




## CHAPTER 5

# Simulink Modeling Essentials

Simulink<sup>1</sup> is the graphical programming package that works in association with MATLAB and interacts with it as one combined package. It is employed for modeling, simulating, and analyzing dynamic systems, control algorithm development, and so forth. It supports linear and nonlinear systems, continuous and discrete systems, and multirate systems. With Simulink, you can model myriad types of systems, processes, and problems, and you can use top-down and bottom-up approaches.

The Simulink package, like MATLAB, is expandable. By using its standard blocks, you can develop your own library of blocks and subsystems and add to and expand existing Simulink libraries. In your Simulink models, you can combine continuous systems with discrete ones. One of the main advantages of using the package from a user's perspective is that it is much easier to model systems via block diagrams because it doesn't require any preliminary programming skills or experience from users. In Simulink, you simply drag, drop, and connect blocks, and of course, adjust parameters, solvers, and other components in the model. Another advantage of the Simulink package is that its models can work interactively with MATLAB and can be manipulated and executed from the MATLAB Command window and the M/MLX-files and function files.

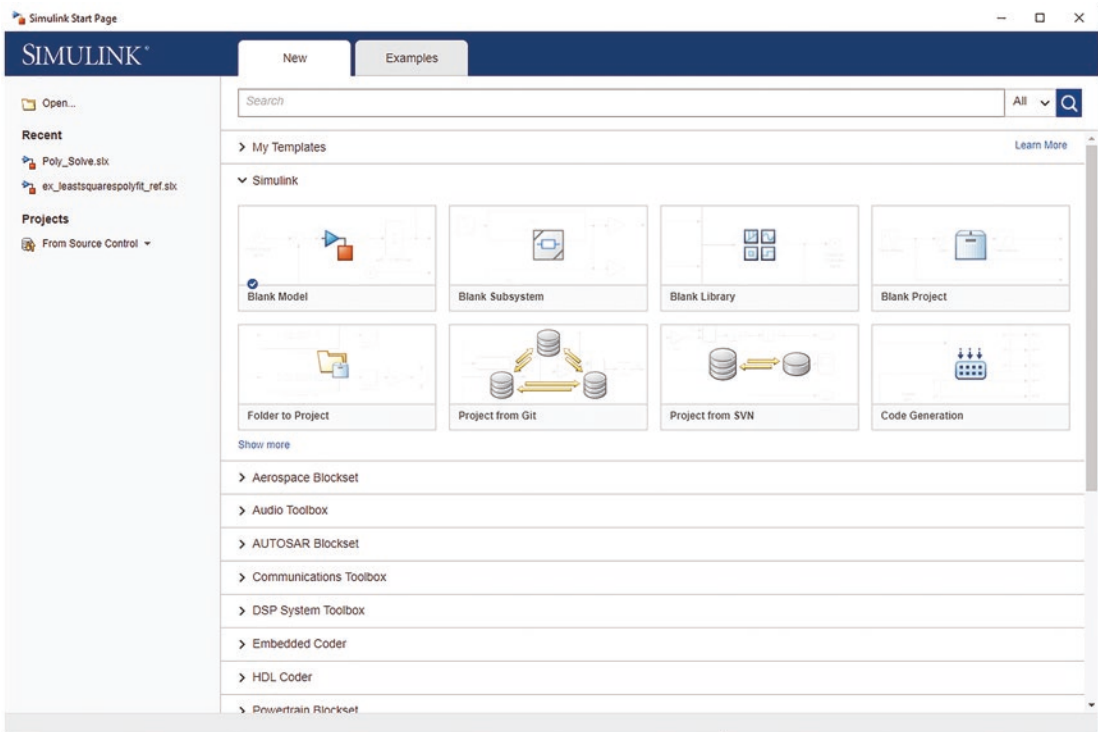
## Simulink Modeling

To launch the package from MATLAB, you just type `>> simulink` in the Command window, click the Simulink icon  in the menu panel, or click  and  Simulink Model icons. The window in Figure 5-1 will pop up. From the Simulink startup window, you can

---

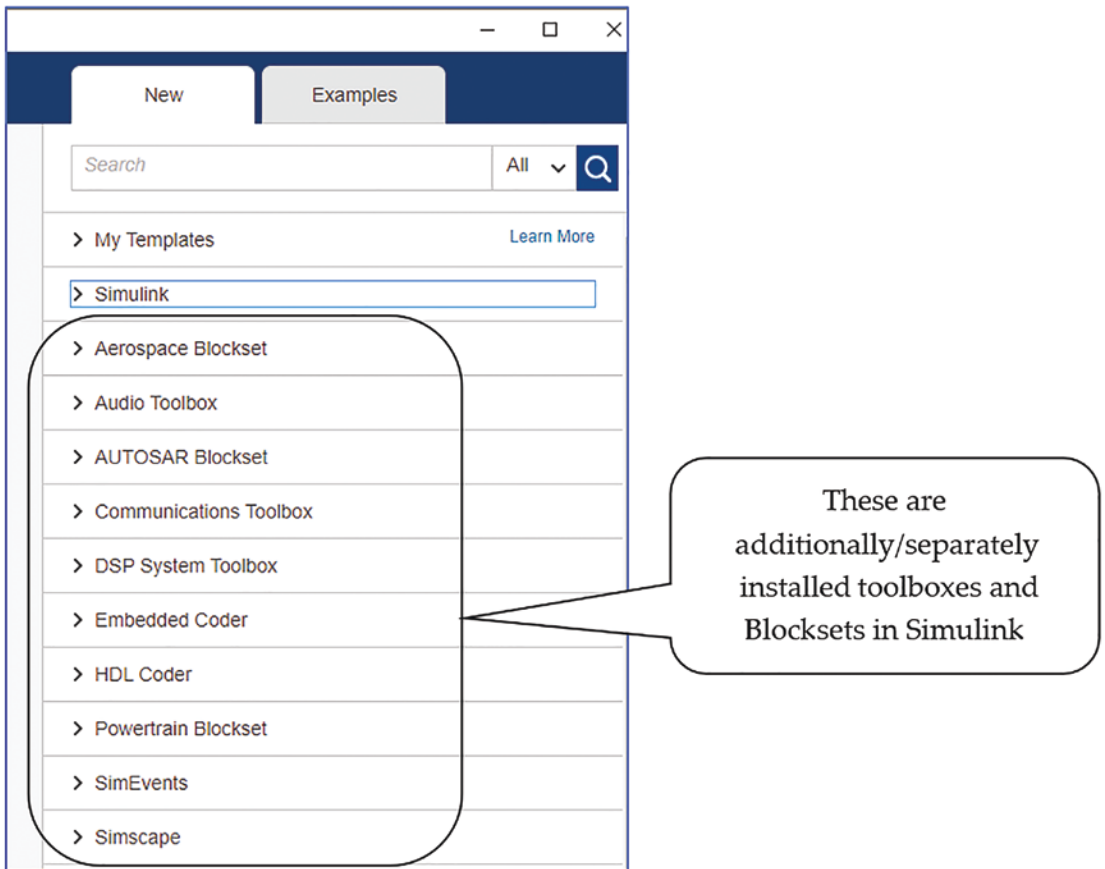
<sup>1</sup> Simulink is a registered trademark of MathWorks Inc.

open existing models (recently worked on ones) from the right-side pane (Open) or create a new model by clicking Blank Model or the other options there. Also, from the Examples tab, users can open, study, and change existing examples. There are dozens of examples from different areas of engineering, physics, computing, image processing, code generation, and so forth.



**Figure 5-1.** Simulink’s startup window

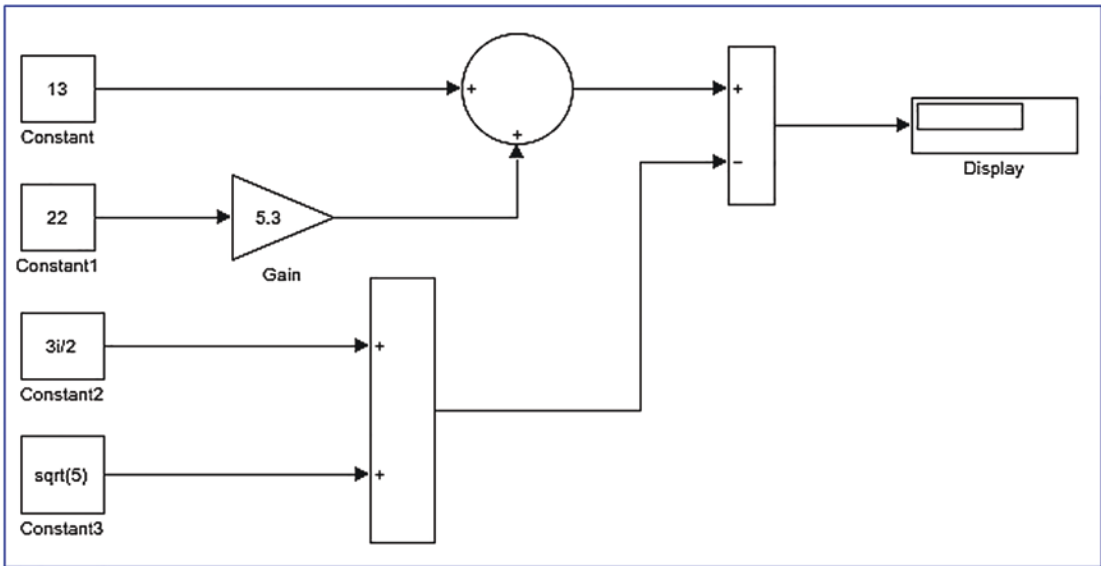
Figure 5-1 shows the default startup window of the Simulink package. Note that Simulink is a stand-alone package, and there are a few toolboxes and add-ons (see Figure 5-2) that can be installed. All of the blocks of the additional libraries and add-ons will be accessible once they’re installed from the Simulink Library browser.




**Figure 5-2.** Simulink's startup window and additional toolboxes installed in it

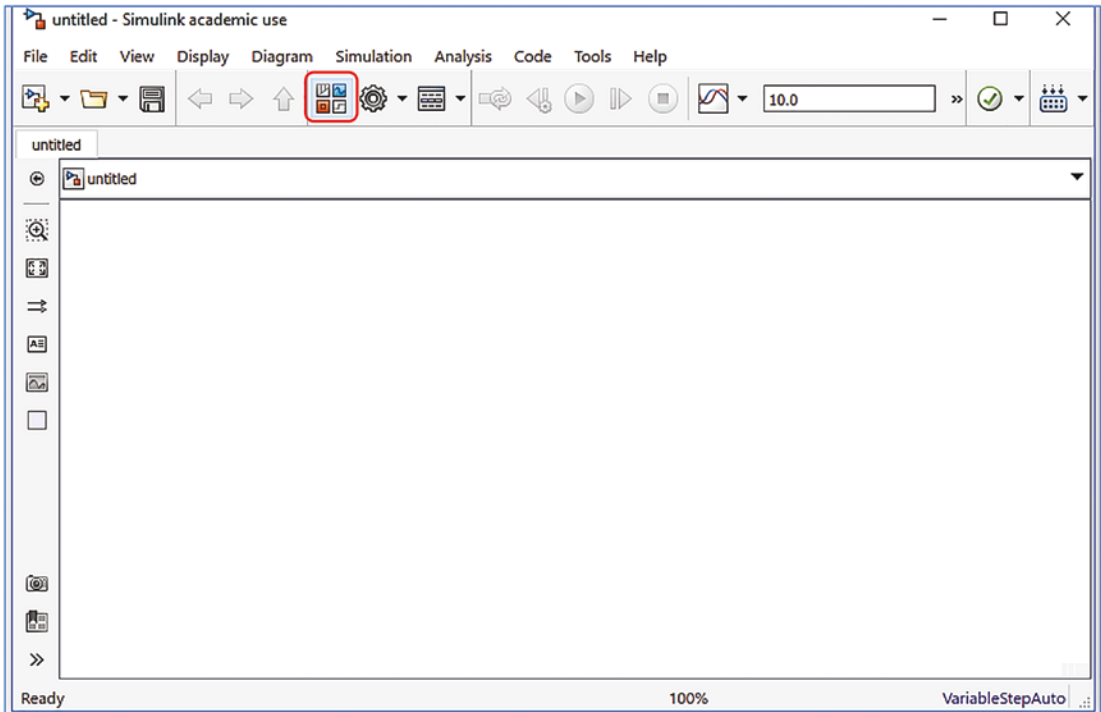
## Example: Arithmetic Calculations

Let's look at a simple example to demonstrate how to build a Simulink model that computes the addition, multiplication, and subtraction of four different scalar numbers. The complete model is shown in Figure 5-3. It's composed of Constant, Gain, Sum, and Display blocks. The scalars entered in constant blocks are 13, 22,  $3i/2$ , and  $\sqrt{5}$ . When this model is executed, it displays its computed results in the Display block. Note that all of the blocks employed in this model are available in the Simulink Library.





**Figure 5-3.** Complete Simulink model example with arithmetic calculations

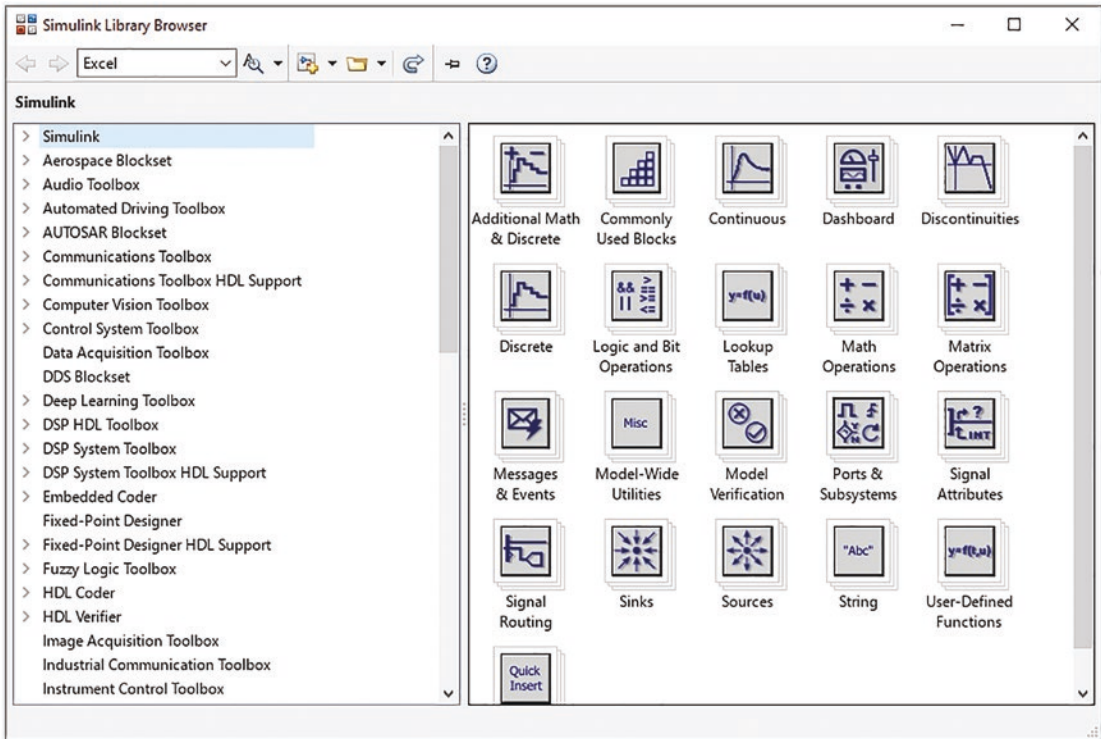
To open a blank model, click Blank Model, as shown in Figure 5-1. Note that a blank model can also be opened from the already opened Simulink window by clicking  or by pressing Ctrl+N on the keyboard. After clicking the Blank Model icon (see Figure 5-1), the model shown in Figure 5-4 opens. This is called untitled.slx by default, just like MATLAB's M/MLX-files.



**Figure 5-4.** *New Simulink model window (untitled by default)*

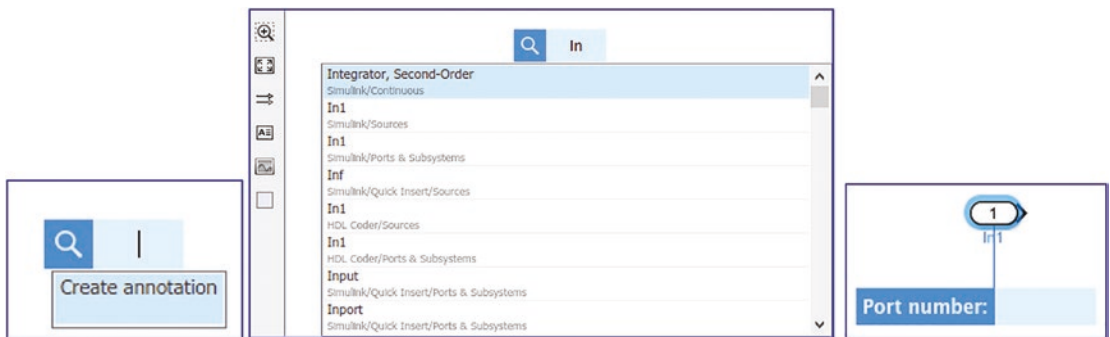
From the opened blank model window, click  (the Simulink Library icon). That will open a Simulink Library window, as shown in Figure 5-5. It must be noted that the toolboxes that are available in the library are defined by which toolboxes are installed and which user developed/created custom libraries are installed. If necessary, you can also create a new blank model by clicking the New Model icon  in the Library Browser (see Figure 5-5). This also creates a blank model window.

The Simulink Library (see Figure 5-5) looks different in different versions, but most of its general blocks function using the same principles. In this regard, it is worth pointing out that the package has been developed and subject to constant improvement with novel add-ons/blocks, and therefore, the models created in recent versions of Simulink are not fully compatible with its older versions. All of models are forward compatible; in other words, models created in older versions of Simulink work in later versions of the package.



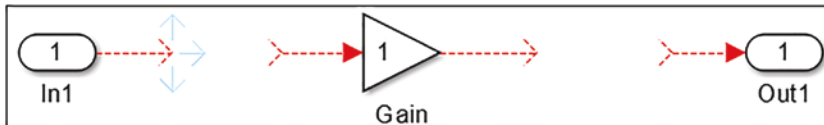
**Figure 5-5.** Simulink Library browser

Let’s build a new model where you drag and drop all the necessary blocks from the library. Dragging and dropping a block into a new model area is the most common practice in Simulink-based modeling or programming; however, in recent versions of Simulink (starting from MATLAB/Simulink 2018a), there is an alternative way of obtaining blocks within a model area. It is also possible to obtain any block by double-clicking a desired spot of the model area. For example, if you want to use an Input block, you can double-click and type in the Create Annotation search box. The prompt drop-down options will appear, as shown in Figure 5-6. From the drop-down option, you can select the desired block name. There is another optional step to specify the port number in this case. For other blocks, this option differs.



**Figure 5-6.** Creating/obtaining blocks within the model area by using the annotation option

Once all of the necessary blocks are placed in the model area, they need to be connected. There are two ways to connect blocks in the Simulink model window. You can connect a block in the blank model window by clicking the left mouse button and holding the Ctrl key on the keyboard. You then click another block to connect the two. An alternative way to link blocks or connect signals in between blocks is to drag a signal arrow (see Figure 5-7) to the block you want to connect.





**Figure 5-7.** Connecting blocks

Some blocks have input and output ports, some have only input ports, and others have only output ports. You can only connect signals from their output port to another block's input port. In other words, it is not possible to connect signals from output port to output port or input to input.

---

**Note** Blocks can be resized easily. You click the block to be resized and drag it from one of the four corners of the rectangular boundary around the block.

