Lab 2

Tutorial: Running Simulink from a MATLAB M-file

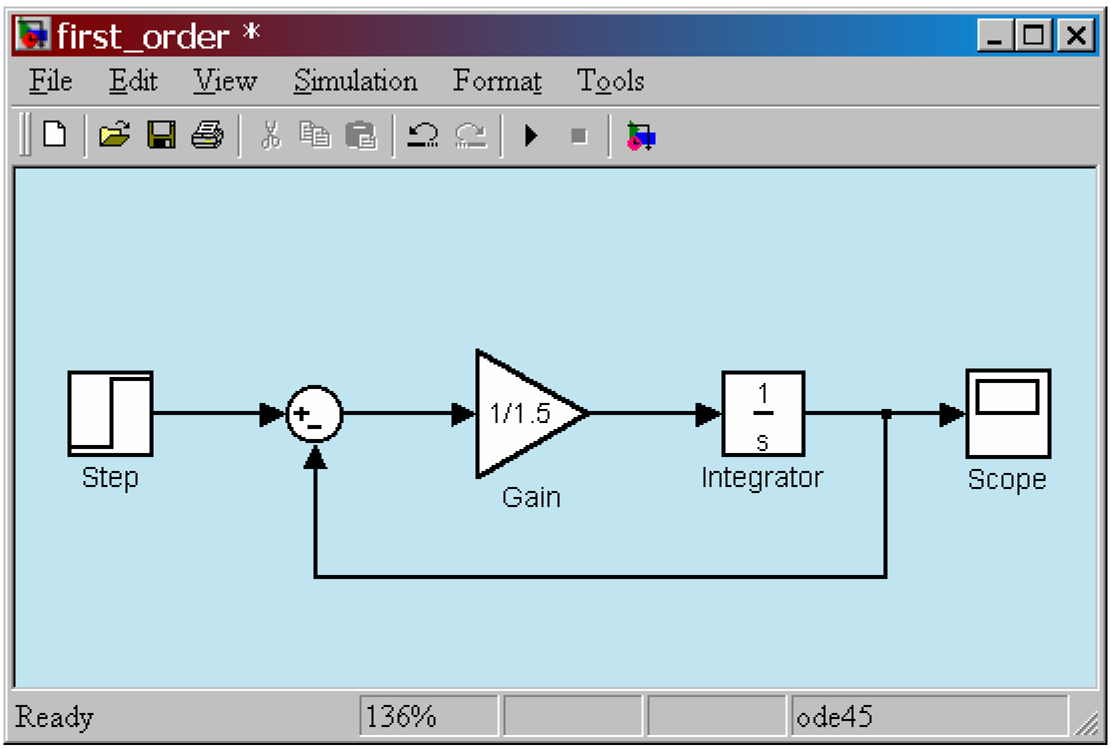
Getting started

Set up a Simulink file to solve the ODE given by

$$1.5\dot{y} + y = 3u\,,$$

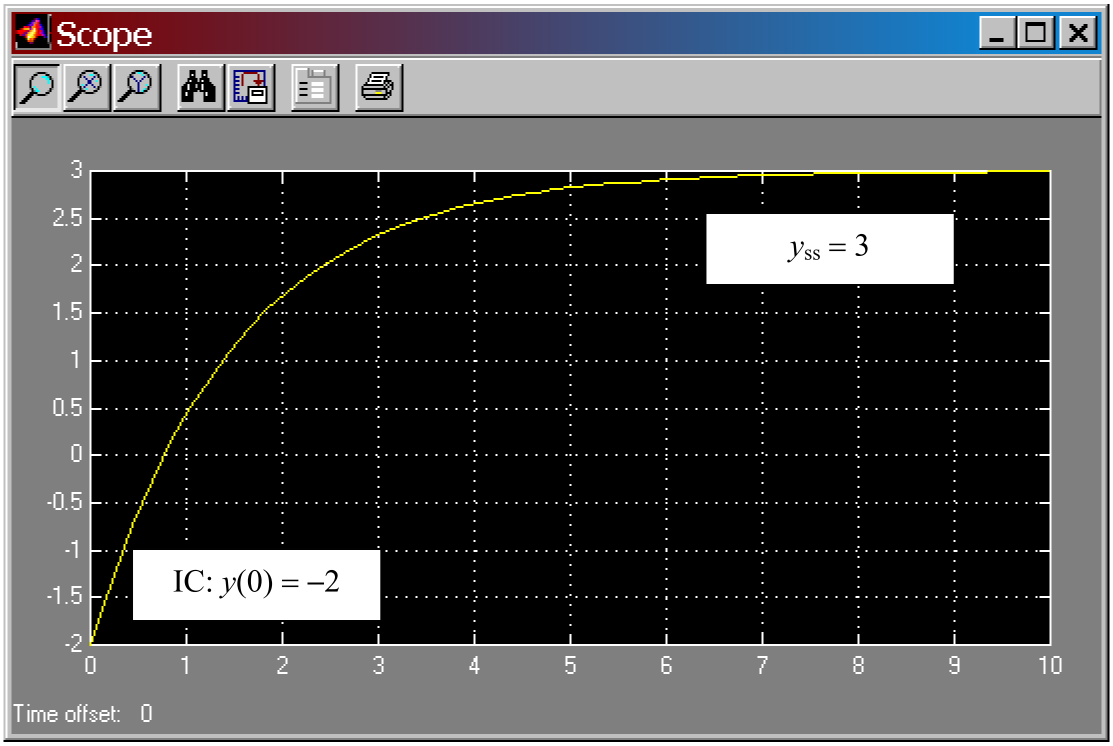
where $y(0) = -2$ and $u(t)$ is a unit step input. Save the model under the filename **first_order.mdl**.

Your simulation file should look like:



Every time you make a change to a MATLAB *M-file*
or a Simulink *model file*,
you have to **File ➔Save**
before running the new simulation.

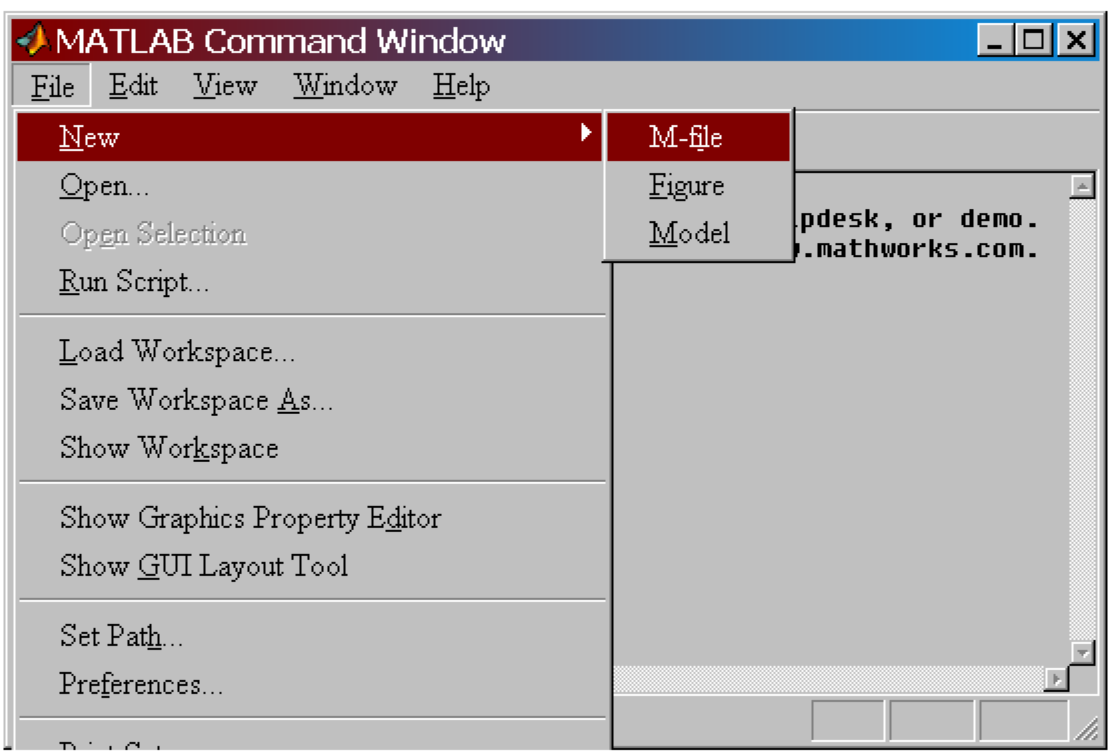
The solution of the ODE should look like:



