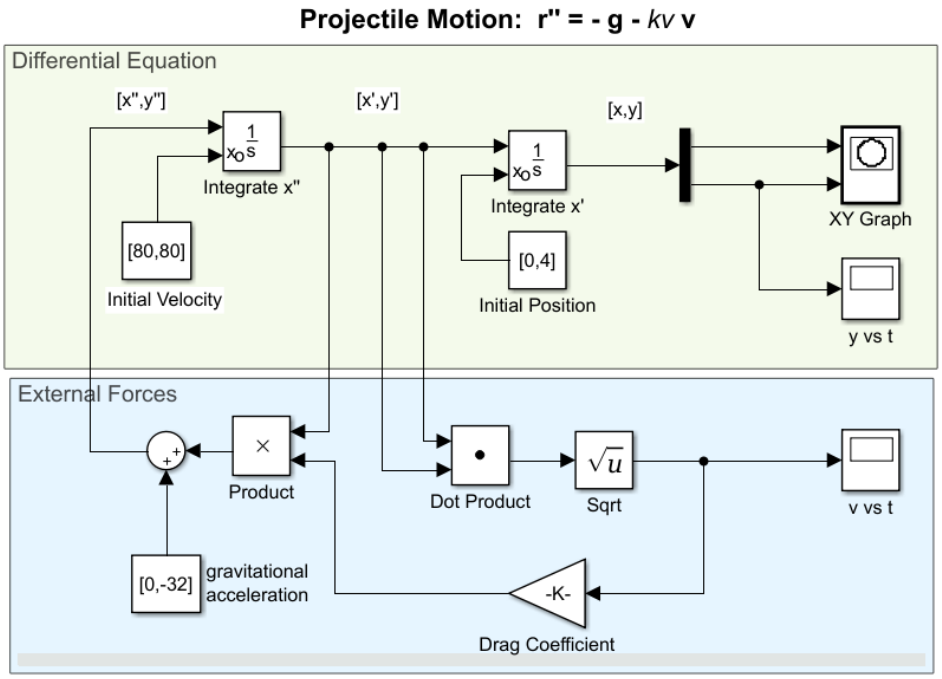
R. HERMAN

SOLVING DIFFERENTIAL EQUATIONS USING SIMULINK



R. L. HERMAN - VERSION DATE: DECEMBER 18, 2020

PUBLISHED BY R. L. HERMAN

This text has been reformatted from the original using a modification of the Tufte-book documentclass in LaTeX. See TUFTE-LATEX.GOOGLECODE.COM.

First printing, 2015

Contents

1 Introduction to Simulink

THERE ARE SEVERAL COMPUTER PACKAGES for finding solutions of differential equations, such as Maple, Mathematica, Maxima, MATLAB, etc. These systems provide both symbolic and numeric approaches to finding solutions. They often require a bit of coding. However, there are graphical environments for solving problems, including differential equations. One such environment is Simulink, which is closely connected to MATLAB. In these notes we will first lead the reader through Simulink examples of solutions of first and second order differential equations usually encountered in a differential equations course. We will then look at examples of more complicated systems.

Most of these models were created using Version 2015. Some changes in Versions 2017-2020 are noted.

Examples of MATLAB solutions of differential equations will also be provided.

1.1 Solving an ODE

SIMULINK IS A GRAPHICAL ENVIRONMENT for designing simulations of systems. As an example, we will use Simulink to solve the first order differential equation (ODE)

$$\frac{dx}{dt} = 2\sin 3t - 4x. \tag{1.1}$$

We will also need an initial condition of the form $x(t_0) = x_0$ at $t = t_0$. For this problem we will let $x(0) = 0$.

We can solve Equation (1.1) by integrating $\dfrac{dx}{dt}$ to formally obtain

$$x(t) = \int (2\sin 3t - 4x(t))\, dt.$$

We will view this as a system in which the input, $x' = 2\sin 3t - 4x$, is fed into an integrator and the output will be $x(t)$. Generally, we have

$$x(t) = \int x'(t)\, dt.$$

This process is depicted in Figure 1.1.

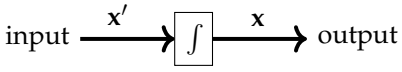


Figure 1.1: Schematic for a general system in which the block takes the input and produces an output.

In order to carry this out, we separately insert the terms $2\sin 3t$ and $-4x$ into the integration procedure. Since we do not know $-4x$, we take

the output from the integrator, multiply it by 4, and subtract that from $2\sin 3t$. This combined set of terms is then feed back into the integrator. This is shown schematically in Figure 1.2.

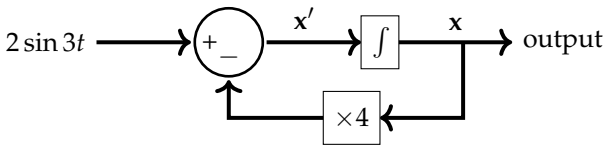


Figure 1.2: Schematic for solving $x' = 2\sin 3t - 4x$. The terms $2\sin 3t$ and $4x$ are fed into the integrator and x is output.

When you have access to Simulink and MATLAB, you can start MATLAB by typing **simulink** on the command line to bring up Simulink. Alternatively, you can select Simulink on the MATLAB icon bar to launch Simulink. Starting in 2017 Simulink opens with a start screen in which there are several selections as shown in Figure 1.3.[1] Pick the Blank Model to begin a new model or select a recently opened model. Then, you will be in the Simulink workspace [see Figure 1.4].

[1] In earlier versions the Simulink Library Browser in Figure 1.5 would appear.

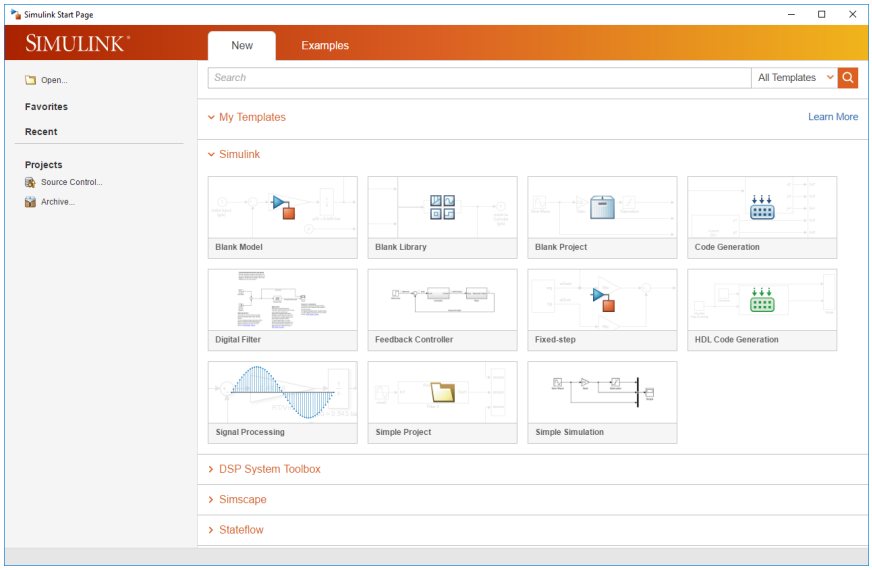


Figure 1.3: The Simulink Start screen. Pick the Blank Model to begin a new model or select a recently opened model.

From the workspace you can open the Simulink Library Browser as shown in Figure 1.5[2]. Next, click the yellow plus to bring up a new model. We build models by dragging and connecting the needed components, or blocks, from groups such as the Continuous, Math Operations, Sinks, or Sources.

[2]

Now we can create the model for simulating Equation (1.1) in Simulink as described in Figure schema2 using Simulink blocks and a differential equation (ODE) solver. In the background Simulink uses one of MATLAB's ODE solvers, numerical routines for solving first order differential equations, such as **ode45**. This system uses the **Integrator** block[3] 

to integrate $\dfrac{dx}{dt}$, producing $x(t)$.

[3] The notation on the **Integrator** block is related to the Laplace transform

$$\mathcal{L}\left[\int_0^t f(\tau)\,d\tau\right] = \frac{1}{s}F(s),$$

where $F(s)$ is the Laplace transform of $f(t)$.

