as a `SimulationOutput` object and can be accessed like a structure.

```

Command Window

>> out

out =

Simulink.SimulationOutput:

    x_dot: [1x1 timeseries]

SimulationMetadata: [1x1 Simulink.SimulationMetadata]
ErrorMessage: [0x0 char]

fx >> |

```

Figure 10: `SimulationOutput` object in MATLAB.

### 3.2. Modelling General Transfer Functions

Simulink works perfectly to perform integration and differentiation, and therefore work with differential equations. In control systems, these are generally handled using the Laplace transforms. If one only knows the transfer function of a system (due to experimental data or analytical considerations), it is still possible to model the system itself in Simulink using the Transfer Fcn block. Fig. 11 shows the block and its parameters. In particular, the numerator and denominator coefficients are arrays of arbitrary length that can adapt to the requirements of the user.

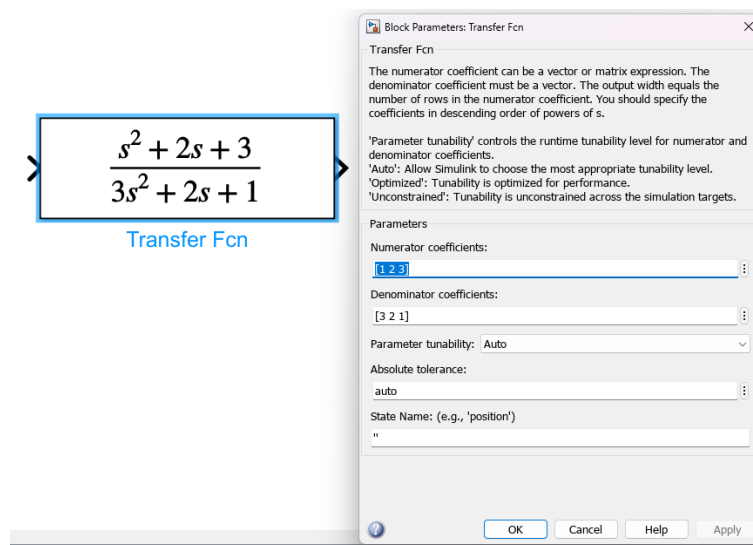


Figure 11: Transfer function block and parameters.

### 3.3. Discrete and Continuous Variables

Continuous variables are those that change smoothly over time. These variables are typically associated with physical systems described by differential equations. In Simulink, continuous variables are represented using continuous-time blocks, which simulate systems using continuous-time signals. Differently, discrete variables change with a fixed step. This usually coincides with the sampling time of a discrete input.

In general, it is possible to create models in Simulink where discrete and continuous variables interact. In this context, it is important to highlight what variables are discrete and which ones are continuous. In particular, when modelling physical phenomena, using continuous variables makes it possible to effectively analyse the behaviour of systems. On the contrary, discrete variables should be used when the response of systems (e.g. sensors, actuators) has a finite resolution in time.

As an example, note how Simulink implements the differentiation of input signals. In Fig. 12, the Derivative, Difference, and Discrete Derivative are all different, modelling continuous, discrete (general), and discrete (fixed sampling time) variables.

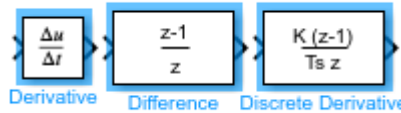


Figure 12: Continuous, discrete, and discrete (sample rate) blocks in Simulink.

### 3.4. Model and Solver Settings

It is possible to access the Model Settings from the Modeling Toolbar. This usually overlooked window allows the user to select and adapt the model and the solver to the simulation requirements. In particular, the Solver selection menu makes it possible to adapt the solver to the problem. A Fixed-step or discrete solver will run and integrate the model with a fixed resolution, while a Variable-step solver will be adapting to the changes in the system to avoid errors in the calculation. An example of this are stiff problems, where there are rapidly changing solutions that require adaptable and small time steps for numerical stability. In this case, a Fixed-step solver or an inadequate Variable-step solver (e.g. ode45) might produce incorrect results. For a comprehensive description of the solver settings refer to the official [documentation](https://www.mathworks.com/help/pdf_doc/simulink/simulink_gui.pdf)<sup>4</sup>.