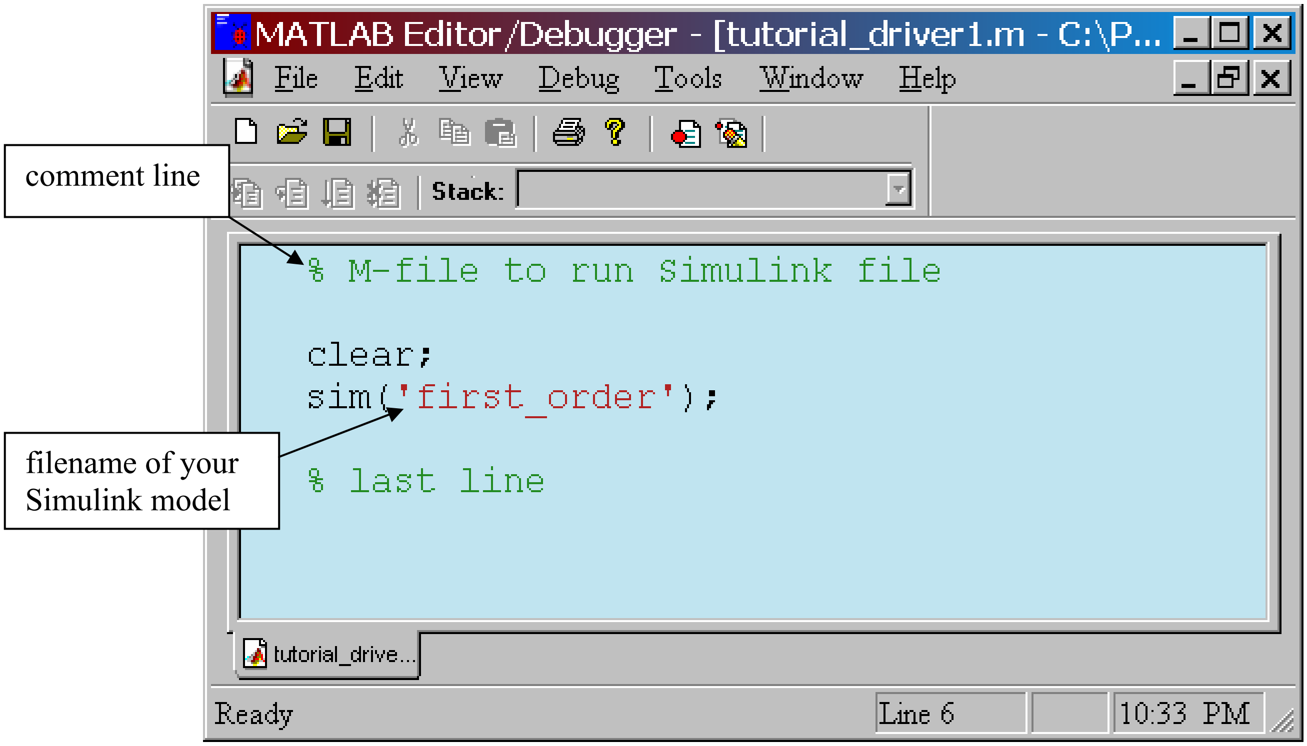
To run the simulation from Matlab

We need to create an M-file. In the MATLAB Command Window, select

File ➔New ➔M-file.

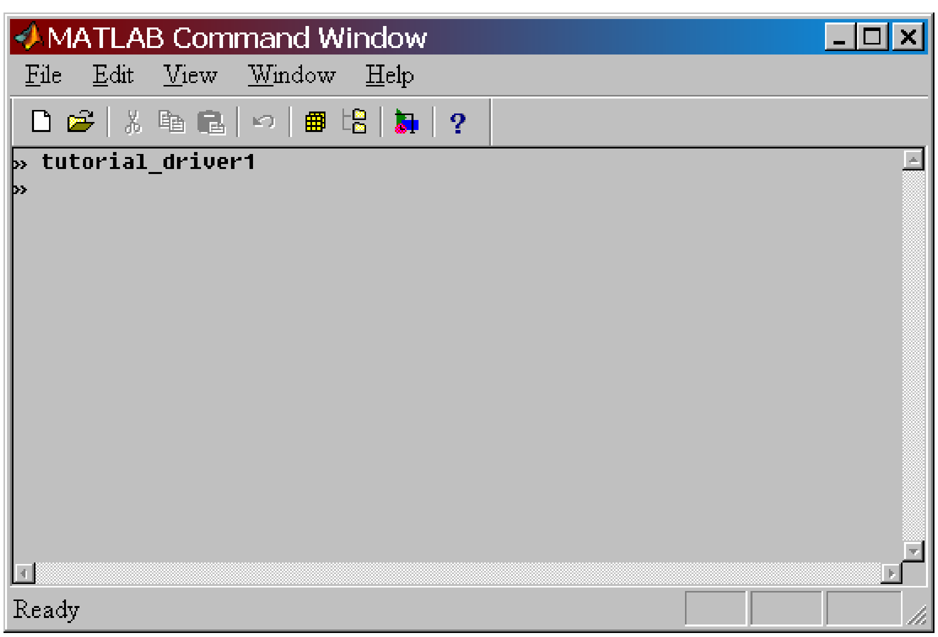


Save the file that opens as **tutorial_driver1.m** . Type the following commands in the M-file. **File ➔Save**. This file is the MATLAB program you will use to control your simulation, change variables, and gain greater control over plotting results.



Run the program by typing its filename **tutorial_driver1**↵ (return) in the MATLAB *Command Window*. Select the *Scope Window* to see the simulation results.

If you get an error message, read the next section.



'Undefined function or variable' error message

(If your simulation ran correctly, continue to the next section.)

If you type the filename in the MATLAB Command window, you might get an error message that says *Undefined function* ... (see below):

```
» tutorial_driver1
??? Undefined function or variable 'tutorial_driver1'.
```

Or you might get an error regarding the use of the `sim` command.

First check that you spelled the filename correctly. If spelled correctly, you probably have to use the *Set Path* command to tell MATLAB the directory to look in so it can find your file.