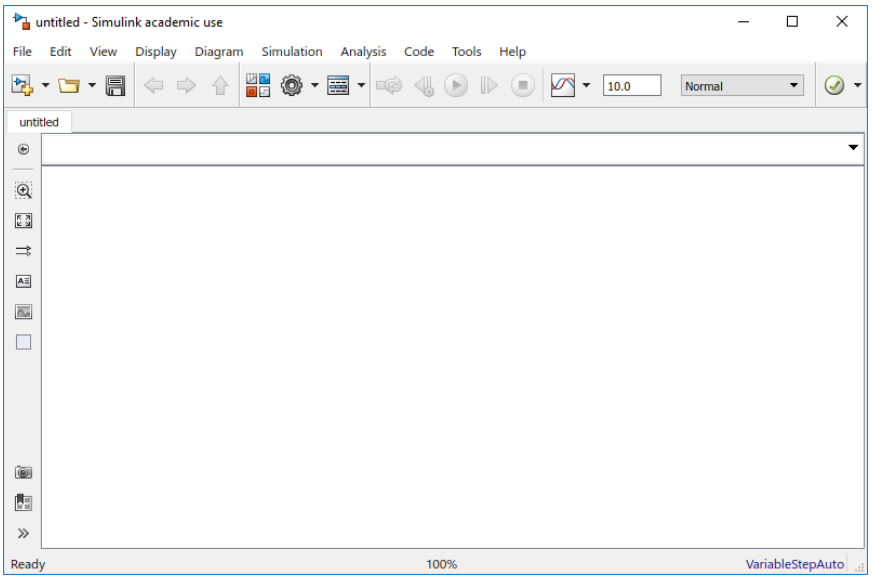


Figure 1.4: A blank model in Simulink.

The input for the **Integrator** is the right side of the differential Equation (1.1), $2\sin 3t - 4x$. The sine function can be provided by using the **Sine Wave** block, whose parameters are set in the **Sine Wave** block. In order to get $4x$, we grab the output of the **Integrator** (x) and boost it by changing the Gain value to "4" Then, using the **Sum** component, these terms are added, or subtracted, and fed into the integrator. The **Scope** is used to plot the output of the **Integrator** block, $x(t)$. That is the main idea behind solving this system using the model in Figure 1.6.

For this example, we implement the following detailed steps in Simulink:

- Drag needed blocks into the model region [Figure 1.7.]:
  - **Integrator** block from the Continuous group;
  - **Sum** block from the Math Operations group,
  - **Gain** block from the Math Operations group,
  - **Sine Wave** block from the Math Operations group; and,
  - **Scope** block from the Sink group.
- Connect the output of the **Sum** block to the input of the **Integrator** block. [Figure 1.8.]
- Connect the **Integrator** to the **Scope** by clicking on the **Integrator** output and dragging to the **Scope** until they are connected. In more recent versions it is easier to double-click the unattached arrow to get a connection.
- Right-click the **Gain** control and choose **Flip Block** under **Rotate & Flip**. Double-click the **Gain** block and change the **Gain** block value from 1 to 4. It should change on the control.

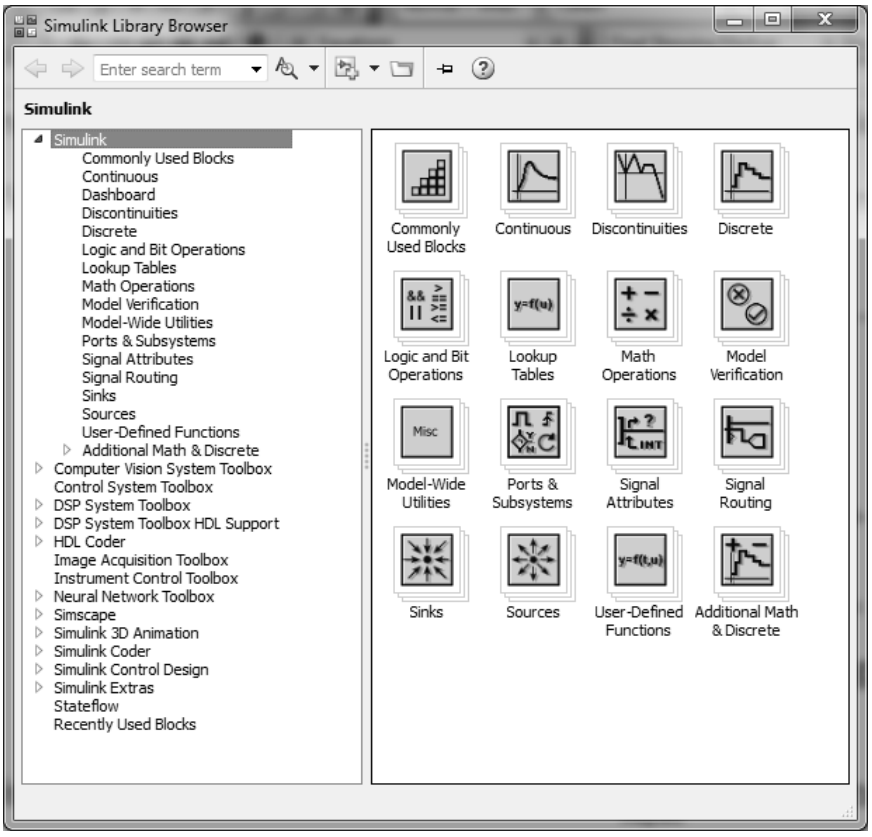


Figure 1.5: The Simulink Library Browser. This is where various blocks can be found for constructing models. [As seen in MATLAB 2015a.]

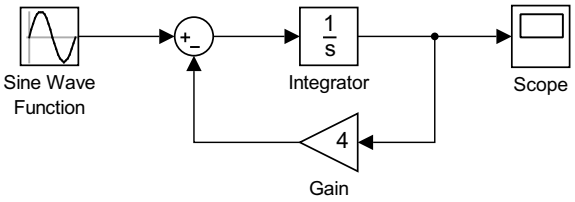


Figure 1.6: System for solving first order ODE $\frac{dx}{dt} = 2\sin 3t - 4x$ as a Simulink simulation.

- Double-click the **Sum** control to bring up **Block Parameters** as shown in Figure 1.9 and change from |++ to |+- in order to set addition/subtraction nodes. [Note that the symbol '|' is a blank node. Also, one can change the block to rectangular form. This is often useful in displaying an overall flow direction to the model. In this case the spacer, |, is not needed.]
- Double-click the **Sine Wave** block and change the frequency to 3 rad/s and the amplitude to 2. [See Figure 1.10] Set the time dropdown menu to **Use Simulation Time**.
- Connect the **Gain** output to the negative input of **Sum** and the **Sine Wave** output to the positive input on the Sum control. [Note: The Gain can be set to a negative value and connected to a + node in the **Sum** block to obtain the same effect.]
- To add a node to route an x value to the **Gain**, hold the **CTRL key** and click on the Output line of the **Integrator** and drag towards the input

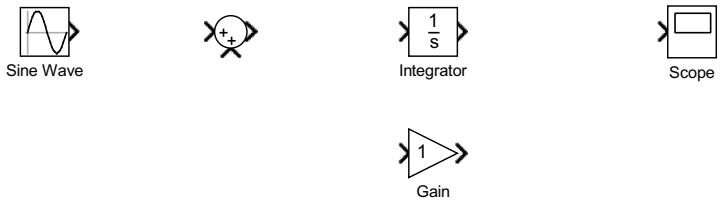


Figure 1.7: Add needed components to the model window.

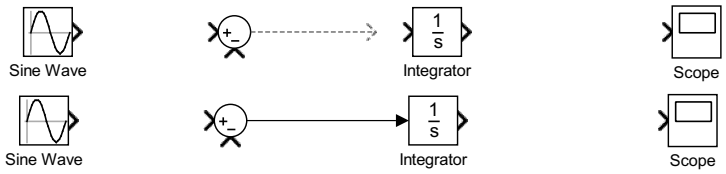


Figure 1.8: Example of connecting two components: Align the components, Click on output of one and drag to another. Then, release to finalize connection. Sometimes it is easier to double-click the temporary arrow to connect the blocks.

of the **Gain**. You can also Right-Click the line where you want the node and drag from there to the **Gain** block. See Figure 1.11.

- The initial value, $x(0)$, of x is inserted by double-clicking the **Integrator** and setting the value. For this example we set $x(0) = 0$.

- One can annotate the diagram by clicking near where labels are needed and typing in the text box. This leads to the model in Figure 1.12. In more current versions the default is to hide block names.

- Save the file under a useable file name. This file can be called in MATLAB, or one can use the run button to run the simulation.