<sup>4</sup> [https://www.mathworks.com/help/pdf\\_doc/simulink/simulink\\_gui.pdf](https://www.mathworks.com/help/pdf_doc/simulink/simulink_gui.pdf)

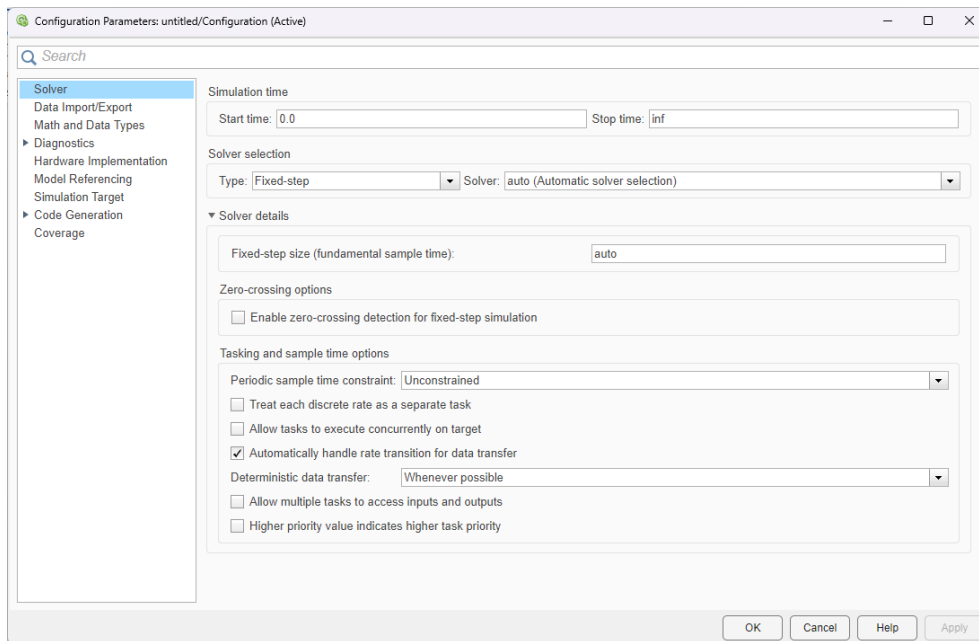


Figure 13: Solver Configuration Parameters.

### 3.5. Modelling Differential Equations

Simulink handles differential equations (and systems of) particularly well, especially in consideration of the feedback loops that can be implemented. In particular, Simulink can provide numerical solutions to systems that might not have an analytical one. In Fig. 14, the simulation of the trajectory of a free falling body is shown.

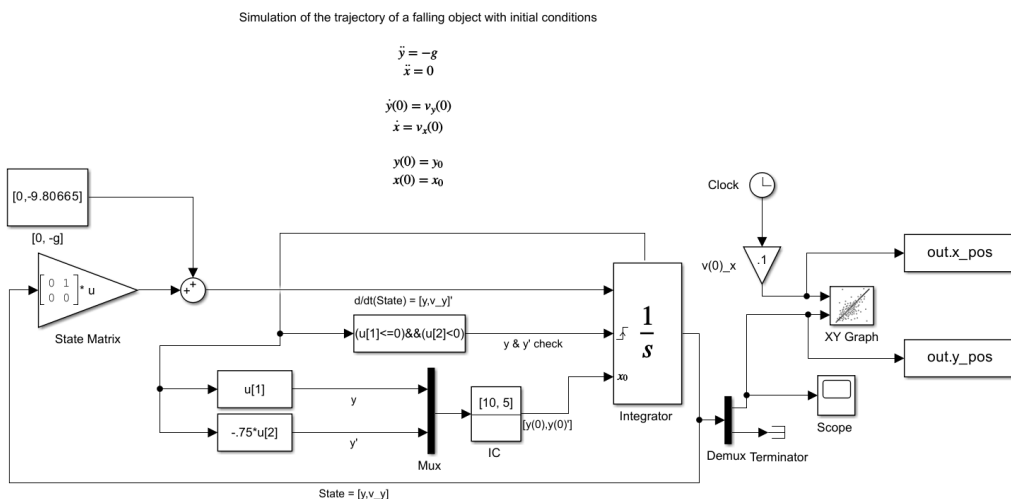


Figure 14: Simulation model of a falling object with initial conditions (FreeFallingStateMatrix.slx).