---

Before you start working with Simulink modeling, you must adjust one tool to ease the process of working with the library of blocks. To keep the Simulink Library on top of all the model windows, the  icon must be clicked to a position of “on top,” which looks like this: . This is done with a single click.

---

**Note** To keep the Simulink Library on top of all the windows, including the model window, click the “stay on top”  icon.

---

Moreover, at the beginning you need to work within the Simulink Library. That can be accessed by clicking the + before Simulink. The Simulink Library contains a number of block sets grouped into Commonly Used Blocks, Continuous, Dashboard, Discontinuities, Discrete, Logic, and Bit Operations. Let’s look at several examples to explore the modeling tools and aspects in the Simulink environment.

## Example: Modeling Simple Arithmetic Operations

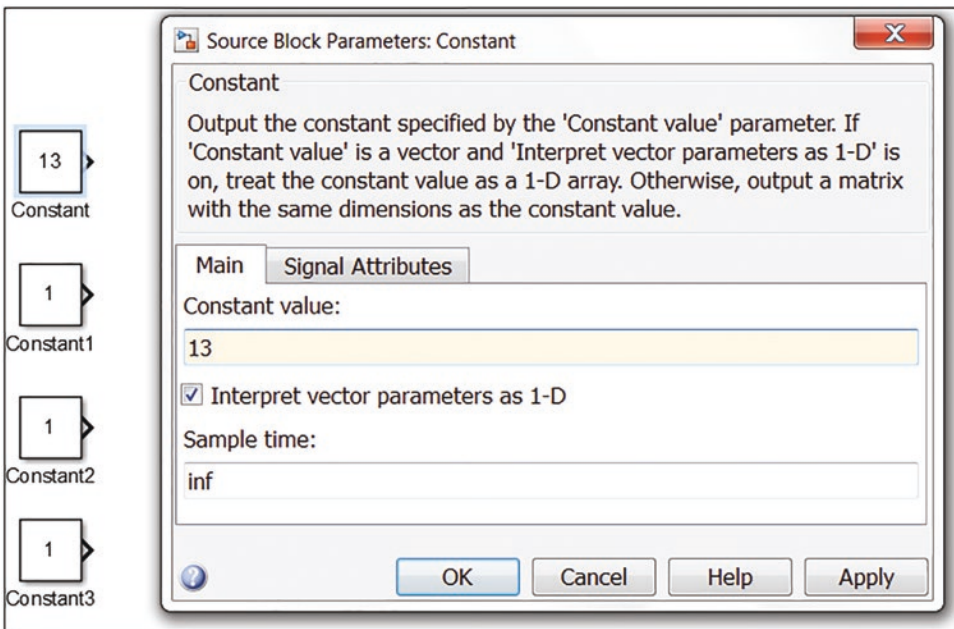
Let’s compute simple arithmetic operations using these +, -, /, \*,  $\sqrt{\quad}$ , etc., operators to compute this  $13 + 22 * 5.3 - (3i / 2 + \sqrt{5})$  and display the result.

To model these arithmetic computations of the exercise, you need the following Simulink/Commonly Used Blocks from the Library: four Constant blocks, three Sum blocks, one Gain block, and one Display block from Simulink/Sinks to output the computation results.

1. Drag and drop all of these blocks in the model area.
2. All of the Constant blocks’ constant values must be changed according to the given task (13, 22,  $3i/2$ ,  $\sqrt{5}$ )—see Figure 5-8. Similarly, add one Gain block (for 5.3) by double-clicking each block, one after another (see Figure 5-9).
3. Link a Constant block called Constant (13) to a Sum block with the + sign. You do this by clicking the block and holding the Ctrl key on the keyboard. Then you click a Sum block.
4. Link a Constant block called Constant1 (22) to a Gain block, which is linked to the Sum block with the + sign, as shown in Figure 5-6.



5. Link a Constant block called Constant2 ( $3i/2$ ) to a second Sum block. This rectangular shape can be changed by double-clicking a Sum block with the + sign. Similarly, to this Sum block, the Constant block called Constant3 ( $\sqrt{5}$ ) is linked using a + sign.
6. Connect two Sum blocks to a third Sum block with the + and - signs. Subsequently, link the third Sum block to the Display block.

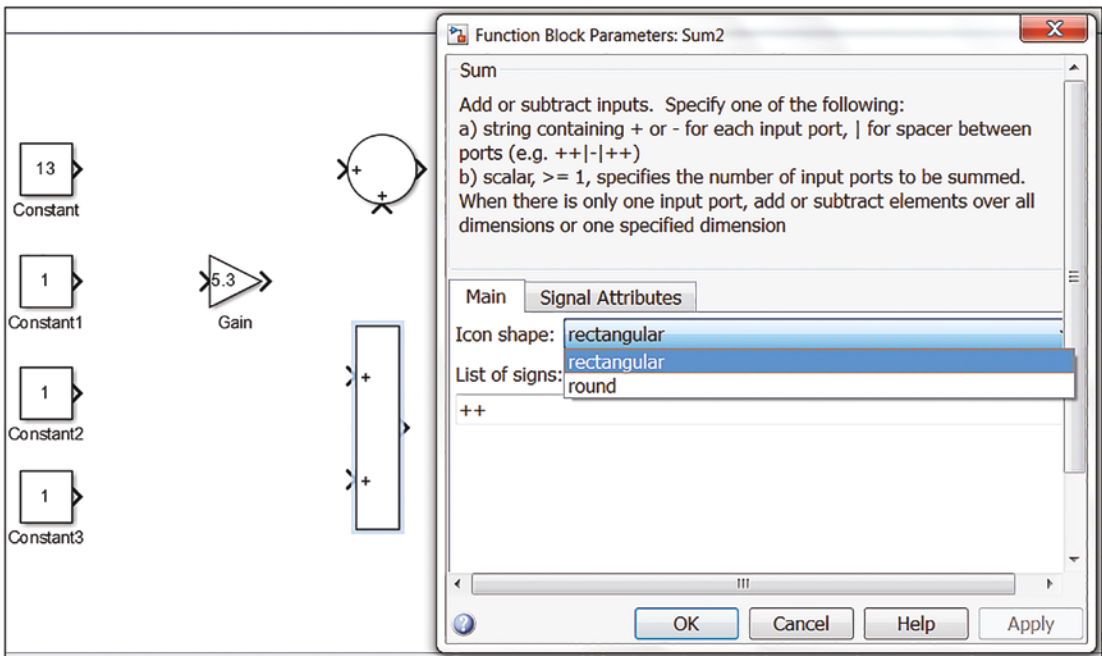


**Figure 5-8.** Blocks

---

**Note** Any block can be copied numerous times by holding Ctrl and clicking the block and then dragging the block over any spot in a model space. This method is faster and more efficient than dragging the same block from the library.

---




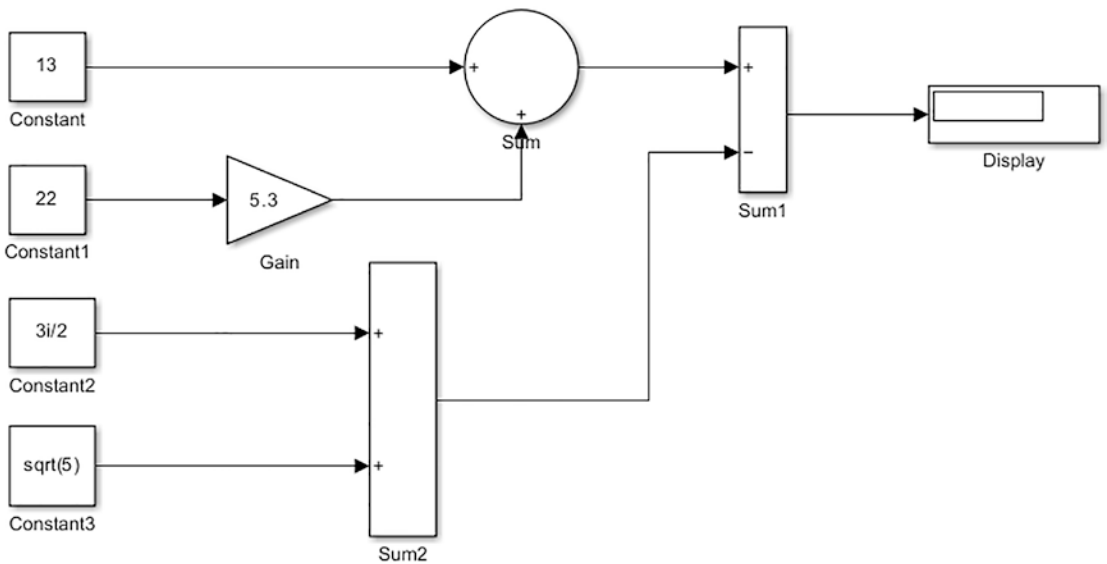
**Figure 5-9.** Altering the Sum block’s signs and icon shape

That completes our simple computation Simulink model.



This model is saved with the filename of `Ex1_Arithmetic_operations.mdl`.

**Note** In the latest versions of Simulink, models can be saved either in `*.mdl` (backward-compatible file format) or in `*.slx` (supported in later versions). Any Simulink model can be exported to previous versions via `File > export model to > previous version`. For the “Save as type” option, you select the appropriate version from the drop-down.

To see the computation results, click the Run button  or press the `Ctrl+T` keys on the keyboard. The results are displayed in the Display block. Figure 5-10 shows the complete model.



**Figure 5-10.** Completed Simulink model called *Ex1\_Arithmetic\_operations.mdl*

**Note** To view a full-screen model, just press the spacebar on the keyboard. To zoom in and out, press the Ctrl++ and Ctrl+- keys (in later versions) on the keyboard, or click  to get a Fit view. You can also use  to zoom in.

Note that you type in  $\sqrt{5}$  as a value of the Constant block called Constant3 by `sqrt(5)` and  $3i/2$  as `3i/2` since Simulink (like MATLAB) recognizes imaginary numbers via the letters `i` and `j` automatically. Any Simulink model can be simulated/executed from MATLAB via the workspace or within any M-file using the `sim()` command. The simulated model must reside in the current working directory. For instance, you can run the previous model with this command:

```
>> sim('Ex1_Arithmetic_operations.mdl')
```

**Note** You can alter the name of any block by clicking its name tag. You can alter the properties and parameters of any block by opening the property window (by double-clicking it) and inserting the necessary changes or selecting necessary options. For instance, in a display block, you can format the data as short, long, short\_e, long\_e, hex, etc.

# Performing Matrix Operations

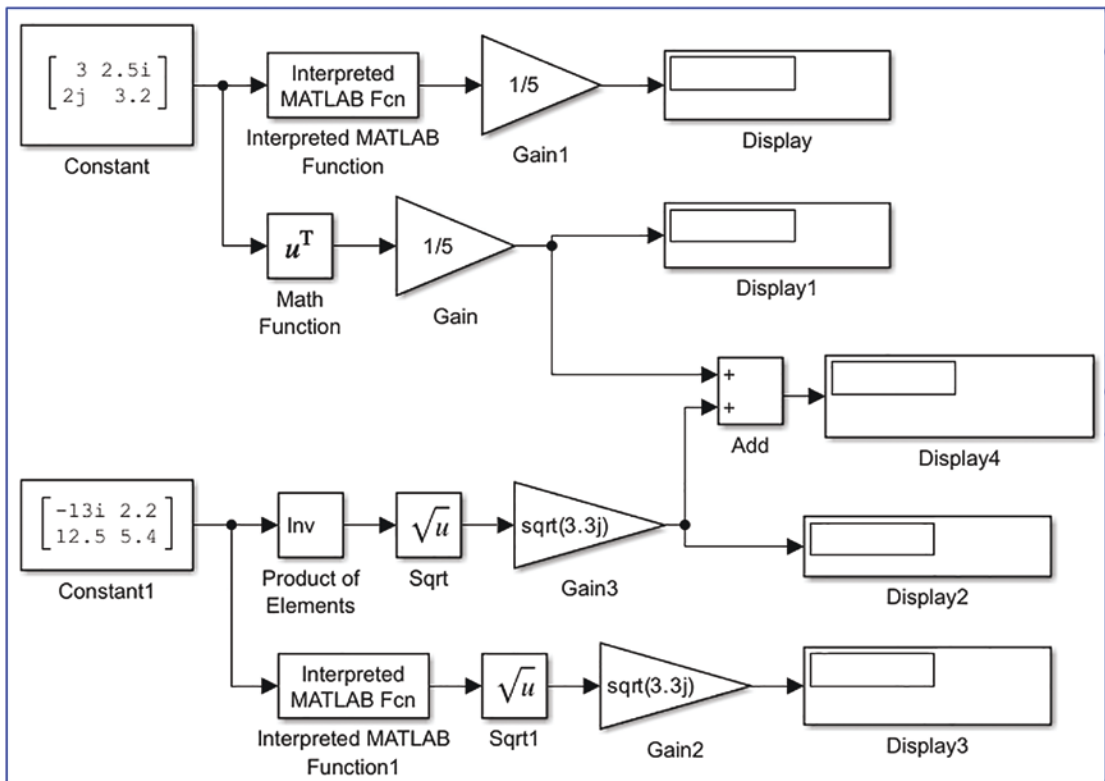
Let's build a Simulink model that performs the following matrix operations:

$$\begin{bmatrix} 3 & 2.5i \\ -2i & 3.2 \end{bmatrix}^T \frac{1}{5} + \begin{bmatrix} \sqrt{-3i} & \sqrt{2.2} \\ \sqrt{2.5} & \sqrt{5.4} \end{bmatrix}^{-1} \sqrt{3.3i}$$

Figure 5-11 shows the complete model. The model performs matrix operations, such as inverse, square root, transpose, sum, and division by a scalar (real and imaginary numbers). The following matrix operations (transpose and inverse) are performed by employing the Interpreted MATLAB Function block:

$$\begin{bmatrix} 3 & 2.5i \\ -2i & 3.2 \end{bmatrix}^T, \begin{bmatrix} \sqrt{-3i} & \sqrt{2.2} \\ \sqrt{2.5} & \sqrt{5.4} \end{bmatrix}^{-1}$$

Math Function and Product of Elements blocks are compared with the computed results and shown via Display blocks.



**Figure 5-11.** The complete model: example 2, matrix operations

To model this exercise, follow these steps:

1. For this model, you need two Constant blocks, four Gain blocks, five Display blocks, an Add block from the Simulink/Commonly Used Blocks library, two Sqrt blocks, one Math Function block, one Product of Elements block from the Simulink/Math Operations library, and two MATLAB Interpreted Function from the Simulink/User-Defined Functions (see Figure 5-12).
2. Enter the elements of two matrices into two Constant blocks as  $\begin{bmatrix} 3 & 2.5j \\ 2j & 3.2 \end{bmatrix}$  (see Figure 5-13) and  $\begin{bmatrix} -3i & 2.2 \\ 2.5 & 5.4 \end{bmatrix}$ . These will be connected with two separate Math Function blocks, one of which is Math Function (transpose of a matrix) and the other is Product of Elements (inverse of a matrix). These two blocks are edited accordingly; for instance, to obtain a transpose operator of the Math Function block, a Function type is chosen to be a transpose from drop-down options. In the Product of Elements block, the “number of inputs” option is changed to be a division (/), and the multiplication option is selected to be a `matrix(*)` multiplication.
3. Link the Math Function (transpose) block to the Gain block (1/5), which is subsequently linked to the Sum block. The Product of Elements block is connected to the Sqrt block, which is linked to the Gain block ( $\sqrt{3.3i}$ ). Finally, signals from the Gain block (1/5) and the Gain block ( $\sqrt{3.3i}$ ) are connected with the Sum block (see Figure 5-13), which is linked to the Display block.

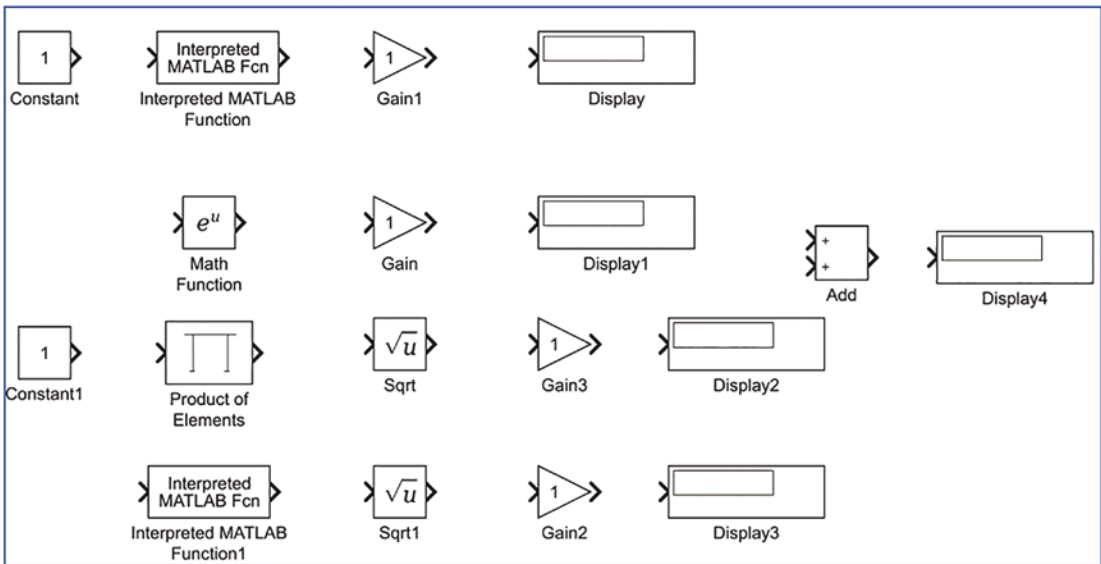


Figure 5-12. All blocks necessary for this model

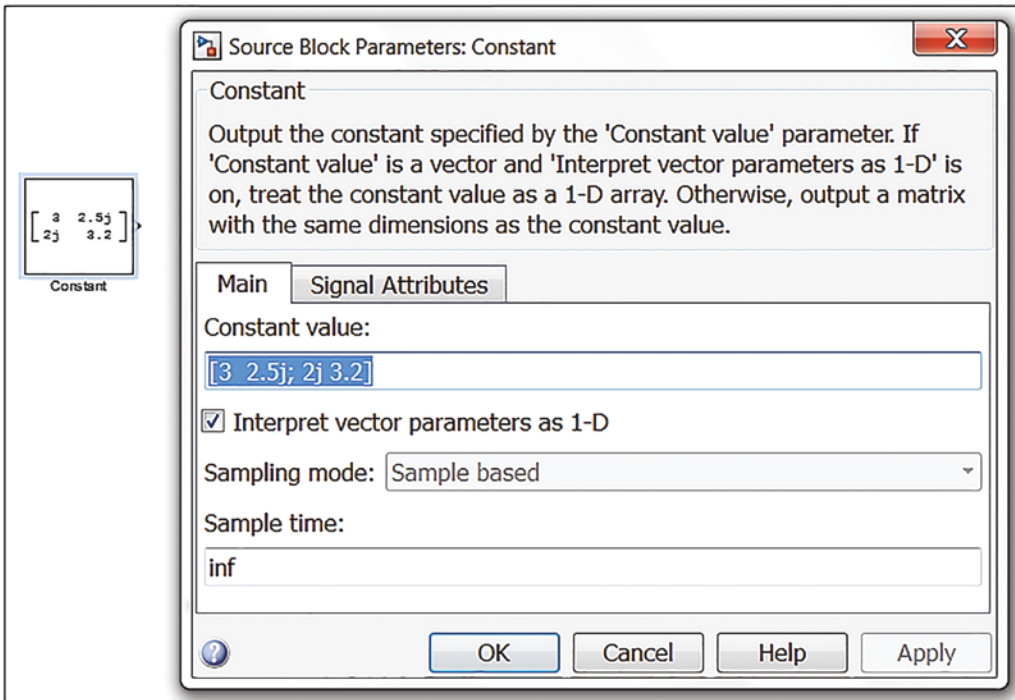
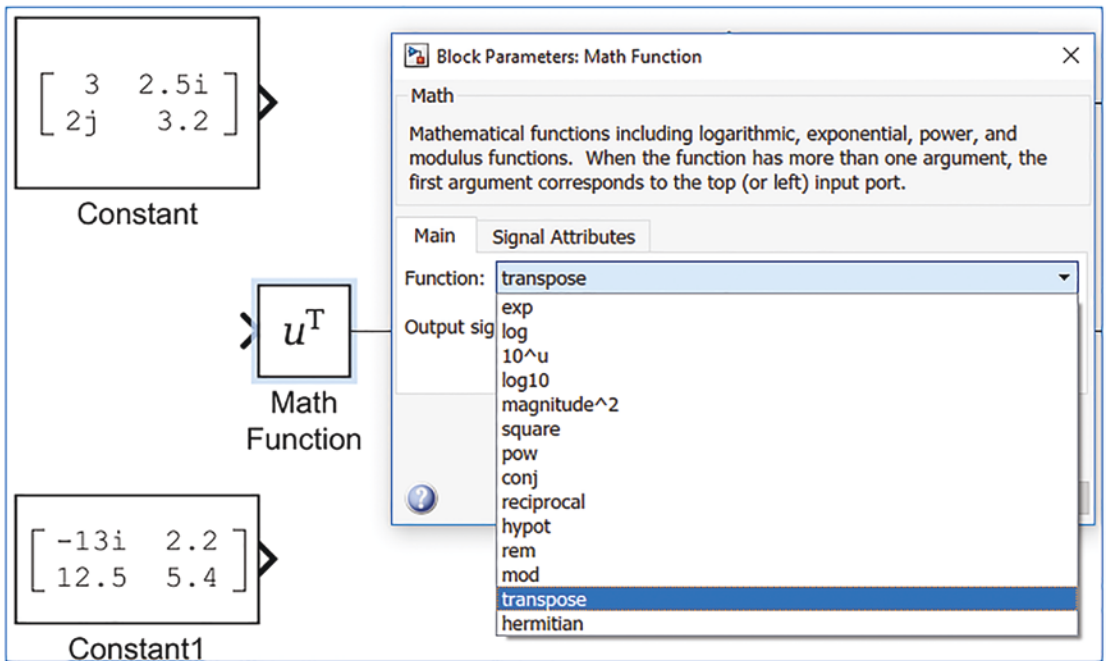


