Simulink Basics Tutorial

Simulink is a graphical extension to MATLAB for the modeling and simulation of systems. In Simulink, systems are drawn on screen as block diagrams. Many elements of block diagrams are available (such as transfer functions, summing junctions, etc.), as well as virtual input devices (such as function generators) and output devices (such as oscilloscopes). Simulink is integrated with MATLAB and data can be easily transferred between the programs. In this tutorial, we will introduce the basics of using Simulink to model and simulate a system.

Simulink is supported on Unix, Macintosh, and Windows environments, and it is included in the student version of MATLAB for personal computers. For more information on Simulink, contact the MathWorks.

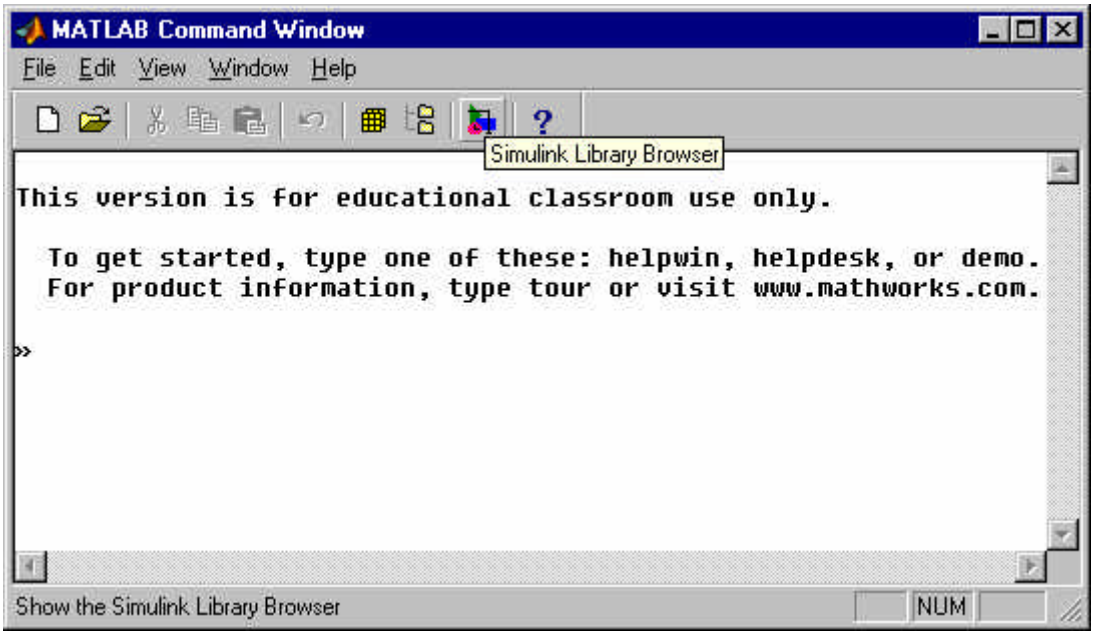
The idea behind these tutorials is that you can view them in one window while running Simulink in another window. Do not confuse the windows, icons, and menus in the tutorials for your actual Simulink windows. Most images in these tutorials are not live - they simply display what you should see in your own Simulink windows. All Simulink operations should be done in your Simulink windows.

Starting Simulink

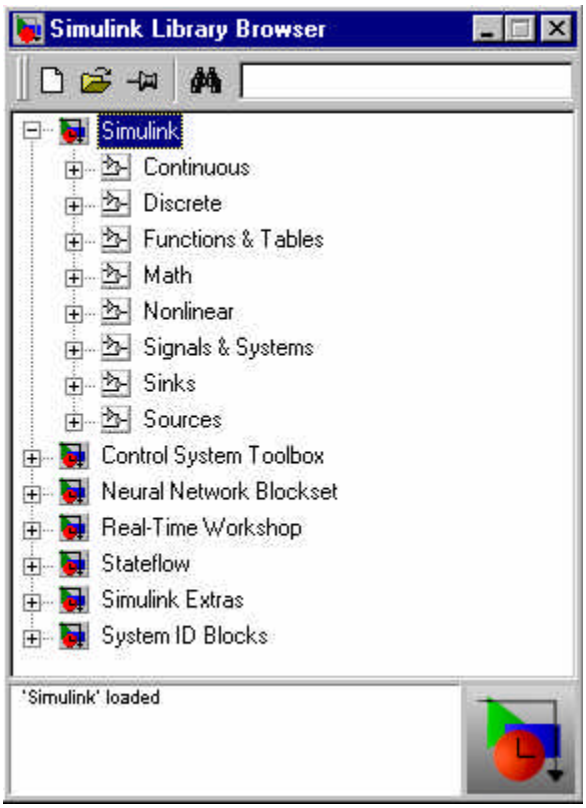
Simulink is started from the MATLAB command prompt by entering the following command:

```
simulink
```

Alternatively, you can click on the "Simulink Library Browser" button at the top of the MATLAB command window as shown below:



The Simulink Library Browser window should now appear on the screen. Most of the blocks needed for modeling basic systems can be found in the subfolders of the main "Simulink" folder (opened by clicking on the "+" in front of "Simulink"). Once the "Simulink" folder has been opened, the Library Browser window should look like:



Basic Elements

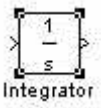
There are two major classes of elements in Simulink: **blocks** and **lines**. Blocks are used to generate, modify, combine, output, and display signals. Lines are used to transfer signals from one block to another.

Blocks

The subfolders underneath the "Simulink" folder indicate the general classes of blocks available for us to use:

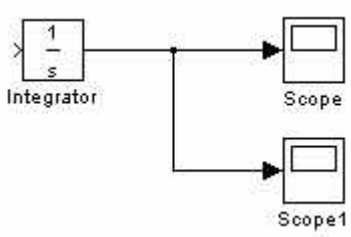
- Continuous: Linear, continuous-time system elements (integrators, transfer functions, state-space models, etc.)
- Discrete: Linear, discrete-time system elements (integrators, transfer functions, state-space models, etc.)
- Functions & Tables: User-defined functions and tables for interpolating function values
- Math: Mathematical operators (sum, gain, dot product, etc.)
- Nonlinear: Nonlinear operators (coulomb/viscous friction, switches, relays, etc.)
- Signals & Systems: Blocks for controlling/monitoring signal(s) and for creating subsystems
- Sinks: Used to output or display signals (displays, scopes, graphs, etc.)
- Sources: Used to generate various signals (step, ramp, sinusoidal, etc.)

Blocks have zero to several input terminals and zero to several output terminals. Unused input terminals are indicated by a small open triangle. Unused output terminals are indicated by a small triangular point. The block shown below has an unused input terminal on the left and an unused output terminal on the right.



Lines

Lines transmit signals in the direction indicated by the arrow. Lines must always transmit signals from the output terminal of one block to the input terminal of another block. One exception to this is that a line can tap off of another line. This sends the original signal to each of two (or more) destination blocks, as shown below:

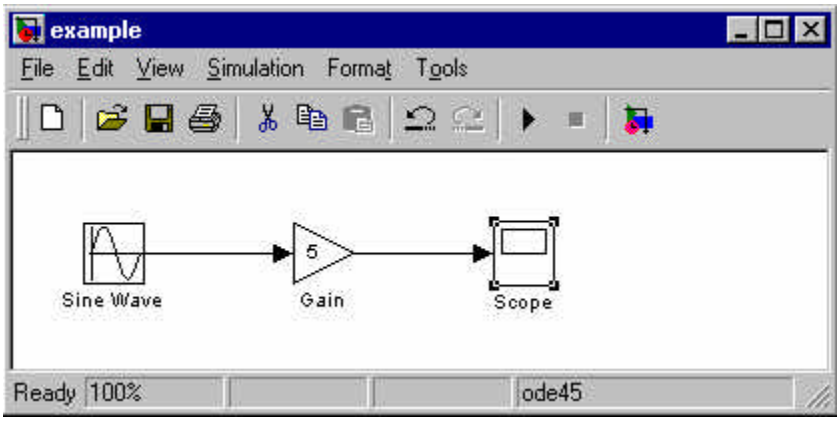


Lines can never inject a signal *into* another line; lines must be combined through the use of a block such as a summing junction.

A signal can be either a scalar signal or a vector signal. For Single-Input, Single-Output systems, scalar signals are generally used. For Multi-Input, Multi-Output systems, vector signals are often used, consisting of two or more scalar signals. The lines used to transmit scalar and vector signals are identical. The type of signal carried by a line is determined by the blocks on either end of the line.

Building a System

To demonstrate how a system is represented using Simulink, we will build the block diagram for a simple model consisting of a sinusoidal input multiplied by a constant gain, which is shown below:



This model will consist of three blocks: Sine Wave, Gain, and Scope. The Sine Wave is a **Source Block** from which a sinusoidal input signal originates. This signal is transferred through a **line** in the direction indicated by the arrow to the Gain **Math Block**. The Gain block modifies its input signal (multiplies it by a constant value) and outputs a new signal through a **line** to the Scope block. The Scope is a **Sink Block** used to display a signal (much like an oscilloscope).