- Double-click the **Scope** to see the solution. Figure 1.13 shows the **Scope** plot after using the autoscale () feature to rescale the scope view. A little effort is needed to change the plot attributes and to import the plots into working documents. This will be discussed in Section 1.4.

- Also, one can make further changes to the system by checking the **Configuration Parameters** under the Simulation menu item. See Figures 1.14-1.15. In particular, changing the **Refine Factor** can lead to smoother solutions. The solution shown in Figure 1.13 had a setting of 1 and that in Figure 1.16 is the result of setting the **Refine Factor** to 10.

In Release 2017b the names of blocks are hidden. One can change this by going to the block Properties and changing the block parameter for 'HideAutomaticName' or change the model parameter 'HideAutomaticNames'.

As noted in setting the initial value, one can double-click the **Integrator** block and set the initial condition. However, sometimes it is useful to externally feed the initial condition into the block. Double-click the **Integrator** block and change the initial condition source from internal to external. This adds another input to the block. Drag a **Constant** block from the Sources group into the model, connect it to the new input, and change the constant value to the desired initial value. This results in the simulation shown in Figure 1.17.

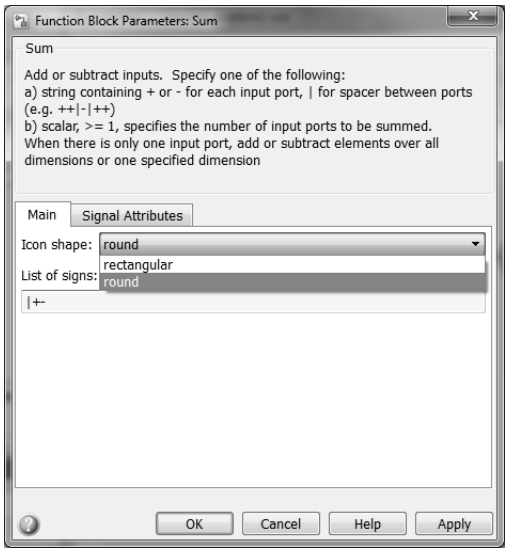


Figure 1.9: Block Parameters for the **Sum** control. In many cases it is best to also select the rectangular shape over the default round shape.

1.2 Handling Time in First Order Differential Equations

IN THIS SECTION WE REVIEW the solutions of first order differential equations, separable first order differential equations and linear first order differential equations involving explicit time dependence. The time dependent functions are obtained using the **Clock** block and a **Math Function** block. Double-clicking the **Math Function** block allows for the selection of a number of common functions.

The independent variable is obtained using the **Clock** block.

Example 1.1. Solve the initial value problem

$$\frac{dy}{dt} = \frac{2}{t}y, \text{ where } y(1) = 1. \tag{1.2}$$

This is a separable equation. Placing y-variables on the left and t-variables on the right side, we have

$$\int \frac{dy}{y} = \int \frac{2}{t}\,dt.$$

Integrating both sides,

$$\ln|y| = 2\ln|t| + C = \ln t^2 + C.$$

Exponentiating, we obtain the general solution,

$$y(t) = At^2,$$

where $A = \pm e^C$.

Using the initial condition, we have the solution, $y(t) = t^2$.

We can set up the problem in Simulink as shown in Figure 1.18 for the initial value problem

$$\frac{dy}{dt} = \frac{2}{t}y,$$

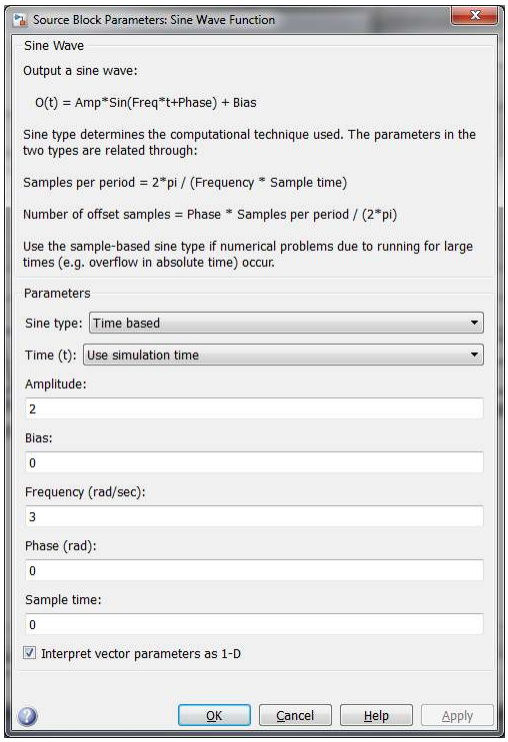


Figure 1.10: Parameters for the **Sine Wave** block. Select the amplitude and frequency desired.

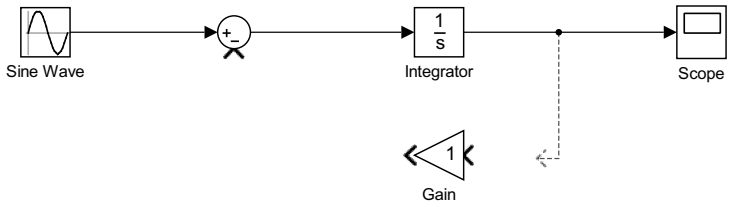


Figure 1.11: Add a node by right-clicking one the line and dragging to the input of a block.

where $y(1) = 1$. Running the simulation, we obtain the solution shown in Figure 1.19.

The solution looks like $y(t) = t^2$. We can verify this by plotting t^2 along with the solution t see if they are the same. Another method would be to compute the difference between the numerical and exact solution, $y(t) - t^2$. In order to do this, we add a **Math Function** block, selecting the square function and connect it to the time route and a **Sum** Block. The solution is also fed into the latter block and the difference is fed into a second **Scope** Block. This is shown in Figure 1.20.

The result of the simulation is shown in Figure 1.21. We note that this is the numerical error, though the solution is only off by 1.4×10^{-5} over the given interval. Considering that the solution at $t = 10$ is $Y(10) = 100$, this is a relative error of roughly 10^{-7}. That seems perfectly acceptable.

It is simple to change the differential equation (1.2) in the previous example to a linear first order differential equation.

$$\frac{dy}{dt} = \frac{2}{t}y + t^2.$$

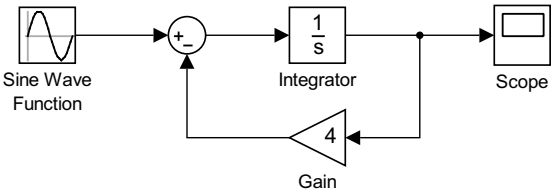


Figure 1.12: Connections for First Order ODE model for $\frac{dx}{dt} = 2\sin 3t - 4x$.

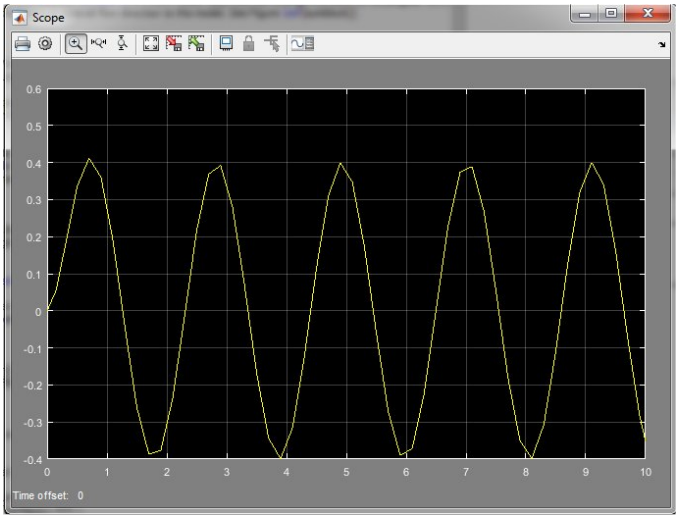


Figure 1.13: **Scope** plot of the solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$, with **Refine Factor**= 1.

Example 1.2. Solve the linear first order differential equation,

$$\frac{dy}{dt} = \frac{2}{t}y + t^2, \tag{1.3}$$

satisfying $y(1) = 1$.