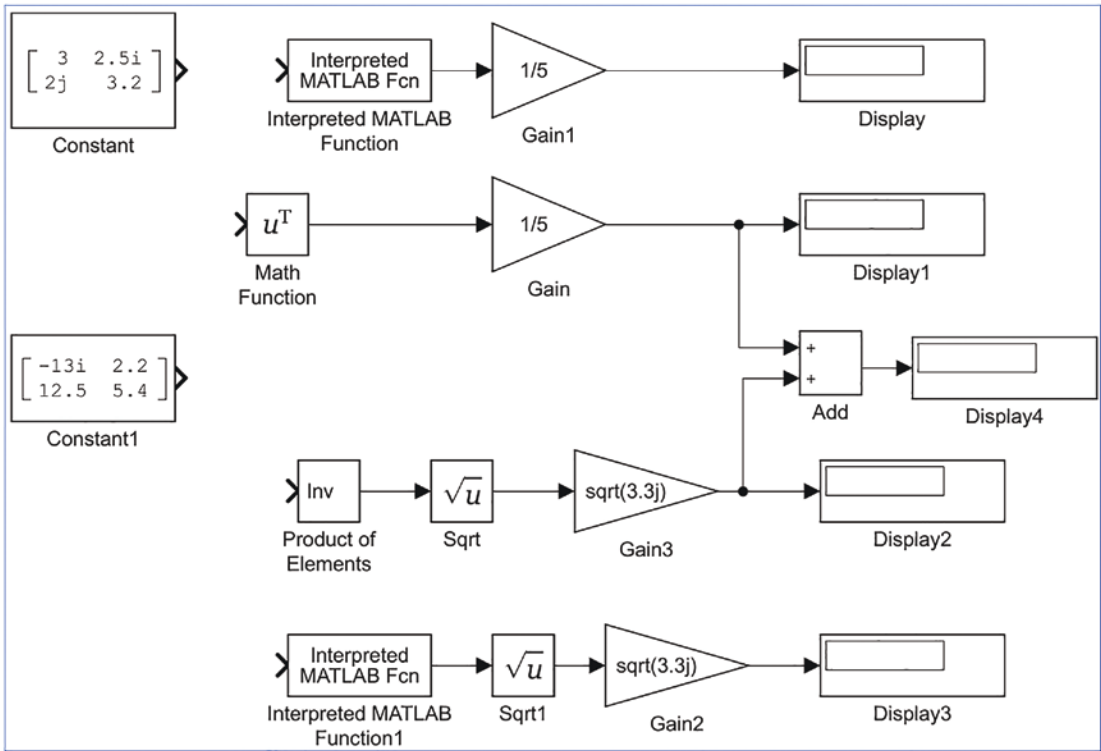
Figure 5-13. Inputting matrix elements into the Constant block's constant value

**Note** In Simulink modeling, you can attain the same results by altering the properties and parameters of blocks. For instance, matrix inverse can be obtained by inserting the MATLAB function called `inv(u)` in the interpreted MATLAB Function block.

Matrix transpose is obtained from the Math Function block's options, as shown in Figure 5-14.



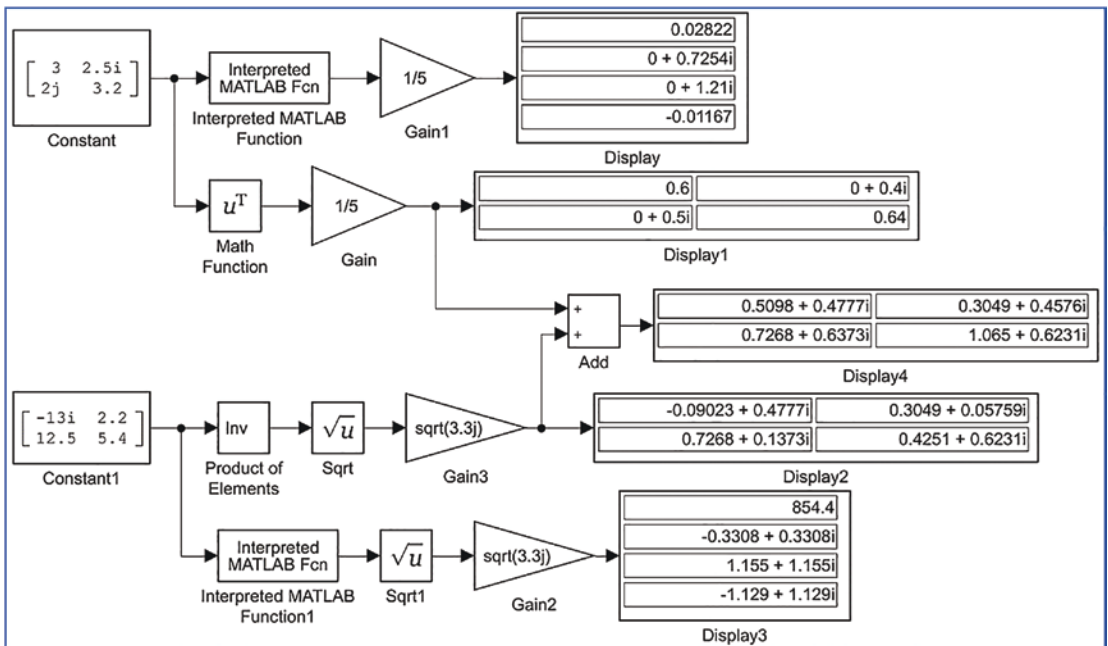
**Figure 5-14.** Selecting the Function type in the Math Function block



**Figure 5-15.** All blocks with adjusted options and entered values/elements

Note that there are two ways to obtain the inverse of matrices demonstrated in the completed model, as shown in Figure 5-16. By running the completed model, you can see that the results in both matrix inverse operations are the same. Note that the Display block is resized to show all calculation results.





**Figure 5-16.** Complete matrix operations and Simulink model called Ex2 MATRIX\_operations.slx

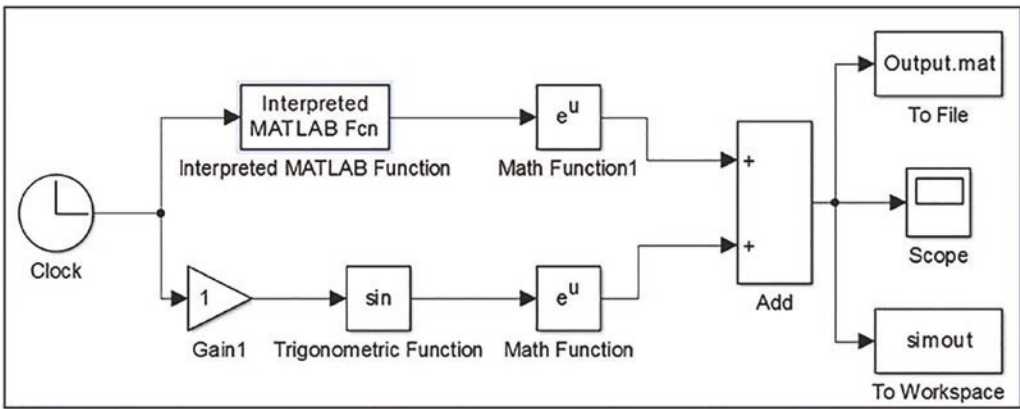
**Note** There are two file formats used to save Simulink models—\*.mdl and \*.slx. The latter model type is supported in later versions (starting with MATLAB 2010) of the Simulink package, and the former format can be opened and simulated by most versions, depending on the blocks used. Via a model export option, you can save models in the previous versions of Simulink.

There are a few new blocks with different properties included in later versions of the Simulink package. Therefore, the models developed by employing such new blocks cannot be simulated by earlier versions of the package.

## Computing Values of Functions

In this section, you learn how to compute values of the following math functions, save the computation results in a separate \*.mat file and MATLAB workspace simultaneously, and display them in a plot figure. Given  $H(t) = e^{\text{sinc}(t)} + e^{\sin(250t)}$ ,  $t = -3\pi \dots 3\pi$ ,  $\Delta t = \pi/3000$ .


There are several ways to build a computation model of the given example. Let's start with a simple and straightforward way. You first take the necessary blocks from the Simulink Library, like the previous two examples. For that, you need the following blocks: Clock, Scope, Math Function, Gain, Trigonometric Function, To File, and Add To Workspace. These are from Simulink/Sources, Simulink/Math Operations, Simulink/Sinks, and Simulink/User-Defined Functions, respectively. The modeling process starts with dragging all of the blocks from the Simulink Library and connecting them in the order of Clock+Interpreted MATLAB Function+Math Function1+Add+Scope2+To File and Clock+Trigonometric Function+Math Function+Scope1+Add+To Workspace. Figure 5-17 shows the completed model that is saved under the file name Ex3\_Function\_Compute.slx.



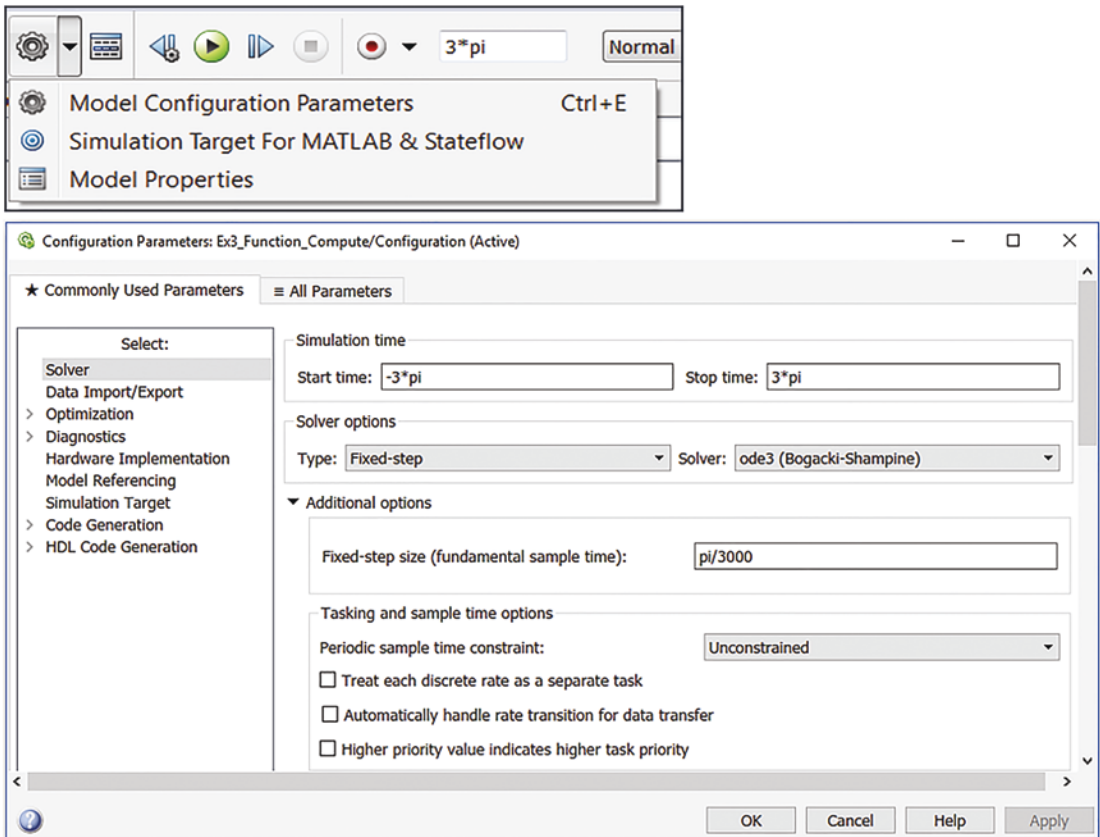
**Figure 5-17.** The complete simulation model called Ex3\_Function\_Compute.slx

This completed model, as it is, cannot be executed because the properties and parameters of several blocks used in the model need to be fixed according to the given tasks of this exercise.

For instance, the simulation period should be in the range of  $t = -3\pi \dots 3\pi$  with the time interval of  $\Delta t = \pi/300$ . The MATLAB function of the block Interpreted MATLAB Function must be altered to `sinc()`. Two optional editing points—the To File and To Workspace blocks—should be renamed as external \*.mat files, and a variable should be saved in the workspace in structure format.

You can start adjusting the simulation time interval and the time step via the Model Configuration Parameters (see Figure 5-18). They can be accessed by clicking the  icon or by pressing the Ctrl+E keys on the keyboard. Insert the start time of  $-3*\pi$  and the

stop time of  $3\pi$ , and change the solver type to a fixed-step from the variable step that is chosen by default. In addition, in the Solver options, you should set Type to Fixed-step and, in the Additional options, a fixed-step size (fundamental sample time) of  $\pi/3000$ . See Chapter 8 for a more detailed explanation on solvers and how to choose their types and parameters, including solver algorithm, step size, relative error, and absolute error tolerances.



**Figure 5-18.** Adjusting the configuration parameters

---

**Note** For many models, the accuracy of the simulation results depends on the chosen solver type and step size.

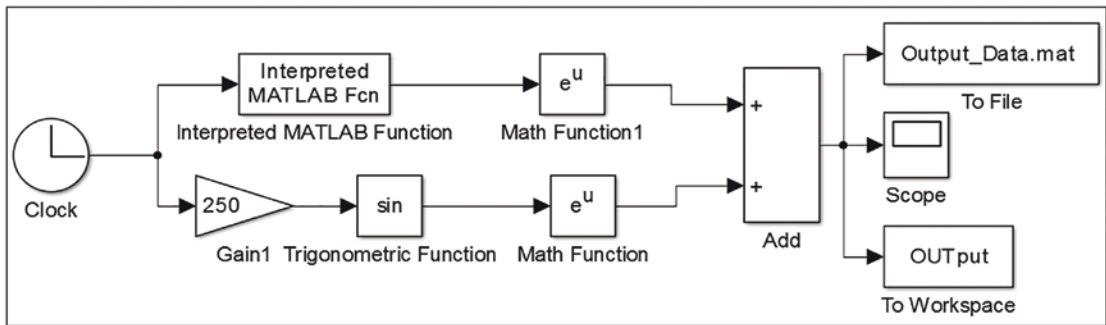
---

Also, before proceeding with the simulations, you need to make the following adjustments to the model:

1. The MATLAB function of the Interpreted MATLAB Function block should be  $\text{sinc}(u)$ . Note that the input variable name  $u$  is defined by default.
2. In both MATH Function blocks, a Function type (from the drop-down options) of  $\text{exp}$  function should be used (it's the default).
3. In the Gain block, the Gain value should be 250 to obtain  $250 \cdot t$ .
4. In the Trigonometric function, make it the  $\text{sin}$  function by default.
5. Even though this step is optional, rename the output file `Output_Data.mat` and the output variable by `output` saved in the MATLAB workspace.

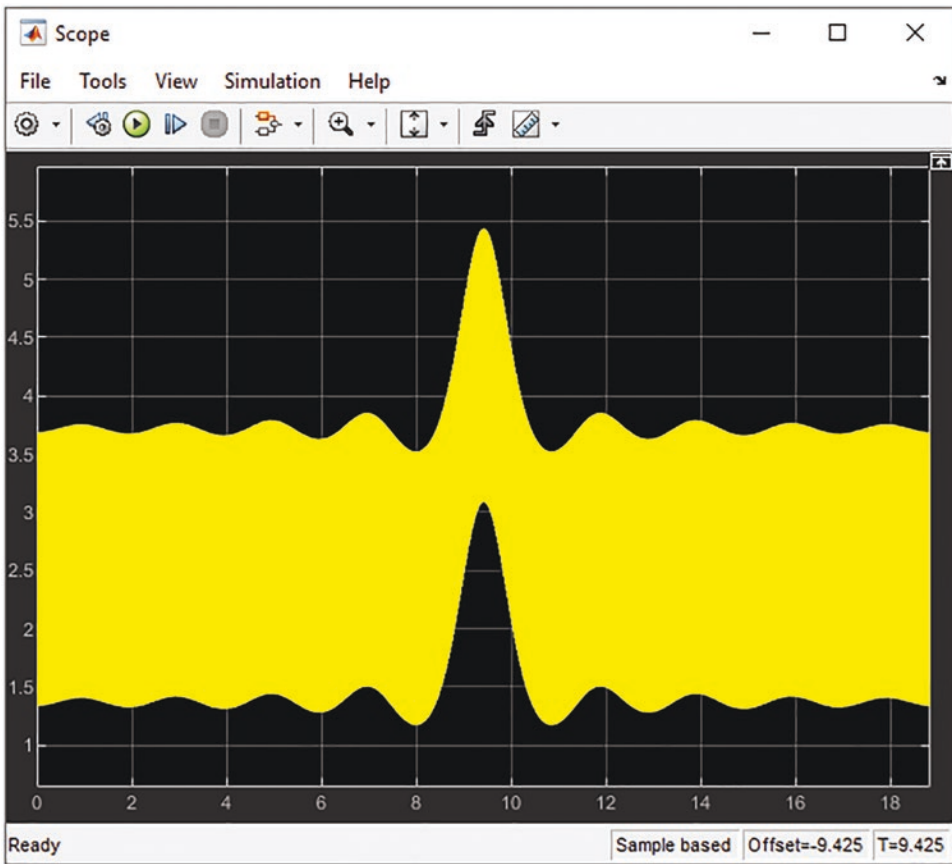
**Note** When you're saving the computation results via an output file in `*.mat` format and an output variable, there are several options (formats) to save data—time-Series, array, Structure, and Structure with time series.