We begin building our system by bringing up a new model window in which to create the block diagram. This is done by clicking on the "New Model" button in the toolbar of the Simulink Library Browser (looks like a blank page).

Building the system model is then accomplished through a series of steps:

1. The necessary blocks are gathered from the Library Browser and placed in the model window.

2. The parameters of the blocks are then modified to correspond with the system we are modelling.

3. Finally, the blocks are connected with lines to complete the model.

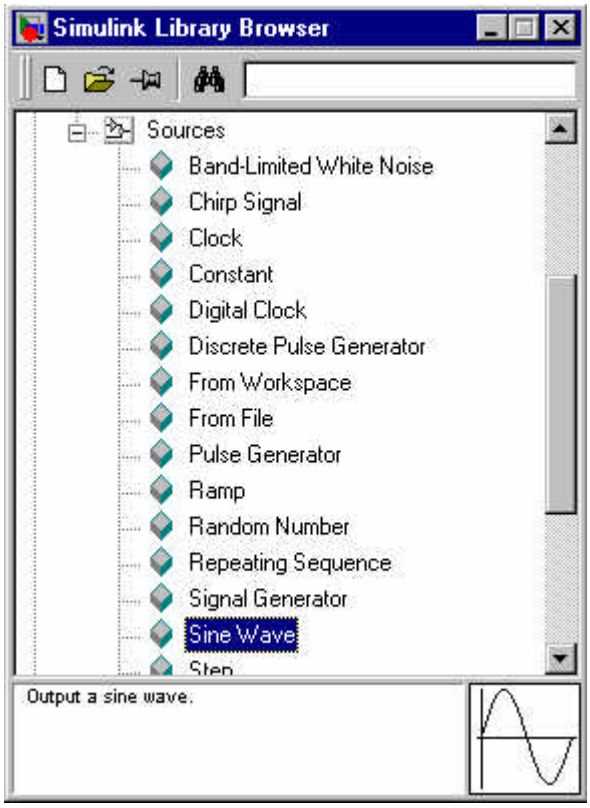
Each of these steps will be explained in detail using our example system. Once a system is built, simulations are run to analyze its behavior.

Gathering Blocks

Each of the blocks we will use in our example model will be taken from the Simulink Library Browser. To place the Sine Wave block into the model window, follow these steps:

1. Click on the "+" in front of "Sources" (this is a subfolder beneath the "Simulink" folder) to display the various source blocks available for us to use.

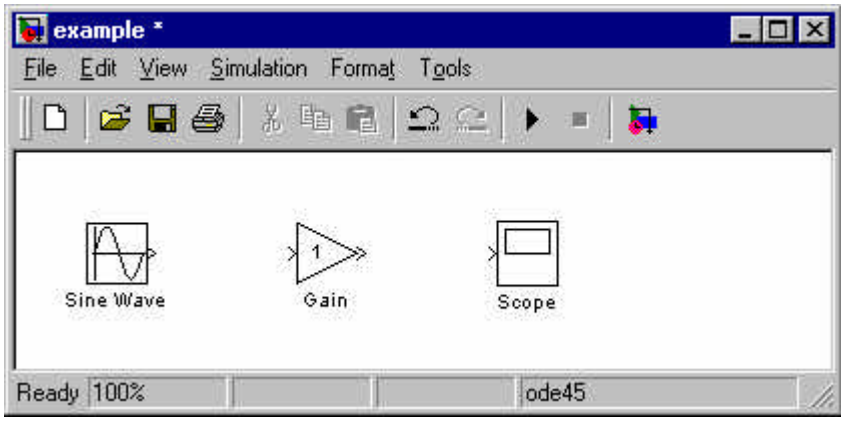
2. Scroll down until you see the "Sine Wave" block. Clicking on this will display a short explanation of what that block does in the space below the folder list:



3. To insert a Sine Wave block into your model window, click on it in the Library Browser and drag the block into your workspace.

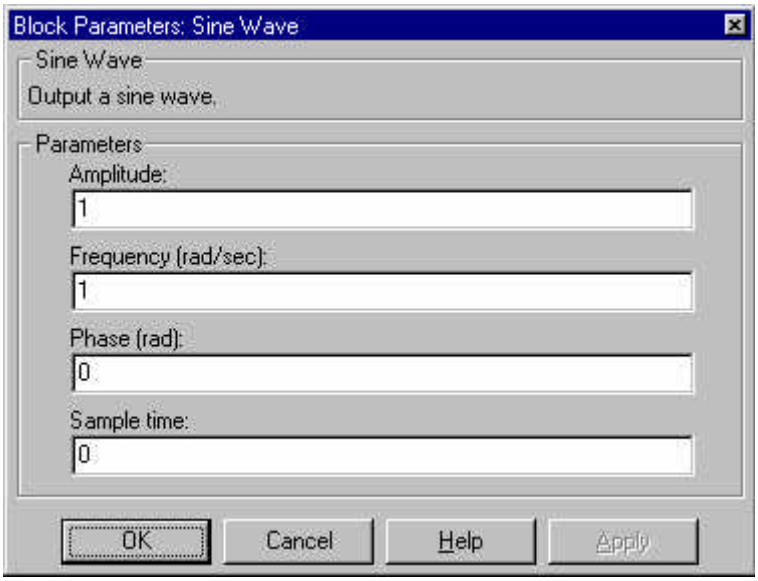
The same method can be used to place the Gain and Scope blocks in the model window. The

"Gain" block can be found in the "Math" subfolder and the "Scope" block is located in the "Sink" subfolder. Arrange the three blocks in the workspace (done by selecting and dragging an individual block to a new location) so that they look similar to the following:



Modifying the Blocks

Simulink allows us to modify the blocks in our model so that they accurately reflect the characteristics of the system we are analyzing. For example, we can modify the Sine Wave block by double-clicking on it. Doing so will cause the following window to appear:



This window allows us to adjust the amplitude, frequency, and phase shift of the sinusoidal input. The "Sample time" value indicates the time interval between successive readings of the signal. Setting this value to 0 indicates the signal is sampled continuously.

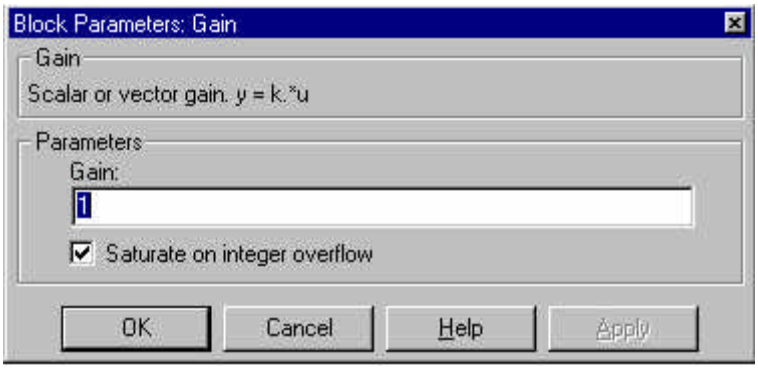
Let us assume that our system's sinusoidal input has:

- Amplitude = 2
- Frequency = pi

- Phase = pi/2

Enter these values into the appropriate fields (leave the "Sample time" set to 0) and click "OK" to accept them and exit the window. Note that the frequency and phase for our system contain 'pi' (3.1415...). These values can be entered into Simulink just as they have been shown.

Next, we modify the Gain block by double-clicking on it in the model window. The following window will then appear:



Note that Simulink gives a brief explanation of the block's function in the top portion of this window. In the case of the Gain block, the signal input to the block (u) is multiplied by a constant (k) to create the block's output signal (y). Changing the "Gain" parameter in this window changes the value of k.

For our system, we will let k = 5. Enter this value in the "Gain" field, and click "OK" to close the window.

The Scope block simply plots its input signal as a function of time, and thus there are no system parameters that we can change for it. We will look at the Scope block in more detail after we have run our simulation.

Connecting the Blocks

For a block diagram to accurately reflect the system we are modeling, the Simulink blocks must be properly connected. In our example system, the signal output by the Sine Wave block is transmitted to the Gain block. The Gain block amplifies this signal and outputs its new value to the Scope block, which graphs the signal as a function of time. Thus, we need to draw lines from the output of the Sine Wave block to the input of the Gain block, and from the output of the Gain block to the input of the Scope block.