We first rewrite Equation (1.3) in standard form,

$$\frac{dy}{dt} - \frac{2}{t}y = t^2. \tag{1.4}$$

We can now determine the integrating factor,

$$\begin{aligned} \mu(t) &= \exp\left[-\int^t \frac{2}{\tau}\,d\tau\right] \\ &= \exp\left[-2\ln t\right] \\ &= t^{-2}. \end{aligned}$$

Multiplying Equation (1.4) by the integrating factor, $\mu(t)$, we can find the solution:

$$\begin{aligned} t^{-2}\left(\frac{dy}{dt} - \frac{2}{t}y\right) &= t^{-2}t^2 \\ \frac{d}{dt}\left(t^{-2}y\right) &= 1 \\ t^{-2}y(t) &= t + C \\ y(t) &= t^3 + Ct^2. \end{aligned} \tag{1.5}$$

Using the initial condition, $y(1) = 1$, we obtain $C = 0$. Therefore, the solution is $y(t) = t^3$.

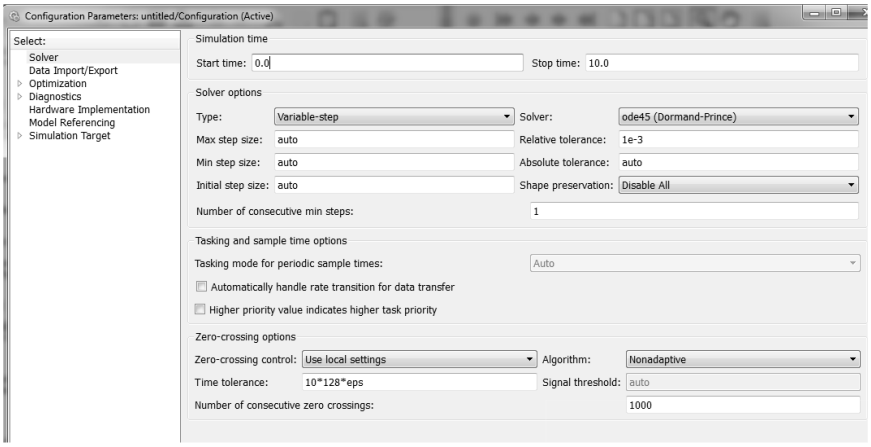


Figure 1.14: System Configuration Parameters.

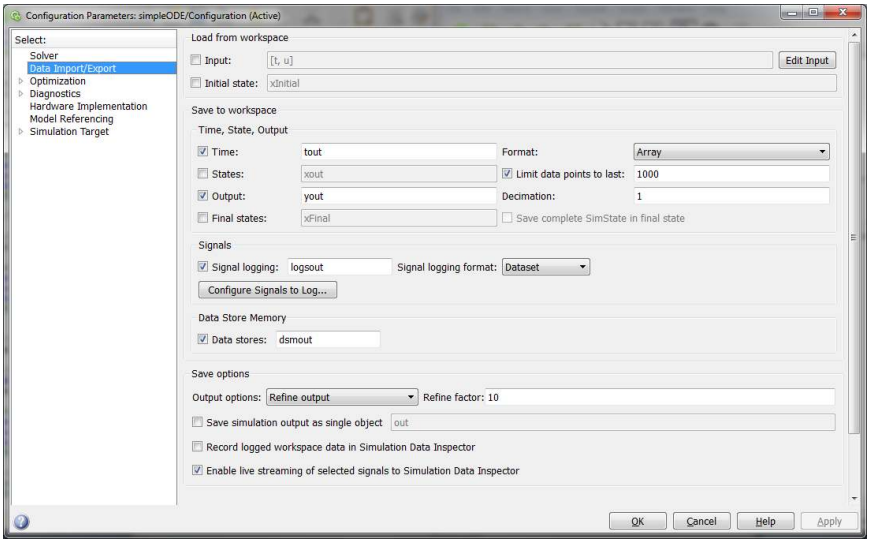


Figure 1.15: Configure Data Import/Export Parameters. Changing the**Refine Factor** can lead to smoother solutions.

The model for this problem is shown in Figure 1.22. Running the simulation, we obtain the numerical solution, $y(t) = t^3$, as shown in Figure 1.23. Computing the difference between the numerical and exact solutions in this case, we find the error is about 6×10^{-5}.

Example 1.3. Consider the initial value problem,

$$\frac{dx}{dt} = 2\sin 3t - 4x, \quad x(0) = 0. \tag{1.6}$$

This is the example that we first solved using Simulink. It is another linear first order differential equation. In standard form is is written as

$$\frac{dx}{dt} + 4x = 2\sin 3t.$$

The integrating factor is found to be

$$\mu(t) = \exp\left[\int 4\,dt\right] = e^{4t}.$$

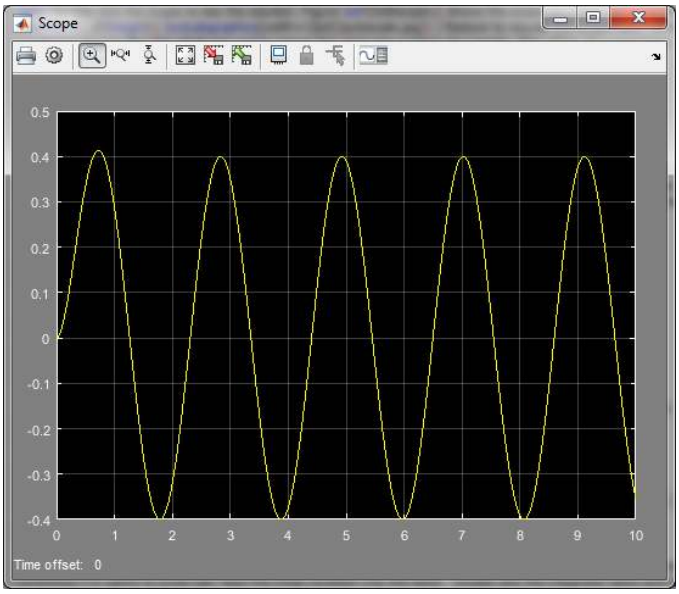


Figure 1.16: **Scope** plot of the solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$, with **Refine Factor**= 10.

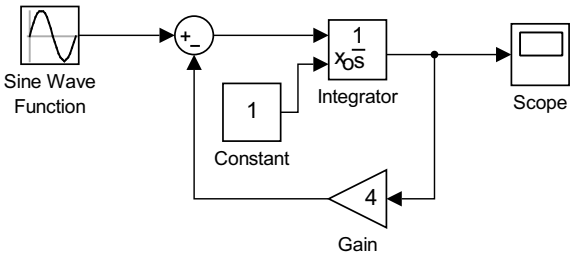


Figure 1.17: Connections for the First Order ODE model for $\frac{dx}{dt} = 2\sin 3t - 4x$ showing how to provide an external initial value.

Multiplying Equation (1.6) by the integrating factor, we can obtain the general solution:

$$\begin{aligned}
\frac{d}{dt}\left(e^{4t}x\right) &= 2e^{4t}\sin 3t \\
e^{4t}x &= 2\int e^{4t}\sin 3t\,dt + C \\
&= \frac{2}{25}e^{4t}\left(4\sin 3t - 3\cos 3t\right) + C \\
x(t) &= \frac{2}{25}\left(4\sin 3t - 3\cos 3t\right) + Ce^{-4t}. \qquad (1.7)
\end{aligned}$$

Using the initial condition, $x(0) = 0$, we find $C = \frac{6}{25}$. Therefore, the particular solution is

$$x(t) = \frac{2}{25}\left(4\sin 3t - 3\cos 3t\right) + \frac{6}{25}e^{-4t}. \qquad (1.8)$$

The solution can be found using Simulink. The model for this exact solution is shown in Figure 1.24. The plot on the scope matches the solution we obtained earlier as seen in Figure 1.13.

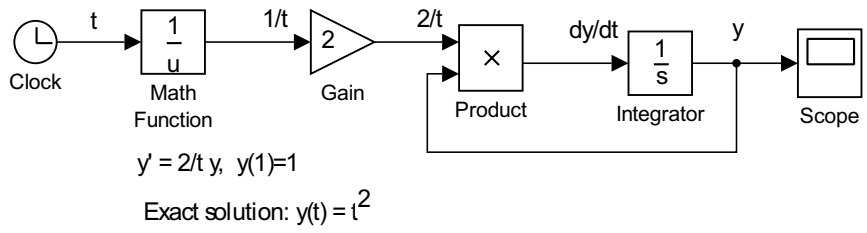


Figure 1.18: First order separable differential equation model.