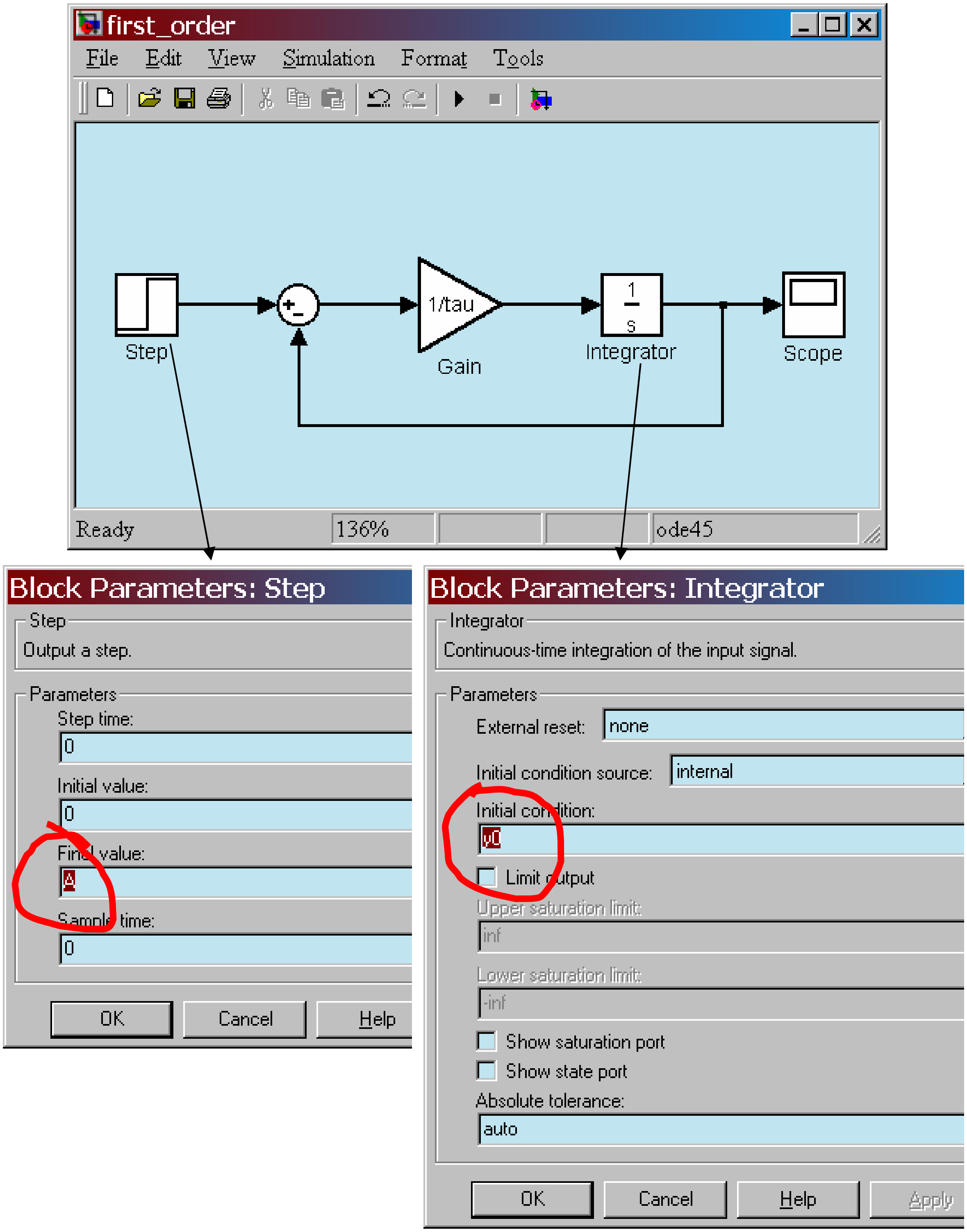
Go to **File ➔Set Path** ..., then select **Browse** to find the directory in which you saved your M-file. Once you've selected the correct folder in the *Path Browser* window, select (highlight) the path name in the *Current Directory* space. Then from the menu, select **Path ➔Add to path**... , select **OK**, and close the *Path Browser*.

Repeat for the directory in which your Simulink models are stored.

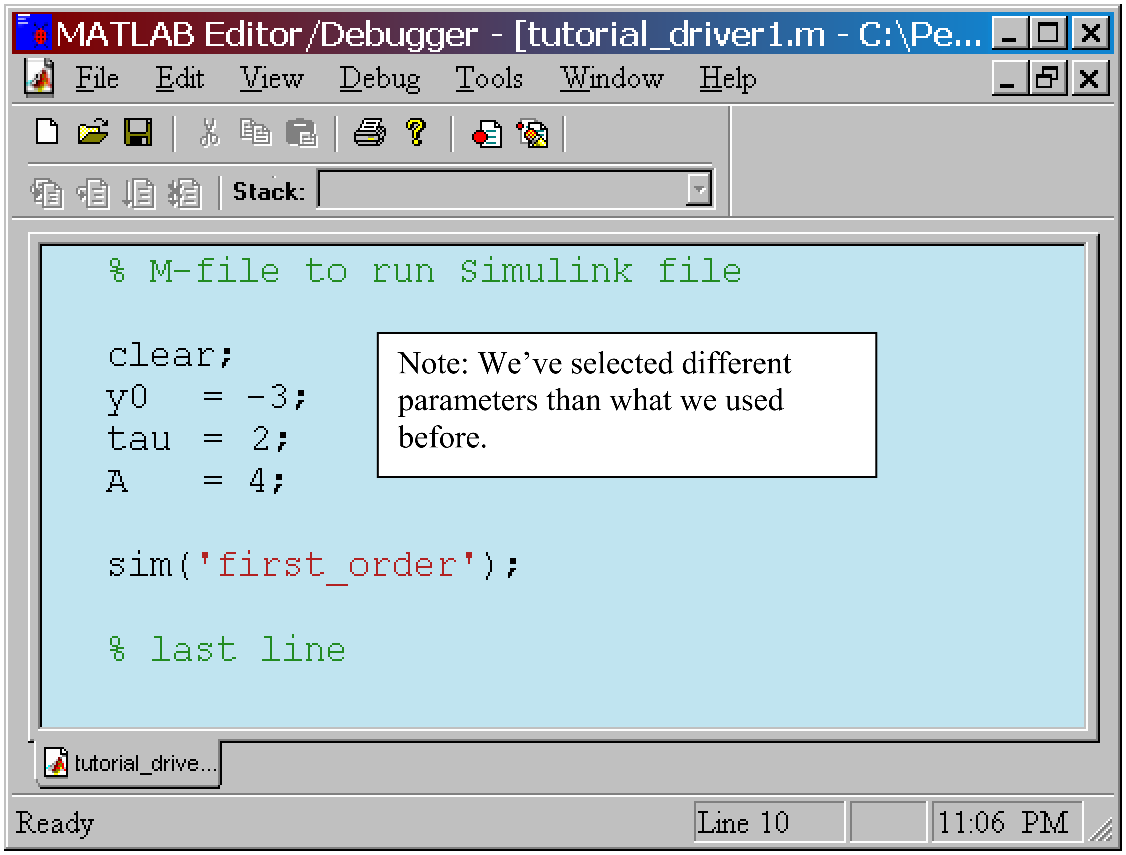
Return to the MATLAB Command window, type the filename of your M-file. Your file should run.