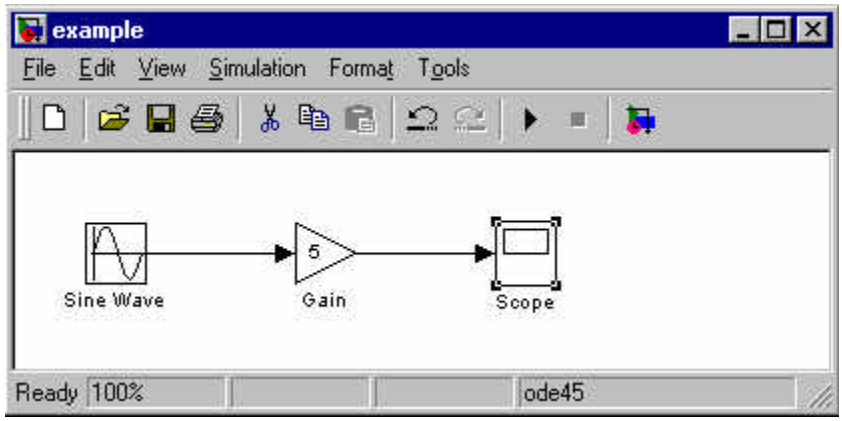
Lines are drawn by dragging the mouse from where a signal starts (output terminal of a block) to where it ends (input terminal of another block). When drawing lines, it is important to make sure that the signal reaches each of its intended terminals. Simulink will turn the mouse pointer into a crosshair when it is close enough to an output terminal to begin drawing a line, and the pointer will change into a double crosshair when it is close enough to snap to an input terminal. A signal is properly connected if its arrowhead is filled in. If the arrowhead is open, it means the signal is not connected to both blocks. To fix an open signal, you can treat the open arrowhead as an output terminal and continue drawing the line to an input terminal in the same manner as

explained before.

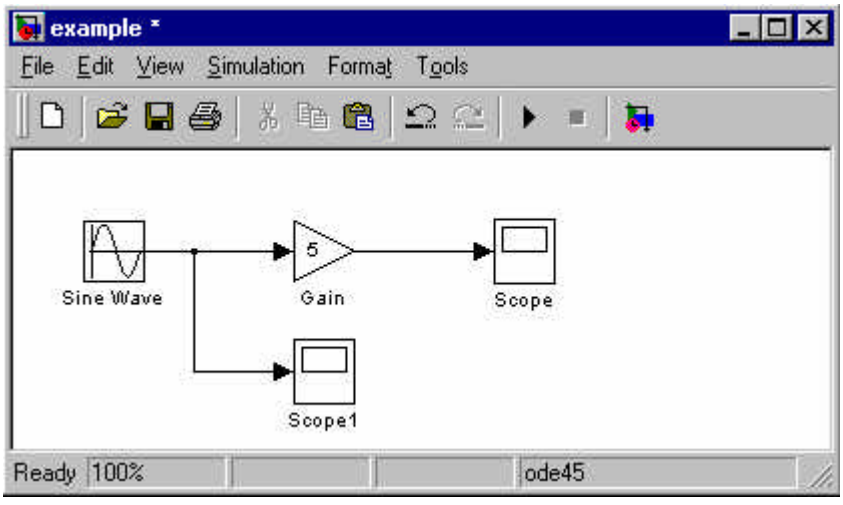
Properly Connected Signal **Open Signal**

When drawing lines, you do not need to worry about the path you follow. The lines will route themselves automatically. Once blocks are connected, they can be repositioned for a neater appearance. This is done by clicking on and dragging each block to its desired location (signals will stay properly connected and will re-route themselves).

After drawing in the lines and repositioning the blocks, the example system model should look like:



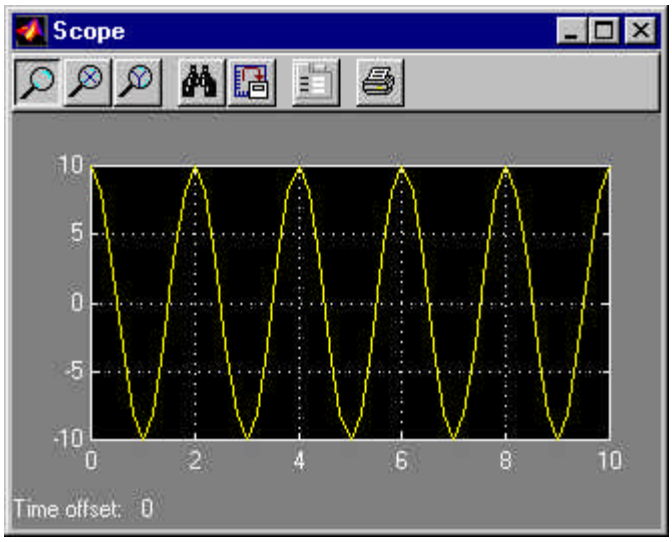
In some models, it will be necessary to branch a signal so that it is transmitted to two or more different input terminals. This is done by first placing the mouse cursor at the location where the signal is to branch. Then, using either the CTRL key in conjunction with the left mouse button or just the right mouse button, drag the new line to its intended destination. This method was used to construct the branch in the Sine Wave output signal shown below:



The routing of lines and the location of branches can be changed by dragging them to their desired new position. To delete an incorrectly drawn line, simply click on it to select it, and hit the DELETE key.

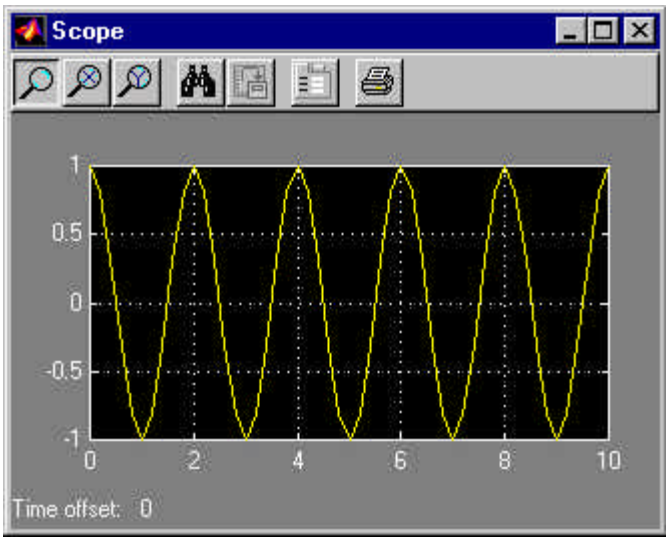
Running Simulations

Now that our model has been constructed, we are ready to simulate the system. To do this, go to the **Simulation** menu and click on **Start**, or just click on the "Start/Pause Simulation" button in the model window toolbar (looks like the "Play" button on a VCR). Because our example is a relatively simple model, its simulation runs almost instantaneously. With more complicated systems, however, you will be able to see the progress of the simulation by observing its running time in the the lower box of the model window. Double-click the Scope block to view the output of the Gain block for the simulation as a function of time. Once the Scope window appears, click the "Autoscale" button in its toolbar (looks like a pair of binoculars) to scale the graph to better fit the window. Having done this, you should see the following:



Note that the output of our system appears as a cosine curve with a period of 2 seconds and amplitude equal to 10. Does this result agree with the system parameters we set? Its amplitude makes sense when we consider that the amplitude of the input signal was 2 and the constant gain of the system was 5 (2 x 5 = 10). The output's period should be the same as that of the input signal, and this value is a function of the frequency we entered for the Sine Wave block (which was set equal to pi). Finally, the output's shape as a cosine curve is due to the phase value of pi/2 we set for the input (sine and cosine graphs differ by a phase shift of pi/2).

What if we were to modify the gain of the system to be 0.5? How would this affect the output of the Gain block as observed by the Scope? Make this change by double-clicking on the Gain block and changing the gain value to 0.5. Then, re-run the simulation and view the Scope (the Scope graph will not change unless the simulation is re-run, even though the gain value has been modified). The Scope graph should now look like the following:



Note that the only difference between this output and the one from our original system is the amplitude of the cosine curve. In the second case, the amplitude is equal to 1, or 1/10th of 10, which is a result of the gain value being 1/10th as large as it originally was.

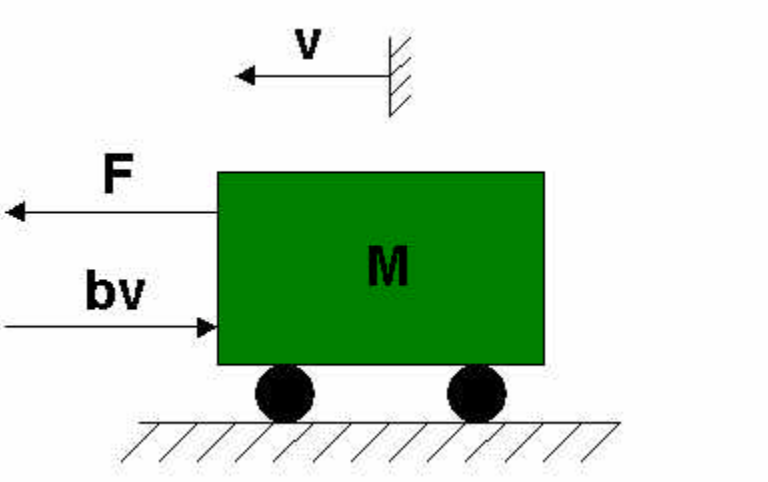
Author: RDM
Updated: 6/13/00

Modeling a First Order System in Simulink

The idea behind these tutorials is that you can view them in one window while running Simulink in another window. Do not confuse the windows, icons, and menus in the tutorials for your actual Simulink windows. Most images in these tutorials are not live - they simply display what you should see in your own Simulink windows. All Simulink operations should be done in your Simulink windows.