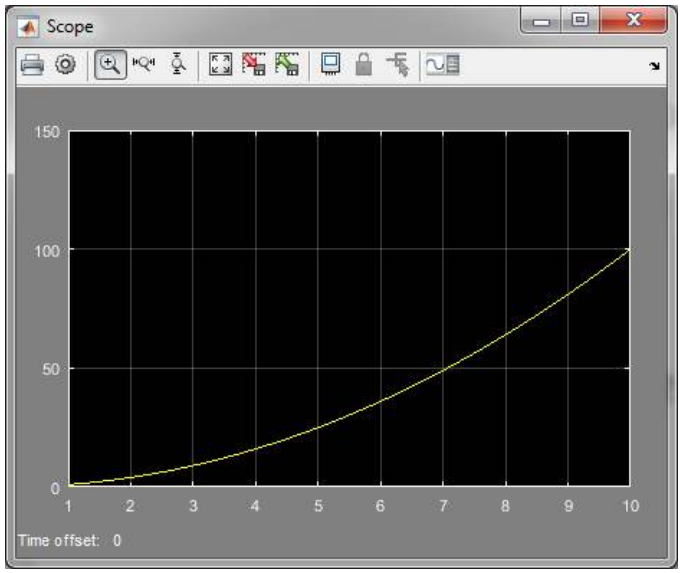


Figure 1.19: **Scope** plot of the solution of initial value problem (1.2), $\frac{dy}{dt} = \frac{2}{t}y$, where $y(1) = 1$.

1.3 Working with Simulink Output

Often we might want to access the solutions in MATLAB. Using the model in Figure 1.18 for a first oder separable equation, we can add the **To Workspace** block. This is shown in Figure 1.25. Double-click and rename the variable as y and change the output type to array. When you run the simulation, it will send the data to MATLAB for further analysis or plotting. This will put **tout** and **y** data into the MATLAB workspace.

In MATLAB you can plot the data using **plot(tout,y)**. You can add labels with **xlabel('t')**, **ylabel('y')**, **title('y vs t')**. Adding the command **set(gcf,'Color',[1,1,1])** makes the plot background white. The result is shown in Figure 1.26.

```
plot(tout,y)
xlabel('t')
ylabel('y')
title('y vs t')
set(gcf,'Color',[1,1,1])
```

Once you have exported your data to the MATLAB workspace and created a plot, then you can use the menu items under **Tools** to annotate the plot. Once you are satisfied with the figure, go to the Edit menu and

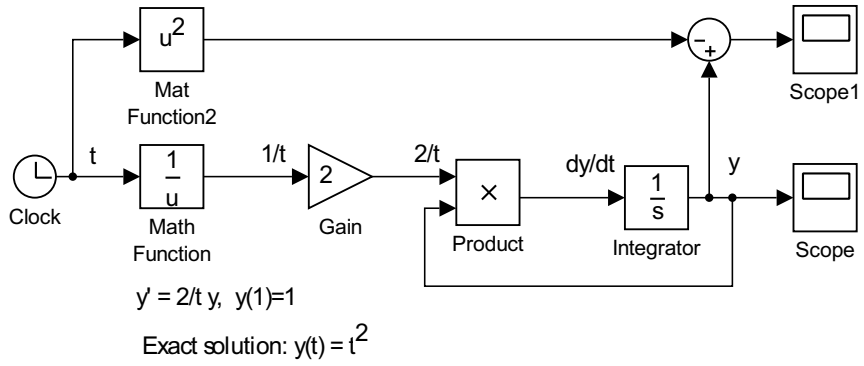


Figure 1.20: First order separable differential equation model with extra blocks to plot the difference between the numerical and exact solution, $y(t) - t^2$, for Equation (1.2).

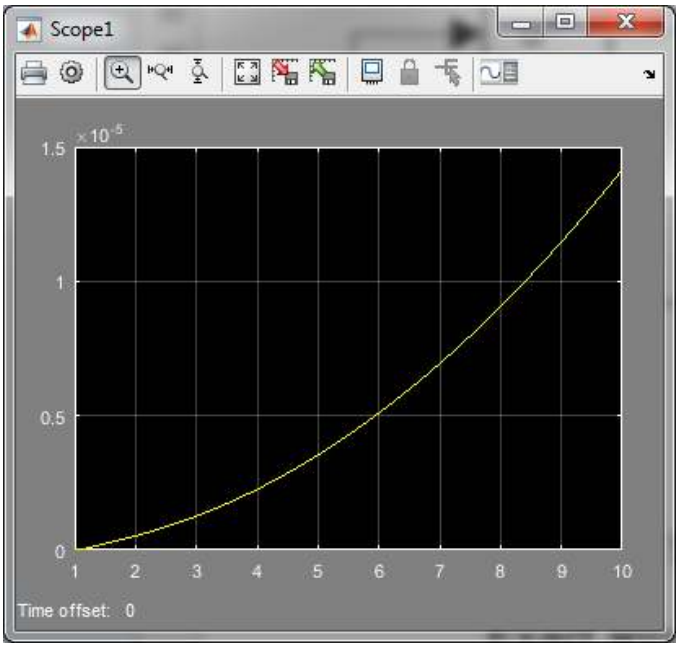


Figure 1.21: **Scope** plot of the difference between the numerical and exact solution, $y(t) - t^2$, for Equation (1.2)

select **Copy Figure**. Go to your report document and **Paste** (CTRL-V) the figure into your document. You can then resize the figure, center it, and add a numbered Figure caption describing the figure. Other methods for recording Simulink Scope images and the Simulink model are described next.

1.4 Printing Simulink Scope Images

In this section we discuss different methods for transferring the plots generated in Simulink models to a document or report. For example, you might want to copy images produced by the scope or your model into an MS Word document. There are several ways you can do this. You might be able to use the Print icon to print to a file or printer, or you can follow one of the following methods. **Note:** In 2015 it was not easy to export plots from Simulink. In the 2017 versions, it is easier to do so and perhaps the

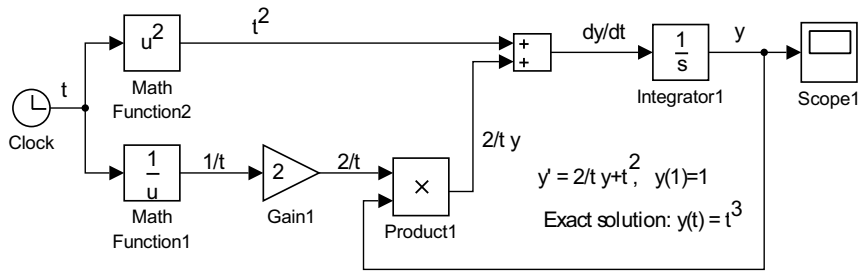


Figure 1.22: Linear first order differential equation model.

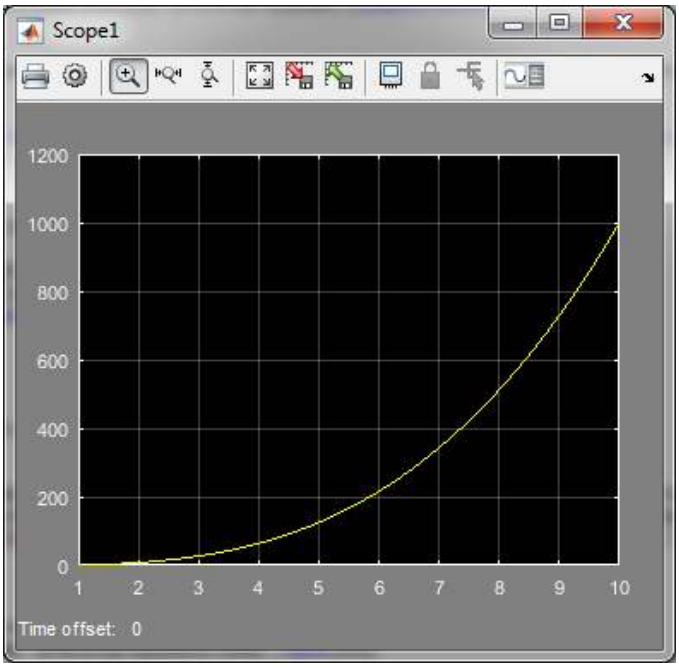


Figure 1.23: **Scope** plot of the difference between the numerical and exact solution, $y(t) - t^2$.

preferred method unless you are using earlier methods.