In this simulation, the object is considered to be set at a height  $h$  with an initial velocity  $v = [v_x, v_y]$ . Using Simulink, it is possible to integrate the trajectory in both coordinates and visualize the position and height as a function of time. Additionally, boundary conditions are set as a function of height and velocity in order to model the dissipation of energy when touching the ground. The system can be summarized as:

$$\begin{cases} \ddot{y} = -g \\ \dot{x} = v_x(0)t \\ \dot{y}(0) = v_y(0) \\ x(0) = x_0 \\ y(0) = h \end{cases}$$

## 1. Vertical Component

For the vertical component, the transfer function can be written in matrix form. In particular:

$$\frac{d}{dt} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} v_y(0) \\ -g \end{bmatrix}$$

Needless to say, the fundamental component of this model is the integrator, which serves as the block required to compute the state (position and velocity) of the object every step of the simulation. In this model, the integrator is set with an external initial condition (to input the vertical velocity and gravity acceleration) and an external reset. The reset is required to simulate the effect of impacting the ground when the computed velocity and position are negative. The three inputs of the integrator are as follow:

- Input 1: The first input is the derivative of the state. This is the integrand in the block, changing the input into the state to be computed.
- Input 2: The second input is the boolean check on the state. The incoming branch to the function block is the state of the object  $[y, \dot{y}]$  which are checked against the condition that  $y \leq 0$  AND  $\dot{y} < 0$ . When this condition is true, the integrator resets into the rising condition and takes the third input as a new initial condition.
- Input 3: The third and final input is defined as an initial condition block, which mean its values are taken as a first iteration, and then the input is passed from the incoming

branches. The Mux before the initial condition block creates an array of position and velocity from the single inputs of position and velocity. Take note of how the energy dissipation is simulated here, with an elastic coefficient of 0.75 that multiplies the input velocity. If the integrator is not reset, this branch is not used.

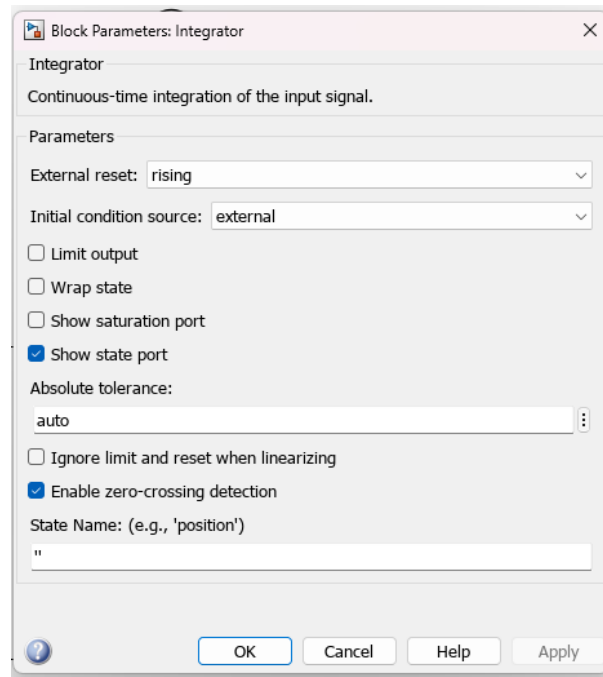


Figure 15: Integrator Block Parameters.

Outside of the integrator, the main part of the model serves to model the transfer function in matrix form presented at the beginning of this section, with the output of the integrator being  $[y, \dot{y}]$ .

## 2. Horizontal Component

The horizontal component can be implemented similarly to the vertical one. In this case to showcase the capabilities of Simulink, the implementation has been simplified. Given the simple equation  $x(t) = v_x(0) t$ , the simulation time is taken using the Clock block, and a Gain block serves as the multiplier.