After making all these adjustments, the model should look like Figure 5-19.







**Figure 5-19.** Finalized model with all necessary adjustments and tunings

Finally, you can simulate the `Ex3_Function_Compute.slx` model by pressing `Ctrl+T` on the keyboard or clicking the Run button. You can view the simulation results by double-clicking the Scope block. Figure 5-20 shows the simulation results, plotted in the Scope block.





**Figure 5-20.** Simulation results




---

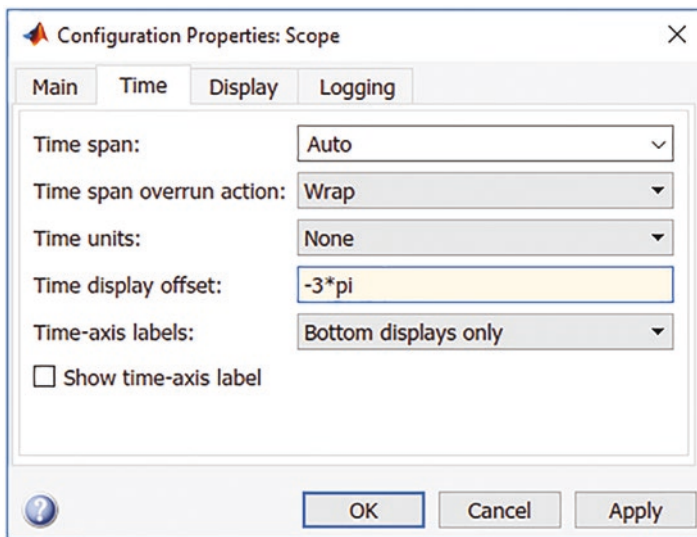
**Note** in the Scope display, the simulation results can be zoomed in or out proportionally, horizontally, and vertically. They can also be auto-zoomed in or out, all by using the , , ,  tools.

---

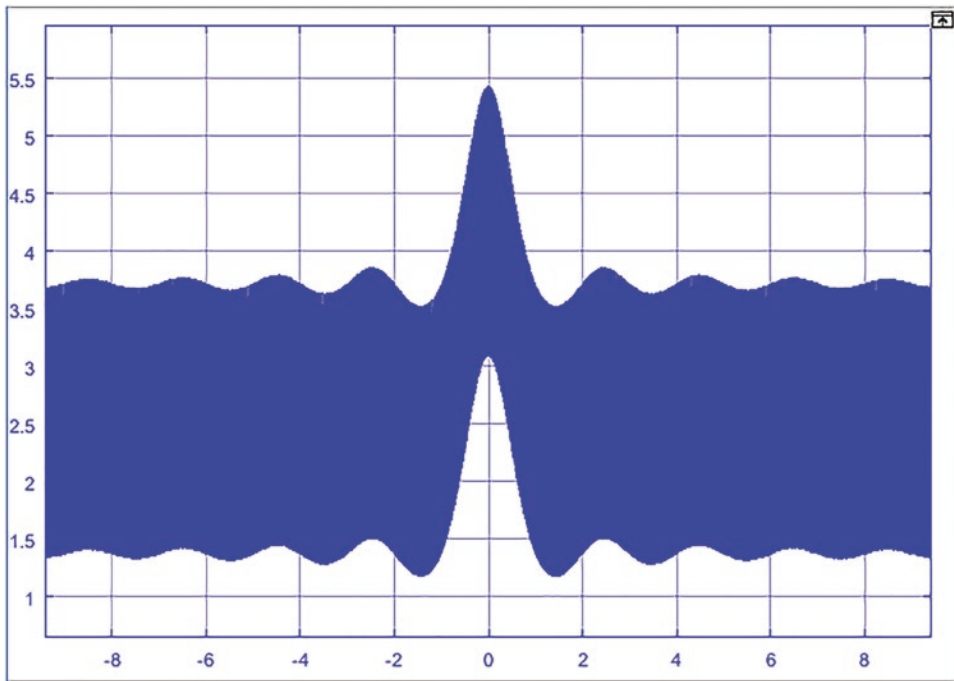
The simulation results, shown in Figure 5-20, are not coherent with the set simulation period that is set to be within  $-3\pi \dots 3\pi$ . The problem is in the Scope block's Time display offset, which is set to 0 by default. The time offset needs to be set to  $-3\pi$ , which can be fixed via the Configuration properties of Scope. The Configuration properties of Scope can be accessed by clicking the  icon from its drop-down options of  Configuration Properties. After opening the Configuration Properties, click the Time tab and enter  $-3\pi$  for "Time display offset." Subsequently, click Apply and OK

(see Figure 5-21) and you'll obtain a correct display of the results in Scope, as shown in Figure 5-22. In addition, the simulation results displayed in the scope can be saved as a variable in the format of structure with time series, array, or structure. You do this by selecting the Log Data to Workspace option on the Logging tab (see Figure 5-21).

Moreover, the Scope block has many graphical display options that can be accessed via the  drop-down options, by clicking Style . Via the Style  options, you can adjust the figure color, axes colors, lines, and markers. See Figure 5-22.



**Figure 5-21.** Setting up the Time display offset at  $-3\pi$  in Configuration Properties: Scope

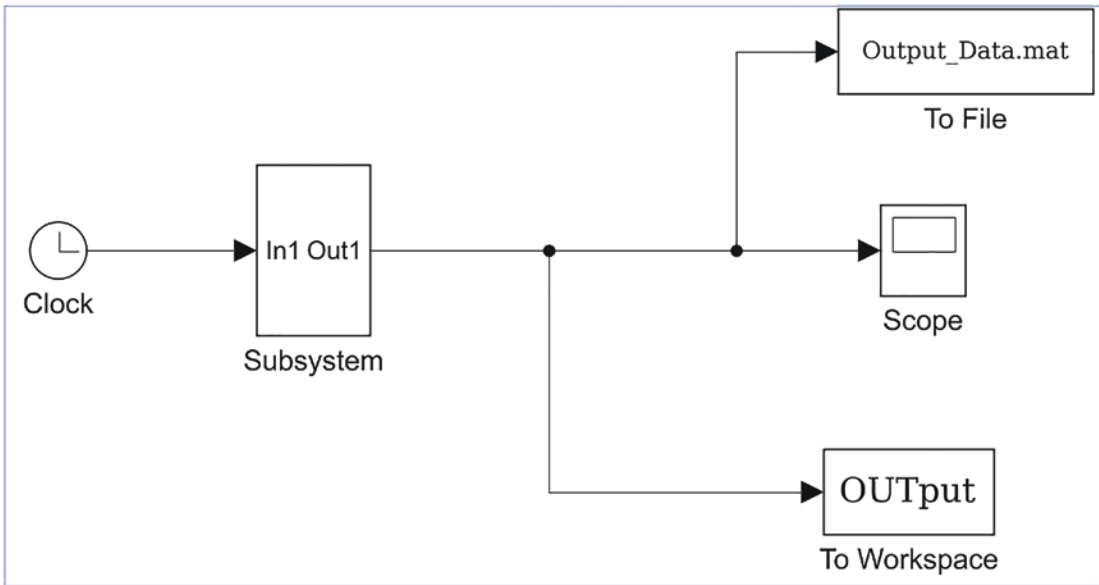


**Figure 5-22.** *A complete view of the simulation results*

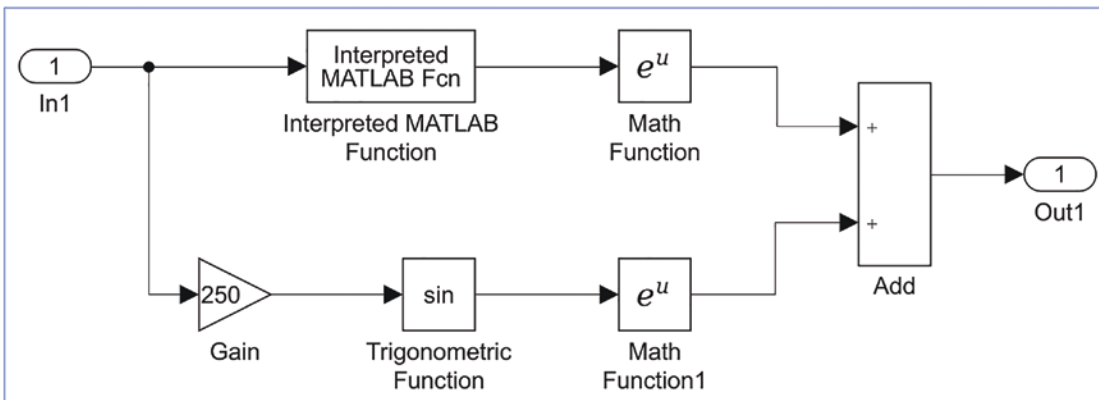
To improve the readability and manageability of large and complex Simulink models, a whole model or part of a model can be associated in one Subsystem block.

It is easy and straightforward to create subsystems from existing models. The easiest way to create subsystems from existing models is to select the blocks (interlinked ones) by holding the left mouse button and dragging the cursor over the desired blocks and then pressing Ctrl+G on the keyboard. To create one subsystem out of a whole model, press Ctrl+A first and then Ctrl+G on the keyboard.

Let's look at the complete model (see Figure 5-19) and create a subsystem out of the whole model, excluding the Input signal to Clock and the Output signal to blocks—To File, Scope, and To Workspace. You first select the model blocks except for the Input block and all the Output blocks, by using the left mouse button, and then press Ctrl+G on the keyboard. The subsystem is created from the selected blocks of the model, as shown in Figure 5-23. To have access to what is under the created subsystem, you need to double-click it. See Figure 5-24.



**Figure 5-23.** Subsystem created out of the completed model (Figure 5-19), excluding Input and Output blocks



**Figure 5-24.** The created subsystem components

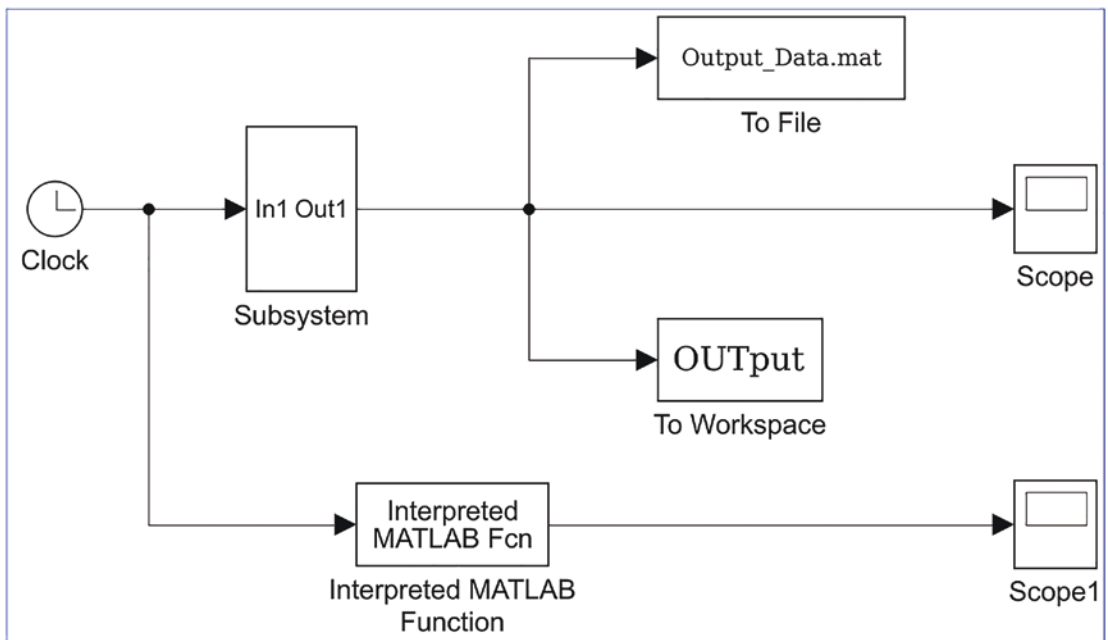
Note that this subsystem in Figure 5-24 has one input block called In1 and one output block called Out1. They are linked with the Input block, Clock, and Output blocks—To File, Scope, To Workspace—to receive and send signals, respectively.

Note that if you create a subsystem from the whole model (see Figure 5-19) excluding only the Input block, then the subsystem will contain only one input block, called In1. If you create a subsystem from the whole model (see Figure 5-19) excluding the output



blocks—To File, Scope, To Workspace—then it will contain one output block Out1. If you create a subsystem from the whole model including all blocks, as well as the Input and Output blocks, the created subsystem does not contain any input and output blocks.



The completed model can be simplified. In other words, the number of blocks used in this model can be reduced by employing the Interpreted MATLAB Function block with appropriately edited function formulation expressed by  $\exp(\text{sinc}(u)) + \exp(\sin(250*u))$  and adjusting Scope parameters that save simulation results in the MATLAB workspace as a structure with time series. Figure 5-25, together with the subsystem created from the model in Figure 5-23, produces the same results as the subsystem in Figure 5-19.

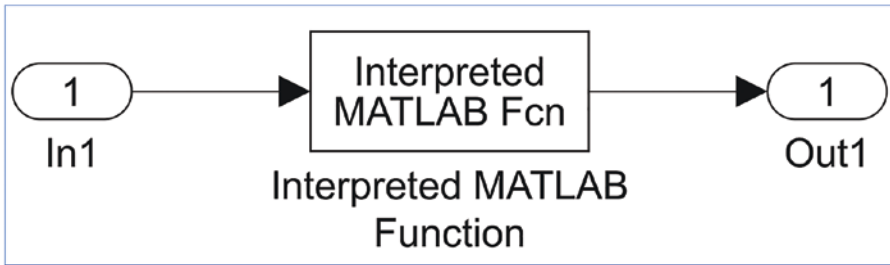


**Figure 5-25.** *Simplified model Ex3\_Function\_Compute\_Simple.slx*


## Input/Output Signals from/to the MATLAB Workspace

As stated, the Simulink model works interactively and flawlessly with the MATLAB workspace. For model development and simulation purposes, input signals can be generated in the MATLAB workspace or within an M-file and then transferred to the Simulink model environment. Similarly, all final simulation results of Simulink models can be sent to the MATLAB workspace. To have an input signal loaded from the MATLAB

workspace and an output signal (simulation results) sent back to the MATLAB workspace at the same time, you use input and output blocks and adjust the model configuration parameters , which can be accessed by clicking the  icon from the menu panel or pressing Ctrl+E on the keyboard. In the previous example shown in Figure 5-25 (simplified part is considered), we substitute the input signal Clock block with In1 (the Input block) and the Scope block with Out1 (the Output block); see Figure 5-26.

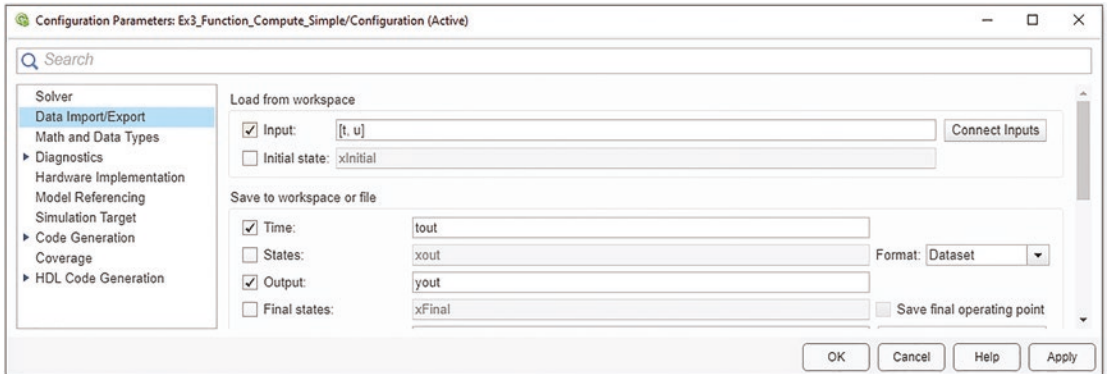


**Figure 5-26.** *Ex3\_Function\_Compute\_In\_Out.slx* model with Input/Output from/to MATLAB workspace

In addition, the Input and Output signal options in the model configuration parameters  are checked (see Figure 5-27) for input as *t*, *u*, and for output as *tout* and *yout*, respectively. Note that before starting the simulation, you must define the input signal in the form of *t*, *u* in the MATLAB workspace as two column vectors. That can be done for this example as follows:

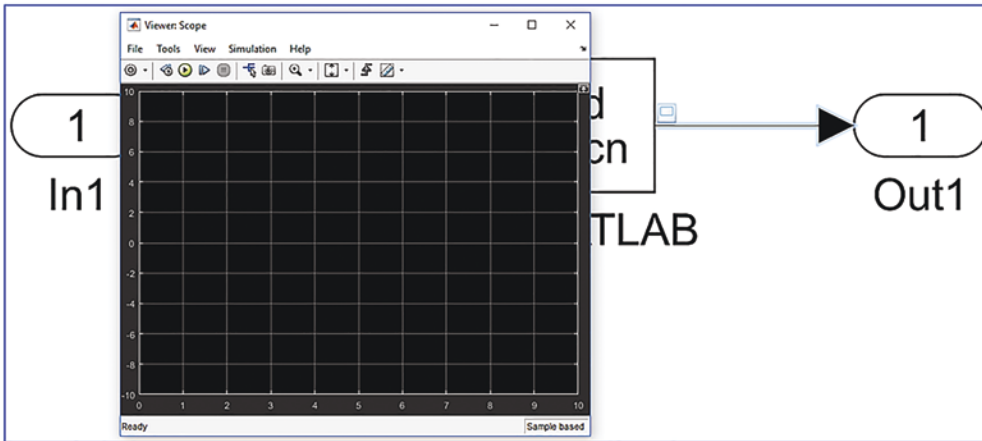
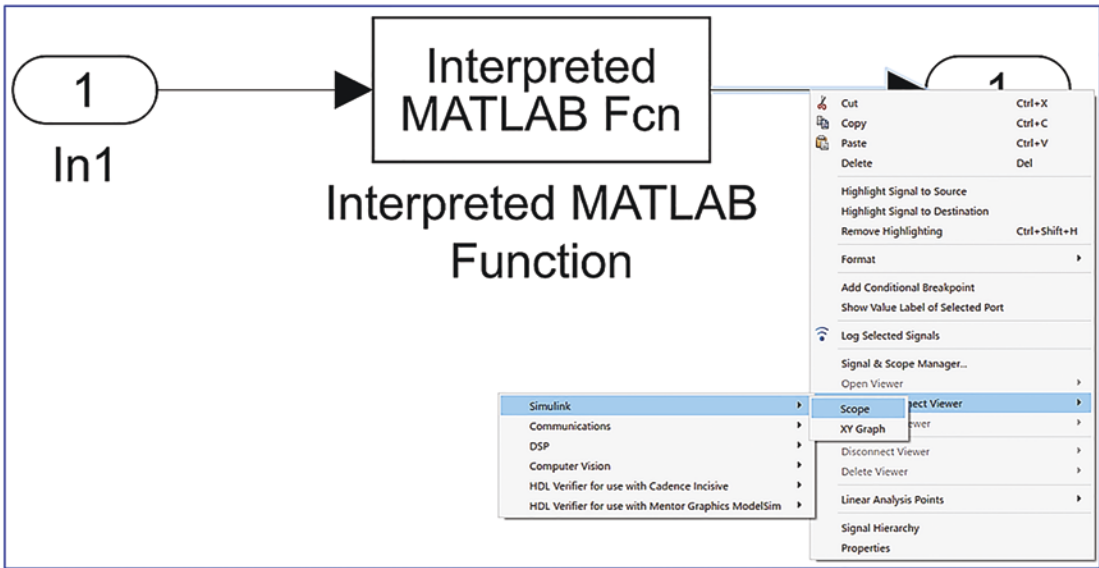
```
>> t=[-3*pi:pi/3000:3*pi]'; u=t;
```

In this example, the input signal is defined by time only, and thus, *t* = *u*. Moreover, we make several adjustments (Start time:  $-3\pi$ ; End time:  $3\pi$ ; Type: Fixed-step; Fixed-step size (fundamental sample time):  $\pi/3000$ ) in the Solver options of this model via the Configuration Parameters window, as shown in the example in Figure 5-18.