Free Body Diagram and System Equation

To demonstrate how Simulink can be used to investigate a real-world system, we will look at a simplified, first-order model of the motion of a car. If we assume the car to be travelling on a flat road, then the horizontal forces on the car can be represented by:



In this diagram:

- v is the horizontal velocity of the car (units of m/s).
- F is the force created by the car's engine to propel it forward (units of N).
- b is the damping coefficient for the car, which is dependent on wind resistance, wheel friction, etc. (units of N*s/m) We have assumed the damping force to be proportional to the car's velocity.
- M is the mass of the car (units of kg).

Writing Newton's Second Law for the horizontal direction thus gives:

$$M\frac{dv}{dt} = F - bv$$

For our system, we will assume that:

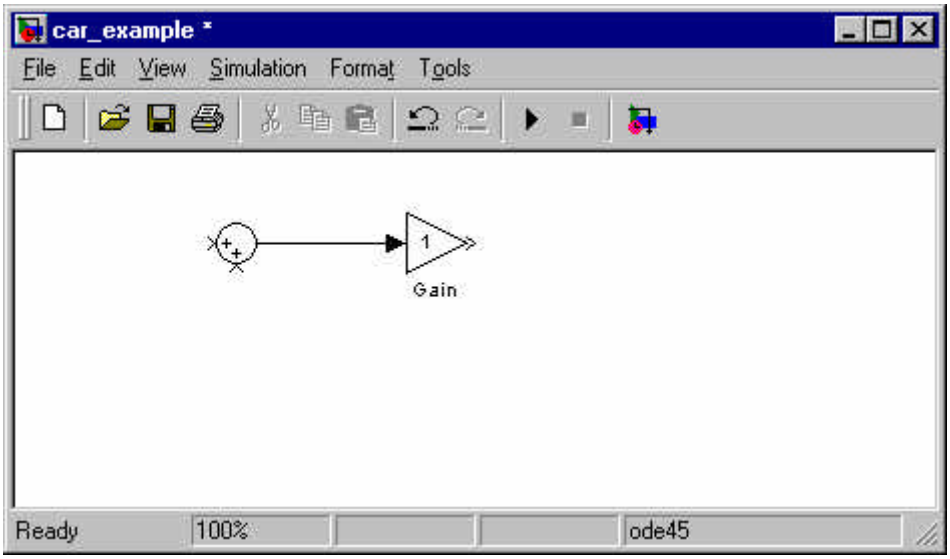
M = 1000 kg (a Dodge Neon has a mass of about 1100 kg)
b = 40 N*sec/m

Building System Model

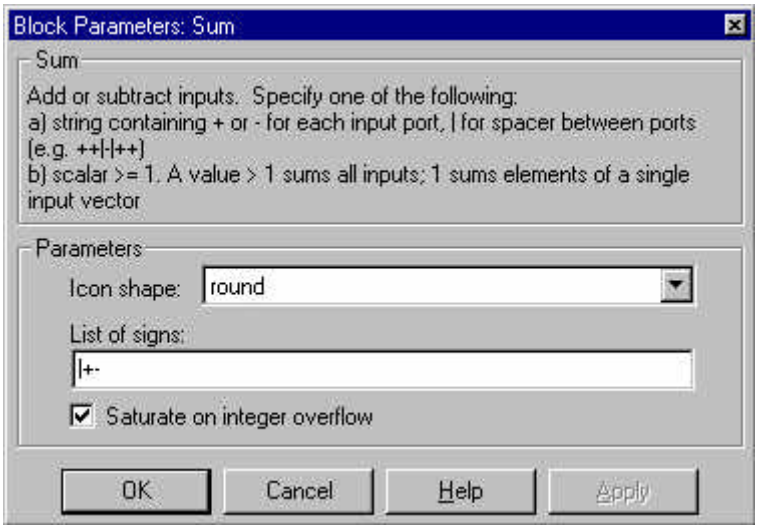
This system will be modeled in Simulink by using the system equation derived above. This equation indicates that the car's acceleration (dv/dt) is equal to the sum of the forces acting on the car (F-bv) divided by the car's mass:

$$\frac{dv}{dt} = \frac{F - bv}{M} = \frac{F - 40v}{1000}$$

To model this equation, we begin by inserting a Sum block and a Gain block (both found in the Math subfolder of the Simulink folder in the Library Browser) into a new model window. The Sum block represents adding together the forces and the Gain block symbolizes dividing by the mass. Connecting the blocks with a line gives the following in the model window:



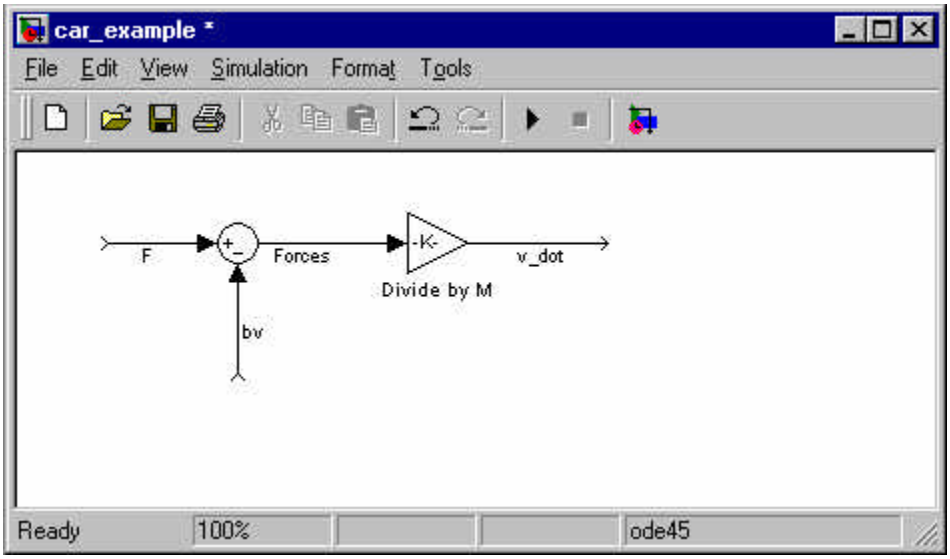
Next, we modify these blocks to properly represent our system. The Sum block needs to add the motor force (F) and subtract the damping force (bv). Thus, we double-click on this block and change the second "+" in the "List of signs" box into a "-". The Sum Block Parameters window should now look like:



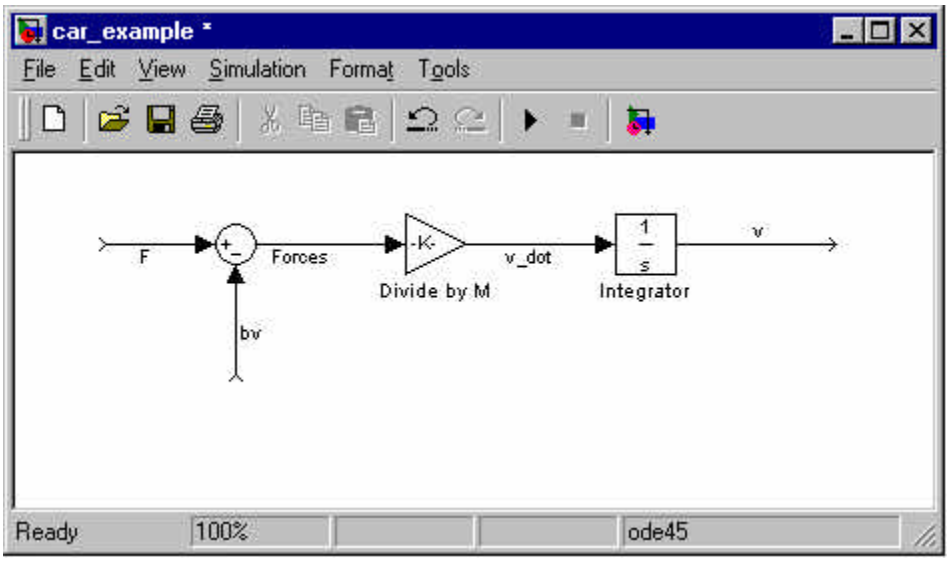
We also modify the Gain block so that it divides by the car's mass. Double-click on the block and change the Gain to 1/1000 (dividing by 1000 is the same as multiplying by 1/1000).