We compare the 2017a **Scope** figure windows to those shown later from 2015a. In Figure 1.27 one might see slight differences in the Scope icons. In Figure 1.28 the **File** menu shows a menu item for **Print to Figure**. Here one can put the scope figure in a MATLAB figure environment and **Save As** a figure file of different types such as the png image in Figure 1.29. Other methods are provided below for producing output useful for reports.

Method 1:

Select the **Scope** figure window in Figure 1.30, then hit **ALT+PrintScrn** to copy the figure to a clipboard and paste the figure into your application.

Changing the scope appearance.

You might want to change the colors before copying the scope image. Click the **Scope Parameters** icon (2^{nd} icon) and go to the **Style** tab as seen in Figure 1.31. Change the Figure Color to black, Axes Colors to white background and black writing, and Line Color to black. The selection of these parameters is shown in Figure 1.31.

Now the **Scope** plot looks like Figure 1.32.

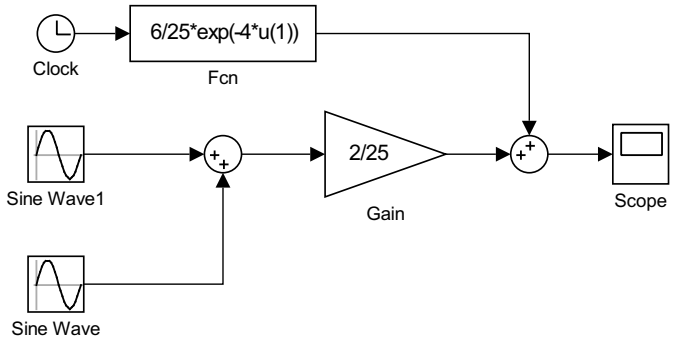


Figure 1.24: Model for plotting the exact solution (1.8) of the initial value problem (1.6).

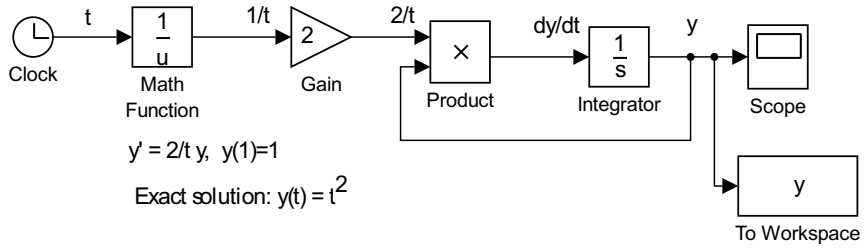


Figure 1.25: Adding To Workspace block for sending output to MATLAB.

Method 2:

Go to **Scope Parameters** and select the **History** tab. Check the **Save data to workspace** box. Note the variable name. Let's change the name to MyScopeData for this example. Saving with **Structure with time** will save the data as a structure. Run the simulation again.

Now, go into the MATLAB command window. You should see the MyScopeData data in the variable list. Type

```
plot(MyScopeData.time, MyScopeData.signals.values)
```

This gives the MATLAB plot in Figure 1.34 which can be manipulated and saved or copied as an image.

Method 3:

You can save the scope image as a jpg image. Create the MATLAB code in Table 1.1. Save this code as an m-file with a name like **prfig.m**. In MATLAB run prfig (type prfig in the Command Window.) It should produce the file 'mypic.jpg in your MATLAB folder. Of course, you can change the name of the image before running **prfig.m** before inserting the figure into your MS Word document as a Picture file.

Method 4:

Add **To Workspace** block.

You can add a **To Workspace** block to your simulation. This will automatically place the data in the MATLAB space. Go to the Simulink library and add a **To Workspace** block to your model as discussed in the last section. Connect this block to the input of the **Scope** (Right-click the input line and drag to connect to the **To Workspace** block.) This will give the connection as shown in Figure 1.36.

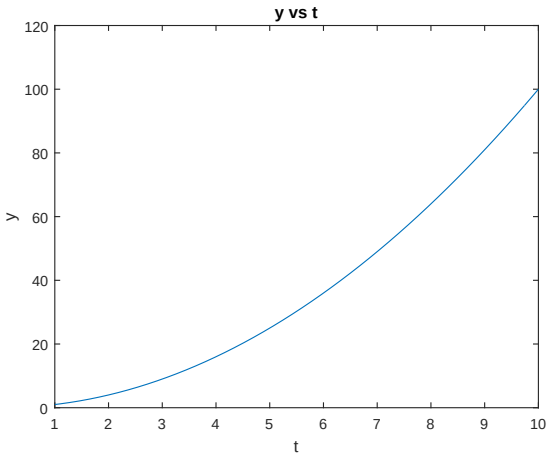


Figure 1.26: Plot of model solution in MATLAB.

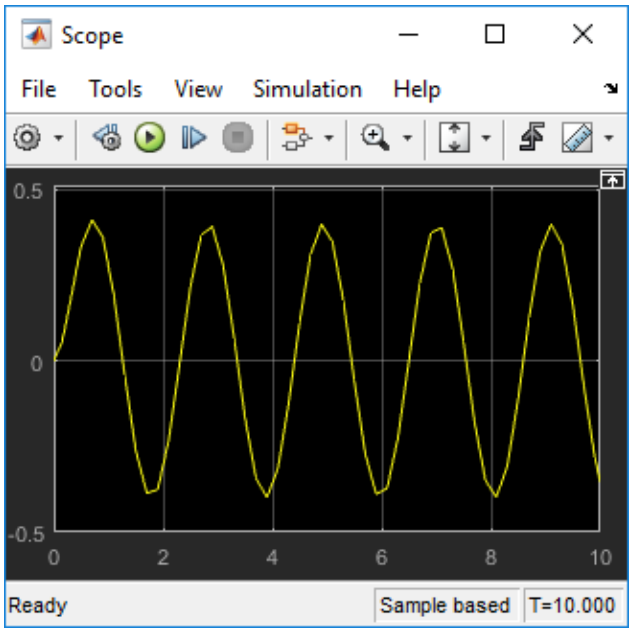


Figure 1.27: **Scope** plot using MATLAB 2017a. Note the differences in some icons.

You can double-click this block and change the variable name that will be saved. Let's assume it is simout. Then, run the simulation. Go into MATLAB and type

```
plot(simout.time,simout.data)
```

This will give you a plot of the **Scope** data. Now you can print, save as an image, or copy (under Edit) to an MS Word document. Below is what you get using Copy Figure under the Edit menu item in the Figure window.

Printing Models

Printing models.

Once you have made a model, you might want to include it in a report. It is easy to capture a model, but a complicated model might not print large enough to see the component annotations.

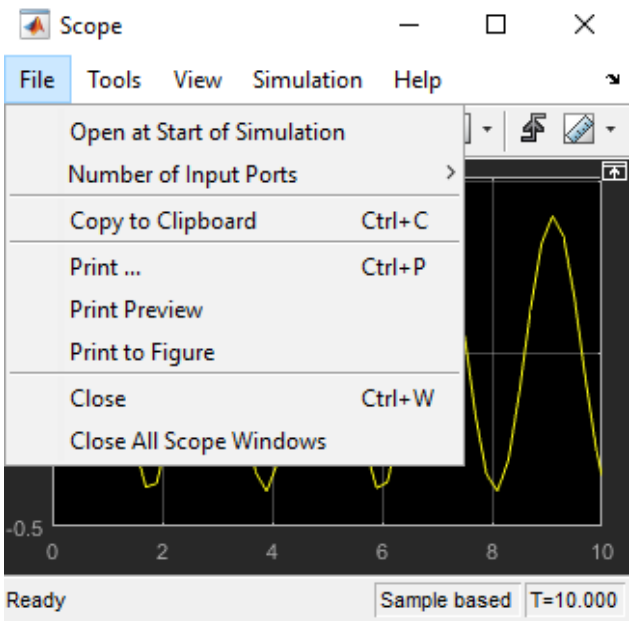


Figure 1.28: **Scope** plot showing File menu where one can Print to Figure.

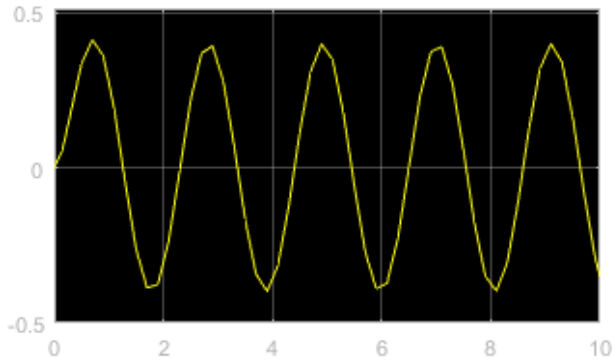


Figure 1.29: MATLAB png image saved as png file. The colors and other attributes can be changed before saving the **Scope** plot.