**Figure 5-27.** Configuration parameters changed to load an input signal from the workspace and export an output signal back to the workspace

In addition, we can add the Scope block to the model by clicking the signal going to the Out1 block and using the right-click options (Create & Connect Viewer ► Simulink ► Scope), as shown in Figure 5-28 (top). After selecting Scope, the scope sign shows up on top of the signal going to the Out1 block, as in Figure 5-28 (bottom).



**Figure 5-28.** Adding a Scope block to a signal

In addition, make one important adjustment (Time display offset:  $-3\pi$ ) in the Scope block to make it display the whole simulation results completely, as demonstrated in the example and shown in Figure 5-21. Now, you save the model (Ex3 Function\_compute\_In\_Out.slx) and simulate it after entering this in the MATLAB workspace:

```
>> t=[-3*pi:pi/3000:3*pi]; u=t;
```

After simulating the three alternative models, Ex3\_Function\_compute.slx, Ex3\_Function\_Compute\_Simple.slx, and Ex3\_Function\_Compute\_In\_Out.slx, the simulation results are identical. Via these examples, you have seen how easily one block can be substituted for another, how easily subsystems can be created from the existing models by associating parts of interconnected blocks, and how models can be simplified by reducing the number of blocks used in them.

## Simulating a Mechanical System

Let's consider a mechanical spring-mass-damper system with Newtonian friction that is formulated by the following differential equation:

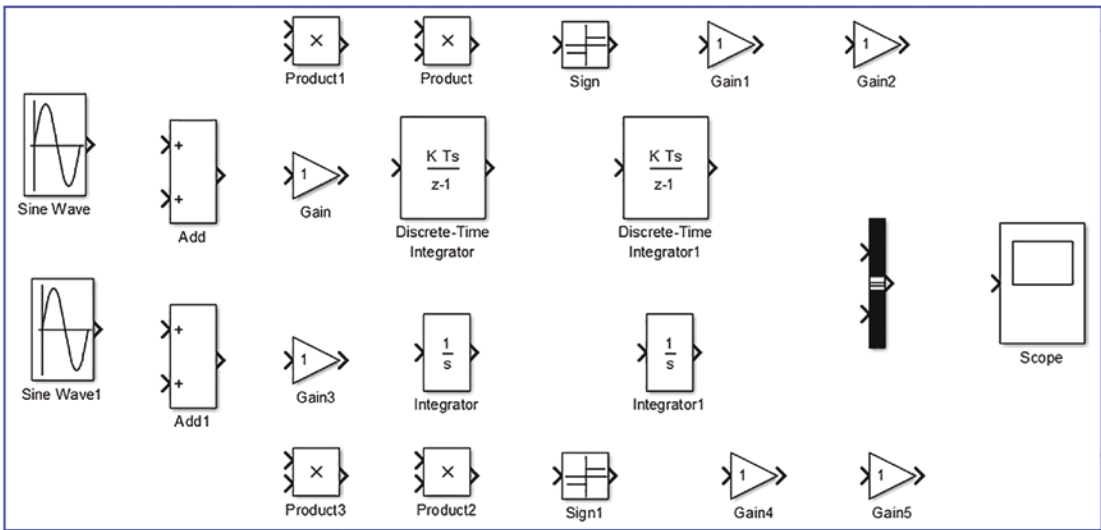
$$m\ddot{x}(t) + b * \text{sign}(\dot{x})\dot{x}^2(t) + kx(t) = f(t)$$

Let's treat the given model of the system as continuous and discrete systems in order to demonstrate how to model and simulate such system in Simulink. Note that solving and simulating differential equations via Simulink modeling is explained more in Chapter 8. Here, we put more emphasis on model building, adjusting block parameters, and interacting Simulink with MATLAB. Moreover, we address the issues of modeling continuous and discrete systems and of creating subsystems.

Given  $m = 0.52$ ;  $b = 0.00525$ ;  $k = 165.5$ ;  $f(t) = A\cos(\omega t)$ ;  $\omega = 131$ ;

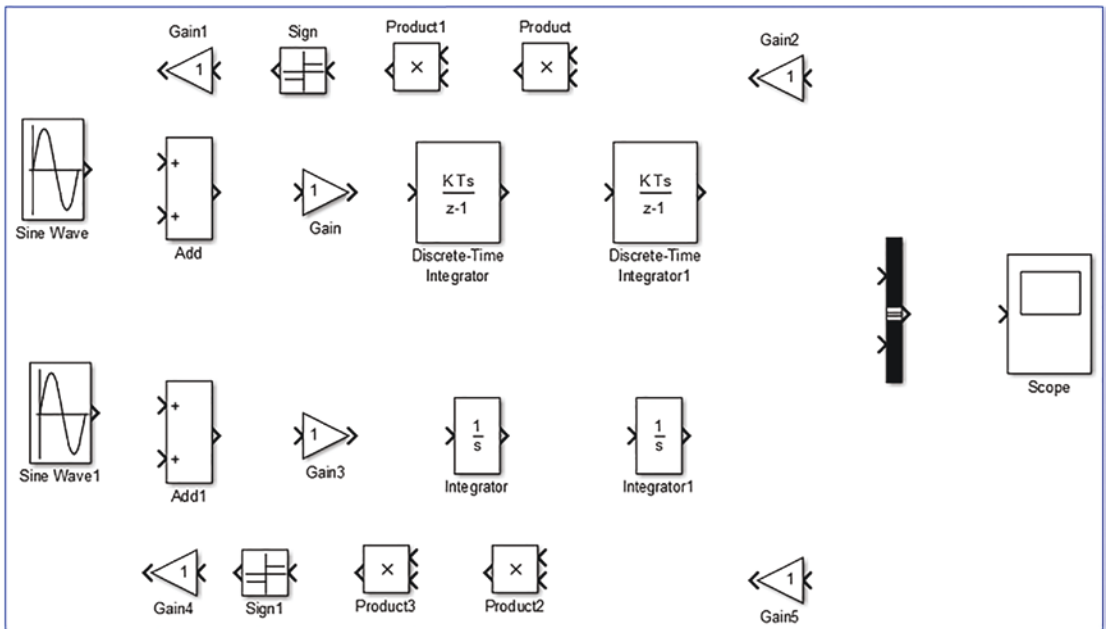
$A = 2.3$  and all initial conditions are "zero." The sampling time is  $ts = 0.01$ . The parameters of the system are  $m$  for mass,  $b$  for the damping coefficient,  $k$  for stiffness,  $f(t)$  for the input force,  $A$  for magnitude, and  $\omega$  for frequency.

1. Collect all the necessary blocks from the Simulink Library by dragging and dropping them in a Blank Model window. Figure 5-29 shows the required blocks, taken from Simulink/Math Operations, Continuous, Discrete, Sources, Signal Routing, and Sinks. Also, the Bus collector block is taken from Simulink/Signal Routing.



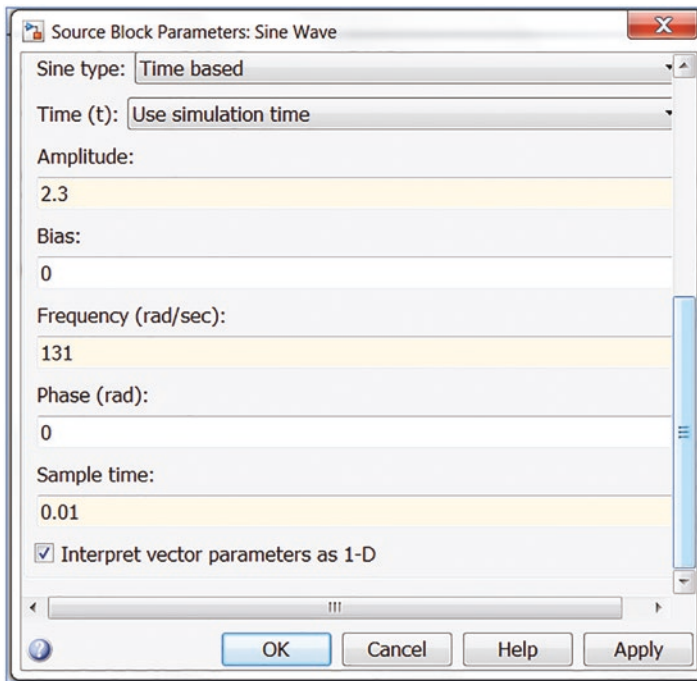
**Figure 5-29.** Necessary the blocks for continuous and discrete systems

2. Adjust the blocks in a more readable order by moving them around and rotating some of them by 180 degrees. The Product, Sign, and Gain blocks need to be rotated. You can do that by selecting a block and pressing the Ctrl+R keys on the keyboard or using the right mouse button's Rotate & Flip options. See Figure 5-30.



**Figure 5-30.** Adjusted blocks

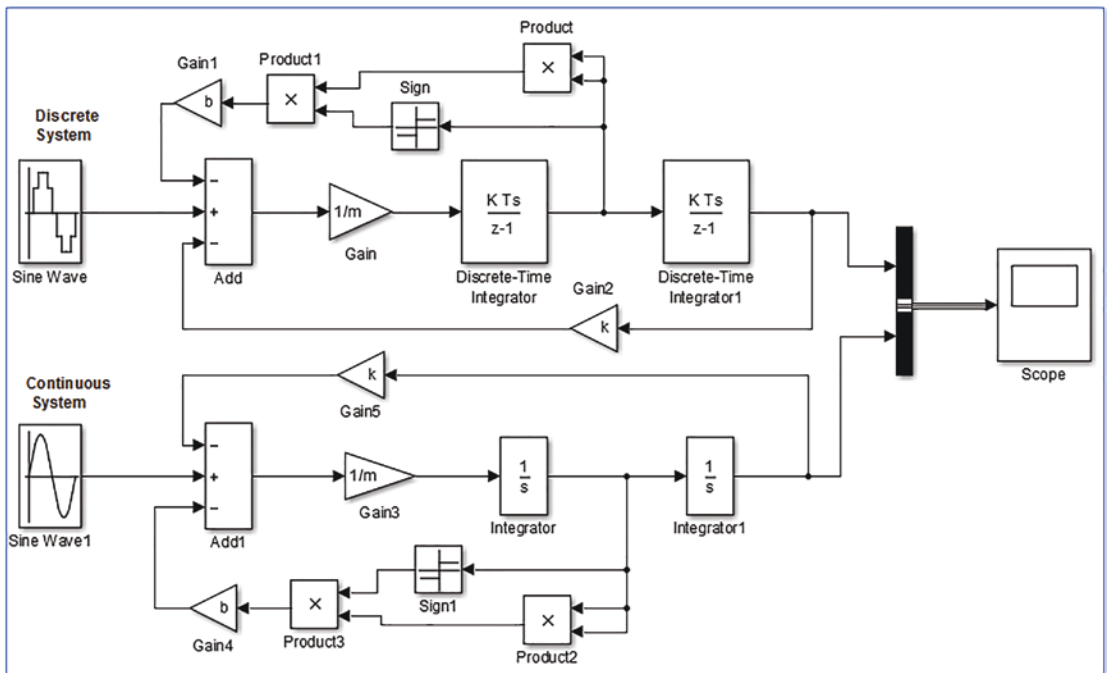
- Adjust the parameters of the blocks. First adjust the amplitude, frequency, and sampling time for the Sine Wave and Sine Wave1 blocks, which are input signals (see Figure 5-31). Note that for Sine Wave1, the sampling time is set to 0 because by default it is used for continuous system modeling.



**Figure 5-31.** Adjusting Sine Wave block's parameters

4. Note how the sine wave sign in the Sine Wave block has changed from a smooth curve to a stairs curve when Sample time is set to be 0.01. Change the Gain values for Gain and Gain3 to 1/m, for Gain1 and Gain4 to b, and for Gain2 and Gain5 to k. Also, change the signs in the Add and Add1 blocks to -+.
5. All blocks are connected, and complete models are attained. Two signals coming from the Discrete Time Integrator1 and Integrator1 are connected to the Bus Creator block that subsequently is connected to the Scope block. In addition, two notations are added: Discrete System and Continuous System; see Figure 5-32. The complete model is saved (Ex4\_Discrete\_Continuous\_Sys.slx).








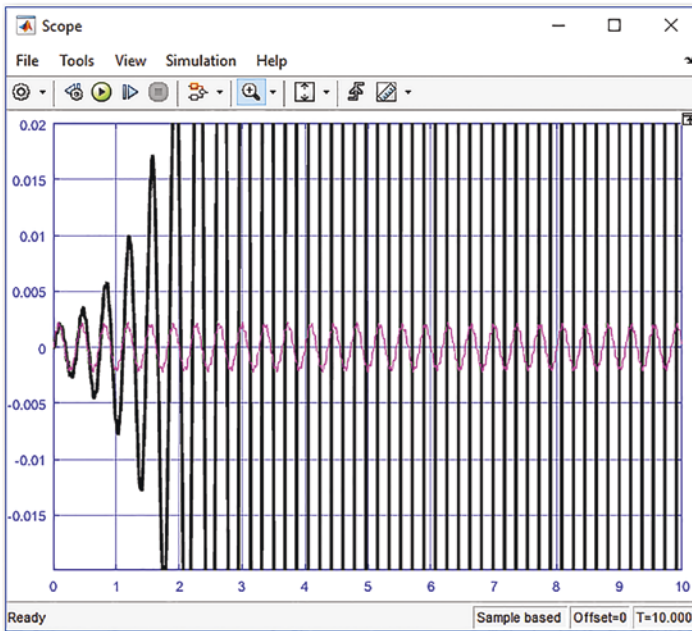
**Figure 5-32.** Complete models of discrete and continuous systems

6. You can execute these models by pressing the Ctrl+T keys on the keyboard or pressing the Run button.

Now the completed models (Ex4\_Discrete\_Continuous\_Sys.slx) seem to be ready for simulation. However, if we execute them, error message windows will be launched, and no results will be attained. The reason for that is the values of three parameters— $m$ ,  $k$ ,  $b$ —are not defined yet. You can define them in several different ways, one of which is to specify the values of the parameters in each block or in the MATLAB workspace. You can also specify this information in the Model properties/Callback/InitFcn, which can be accessed via File ► Model Properties. Enter the following in the Command window:

```
>> m=0.52; k=155; b=.00144.
```


Subsequently, click the  Run button in the Simulink model window. You can change the Scope block's Style settings (via  drop-down options ► Style ) and obtain the results of the simulation displayed in the Scope block; see Figure 5-33.

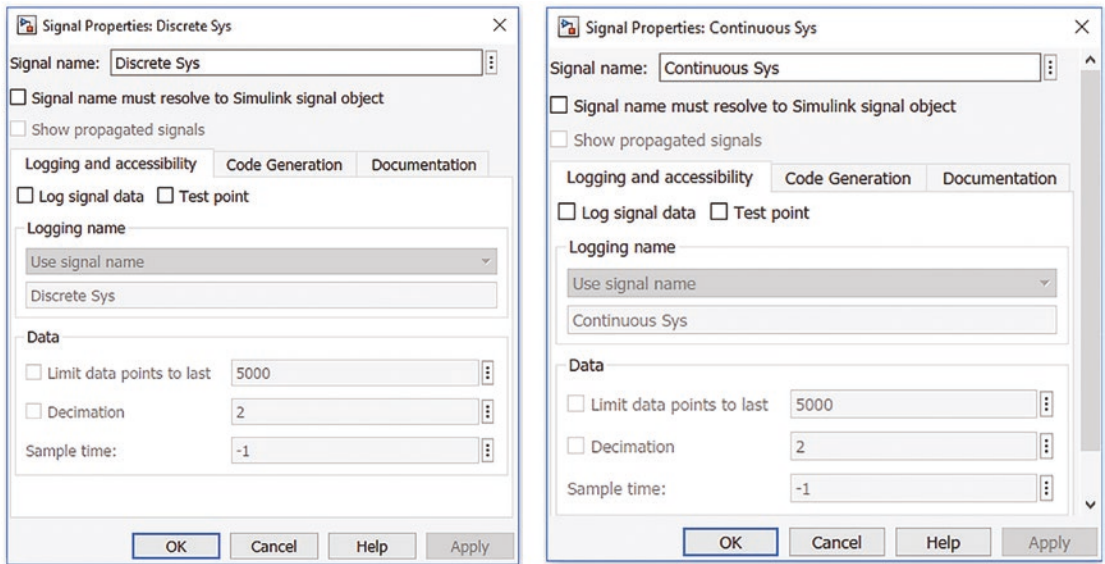


**Figure 5-33.** Simulation results

From the simulation results, you can see that one of the systems (the discrete one) is not stable and should be fixed. The problem with this discrete system modeling resides in the sampling time, which has to be adjusted. That can easily be fixed from the input signal block, i.e., the Sine Wave. In this block’s sample, the time value is changed from 0.01 to 0.0001, which is 100 times smaller than initially set.

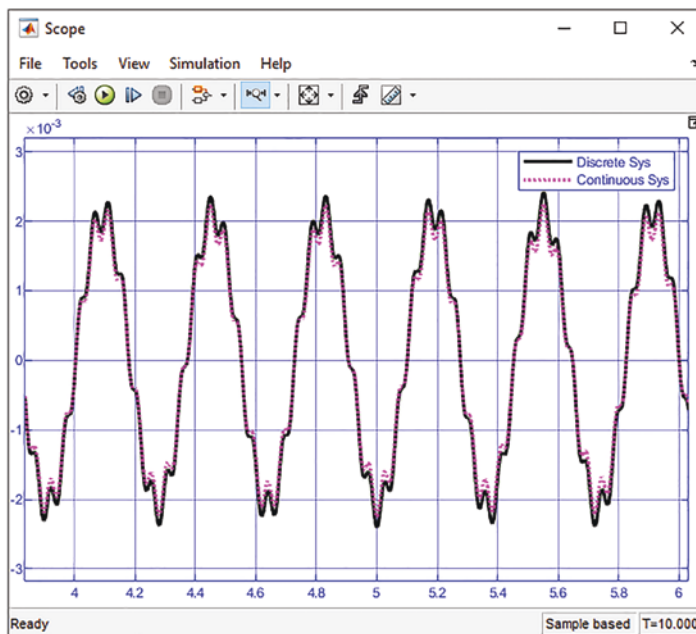
**Note** If the sample time is set to -1 in a Discrete Integrator block, that makes the sample time be inherited automatically from an input signal source.

In addition to making the plot in the Scope more readable with legends displayed for the discrete and continuous system models, let’s check the Legends option of the Scope block’s properties . Second, click the signal going from Discrete-Time Integrator1 to Bus Creator and use the right mouse button’s option to access Properties (Signal Properties). There, you specify the signal name to be Discrete Sys and then click OK. Similarly, change the signal name for the signal going from Integrator1 to Bus Creator and name the signal Continuous Sys. Then click OK, as shown in Figure 5-34.