## 3. Model Output

The output of models can be tuned to serve different purposes, from showing results, real-time simulations, or outputting data for further processing in MATLAB. In this case, the model passes the  $x$  and  $y$  coordinates to the MATLAB workspace as a structure with the  $x\_pos$  and  $y\_pos$

fields. Additionally, the signals are scoped inside of the model using the appropriate blocks. This allows users to inspect data quickly instead of iterating and switching between MATLAB and Simulink.

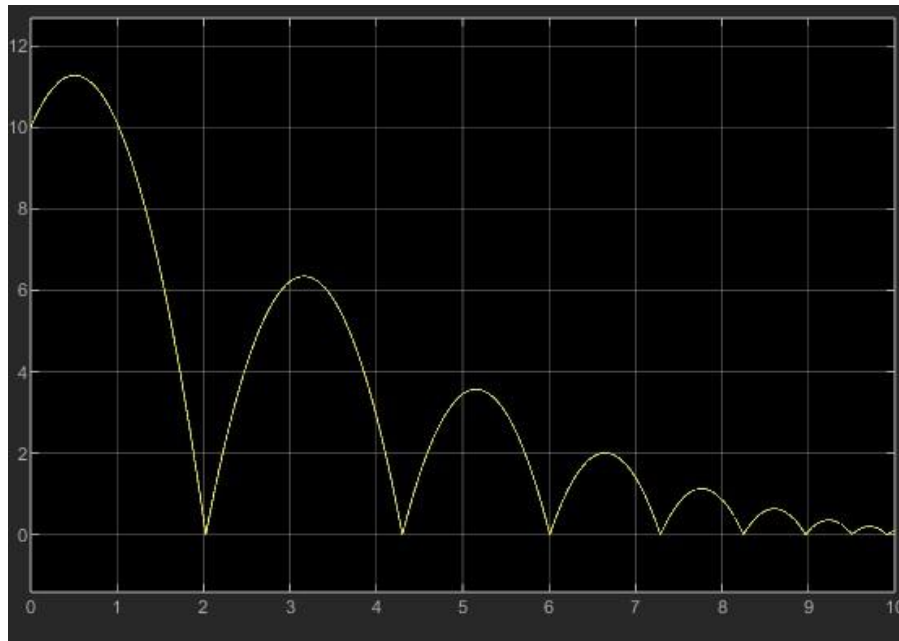


Figure 16: Model scope. Y component (y-axis) vs simulation time (x-axis).

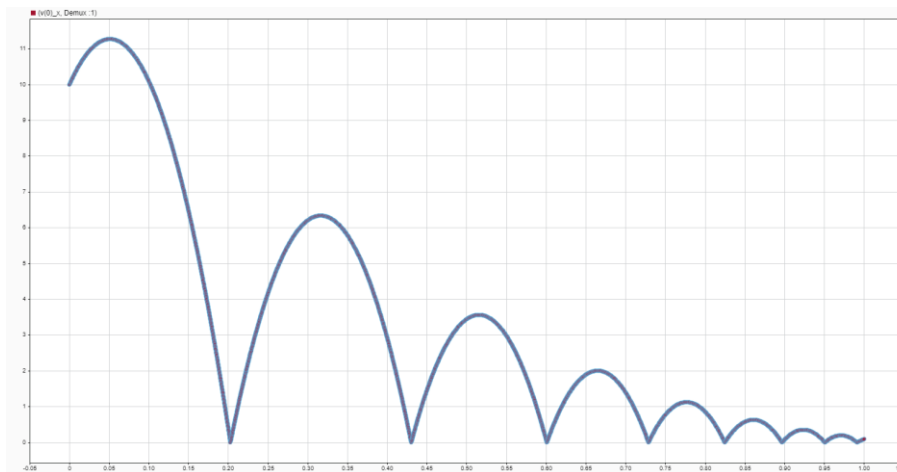


Figure 17: Model XY Graph. Y component (y-axis) vs X component (x-axis).

## 4. MATLAB-Simulink Interactions

MATLAB and Simulink can work together to streamline the processing pipeline. As shown in the model examples, Simulink can interface with the MATLAB workspace by getting and storing variables. In general, a variable **var** can be called within Simulink (e.g. the Gain of a block is set as **var**) or data imported through MATLAB can be passed as signals to Simulink (e.g. using the From Workspace block). Finally, it is worth noting that the Simulink GUI is not required to run simulations with it, as models can be run directly within MATLAB using the `sim()` command<sup>5</sup>.