First open the desired model. Then, in MATLAB you can use the **print** command to print the model. For example, typing the following in the MATLAB command window prints the open model to an encapsulated postscript file:

```
print -s -deps -r300 mymodel.eps
```

For jpg files, you can use

```
print -s -djpeg -r300 mymodel.jpg
```

For other formats, consult the MATLAB help system.

1.5 Scilab and Xcos

THERE ARE ALTERNATIVES TO USING MATLAB. One example is Xcos. Xcos is part of Scilab. Scilab is free and open source software for numerical

Xcos is part of Scilab, an open source alternative.

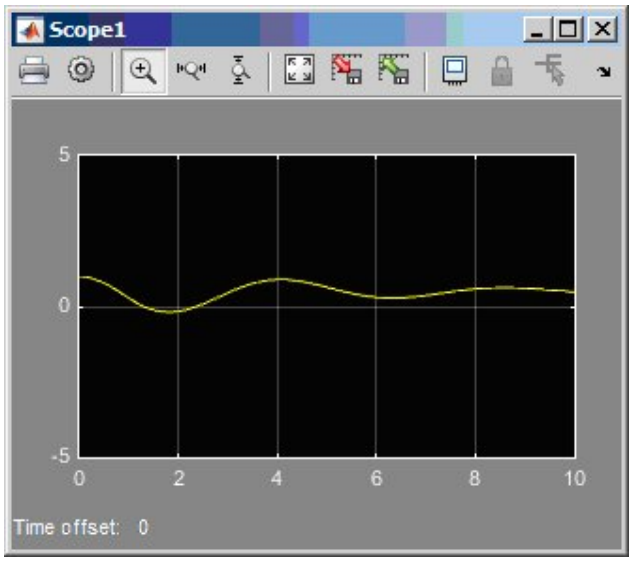


Figure 1.30: **Scope** plot. Note that the plots in this section are generated by the oscillator model in the next chapter.

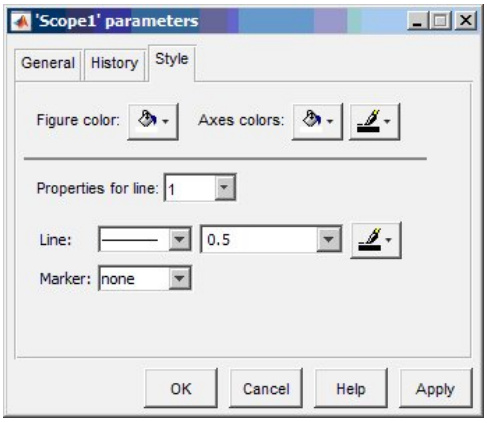


Figure 1.31: **Scope** color parameters.

computation similar to MATLAB. Xcos is a graphical design environment. The Xcos environment is shown in Figure 1.38.

After downloading and installing Scilab from `http://www.scilab.org/`, one can type **xcos** or click on the icon  to launch Xcos. This brings up the Xcos Palettes browser and Xcos workspace as shown in Figures 1.39 and 1.40. This looks similar to Simulink's Library Browser as shown in Figure 1.5.

In Figure 1.41 we show the model for solving the first example of this chapter:

$$\frac{dx}{dt} = 2\sin 3t - 4x, \quad x(0) = 0.$$

This is equivalent to the Simulink model in Figure 1.6. We see that this model is similar to the Simulink construction. However, there are are some differences. First of all, the block have a different appearance.

Next, there are some differences in setting up the block parameters. The **Sum** block is set up by double-clicking the block and entering the signs and number of input ports as [1;-1]. This indicates that the **Sum** block has

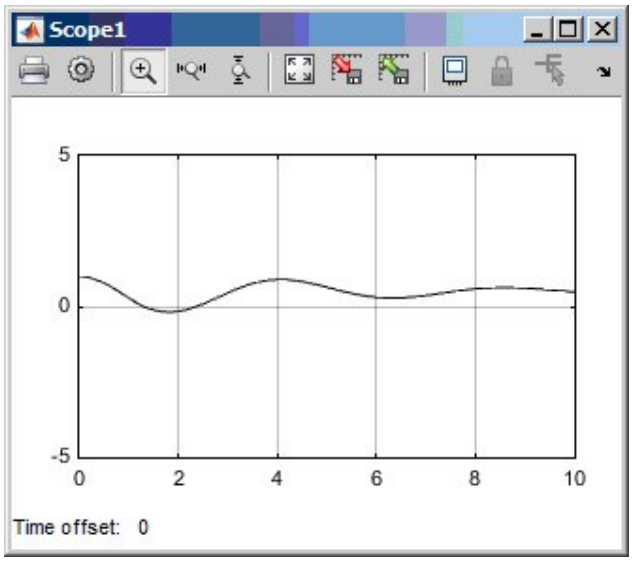


Figure 1.32: **Scope** plot with a white background.

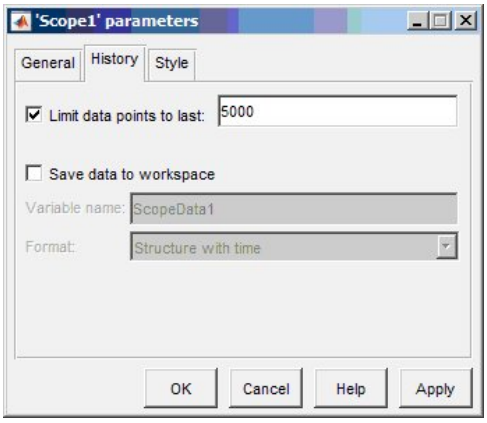


Figure 1.33: History tab in **Scope parameters**.

two inputs. The first is positive and the second is negative.

The scope requires an additional input. Namely the time is entered using a clock. In Simulink this is automatic, though we had also used the clock to introduce time as an independent variable when needed.

The initial condition and the sine function parameters are entered by double-clicking the integrator and sine block, respectively.

In order to run the simulation, one can click the "play" icon or select **Start** under the **Simulation** menu item. The ODE solver can be changed through **Setup** under the **Simulation** menu. The solution is shown in Figure 1.42.

The blocks are not labeled like Simulink. One can label the blocks by right-clicking and selecting **Edit** under the **Format** item. There one can enter text to appear with the block. Annotation of the workspace is done by selecting a **Text_f** block and adding text to it and changing the fontsize. Sample annotations are shown in Figure 1.43.[4]

We spent time earlier discussing how to capture images of the output and models for reports. In Xcos it is a simple matter to Export the

[4] In newer Simulink versions the block labels disappear. This can be changed by going into the block Properties and changing the model parameters 'HideAutomaticName' or 'ShowName.'

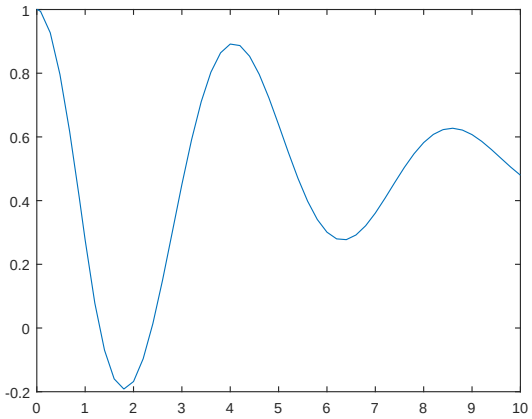


Figure 1.34: Plot generated by Method 2.

```
shh = get(0,'ShowHiddenHandles');
set(0,'ShowHiddenHandles','On')
set(gcf,'PaperPositionMode','auto')
set(gcf,'InvertHardcopy','off')
saveas(gcf,'mypic.jpg')
set(0,'ShowHiddenHandles',shh)
```

Table 1.1: MATLAB code for saving the scope image as a jpg image.

model or the solutions by selecting **Export** under the **File** menu. There are options for saving these to different formats. The images can also be modified by changing the axis range, fonts, colors, etc.

1.6 First Order ODEs in MATLAB

One can use MATLAB to obtain solutions and plots of solutions of differential equations. This can be done either symbolically, using **dsolve**, or numerically, using numerical solvers like **ode45**. In this section we will provide examples of using these to solve first order differential equations. We will end with the code for drawing direction fields, which are useful for looking at the general behavior of solutions of first order equations without explicitly finding the solutions.