To change variables from the M-file

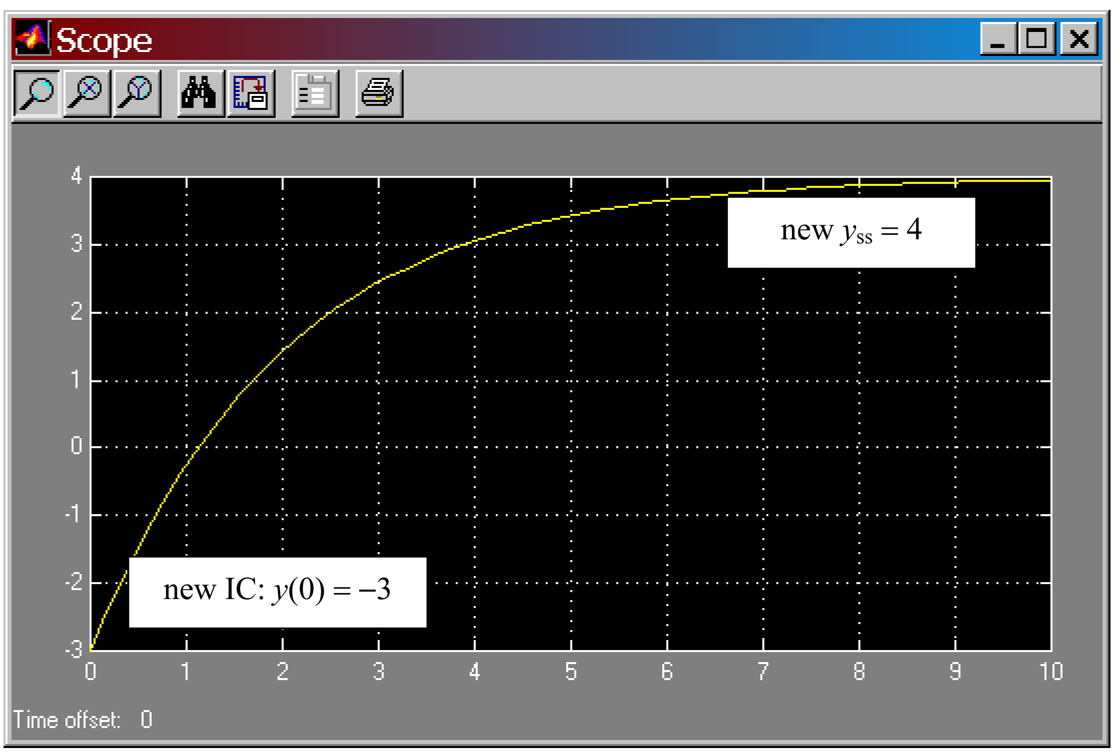
Put variable names in the Simulink simulation diagram. Replace the gain with **1/tau**. Change the IC on the integrator to the variable name **y0**. Change the magnitude of the step input to **A**. Don't forget to **File ➔Save.**



Now, let's assign values to these variables in the M-file.



File ➔**Save**. Again, run the program by typing **tutorial_driver1**↵ in the MATLAB Command Window. The Scope should show a new plot, with a new IC and a new final value.

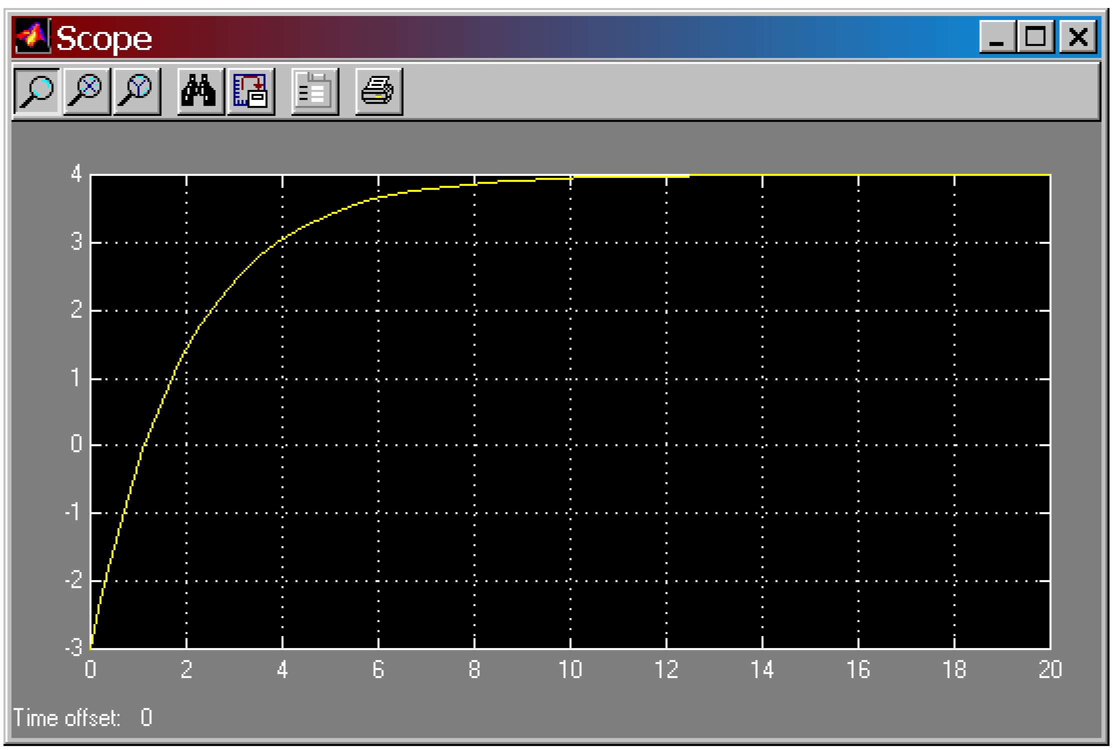


Changing the simulation time span

In the M-file, add a new argument to the **sim** command. The bracketed expression [0 20] tells Simulink to run the simulation for the time interval $0 \le t \le 20$. The M-file should now look like:

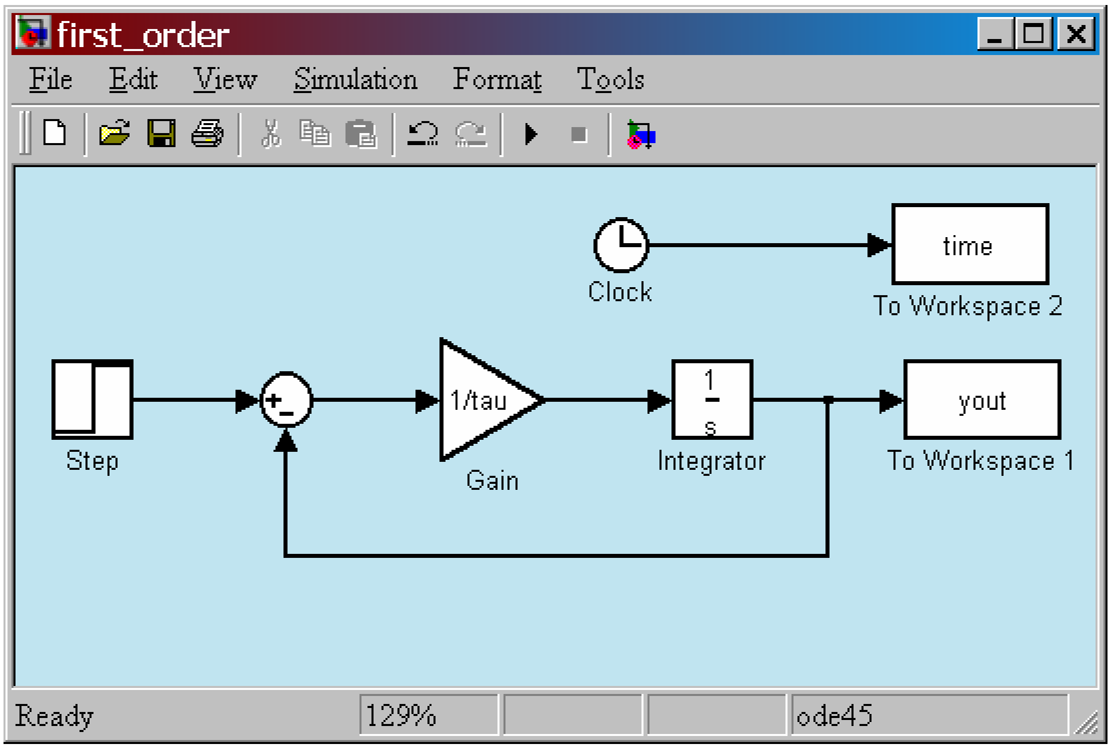
```
% M-file to run Simulink file
clear;
y0    = -3;
tau   =  2;
A     = 4;
sim('first_order',[0 20]);
% last line
```

The solution should look as follows. Note that time now runs from 0 to 20 sec.