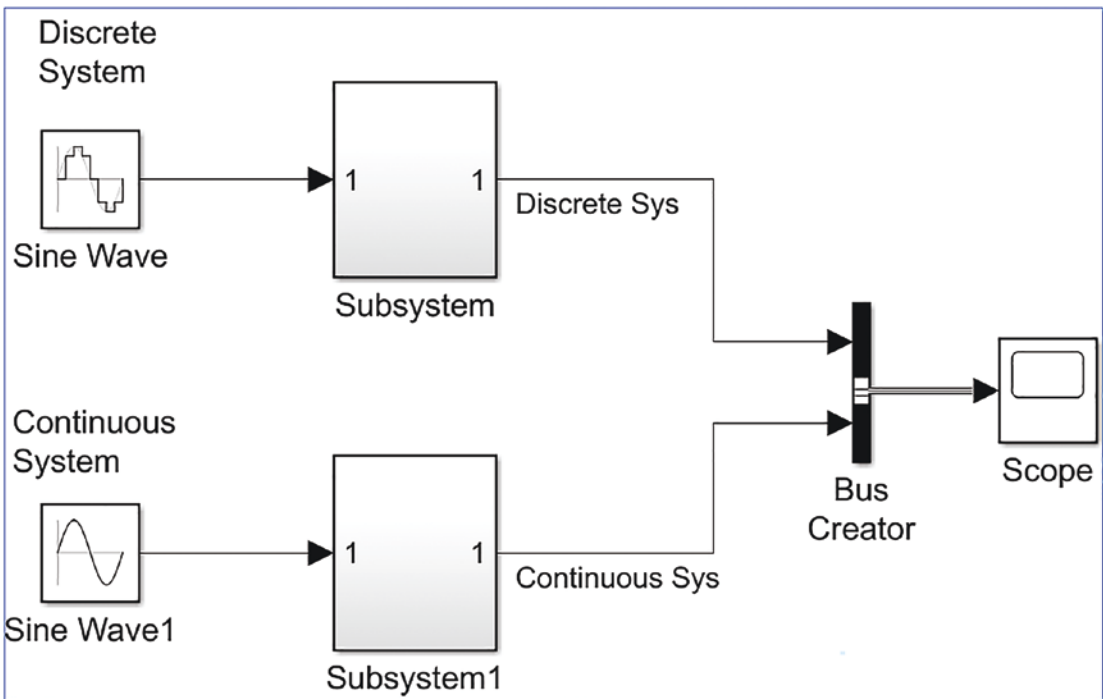
**Figure 5-34.** Signal name change

After these three changes, the whole model runs, and the next result is obtained, as shown in Figure 5-35. Note that it is a zoomed-in view along the horizontal axis.



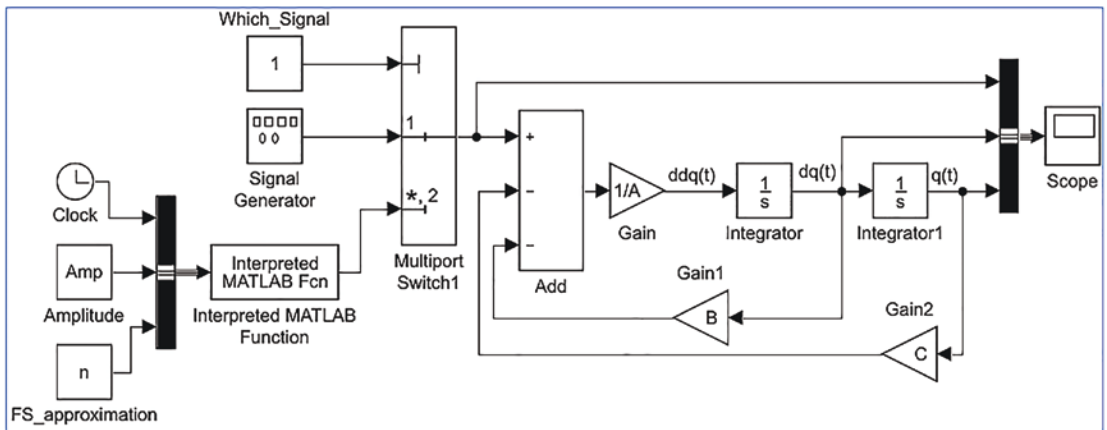
**Figure 5-35.** Simulation results displayed in Scope with its adjusted properties

You have completed and verified the system models that can be simplified by using a subsystem option, as shown in the previous example (see Figure 5-25; Ex3\_Function\_Compute\_Simple.slx). Any system containing more than two blocks can be simplified or rather substituted by employing a subsystem block. There are several ways to create subsystems from system models. The easiest way is to select model blocks meant to be under one subsystem and then use the right mouse button options of Create Subsystem from Selection or press Ctrl+G on the keyboard. You create the two subsystems (Subsystem and Subsystem1) from the discrete and continuous system models (see Figure 5-36).



**Figure 5-36.** The model containing two subsystem models

A model under any subsystem can be accessed by double-clicking it. The subsystem representing the discrete system contains the model shown in Figure 5-37.



**Figure 5-37.** Subsystem composed of this model

Also, it is possible to reverse the process of subsystems by clicking a subsystem block and using the right mouse button option of Subsystem & Model Reference ► Expand Subsystem. You can also do this by pressing the Ctrl+Shift+G buttons after clicking a subsystem block.

In this exercise, you learned how to build discrete and continuous systems and how Simulink handles such composition of systems, but you have not made any adjustments to solver parameters. The accuracy and efficiency of simulation processes can be improved by adjusting solver type (variable step solver selected by default or fixed step solver) and parameter settings (solver, error tolerances, solver algorithm, step size if fixed step solver chosen, zero-crossings, and so forth).

## Working with a Second-Order Differential Equation

Now let's build a Simulink model of the given system expressed by the second-order differential equation  $A\ddot{q}(t) + B\dot{q}(t) + Cq(t) = F(t)$ , where  $F(t)$  is applied force (input signal to the system) associated with the MATLAB's function file.  $F(t)$  is a rectangular pulse approximated by the Fourier series  $F(t) = \sum_{n=1}^N \left( \frac{Amp}{\pi} \right) (1 - \cos(n\pi)) \sin(n\pi t)$ , which is implemented in MATLAB via the `function_pulse.m` function file:

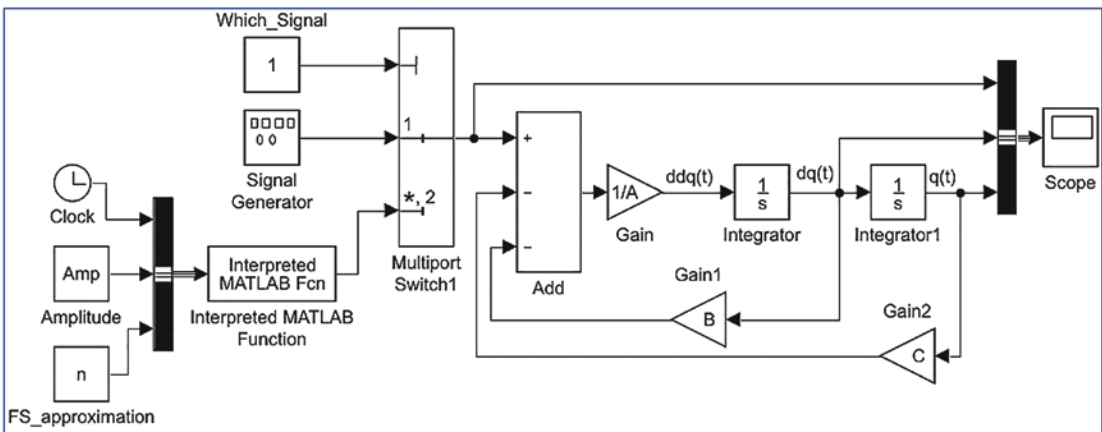
```
function F=function_pulse(t, Amp, n)
% HELP. Two input arguments, viz. t, Amp, and n are needed for
```

% simulation, where t is time vector, Amp is amplitude of a pulse and n % number of approximation terms in Fourier series.

```
F(1,:)=(Amp/pi)*(1-cos(pi))*sin(pi*t); for ii=2:n
    F=F+(Amp/(ii*pi))*(1-cos(ii*pi))*sin(ii*pi*t);
end
```

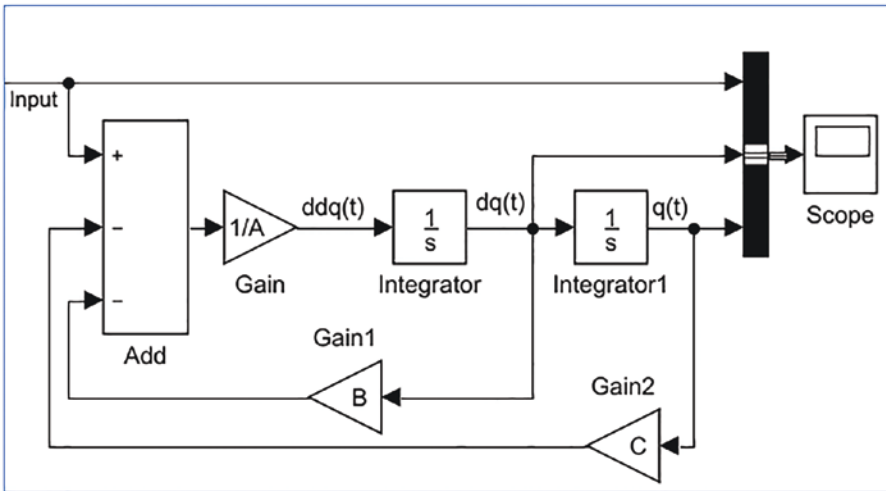
Let’s build a Simulink model associated with the function file `function_pulse.m`. Moreover, you’ll employ Simulink blocks—Repeating Sequence and Signal Generator—to generate a rectangular pulse signal. You’ll explore the options with MATLAB-associated function files and Simulink blocks for input signal generation. In addition, you’ll explore a few key Simulink modeling tools and aides, such as the model explorer, model advisor, code generation in C/C++, report generation, and so forth. For numerical simulations, the following values are used:  $A = 2$ ;  $B = 4$ ;  $C = 200$ ;  $Amp = 10$ ;  $n = 25$ .

Figure 5-38 shows the complete model of this exercise. It associates the given function file called `function_pulse.m` via the Interpreted MATLAB Fcn block. The input signal  $F(t)$  rectangular pulses are generated via two ways—one Interpreted MATLAB Fcn with three input variables ( $Amp$ ,  $n$ ,  $t$ ) and a signal generator with two inputs ( $amp$  and frequency). You can compare the simulation results from the two input signal sources, i.e., the MATLAB associated input signal generation (`pulse_function.m` embedded via Interpreted MATLAB Fcn) versus Simulink Library block (the signal generator). Moreover, you’ll see how to use Simulink’s Model properties to enter the model parameters, such as  $Amp$ ,  $n$ ,  $A$ ,  $B$ ,  $C$ , and how to use Multiport Switch block.



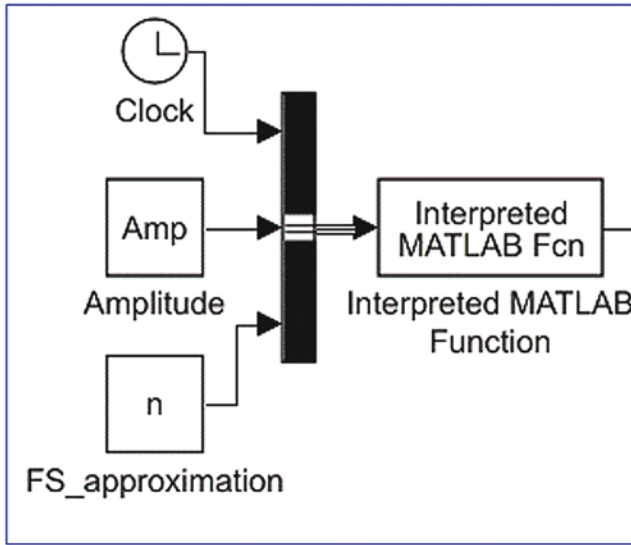
**Figure 5-38.** Complete model of the second-order ODE:  $A\ddot{q}(t) + B\dot{q}(t) + Cq(t) = F(t)$

The necessary blocks to model the given exercise are Add, Gain, Integrator, Bus Creator, and Scope, which you can drag and drop to a Blank Model window. The blocks are connected, as shown in Figure 5-38. Note that the three signals connected with Bus Creator and Scope blocks are called Input,  $dq(t)$ , and  $q(t)$ , and they represent input signal, pulse, velocity, and displacement. This is displayed via legends in the Scope block plot.



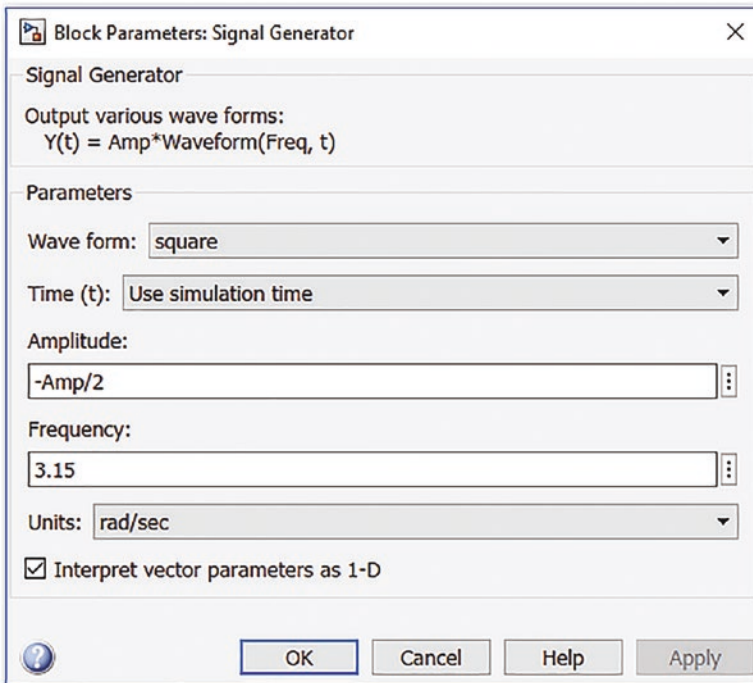
**Figure 5-39.** Simulink model of the given second-order differential equation without Input force signal generators

The function file (called `function_pulse.m`) is associated via the Interpreted MATLAB Fcn block with the Simulink model (see Figure 5-38). Interpreted MATLAB Fcn is modified to call the MATLAB function: `function_pulse(u(1), u(2), u(3))`, where  $u(1)$  calls the time signal,  $u(2)$  calls the amplitude Amp, and  $u(3)$  calls  $n$  number of the Fourier series approximation. Thus, the Interpreted MATLAB Fcn block requires three input signals simultaneously. That can be done via the Bus Creator block, as shown in Figure 5-40. Note that you can change block names by clicking the name of each block and typing the new name. Note that block tags are not considered during the model simulation, and thus, they have only an informative character for the user/programmer.



**Figure 5-40.** MATLAB Fcn block with three inputs

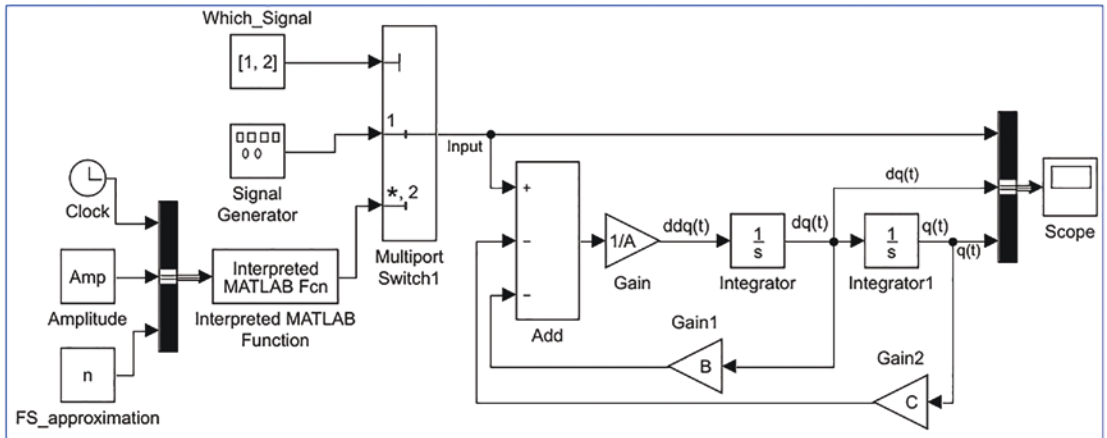
Subsequently, one Simulink block, called Signal Generator, is added to the model and its parameters to generate pulses (Amplitude =  $-Amp/2$  and Frequency = 3.15 [rad/sec]). It's then adjusted according to the given pulse parameters, as shown in Figure 5-41.



**Figure 5-41.** Adjusted parameters of the Signal Generator block

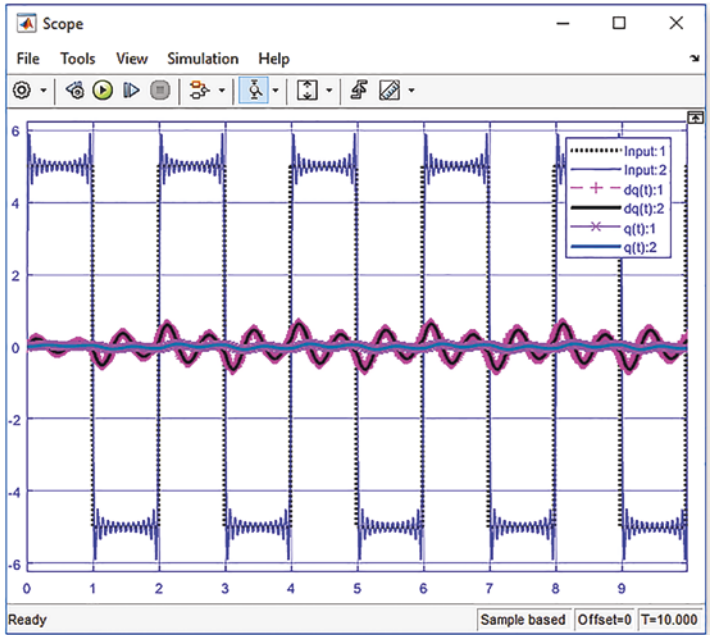


To connect two input signal sources (Signal Generator and MATLAB Interpreted Fcn), add the Multiport Switch block and connect it to another Constant block to specify a source signal block for selection. Finally, name the three signals going to the Scope block via the Bus selector block. Moreover, adjust the Scope block's Style options to make the output signals readable. Figure 5-42 shows a completed model.



**Figure 5-42.** Complete Simulink model called *Ex5\_Function\_PULSE.slx*

Before you start the simulation, you have to specify the values for  $A$ ,  $B$ ,  $C$ ,  $Amp$ , and  $n$ . You can do that via the Callbacks option, from File ► Model Properties ► Model Properties ► Callbacks (Model callbacks) ► InitFcn. In the Model initialization function window (or alternatively, in the Command window), type  $A=2$ ;  $B=4$ ;  $C=200$ ;  $n=25$ ;  $Amp=10$  and click OK. When you execute the model, both input signals are taken in the order of first and second with respect to the Constant block (called *Which\_Signal*) values 1, 2. They correspond to Input 1 – Signal Generator and Input 2 is MATLAB Interpreted Fcn. The simulation results in Figure 5-43 show that the two Input signals and two pairs of Output signals ( $dq(t)$ ,  $q(t)$ ), which represent velocity  $\dot{q}$  and displacement  $q$ , respectively, are well converged. Note that Input: 1 is Signal Generator and Input: 2 is MATLAB Interpreted Fcn.



**Figure 5-43.** Simulation results: a) from Signal Generator, b) from Repeating Sequence, and c) from Interpreted MATLAB Fcn

It is clear from the simulation results displayed in Figure 5-43 that all three blocks generating pulse input signals have resulted in approximately the same excitation in the system. Simulink model blocks can be associated with MATLAB files if they are correctly modeled and adjusted. It should be noted that the second input signal generation approach via Interpreted MATLAB Fcn is less accurate. It approximates the Fourier Series from the M-file function\_pulse.m. Moreover, it is slower since it calls an external M-file to generate the signal.