To keep our block diagram organized and easy to understand, we next add labels to the signals and blocks we have included so far. A signal is labeled by double-clicking on its line and entering the desired description into the text box that appears. These labels can be moved by dragging the text boxes to their desired location on the lines. A block is labeled by clicking on the text underneath it and editing the description.

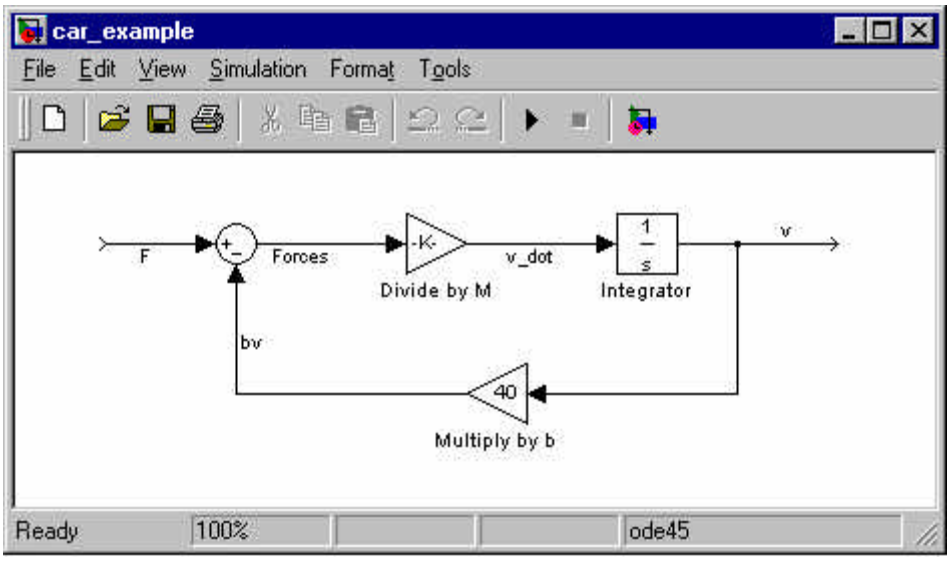
Draw lines to the open input terminals of the Sum block and open output terminal of the Gain block and label the signals and blocks in the model so that they look like:



To relate the car's acceleration (v_dot in the Simulink model) to its velocity-dependent damping force, we will integrate the v_dot signal. Place an Integrator block (from the Continuous subfolder) in the model (you do not need to change its parameters), and draw and label the velocity signal so that the model looks like:



To obtain the damping force from the velocity, we need to branch the velocity signal and multiply it by the damping coefficient (b). Branching the velocity signal is done by clicking the right mouse button anywhere on its line (or hold down CTRL and use the left mouse button) and dragging away a new signal. A Gain block is then used to multiply the velocity by the damping coefficient. Add this block to the model (from the Math subfolder) and flip it to so that it outputs to the left by clicking on **Format** then **Flip Block** (the Gain block must be selected in the model window when this is done). Finally, edit the Gain block's parameters so that its gain equals the damping coefficient of the system (40 N*sec/m). These additions to the model should cause it to look like:

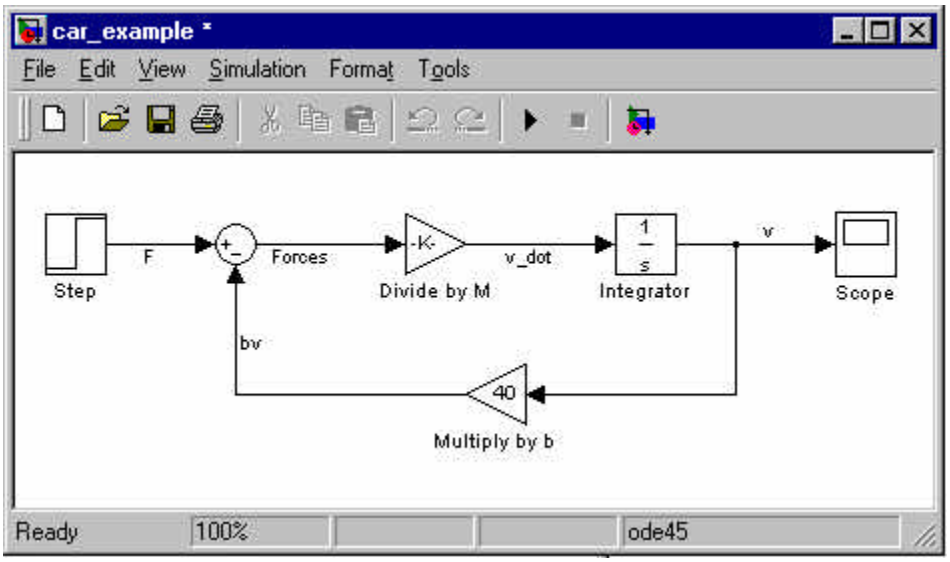


Note that the block diagram is now set up with input F (engine force) and output v (car velocity).

System Response to Step/Pulse Inputs

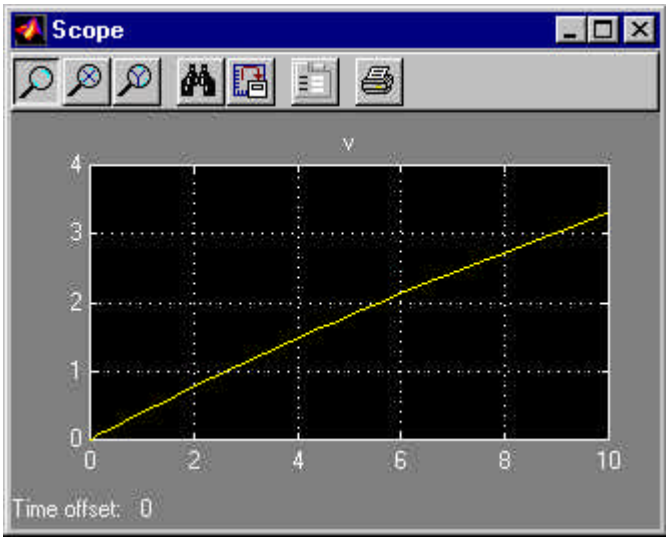
Step Input

To be able to successfully simulate the system, we need to specify an applied input, F. Let us assume the car is initially at rest, and that the engine applies a step input of F = 400 N at t = 0. This is approximately equivalent to the car's driver quickly pushing down and holding the gas pedal in a steady position starting from a stoplight. Insert a Step block from the Sources subfolder into the model, and also add a Scope block from the Sinks subfolder to monitor the system's velocity, v. The Simulink model window should now look like:

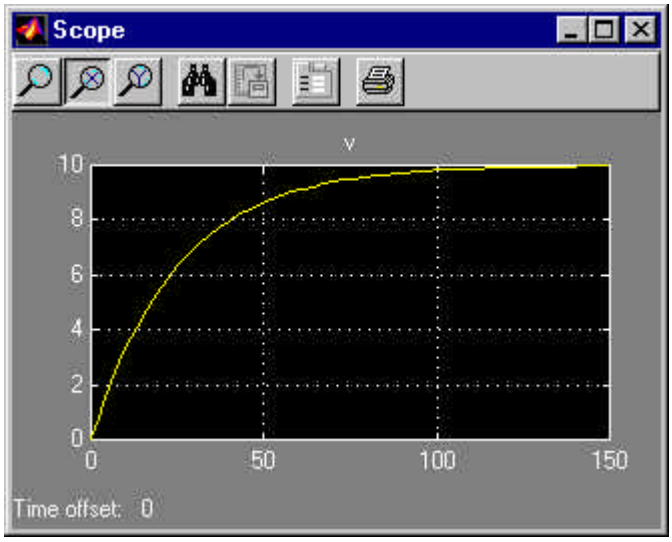


The Step block must be modified to correctly represent our system. Double-click on it, and change the Step Time to 0 and the Final Value to 400. The Initial Value can be left as 0, since the F step input starts from 0 at t = 0. The Sample Time should remain 0 so that the Step block's input is monitored continuously during simulation.