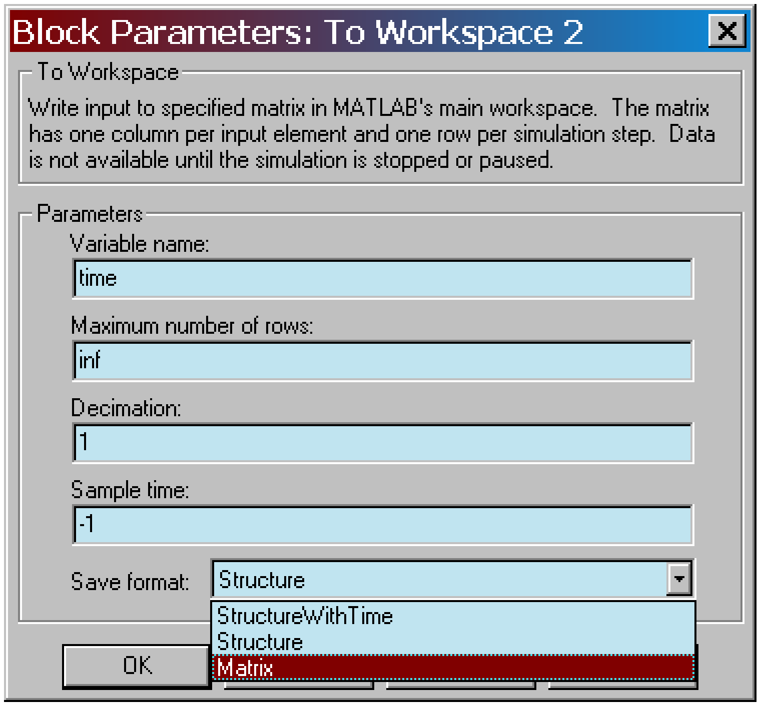


Plotting in MATLAB

To bring the variables from the Simulink workspace into the MATLAB workspace for better control of plotting, we have to assign variable names to the output variables. In the Simulink window, delete the *Scope* block and replace it with a *To Workspace* block from the *Sinks* library. In the *Block parameters* window, change the name of the variable name to **yout**. Add a *Clock* from the *Sources* menu connected to a second *To Workspace* block. Name this variable **time**. You have now created two new variables, **time** and **yout**, which are available for manipulation in the MATLAB environment.



You have to assign a **structure-type** to these two output variables. In the *Block parameters* window of both *To Workspace* blocks, set the Save format selector to *Array* format. See the example below. (In Matlab version 5, select the format called *Matrix*.)



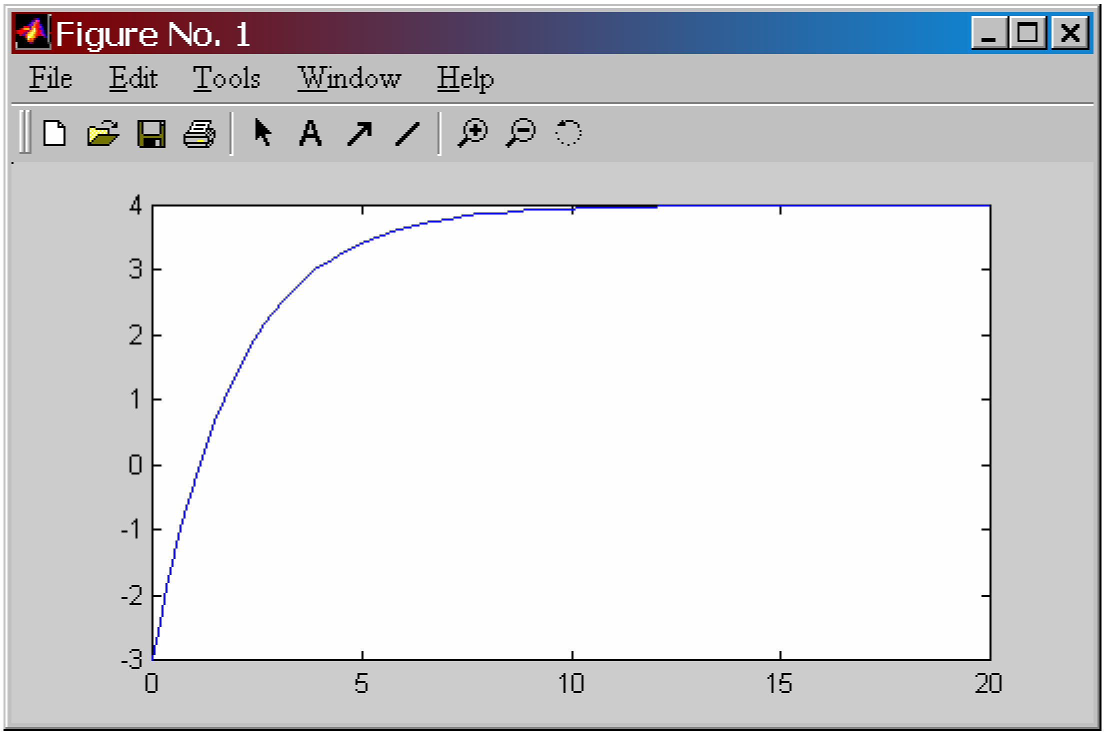
Don’t forget to **File ➔Save** the Simulink model.

Add a plot command to the M-file as follows.

```
% M-file to run Simulink file

clear;

y0    = -3;
tau   =  2;
A     = 4;

sim('first_order',[0 20]);

plot(time, yout)

% last line
```

Don’t forget to **File ➔Save** the M-file.

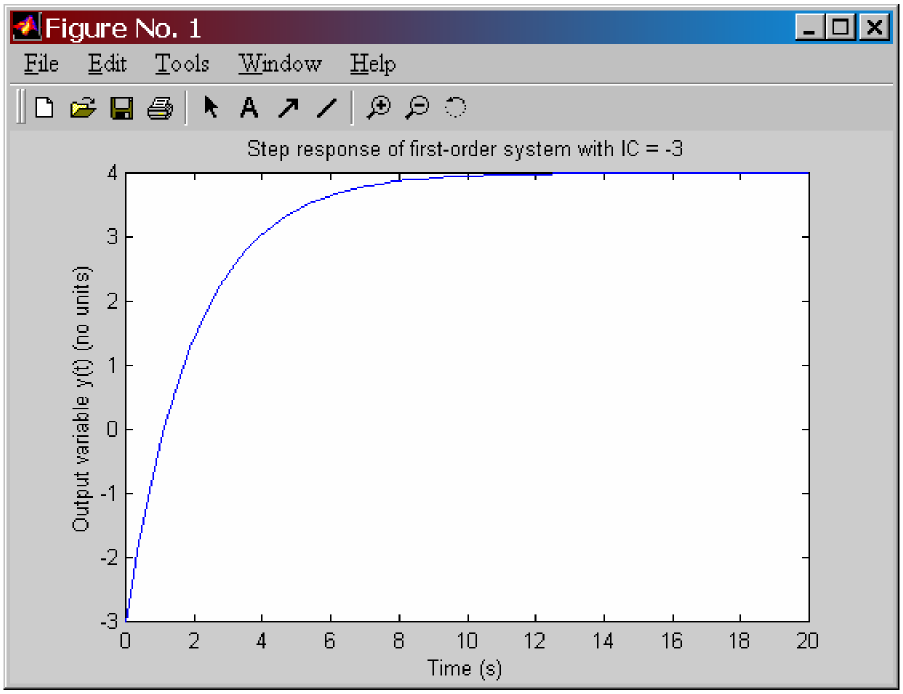
Again, run the program by typing **tutorial_driver1**↵ in the MATLAB Command Window. The resulting plot is in the *Figure* window.



Add a title and label the axes by adding the following commands to the M-file.

```
% M-file to run Simulink file

clear;
y0    = -3;
tau   =  2;
A     = 4;

sim('first_order',[0 20]);

plot(time,yout)
title('Step response of first-order system with IC = -3')
xlabel('Time (s)')
ylabel('Output variable y(t) (no units)')

% last line
```

The plot is shown below.