## Subsystem in Simulink Modeling

Simulink has a handy function to create a subsystem from the existing Simulink model components or create model components inside the Subsystem block. The Subsystem block helps you make your created model in a more well-structured way. The Subsystem-based model and a model without it do not have any differences in terms of simulation speed.

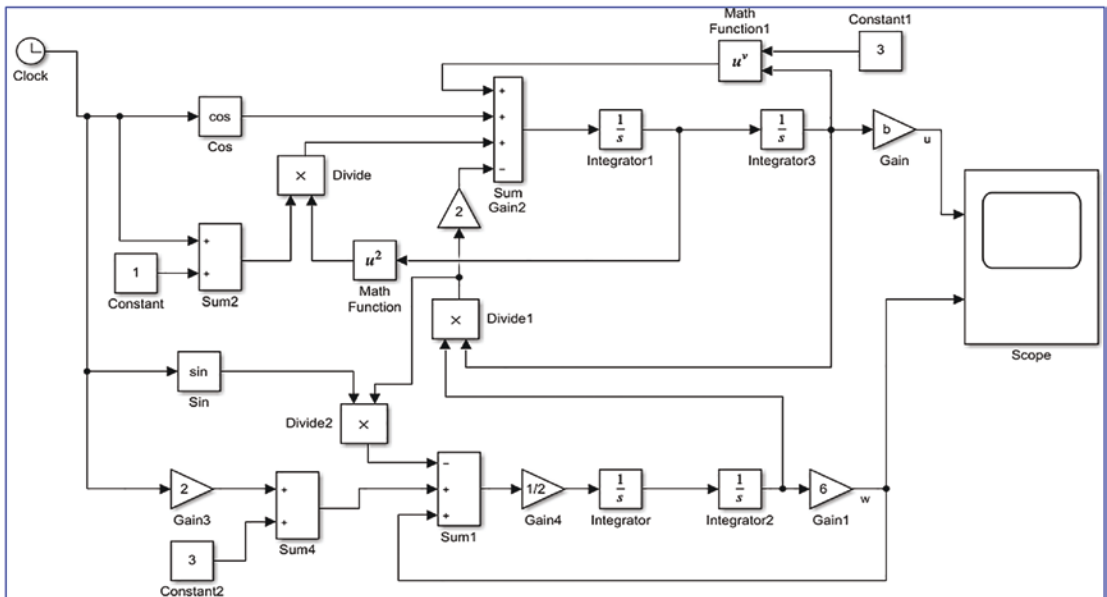
Let’s look at the following example to show how to employ the Subsystem block:

$$\ddot{u} - (t + 1)\dot{w}^2 + 2uw - u^3 = \cos(t)2\ddot{w} + \sin(t)\dot{u}\dot{w} - 6u = 2t + 3$$

With the initial conditions of  $u(0) = 1; w(0) = 3; \dot{u}(0) = 2; \dot{w}(0) = 4$  and parameter values  $a = 2, b = 10$ . The  $u$  and  $w$  are the functions of time. To build a Simulink model of this exercise, we will follow the steps and procedures given in the previous two sections on solving second-order differential equations. Thus, all model building steps are skipped here.

To build a Simulink model of this given system of coupled differential equations, Sum, Integration, Gain, and Scope blocks are needed. Here again we follow the steps of building a Simulink model of a second-order ODE as explained previously.

In addition, the subsystem block will be used once the model is complete. The initial version of the complete Simulink model is `Ex6_Coupled_ODE_ver1.slx`, as shown in Figure 5-44.

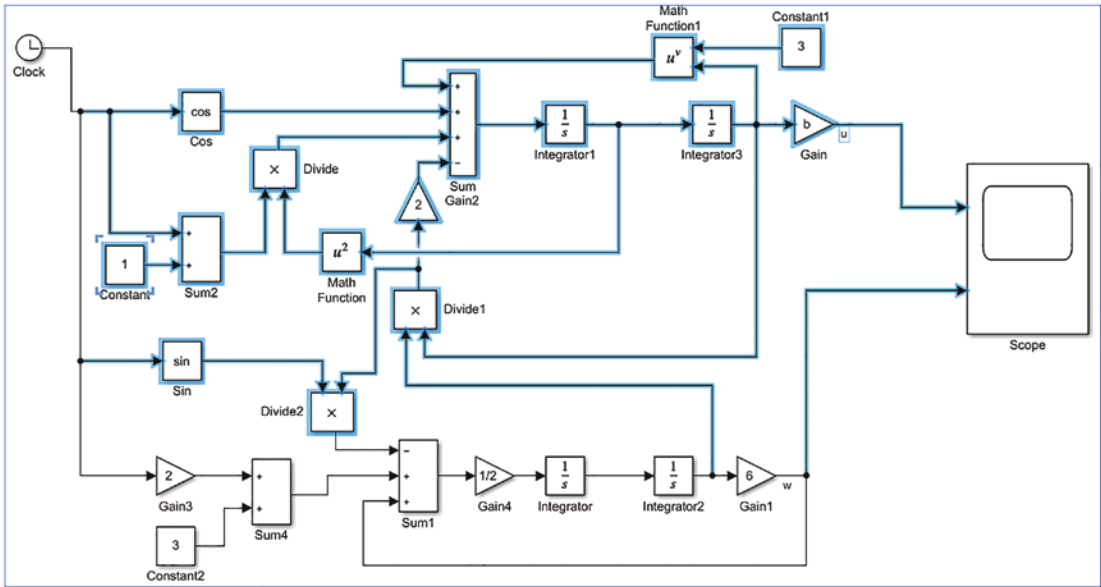


**Figure 5-44.** Simulink model, `Ex6_Coupled_ODE_ver1.slx`

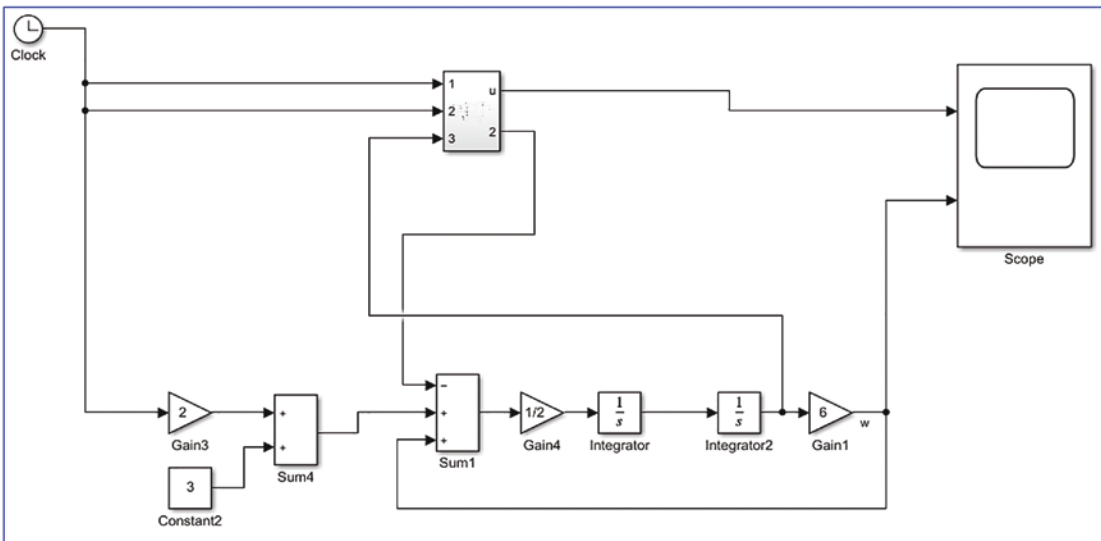
The initial model called `Ex6_Coupled_ODE_ver1.slx` shown in Figure 5-44 is a bit complicated and not easy to read. Therefore, to make it more readable, we use the Subsystem block to re-create a new and simplified model out of this model.

You can create a subsystem out of the existing model in two different ways. The first way is to select blocks using the left mouse button as highlighted in Figure 5-45 and then press `Ctrl+G` keys on the keyboard or right-click and select `Create Subsystem` from

Selection. Once this step is completed, the subsystem is created from the selection; see Figure 5-46.

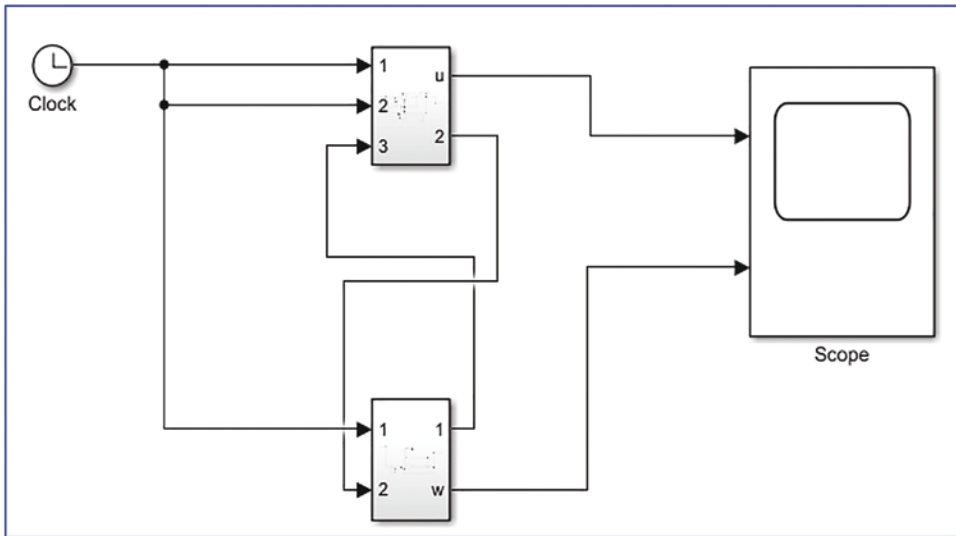


**Figure 5-45.** Simulink model, *Ex6\_Coupled\_ODE\_ver1.slx*, step 1: how to create a Subsystem



**Figure 5-46.** Simulink model, *Ex6\_Coupled\_ODE\_ver1.slx*, step 2: Subsystem created

Similarly, you can select the blocks Gain3, Constant2, Sum4, Sum1, Gain4, Integrator, Integrator2, and Gain1 for  $w$  signal using the left mouse button and press Ctrl+G (Create Subsystem from Selection) on the keyboard. Subsequently, you obtain the simplified model with two subsystems, as shown in Figure 5-47, that you can save with a new file name: Ex6\_Coupled\_ODE\_ver2.slx.



**Figure 5-47.** Simplified Simulink model, Ex6\_Coupled\_ODE\_ver2.slx with two subsystems

To see or edit the model block parameters or connections, you can double-click the Subsystem block or select the subsystem block with the left mouse button and then use the right-click option of Open.

Let's simulate both models, Ex6\_Coupled\_ODE\_ver1.slx and Ex6\_Coupled\_ODE\_ver2.slx, for  $t = [0, 5]$ , from MATLAB using the `sim()` function and compare their simulation results. In addition, you should adjust the settings of the Scope block  Log data to workspace and adjust the output data variable name ► OUT and format type ►

dataset Save format: Dataset in both models: Ex6\_Coupled\_ODE\_ver1.slx and Ex6\_Coupled\_ODE\_ver2.slx. Now, from MATLAB, you can recall these models and simulate them for five seconds, for instance, using the following short script (Sim\_Models.m) for ten times to find out if there is any difference between their simulation time:

```

clc;
%%
clearvars
%%
for ii = 1:10
tic;
OUT2 = sim('Ex6_Coupled_ODE_ver2.slx', 5);
Sim_Time2(ii) = toc;
tic;
OUT1 = sim('Ex6_Coupled_ODE_ver1.slx', 5);
Sim_Time1(ii) = toc;
end
%% Compare simulation time
fprintf('Simulation time of Model 1: %f \n ', mean(Sim_Time1))
fprintf('Simulation time of Model 2: %f \n ', mean(Sim_Time2))
%% Compare simulation results
figure(1)
plot(OUT1.OUT{1}.Values.Time, OUT1.OUT{1}.Values.Data, 'r*')
hold on
plot(OUT2.OUT{1}.Values.Time, OUT2.OUT{1}.Values.Data, 'b-',
'linewidth', 2)
xlabel('Time, [s]')
ylabel('u(t)')
legend('Model 1', 'Model 2', 'location', 'NE')
figure(2)
plot(OUT1.OUT{2}.Values.Time, OUT1.OUT{2}.Values.Data, 'r*')
hold on
plot(OUT2.OUT{2}.Values.Time, OUT2.OUT{2}.Values.Data, 'b-',
'linewidth', 2)
xlabel('Time, [s]')
ylabel('w(t)')
legend('Model 1', 'Model 2', 'location', 'NE')

```

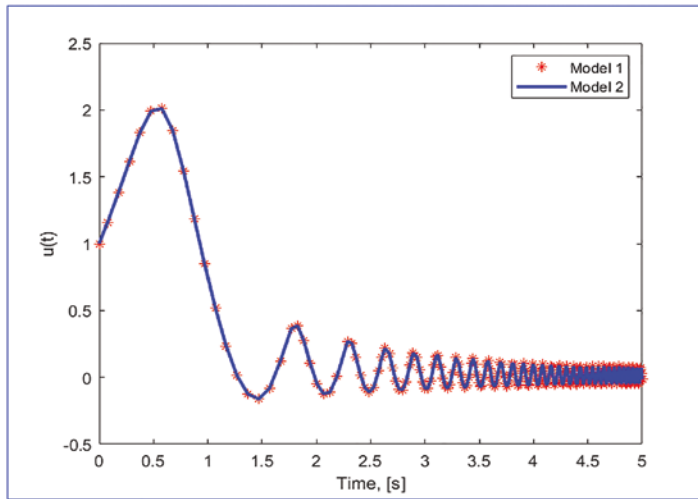
Note that these models are stored in your current MATLAB directory or you should have added their location directory to the MATLAB's path directory list using `addpath()`.

Once the simulation is finished, the following output will be displayed in the Command window:

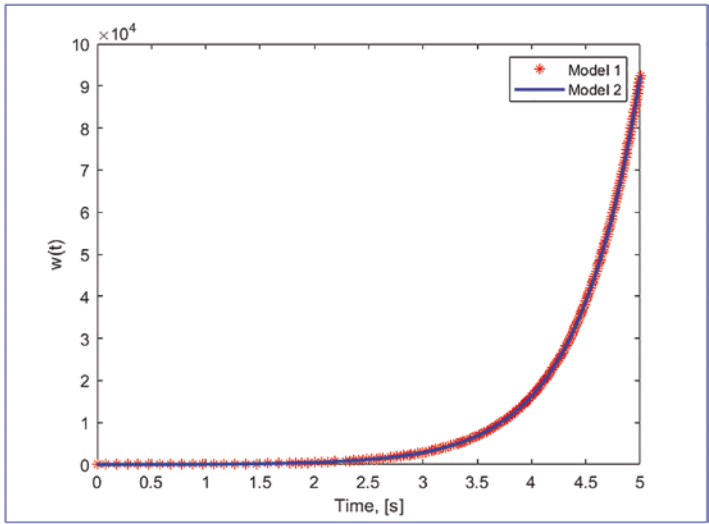
Simulation time of Model 1: 0.341179

Simulation time of Model 2: 0.384217

Also, these two plot figures shown in Figure 5-48 and Figure 5-49 will be displayed.



**Figure 5-48.** Comparison of simulation results of *Ex6\_Coupled\_ODE\_ver1.slx* and *Ex6\_Coupled\_ODE\_ver2.slx* for  $u(t)$





**Figure 5-49.** Comparison of simulation results of *Ex6\_Coupled\_ODE\_ver1.slx* and *Ex6\_Coupled\_ODE\_ver2.slx* for  $w(t)$

The simulation results from the script `Sim_Models.m` for ten times show that there is not a significant difference in simulation time of these two models.



## Simulink Model Analysis and Diagnostics

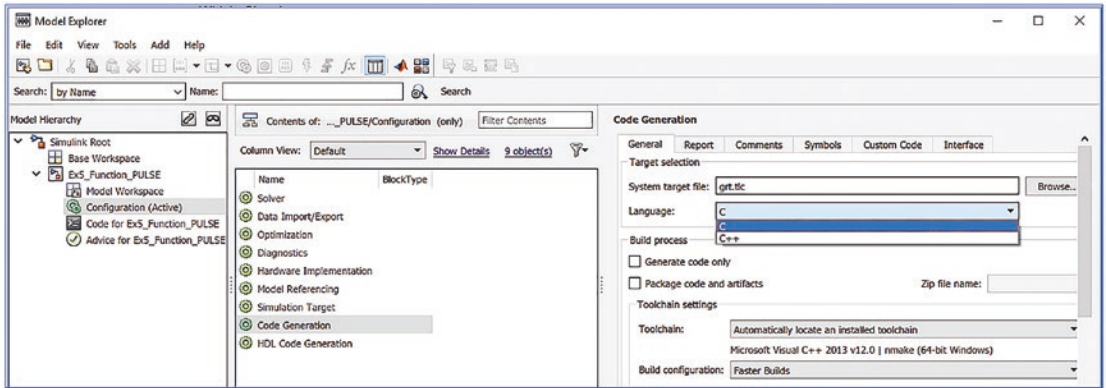
Simulink Model Analysis and Diagnostics tools provide good assistance to programmers for improving their models in terms of simulation speed, efficiency, and elimination of inaccurate and inefficient simulations. Therefore, it is recommended to perform analysis and diagnostics for efficiency and adequacy of employed blocks and combinations, chosen solver type, and many other options. All of these options can be explored via the

Model Explorer  and the Model Advisor  tools. Via the Model Explorer tools, you can generate C/C++ code of a Simulink model, obtain a profile report of a model, start a model advisor, reset the configuration parameters of solver, view input/output, optimize the models, generate code, and much more. Let's look at some of the tools within the Model Explorer, considering the previous example.



## Code Generation


Code generation (see Figure 5-50) can be accessed via the Model Explorer  or Model Configuration Parameters  buttons.

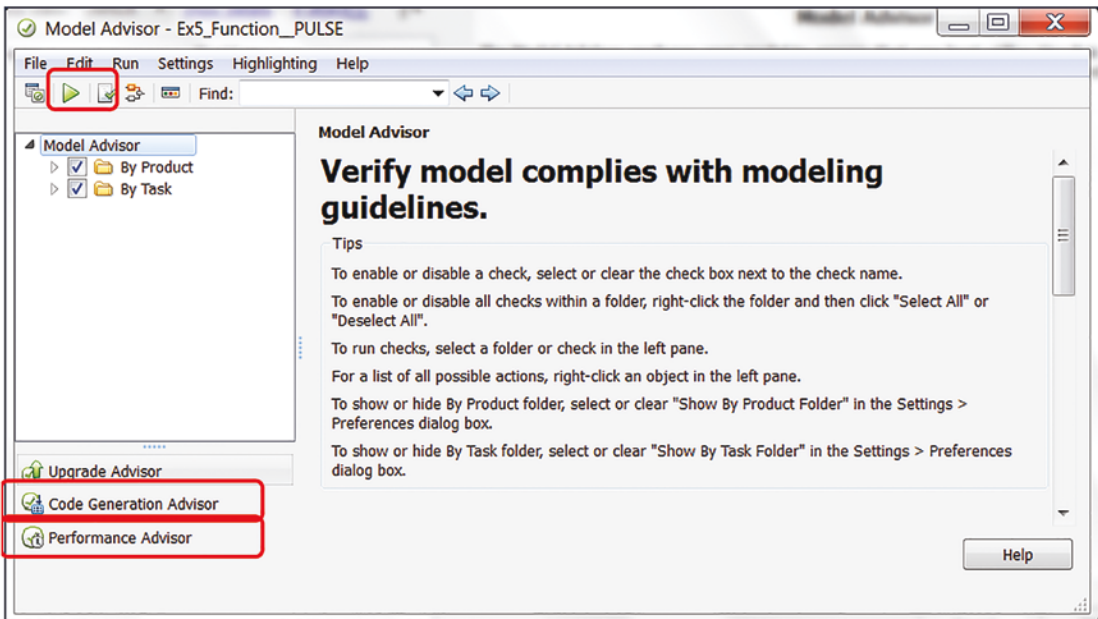


**Figure 5-50.** Model Explorer tools


After clicking Generate Code Only and Package Code and Artifacts, click the Generate Code button. Subsequently, the C code (C is the chosen language) will be generated. Note that there are some constraints in code generation; for instance, a chosen solver has to be a fixed step, and not all blocks used are compatible with code generation in C/C++. If these or other such requirements are not satisfied, the C/C++ code cannot be generated. Also, the code generation process depends on the installed compiler type and version.