Next, run a simulation of the system (by clicking the "Start/Pause Simulation" button or selecting **Simulation**, **Start**). Once the simulation has finished, double-click on the Scope block to view the velocity response to the step input. Clicking on the "Autoscale" button (looks like a pair of binoculars) in the Scope window will produce the following graph:



Note that this graph does not appear to show the velocity approaching a steady-state value, as we would expect for the first-order response to a step input. This result is due to the settling time of the system being greater than the 10 seconds the simulation was run. To observe the system reaching steady-state, click **Simulation, Parameters** in the model window, and change the Stop Time to 150 seconds. Now, re-run the simulation and note the difference in the velocity graph:



From this graph, we observe that the system has a steady-state velocity of about 10 m/s (about 22

mph), and a time constant of about 25 seconds. Let's check these results with our original equation. For a step input of F = 400 N, the system equation is:

$$1000\frac{dv}{dt} + 40v = 400$$

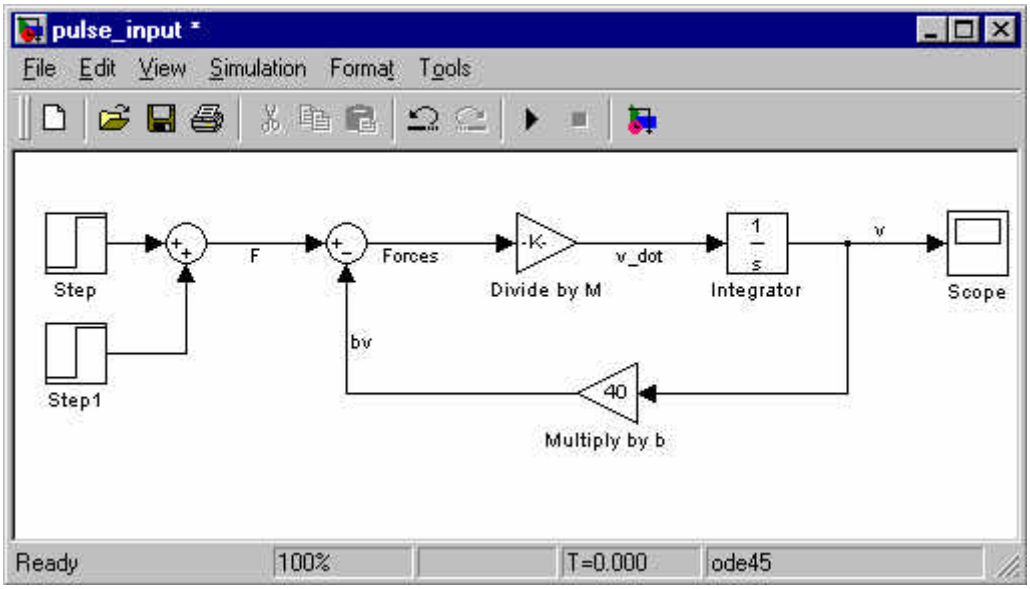
Setting dv/dt = 0 gives a steady-state velocity of 10 m/s, a result which agrees with the velocity graph above. Next, we find the time constant of the system using the characteristic equation, which is:

$$1000s + 40 = 0$$

Solving this gives the characteristic root, s = -0.04, and thus the time constant is indeed 25 seconds (tau = -1/s), as we predicted using the graph.

Pulse Input

Now, we consider the response of the system if a pulse, instead of a step, input is applied. This is approximately equivalent to the car's driver pressing and holding the gas pedal in a constant position for a specified period of time, and then releasing the pedal. To model a pulse input using Simulink, insert another Step block and a Sum block in the system as shown:

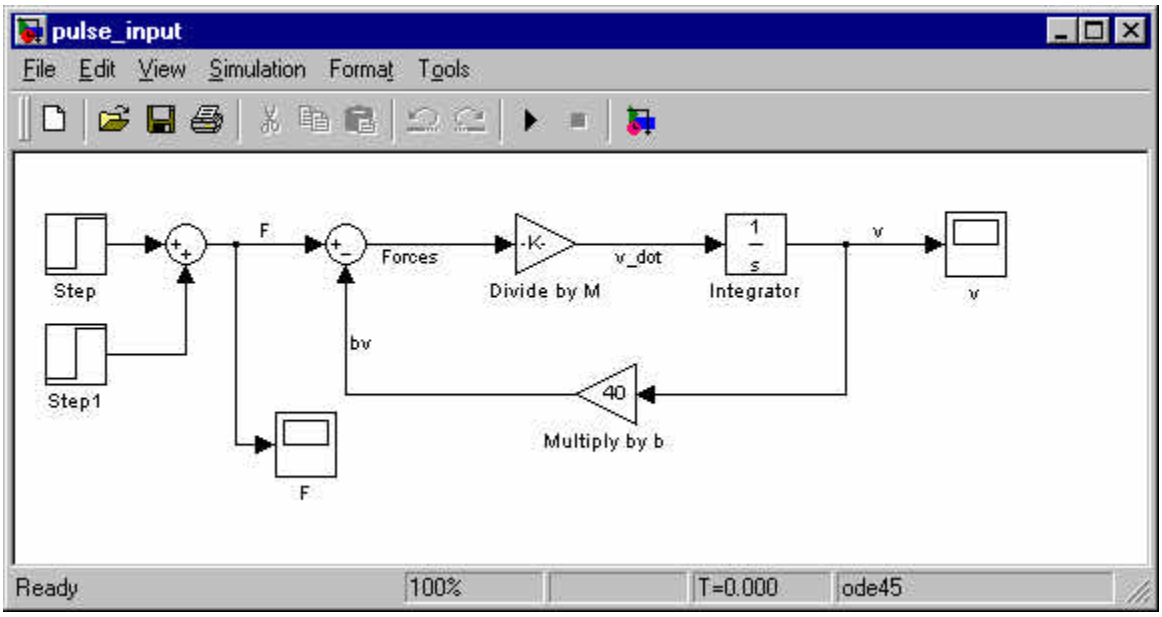


The parameters for the original "Step" block can be left as they were before. Modify the "Step1" block parameters to the following:

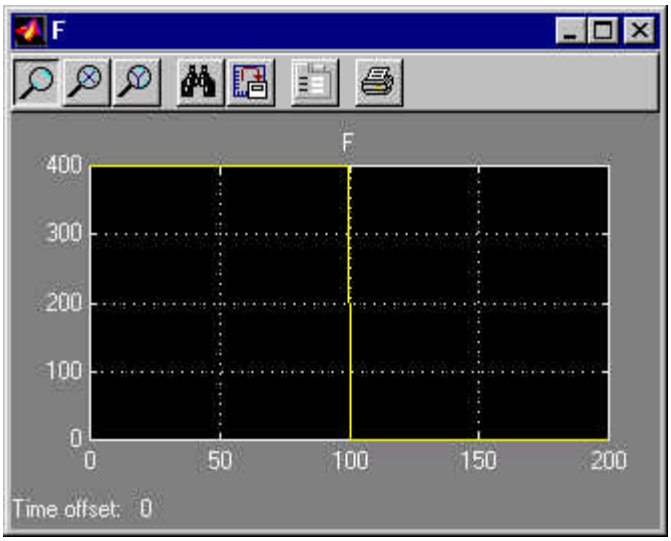
Step Time = 100
Initial Value = 0
Final Value = -400

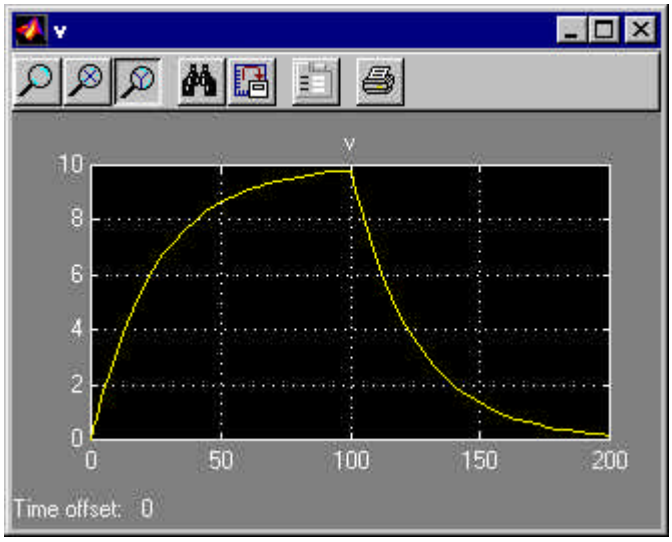
These settings enable the "Step1" block to cancel out the input from the "Step" block starting at t = 100.

To monitor the input of the system, F, we insert another scope into the model window as shown below:



Modify the simulation time (found by going to **Simulation,** then **Parameters**) to 200 seconds, then run the simulation. After autoscaling the scopes recording the F and v signals, you should see graphs that look like:





A couple of important features of the system are visible from comparing the input and output plots above. First of all, as we stated previously, the time constant of the system is 25 seconds. Thus, the time it will take for the system to reach steady-state (the settling time) should be about 100 seconds (4 * time constant). At t = 100 seconds in the simulation, the system is within 2% of its steady-state response to the original step input of 400 N. When F is instantaneously reduced to 0 at t = 100 seconds, it takes the system another 100 seconds (until t = 200 seconds) to respond to this new input. Also, notice that the steady-state response of the system to an input of F = 0 is v = 0. Physically, this means that if the driver of the car drops the accelerator while moving at any speed, the car will eventually come to rest.