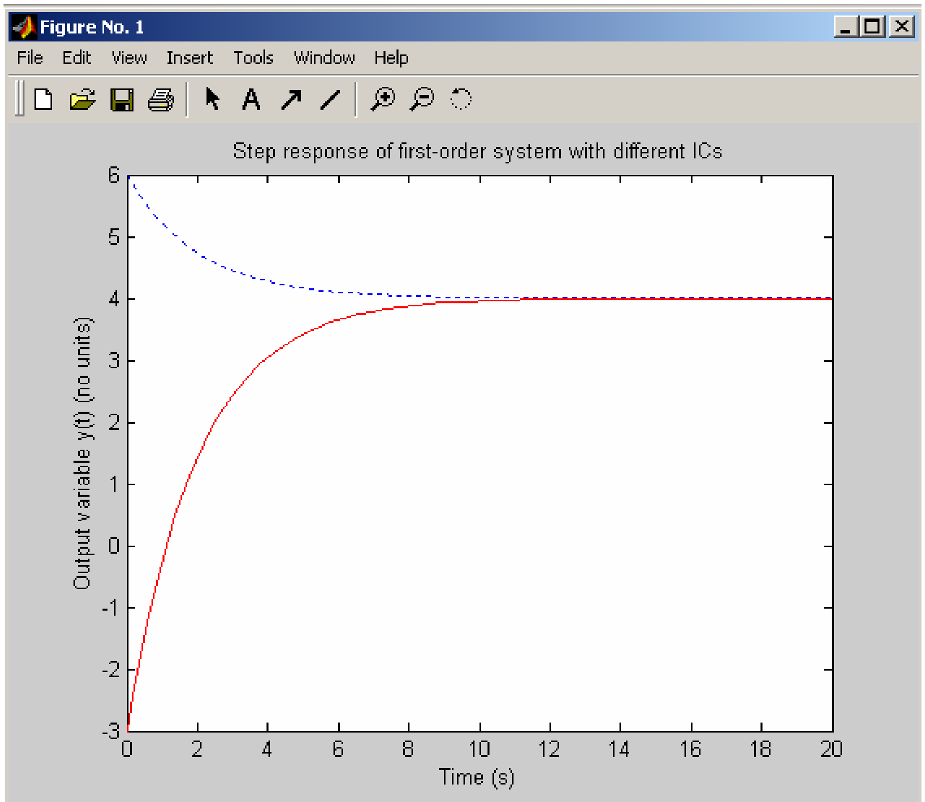


Multiple curves on the same figure

Suppose we want to compare simulation results of the same system with two different ICs. In case 1, $y(0) = -3$ (as before) and in case 2, $y(0) = +6$. Don't forget to **File ➔Save** the M-file.

```
% M-file to run Simulink file
clear;
y0    = -3;
tau   =  2;
A     = 4;

sim('first_order',[0 20]); % run the simulation with IC = -3
t1 = time; % assign time to a new variable name
y1 = yout; % assign yout to a new variable name

y0 = 6; % assign a new IC
sim('first_order',[0 20]); % run the simulation with IC = +6
t2 = time; % assign the new time to a new variable name
y2 = yout; % assign the new yout to a new variable name

plot(t1,y1,'r-',t2,y2,'b:')
title('Step response of first-order system with different ICs')
xlabel('Time (s)')
ylabel('Output variable y(t) (no units)')

% last line
```

The resulting plot:



On this plot the character fields 'r-' and 'b:' in the plot command line

```
plot(t1,y1,'r-',t2,y2,'b:')
```

were used to designate the solid red and dotted blue characteristics of the lines.

Other colors, line types, and marker types which could have been specified in these character fields are included on the table below.

Data markers		Line types		Colors	
Dot	.	Solid line	-	Black	k
Asterisk	*	Dashed line	--	Blue	b
Cross	x	Dash-dotted line	-.	Cyan	c
Circle	o	Dotted line	:	Green	g
Plus sign	+			Magenta	m
Square	s			Red	r
Diamond	d			White	w
Five-pointed star	p			Yellow	y

Either a line type or marker type may be specified in any one character field, but not both.

If no characters are specified, the default line type and color will be a solid and blue.

Some additional plotting commands:

If you wish to overlay a background grid line structure to the plot use the commands

>> **`grid on`**

and

>> **`grid off`**

It is also useful to include a legend to identify the different lines of a multi-line plot. The **`legend`** command adds a dragable legend to the plot that identifies each of the lines.

```
>>legend('line 1 description', 'line 2 description',...)
```

Sometimes it will be useful to control the scale or extent of the axes. Use of the **`axis`** command gives control over the size or other aspects of the axes The following command show an example that explicitly define the axes range.

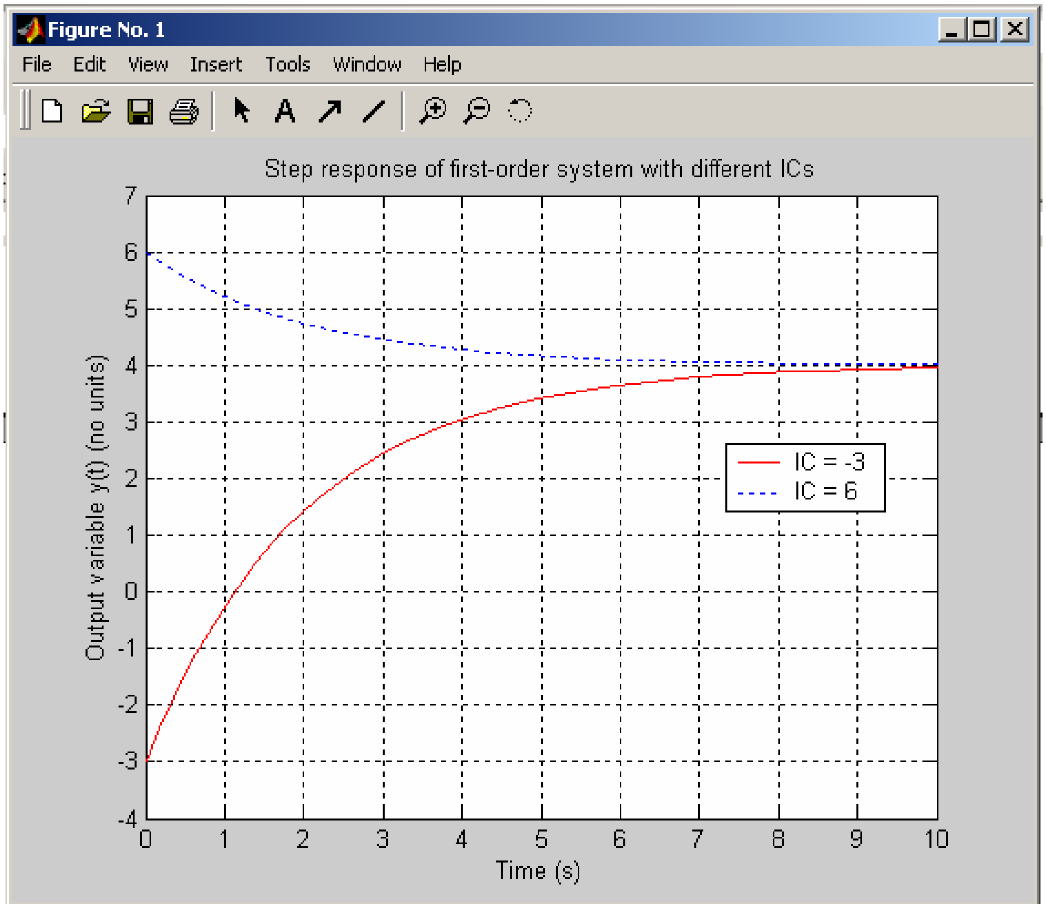
```
>> axis ( [Xmin Xmax Ymin Ymax] )
```

For more information on each of these commands, refer to the MATLAB **help** command.