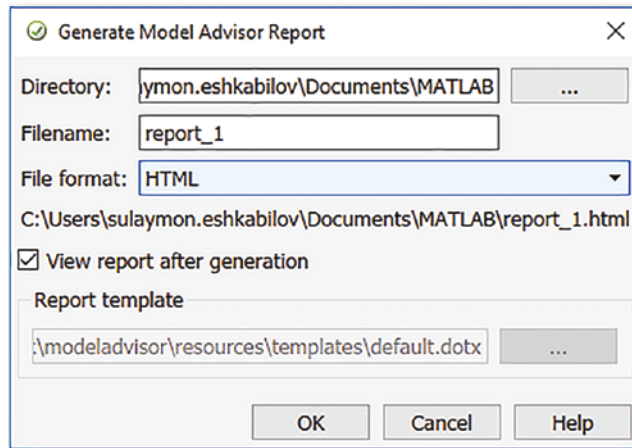
## Model Advisor

Model Advisor  tools (see Figure 5-51) can be helpful in identifying where problems have occurred within a model and where optimization is required. It identifies problems with code generation and model performance by product, by task types, or both.



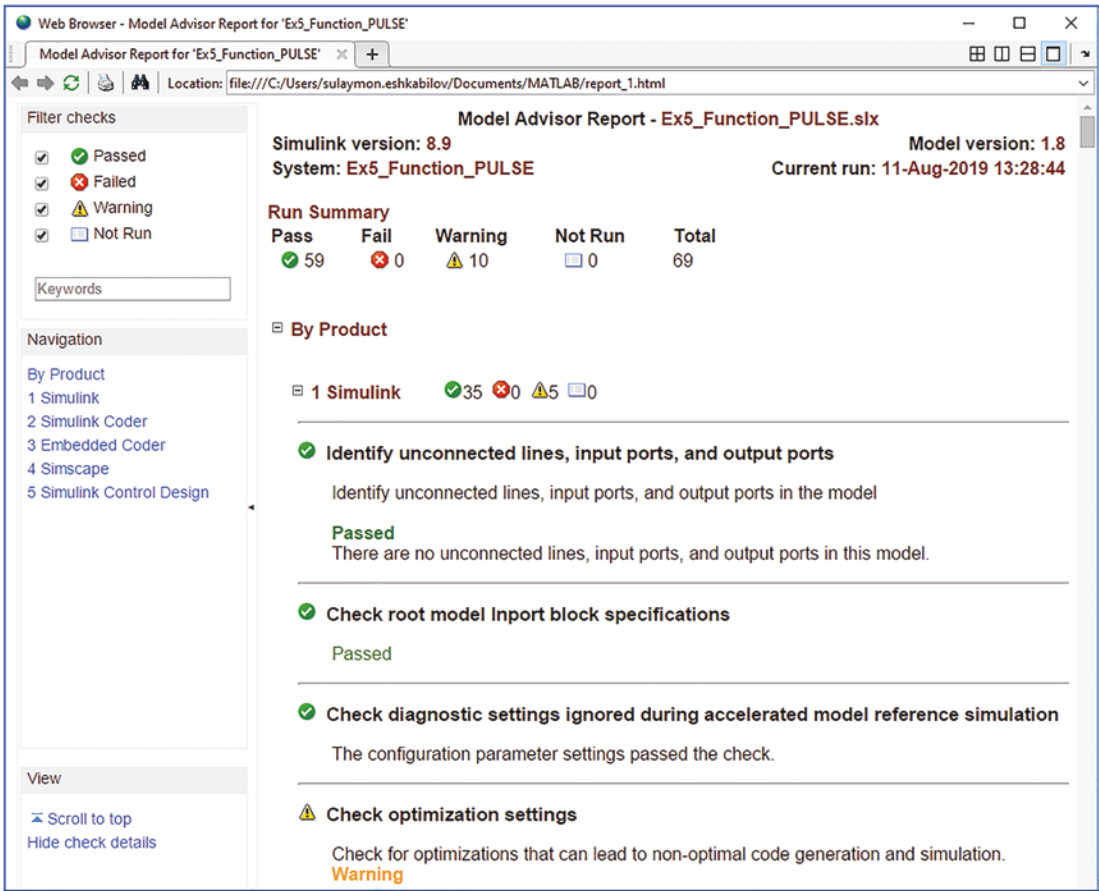
**Figure 5-51.** Model Advisor tools

You first choose which process to get help/advice from in the Model Advisor and then click the Run  button. In this example, we chose the Model Advisor with By Product and By Task. Once the Model Advisor is launched, all diagnostic checks of the model (Ex5\_Function\_PULSE.slx) are run, and the report of all passed, failed, warning, and not run points is prepared. You can view the report by clicking the Generate Report button in the Model Advisor window. The Generate Report button opens the Generate Model Advisor Report window, from which you can select the directory (where to save the generated report), file name, file format (HTML by default, PDF, or Word), and check mark option to view the report after it's generated; see Figure 5-52. We chose HTML, which is the default report format of the Model Advisor.



**Figure 5-52.** *The Generate Model Advisor Report window*

Moreover, there are warnings concerning double precision operations used by the blocks of the model. The blocks (Interpreted MATLAB Function) are not supported by code generation. In addition, the Clock, Integrator, Integrator1, Signal Generator, and Interpreted MATLAB Function blocks are not recommended for C/C++ production deployment.



**Figure 5-53.** Model Advisor Report

Figure 5-53 shows part of the Model Advisor Report for Ex5\_Function\_PULSE.slx. The report is created in HTML format and shows 59 Pass, 0 Fail, 10 Warning, and 0 Not Run, for a total of 69 Run. By scrolling down the report, you can see where the model passed and where it had some warning issues, such as optimization settings. It's recommended to set the parameter of "Remove Code from Floating-Point to Integer Conversions That Warms Out-Of-Range Values (EfficientFloat2IntCast)" to on. Another recommendation is to set the parameter "Inline invariant signals (InlineInvariantSignals)" to on. Furthermore, another warning is Check Data Store Memory blocks for multitasking, strong typing, and shadowing issues. Duplicate data store names checking is not set to error. Duplicate usage of data store names can lead to unintended shadowing of data stores of higher model scope. For this reason, consider changing the duplicate data store names setting to error.

Another interesting warning is linked to bus signals. The warning says: “Check bus signals treated as vectors.” The Bus signal was treated as a vector by the Simulink software. Identify bus signals in the model that are treated as vectors by the Simulink software.

Bus signal feeding input port 1 of the block: Ex5\_Function\_PULSE/Interpreted MATLAB Function. Bus signal is feeding input port 1 of the block in Ex5\_Function\_PULSE/ Scope.

**Recommended Action:** The model contains bus signals that the Simulink software implicitly converts to vectors. However, the model is not configured to explicitly convert these signals to vectors. To fix this issue, insert Bus To Vector blocks at the imports of the blocks listed earlier.

You can do this automatically, by either pressing the modify button below or running the `Simulink.BlockDiagram.addBusToVector` function. You can do this manually using the Simulink ► Signal Attributes library.

By studying the Model Advisor’s reports, you can improve your model by removing bugs, simulation bottlenecks, and unwanted warnings, and substituting some of the inefficient blocks in the model.

In addition to the Model Advisor, you can also employ the Optimization tools under Model Explorer or the Configuration parameters to optimize parameters and blocks in our model. In addition, to locate bugs or bottlenecks, you can use debugging tools. They can be accessed via the menu bar: Simulation ► Debug ► Debug Model. Another way to learn about the model’s performance is from the menu bar: Analysis ► Performance Tools ► Show Profile Report (select) and then ► Performance Tools ► Performance Advisor. The options are displayed in Figure 5-54.

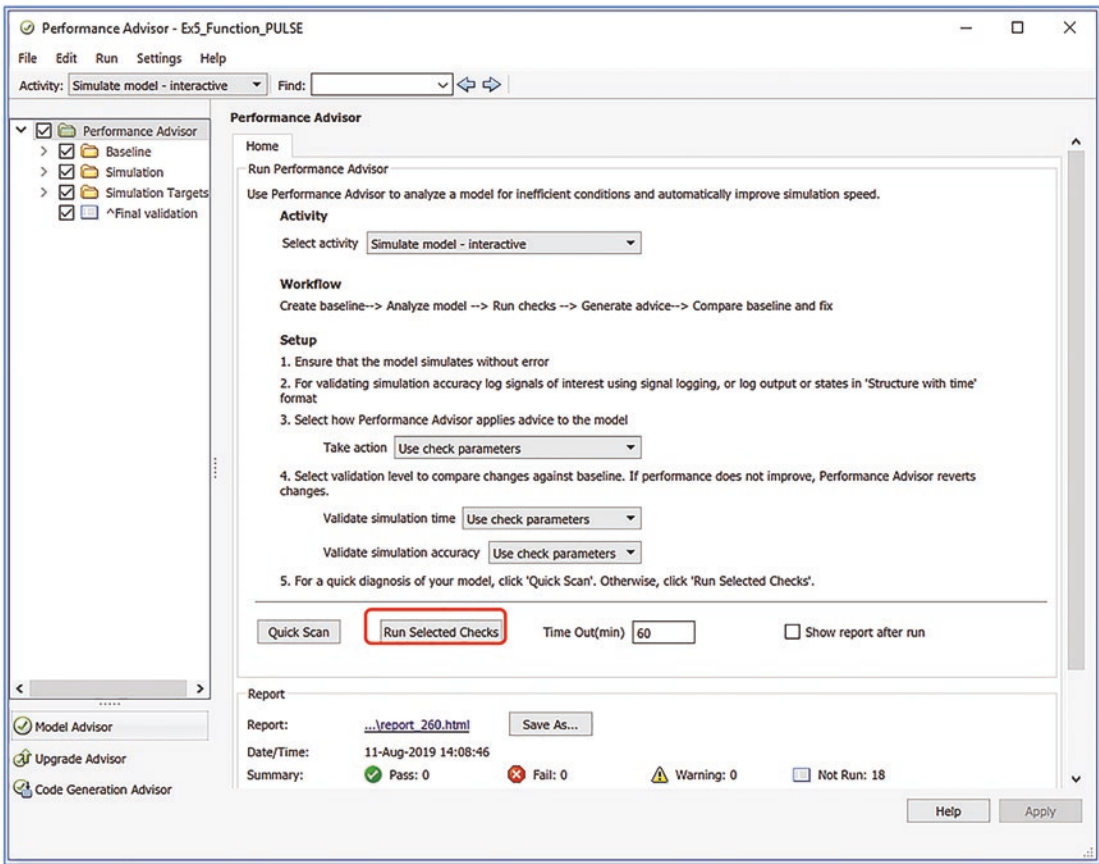
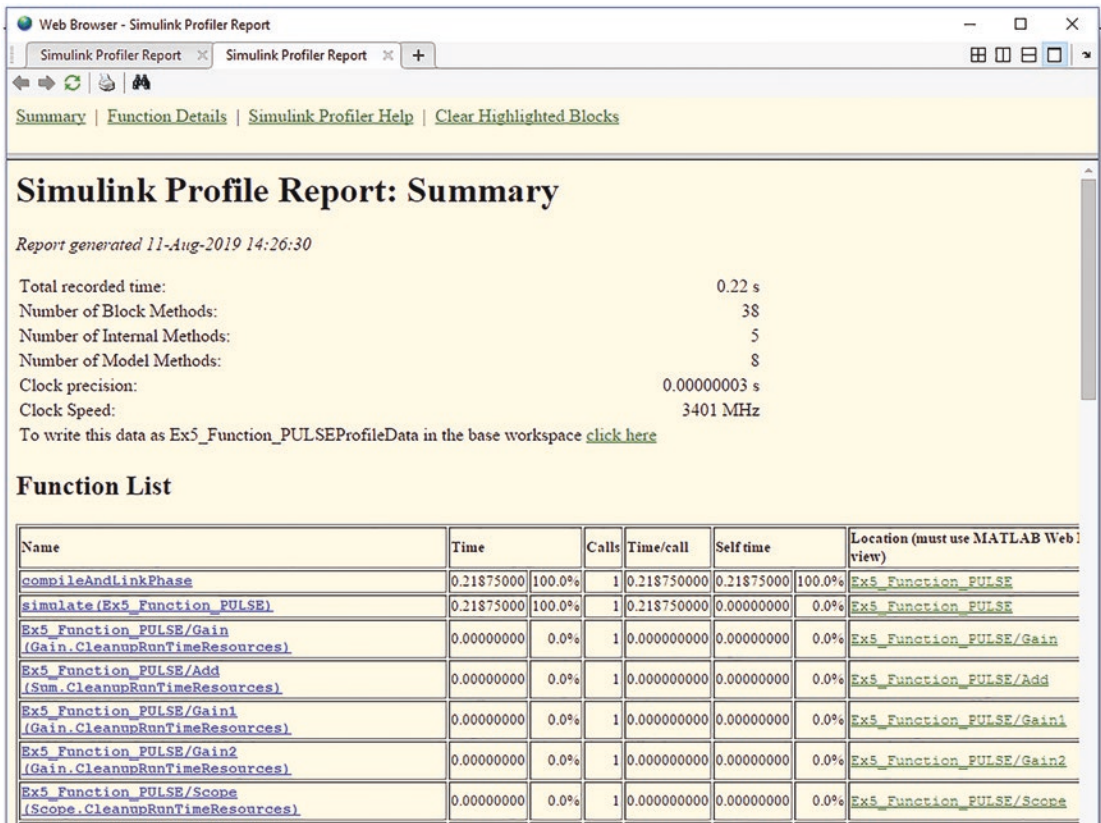


Figure 5-54. Performance Advisor options

After clicking Run Selected Checks (see Figure 5-55), the Simulink Profile Report is displayed. It's composed of the Summary and Simulink Performance Advisor Report, and it displays a complete picture of the model and its simulation processes.



**Figure 5-55.** Simulink Model Profile Report Summary

Thus, it is recommended that you run the Model Advisor and Performance Advisor options to obtain the many help hints to improve your model's performance. Also, the profile report generator can be recalled and executed using commands in order to locate inefficient operations and blocks of the model.

```
>> profile on; sim('Ex5_Function_PULSE.mdl'); profile viewer
```

The profile report generator works well with all M-files and Simulink models and provides comprehensive reports including bottlenecks within a code/script/model in terms of computation and execution time spent on each command and operation.

## Summary

In this chapter, we covered most of the essential graphical programming tools and some common blocks in the Simulink package, including signal sources, matrix operations, integration, visualization, signal routing, and C code generation. Moreover, the chapter highlighted and demonstrated, via numerical simulation examples, a few salient points on how to adjust parameters, use solver tools, set error tolerances, and improve the performance of Simulink models. In addition, you learned how to select and adjust solvers in Simulink.

You learned how to create subsystems from existing models and how to associate Simulink models with MATLAB scripts and function files. Moreover, you learned how to execute Simulink models from MATLAB scripts and acquire the Simulink model simulation results into the MATLAB workspace. In addition, you worked with the Simulink Model Analysis and Diagnostics tools and learned how to obtain Model Advisor and Performance Reports.

## Exercises for Self-Testing

### Exercise 1

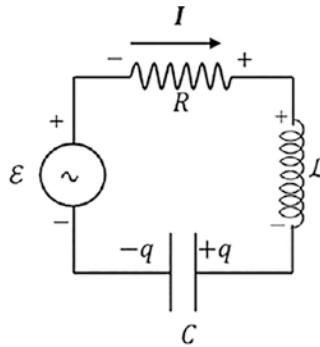
Build a Simulink model to compute values of the cosine function  $g(t) = \cos(\omega t)$  for  $t = 0 \dots 3$  with 3,000 incremental steps and  $\omega = [\pi, 2\pi, 3\pi, 5\pi, 7\pi]$  specified in MATLAB. Simulate your Simulink model using MATLAB with the `sim()` command using a `[for ... end]` or `[while .. end]` loop for all values of  $\omega$ .

### Exercise 2

The equation for charge in a resistor-inductance-capacitor (RLC) circuit (shown in the below figure) in a series is determined by Kirchhoff's law:

$$L\dot{q} + R\dot{q} + q/C = \varepsilon_{\max} \cos(\omega t)$$





Create a Simulink model to simulate the given RLC system expressed by the second-order differential equation for  $q(t)$  with the input arguments of  $R$ ,  $L$ ,  $C$ ,  $\omega$ , and  $t$ . Create a Simulink model to simulate the given RLC system.

Take  $R = 100 \Omega$ ,  $L = 200 \text{ H}$ ,  $C = 0.02 \mu\text{F}$ ,  $\omega = 60 \text{ rad/s}$ .

### Exercise 3

The acceleration of a skydiver is determined by the following:

$$a = g \left( 1 - \frac{v^2}{3600} \right)$$

where  $g = 9.81 \text{ m/sec}^2$ .

Create a Simulink model to simulate the acceleration of a skydiver.

### Exercise 4

A truck of mass  $m$  accelerates from rest at  $t = 0$  with constant power  $P$  along a level road.

The speed of the truck as a function of time is given by  $v(t) = \left( \frac{2P}{m} \right)^{\frac{1}{2}} \sqrt{t}$ . If  $x = 0$  at

time  $t = 0$ , the position function  $x(t)$  is given by  $x(t) = \left( \frac{8P}{9m} \right)^{\frac{1}{2}} \sqrt{t^3}$ , where  $P = 550 \text{ kW}$  and  $m = 15000 \text{ kg}$ .

1. Write an inline function to compute the position of the truck from the function  $x(t)$  as a function of time  $t$ .

2. Create a Simulink model to obtain numerical values of  $v(t)$ ,  $x(t)$  as a function of  $t$  in the MATLAB workspace and compare the results with the ones obtained from the function handle and inline function.
3. Build plots of  $x(t)$  versus  $t$  and  $v(t)$  versus  $t$  in two separate plot figures.

## Exercise 5

Create a Simulink model with three subsystems to compute numerical solutions ( $x_A(t)$ ,  $x_B(t)$ ,  $x_C(t)$ ) of the following second-order coupled differential equations.

$$kx_k + kx_A(t) + m\ddot{x}_A(t) = F_D(t) + kx_B(t)$$

$$2kx_B(t) + m\ddot{x}_B(t) = k(x_C(t) + x_B(t))$$

$$m\ddot{x}_C(t) = k(x_k + x_B(t) - x_C(t))$$

$$x_A(0) = 0, x_B(0) = x_k, x_C(0) = 2x_k, x_k = 1.5,$$

$$\dot{x}_A(0) = \dot{x}_B(0) = \dot{x}_C(0) = 0$$

$$F_D(t) = F_0 \cos(t\omega), F_0 = 5.75, \omega = 3.20;$$

$$t = [0, 25], m = 2; k = 32.$$

Hint: For more information about ordinary differential equations, see Chapter 8.