Additional Examples

Variations of the mass-damper problem we have been studying can be found by following the links below:

First Order System: Ramp Response
First Order System: Linearizing System Equation
First Order System: System Identification

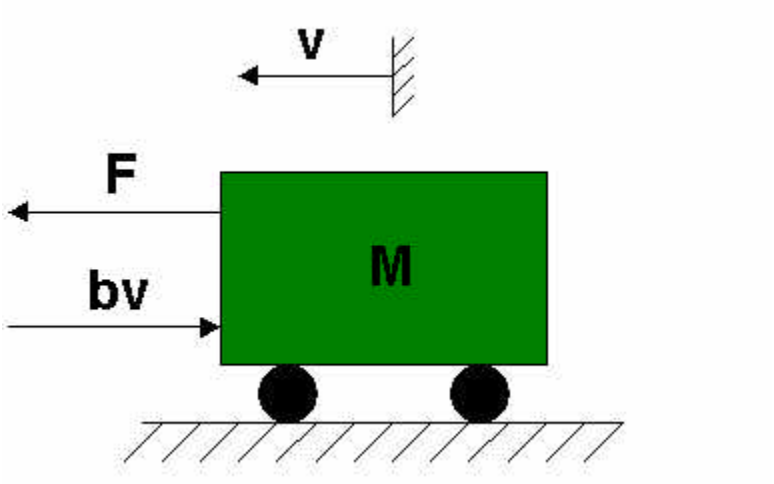
Author: RDM
Updated: 5/18/00

First Order System: Ramp Response

The idea behind these tutorials is that you can view them in one window while running Simulink in another window. Do not confuse the windows, icons, and menus in the tutorials for your actual Simulink windows. Most images in these tutorials are not live - they simply display what you should see in your own Simulink windows. All Simulink operations should be done in your Simulink windows.

Developing Model

Recall the free body diagram developed previously for a car traveling on a flat road:



Where:

- v is the horizontal velocity of the car (units of m/s).
- F is the force created by the car's engine to propel it forward (units of N).

- b is the damping coefficient for the car, which is dependent on wind resistance, wheel friction, etc. (units of N*sec/m) We have assumed the damping force to be proportional to the car's velocity.
- M is the mass of the car (units of kg).

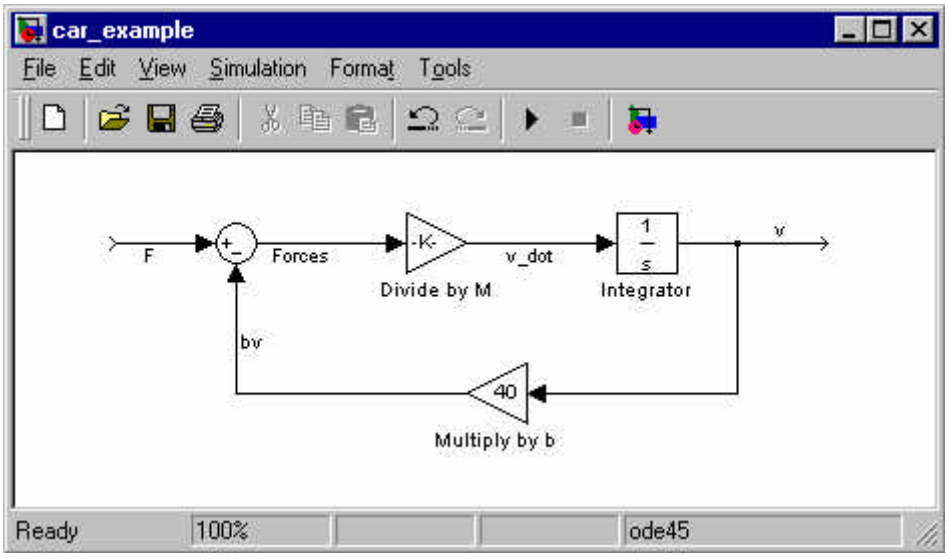
For our system, we assumed that:

M = 1000 kg (a Dodge Neon has a mass of about 1100 kg)
b = 40 N*sec/m

With these parameters, the system equation was:

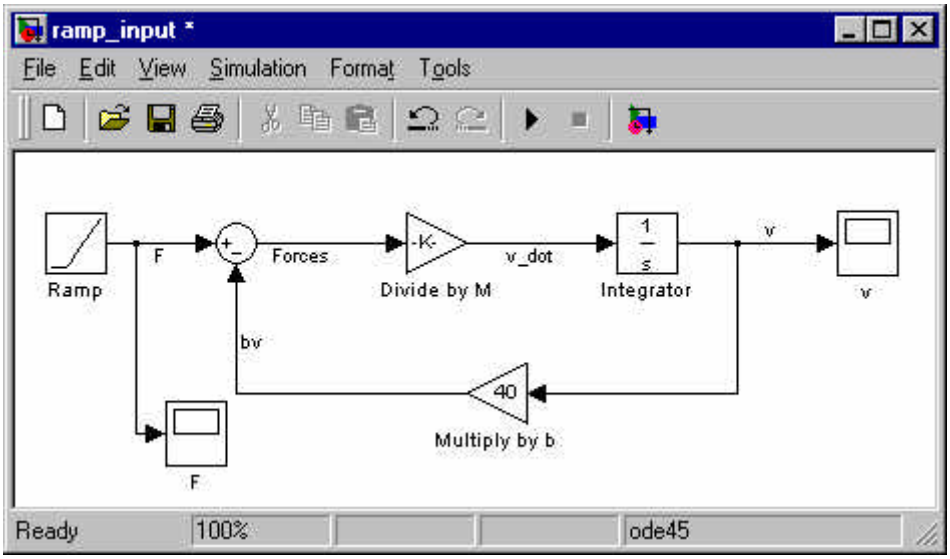
$$\frac{dv}{dt} = \frac{F - bv}{M} = \frac{F - 40v}{1000}$$

And, the system's basic block diagram looked like:

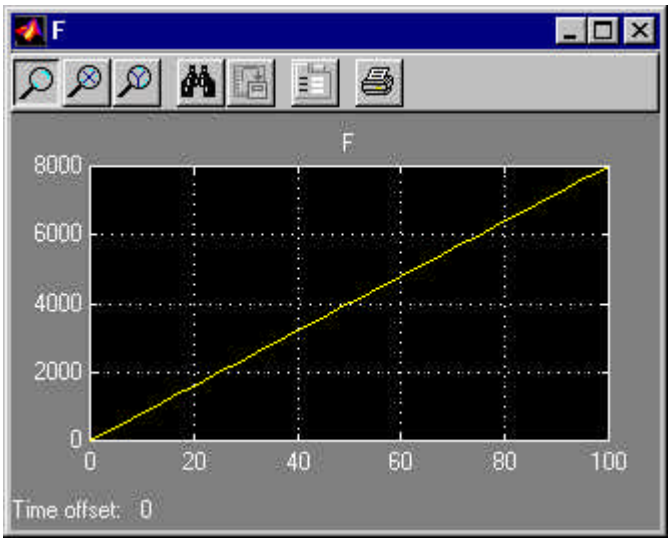


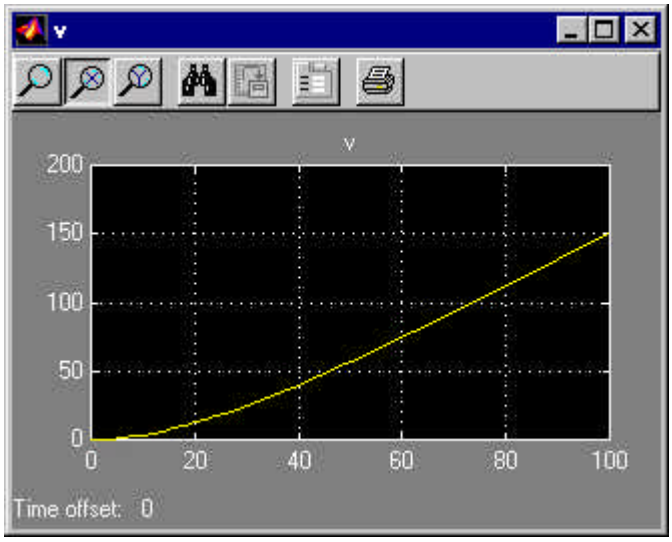
Simulating a Ramp Input

We now apply a ramp input, F, to the system starting from rest. This is approximately equivalent to the car's driver steadily depressing the gas pedal as the vehicle accelerates from a stop light. To model this, insert a Ramp block from the Sources subfolder and connect it so that it produces the system input signal, F. Also, insert Scope blocks (Sinks subfolder) into the model to monitor the engine force, F, and the car's velocity, v. These additions make the block diagram look like:



Double-click on the Ramp block to modify it, and set the Slope equal to 80 N/s (you can leave the Start Time and Initial Output as 0). These settings cause the engine force to steadily increase 80 N every second, starting from F = 0 at t = 0. Also, set the simulation Stop Time to 100 seconds (click on **Simulation**, then **Parameters**), and run the simulation. Opening the two Scope blocks should produce the following graphs:





These plots show us that if the input force of the engine, F, is increased steadily, the velocity of the car, v, will continue to rise, and thus does not approach a specific steady-state VALUE. Also note that as time passes, the velocity curve eventually settles into a straight line. So, the steady-state RESPONSE to the ramp input is linear, and it has a positive slope (i.e. the velocity of the system goes to infinity as time goes to infinity).