Let's rerun the multi-line plot including some of these additional details.

```
% M-file to run Simulink file
clear;
y0   = -3;
tau  =  2;
A    = 4;

sim('first_order',[0 20]); % run the simulation with IC = -3
t1 = time; % assign time to a new variable name
y1 = yout; % assign yout to a new variable name

y0 = 6; % assign a new IC
sim('first_order',[0 20]); % run the simulation with IC = +6
t2 = time; % assign the new time to a new variable name
y2 = yout; % assign the new yout to a new variable name

plot(t1,y1,'r-',t2,y2,'b:')
title('Step response of first-order system with different ICs')
xlabel('Time (s)')
ylabel('Output variable y(t) (no units)')
grid on
legend('IC = -3','IC = 6')
axis([0 15 -4 7])

% last line
```

The resulting plot:

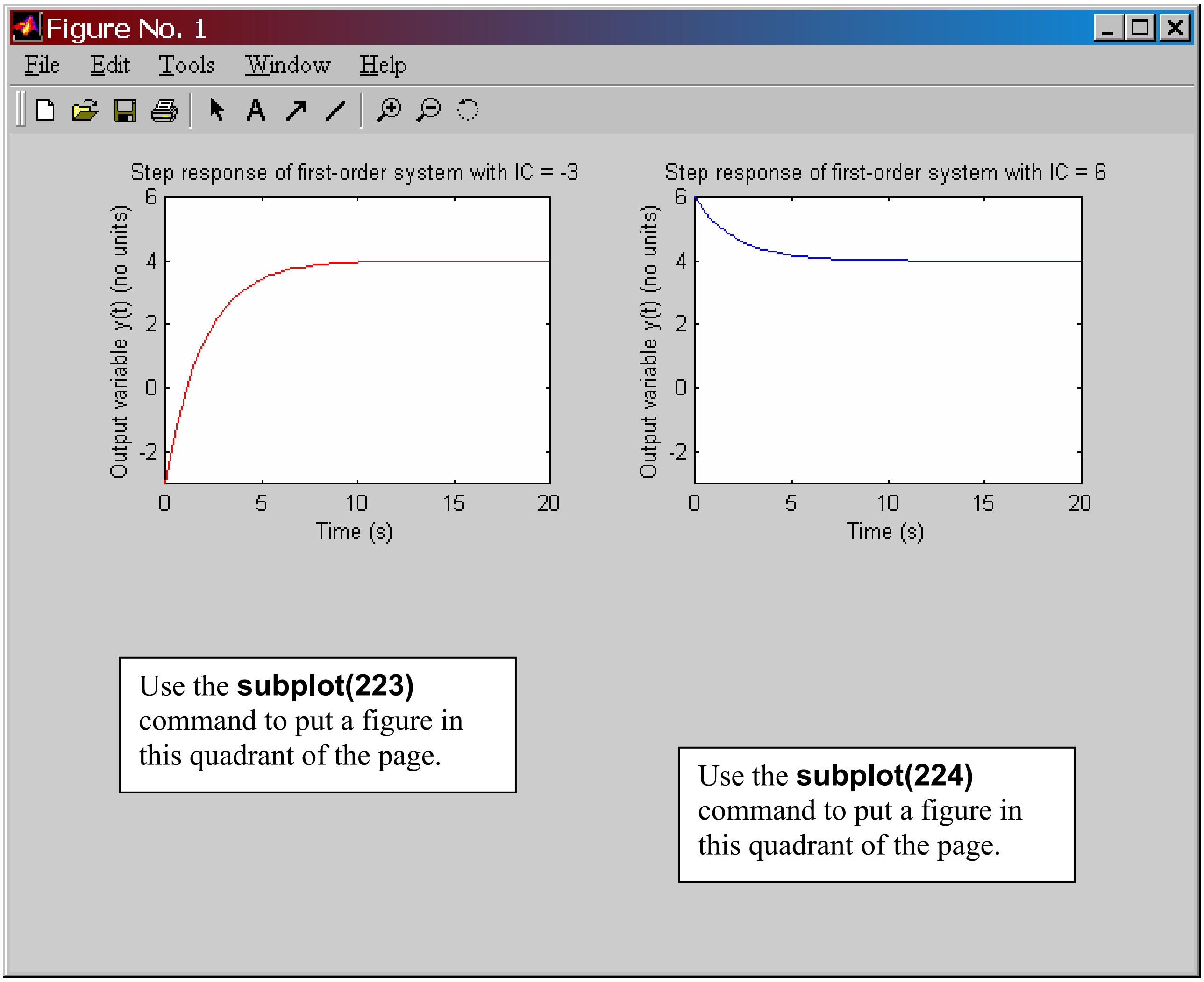


More than one figure on a page

Here, we use the subplot command to create more than one figure on a page. Replace the plotting section of the M-file with the following. We've added the **subplot** command and the **axis** command. Don't forget to **File ➔Save** the M-file.

```
subplot(221)
plot(t1,y1,'r')
title('Step response of first-order system with IC = -3')
xlabel('Time (s)')
ylabel('Output variable y(t) (no units)')
axis([0 20 -3 6])

subplot(222)
plot(t2,y2,'b')
title('Step response of first-order system with IC = 6')
xlabel('Time (s)')
ylabel('Output variable y(t) (no units)')
axis([0 20 -3 6])
```

Again, run the program by typing **tutorial_driver1**↵ in the MATLAB Command Window. The resulting plot is in the *Figure* window.



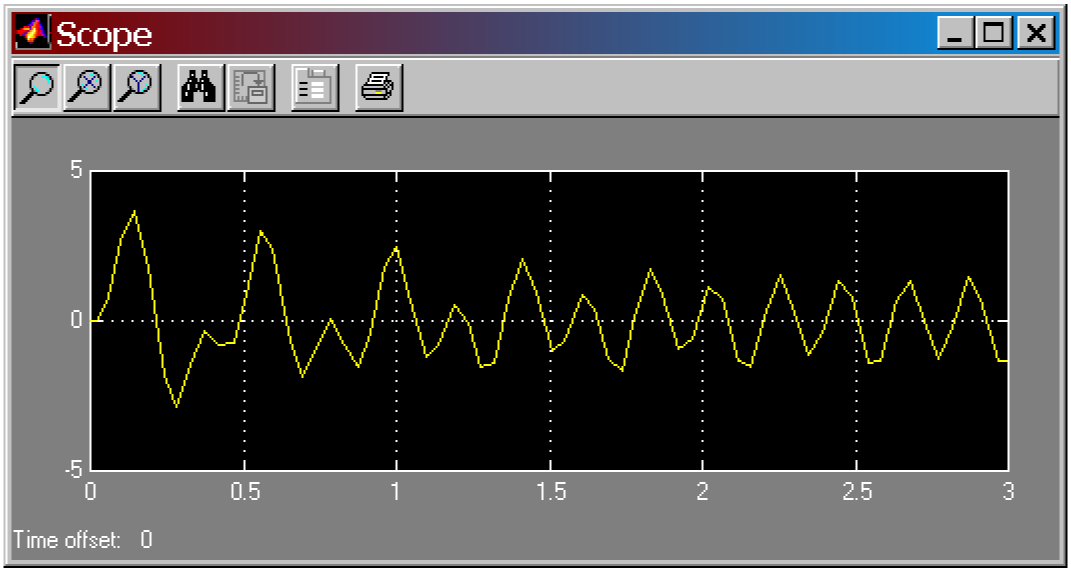
The command **subplot(221)** tells MATLAB to set up a 2x2 grid of figures and to put the next plot in position 1. The command **subplot(312)** tells MATLAB to set up a 3x1 grid of figures and to put the next plot in position 2.

For additional help with plotting in MATLAB, in the *Command window*, type **help plot**. Similarly, for help with any MATLAB command *name*, type **help *name***.