## 5. Model Examples

- [MATLAB-Simulink Connection Example](#)
- [1st order ODE solver](#)
- [Simple Harmonic Oscillator](#)
- [Damped Oscillator](#)
- [Free Falling Object \(Beginner\)](#)
- [Free Falling Object \(Advanced\)](#)
- [Simulink Models Official Examples Library](#)

## 6. Where to go from here...

In this brief document we have seen how to navigate the Library Browser, insert and connect blocks, and setup the Solver for our model. As self-paced courses and official documentation are already available to users, this introductory guide serves mainly as a starting point to speed-up the understanding of the very fundamental concepts of Simulink, while also providing information on where to get further training. Novice users can start the [Simulink Onramp](#)<sup>6</sup> self-paced course, and the [Simulink Fundamentals](#)<sup>7</sup> if possible. A great way to explore Simulink is by browsing the [Simulink Models Example List](#)<sup>8</sup>, which can all be imported and run by opening their page and selecting “Copy Command”. These commands (e.g. `openExample('simulink_general/sldemo_bounceExample')`) can be run in the MATLAB command window to automatically download and open the model to explore and test. Intermediate and advanced users can refer to the official documentation to learn more about Simulink potential. The [Simulink: Getting Started Guide](#) contains all basic information about the software. The [Solving Differential Equations Using Simulink](#)<sup>9</sup> book provides clear and comprehensive examples on how to simulate dynamical systems for users interested in that. Else, users looking to simulate

---

<sup>5</sup> <https://www.mathworks.com/help/simulink/slref/sim.html>

<sup>6</sup> <https://www.mathworks.com/help/simulink/slref/simulinkonramp.html>

<sup>7</sup> <https://www.mathworks.com/help/simulink/slref/simulinkfundamentals.html>

<sup>8</sup> [https://www.mathworks.com/help/simulink/examples.html?category=getting-started-with-simulink&tid=CRUX\\_topnav](https://www.mathworks.com/help/simulink/examples.html?category=getting-started-with-simulink&tid=CRUX_topnav)

<sup>9</sup> [https://people.uncw.edu/hermanr/mat361/simulink/ODE\\_Simulink.pdf](https://people.uncw.edu/hermanr/mat361/simulink/ODE_Simulink.pdf)



specific applications can consider using the [suggested books](#)<sup>10</sup> as reference. Finally, keeping the [Simulink: Graphical User Interface](#)<sup>11</sup> and [Simulink: User's Guide](#)<sup>12</sup> at hands can be incredibly beneficial to quickly debug and understand contextual menus and parameter settings (e.g. page 3-2 of Simulink: Graphical User Interface describes all the details about the Solver settings).

## 7. Additional Resources

Examples Library

[Simulink Models Official Examples Library](#)

Interactive Courses:

[Simulink Onramp](#)

[Simulink Fundamentals](#)

Documentation:

[Simulink: Getting Started Guide](#)

[Simulink: Graphical User Interface](#)

[Simulink: User's Guide](#)

[Simulink: Reference](#)

[Get Started with Simulink](#)

[Simulink Models](#)

[General Considerations When Building Simulink Models](#)

[Model Differential Algebraic Equations](#)

[How to draw ODEs in Simulink](#)

Video:

[Getting Started with Simulink, Part 1](#)

[MATLAB and Simulink Livestreams](#)

Other resources:

[Introduction to Simulink](#)

---

<sup>10</sup> <https://www.mathworks.com/academia/books/search.html?q=simulink&page=1>

<sup>11</sup> [https://www.mathworks.com/help/pdf\\_doc/simulink/simulink\\_gui.pdf](https://www.mathworks.com/help/pdf_doc/simulink/simulink_gui.pdf)

<sup>12</sup> [https://www.mathworks.com/help/pdf\\_doc/simulink/simulink\\_ug.pdf](https://www.mathworks.com/help/pdf_doc/simulink/simulink_ug.pdf)

[An Introduction to Using Simulink](#)

[Solving Differential Equations Using Simulink](#)