Let's verify these results by solving the differential equation for the ramp input. Rewriting the system equation with F = 80t yields:

$$1000\frac{dv}{dt} + 40v = 80t$$

Taking the Laplace transform of this equation (with initial condition v(0) = 0) and solving for V(s) yields:

$$1000\,s\,V(s) + 40\,V(s) = \frac{80}{s^2}$$

$$V(s) = \frac{0.08}{s^2(s+0.04)}$$

$$V(s) = \frac{2}{s^2} - \frac{50}{s} + \frac{50}{s+0.04}$$

Converting back to the time domain gives us the solution for v(t):

$$v(t) = 2t - 50 + 50\,e^{-0.04t}$$

In this solution, the exponential term dies out as t increases. We can estimate how long it takes for this transient part of the solution to be negligible by noting that its time constant is 1/0.04 =

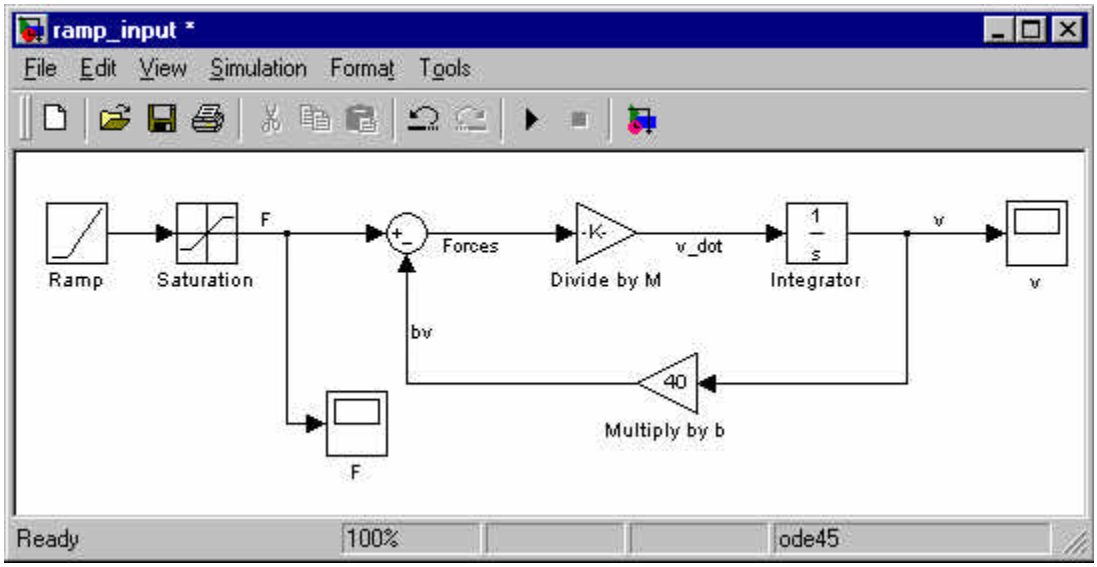
25 seconds. Thus its settling time, which is about 4 times as large as the time constant, is about 100 seconds. So, after 100 seconds, the solution is approximately equivalent to the straight line (2t - 50). The results of our simulation thus agree with the analytic solution to the equation.

Note that in this example, the velocity of the car increases without bound, a result of the steadily-increasing engine force. Intuition tells us that this type of behavior is unrealistic. A Dodge Neon is not capable of going 100 m/s (about 225 mph), yet our simulation shows the car reaching velocities well over this value. In reality, there is a maximum force that the car's engine is capable of producing (and thus a maximum velocity that the vehicle can attain), so F cannot take a value above this upper limit. This idea of a saturated input is the topic of the next section of this tutorial.

Ramp Input with Saturation

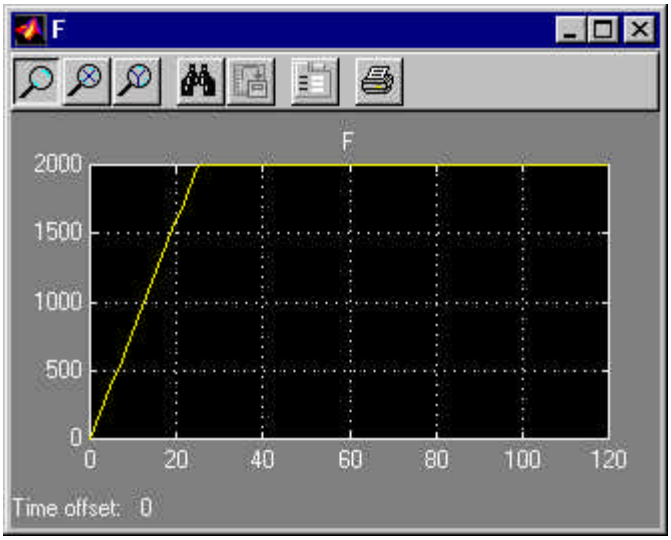
In this section, we will apply a ramp input to our system (as before), with the following alteration: The engine force, F, will not be allowed to exceed 2000 N. Thus, the system's input will appear as a ramp until its value reaches 2000 N. From that time forward, the saturated input will remain at 2000 N. This situation can be thought of as being similar to the car's driver, with the vehicle starting from rest, steadily pushing down on the gas pedal until it reaches the floor (i.e. the maximum force that the engine can provide), and then holding the pedal there for an indefinite time.

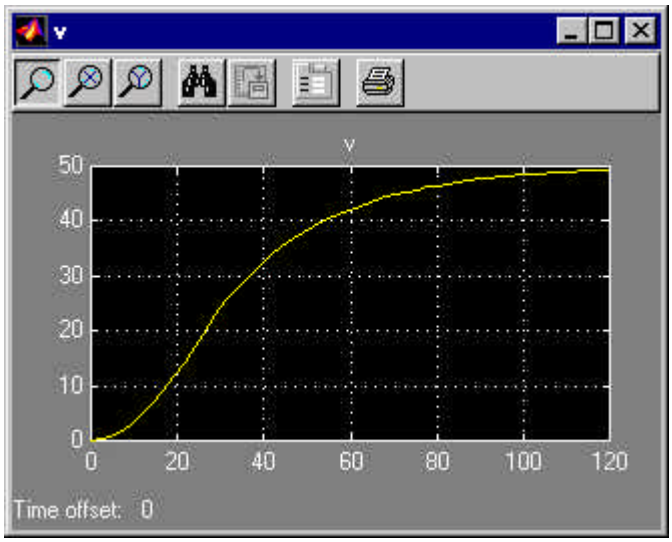
To model this input in Simulink, we insert a Saturation block (from the Nonlinear subfolder) right after the Ramp block in the model window. With this addition, our sytem model should now appear as below:



The Saturation block allows us to set an upper and lower limit for its input signal. If the signal to the block is between the minimum and maximum values we have set, the Saturation block passes it through unaltered. If the input signal is greater than the maximum, however, it outputs the set maximum value. Similarly, if the input signal is less than the minimum, the Saturation

block simply outputs this user-defined minimum value. Double-click on the Saturation block to modify its parameters, and change its Upper Limit to 2000 and its Lower Limit to 0. Now, run the simulation (change the Stop Time to 120 seconds), and view the F and v scope blocks. They should look like:





We notice a number of interesting features of the system by analyzing these graphs. First of all, note that the engine force plot, F, looks like we predicted it would. It appears as a ramp input until it reaches 2000 N, and then stays at that maximum as time continues to pass. Also notice that up to about t = 25 seconds, the velocity response of the car, v, is identical to what it was for the ramp input we analyzed previously. Beyond that time, the two plots take on very different shapes, and in this example, the velocity appears to level off at 50 m/s (about 110 mph). This result is of course due to the input, F, reaching its saturation value of 2000 N at t = 25 seconds.

What would you expect the steady-state velocity of the system to be if a step input of 2000 N were applied at t = 0? Setting dv/dt equal to 0 in the system equation, you should find the answer to also be 50 m/s.

Why do this step input and the ramp input with saturation we simulated have the same steady-state velocity? The two systems are similar in that, after appearing very different at smaller t values, their engine force inputs stay constant at 2000 N as time goes to infinity. Thus, it is logical that both would have the same steady-state velocity.

Using which input, the step or the ramp with saturation, would you expect the system to reach its steady-state velocity more quickly? If the answer to this is not clear to you, think of the physical system we are modeling. If you were driving a car, would you expect to reach your top speed more quickly by stomping the gas pedal to the floor and holding it there (step input) or by gradually pressing down on it until you reach its maximum position (ramp with saturation)? Clearly, the step input would give the smaller settling time.

Author: RDM
Updated: 5/18/00