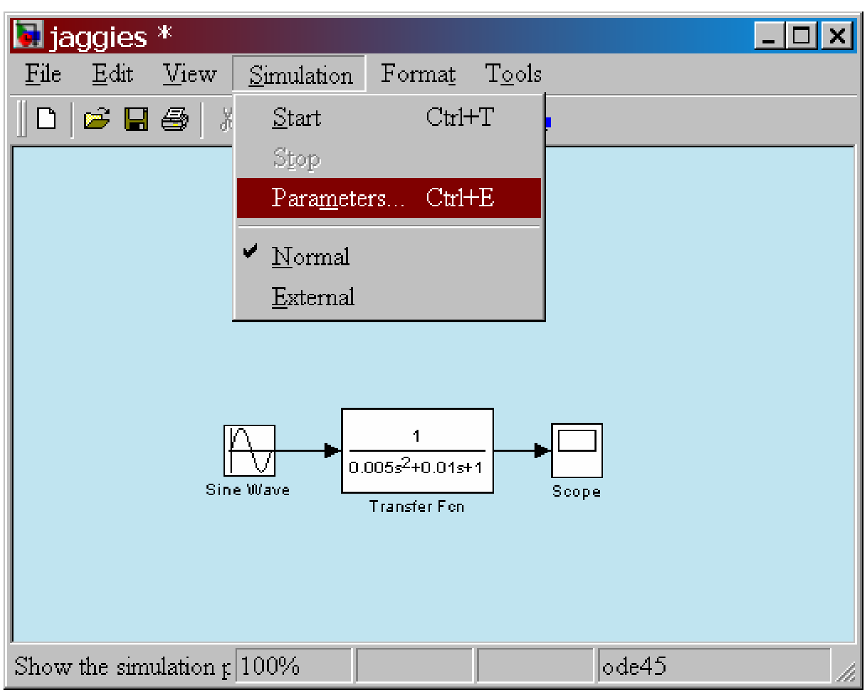
Some simulation stuff

Output 'jaggies'

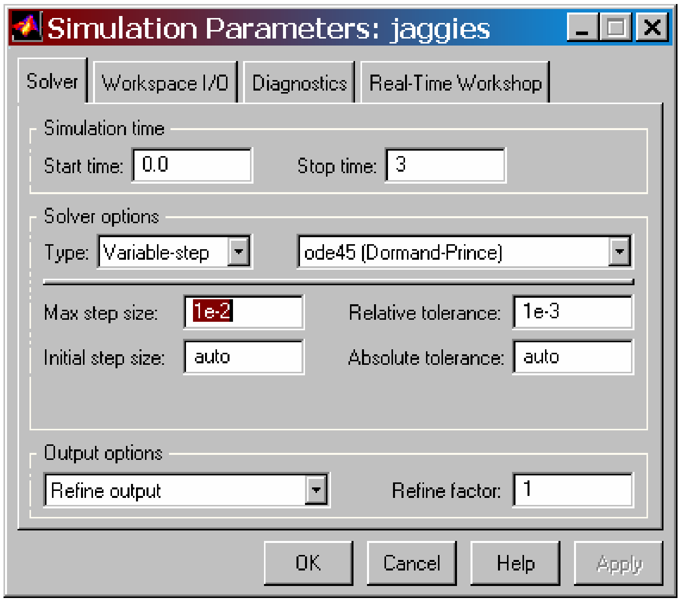
If your output looks like the one below, it represents a poor numerical integration and it is not a good representation or prediction of system behavior.



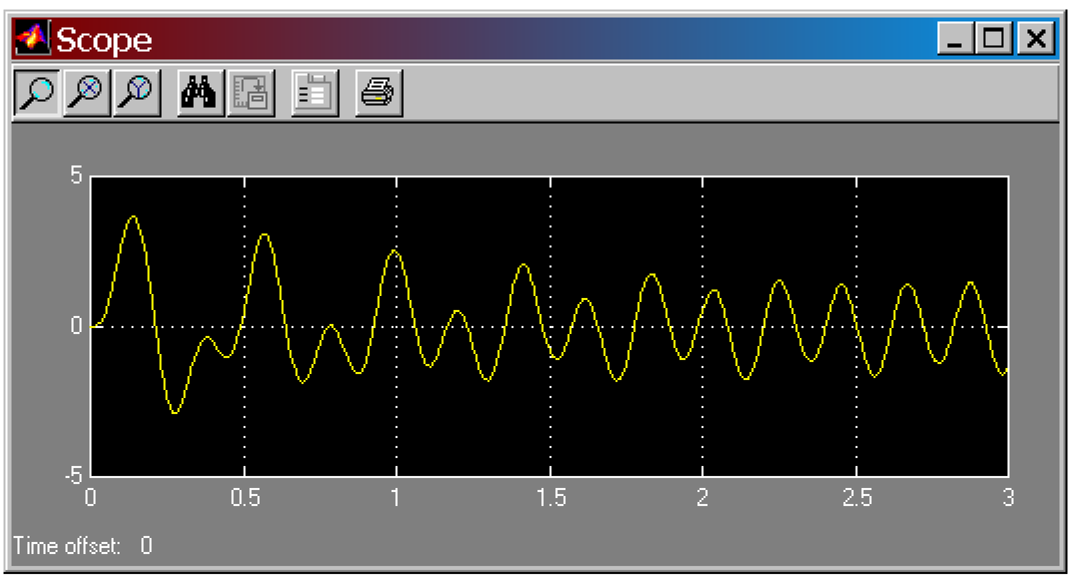
To correct the problem, you have to modify the parameters of the numerical integration. The first parameter to change is the maximum step size. In the Simulink window, select the *Parameters* menu as shown below.



The window that comes up is shown below. Change the *Max step size* from **Auto** to something smaller, for example, 0.01, or **1e-2** as shown.



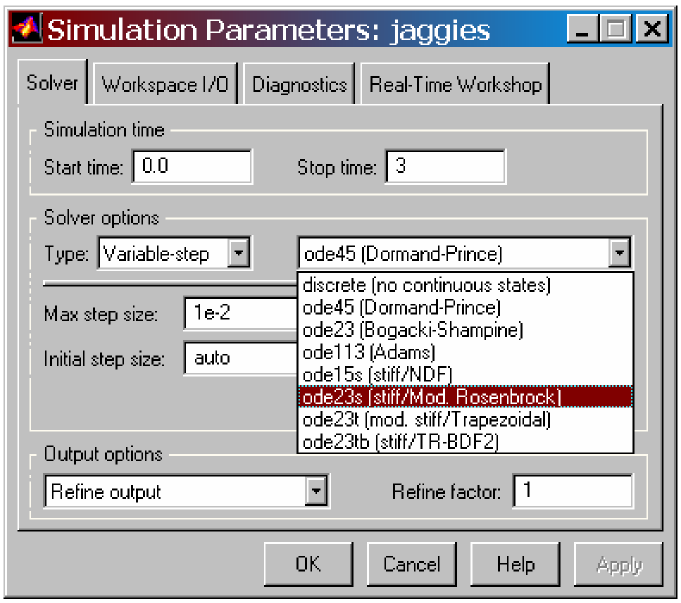
Rerun the simulation, and you get a response that looks much smoother (see below), and probably is a much better prediction of system behavior than the first one.



Stiff systems

Sometimes (particularly in electromechanical systems) a system has both very fast and very slow time constants. Such systems are called *stiff* systems. In integrating the equations of motion of stiff systems the simulation results might not appear to be realistic. For example, you might get high-frequency oscillations or unstable behavior that doesn't make physical sense.

The solution to your problem might be that you have to select a numerical-solver algorithm specialized for stiff systems. To do this, In the Simulink window, select the *Parameters* menu, and select one of the stiff solvers as shown below.



Welcome to Matlab!