Symbolic Solutions

The function **dsolve** obtains the symbolic solution and **ezplot** is used to quickly plot the symbolic solution. As an example, we apply **dsolve** to solve the main model in this chapter.

At the MATLAB prompt, type the following:

```
sol = dsolve('Dx=2*sin(t)-4*x','x(0)=0','t');
```

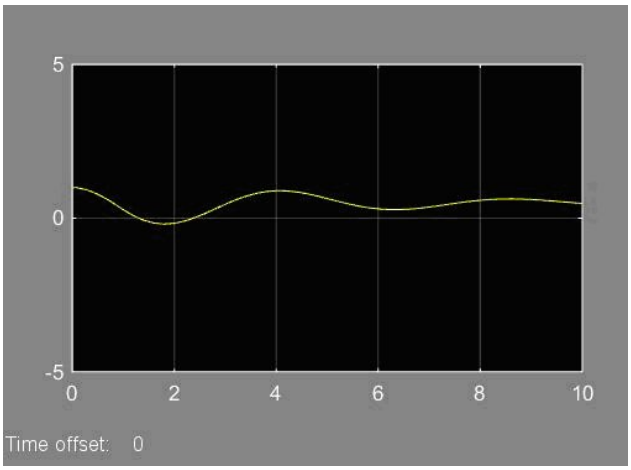


Figure 1.35: Scope plot from Method 3.

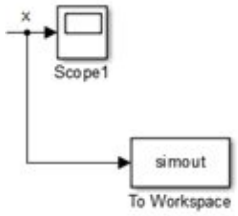


Figure 1.36: Use of a **To Workspace** block.

```
ezplot(sol,[0 10])
xlabel('t'),ylabel('x'), grid
```

The solution is given as

```
sol =

 (2*exp(-4*t))/17 - (2*17^(1/2)*cos(t + atan(4)))/17
```

Figure 1.44 shows the solution plot.

Another approach to symbolically solving a differential equation is provided in the following code snippet. The dsolve command symbolically solves the equation given by 'eq' and using the initial condition in 'cond.' It then plots the solution. This results in the same plot and solution obtained earlier in the chapter.

```
syms t y(t)
dy = diff(y,t);
eq = dy + 4*y == 2*sin(3*t);
cond = y(0) == 0;
sol(t) = dsolve(eq,cond)
fplot(sol,[0 10])
xlabel('t')
ylabel('y')
title('dy/dt-4y = t')
```

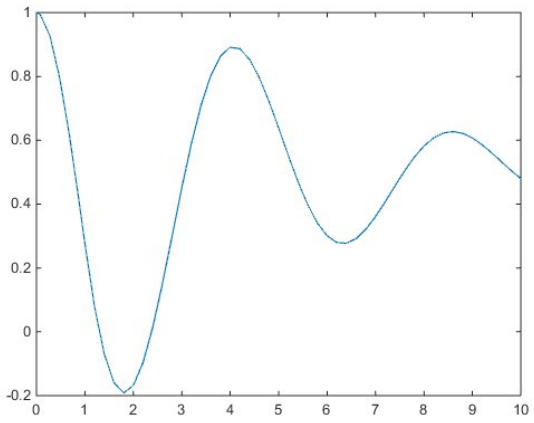


Figure 1.37: **Scope** plot from Method 4.

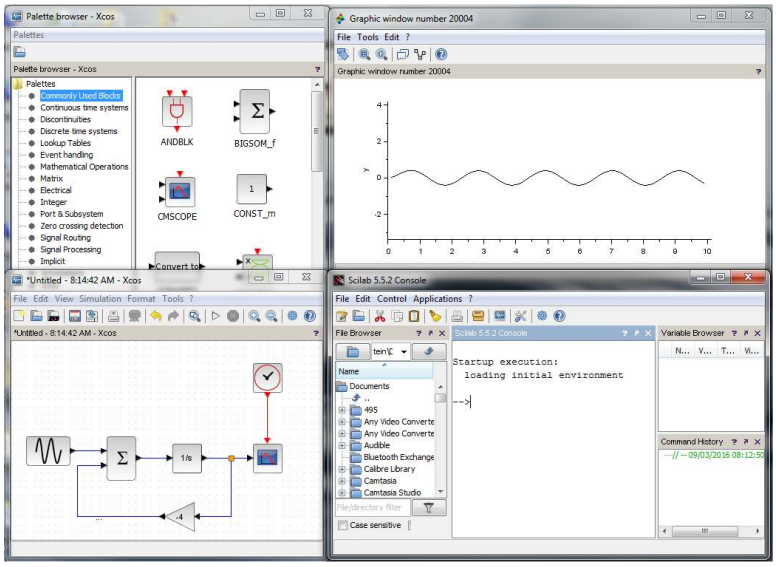


Figure 1.38: The Xcos workspace.

ODE45 and Other Solvers.

THERE ARE SEVERAL ODE SOLVERS in MATLAB, implementing Runge-Kutta and other numerical schemes. Examples of its use are in the differential equations textbook. For example, one can implement **ode45** to solve the initial value problem

$$\frac{dy}{dt} = -\frac{yt}{\sqrt{2-y^2}}, \quad y(0) = 1,$$

using the following code:

```
[t y]=ode45('func',[0 5],1);
plot(t,y)
xlabel('t'),ylabel('y')
title('y(t) vs t')
```

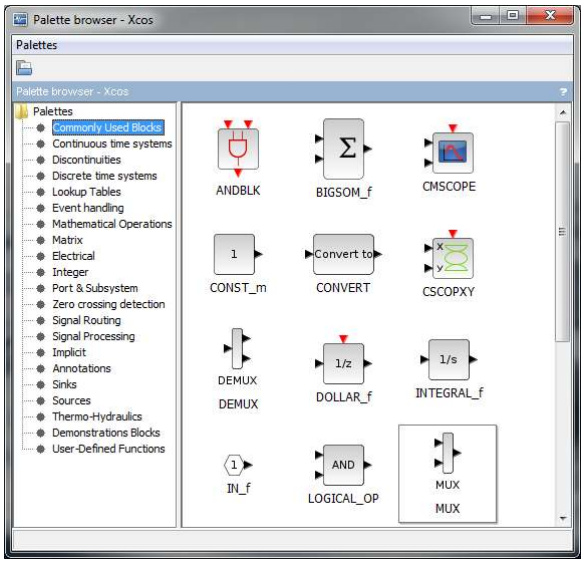


Figure 1.39: The Xcos Palette browser.

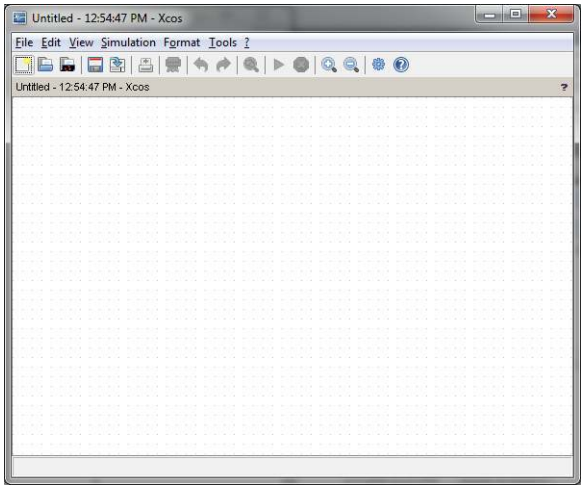


Figure 1.40: The Xcos workspace.

One can define the function **func** in a file **func.m** such as

```
function f=func(t,y)
f=-t*y/sqrt(2-y.^2);
```

Running the above **ode45** code produces Figure 1.45.
One can also use **ode45** to solve higher order differential equations. Second order differential equations are discussed in Chapter 3 Section 5. See MATLAB help for other examples and other ODE solvers.

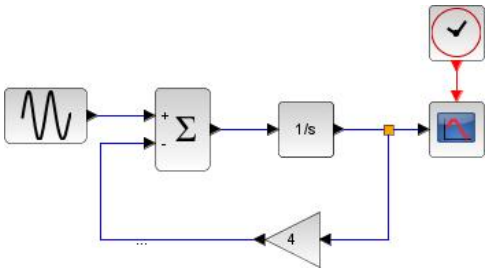


Figure 1.41: The Xcos model for solving the first order ODE $\frac{dx}{dt} = 2\sin 3t - 4x$.

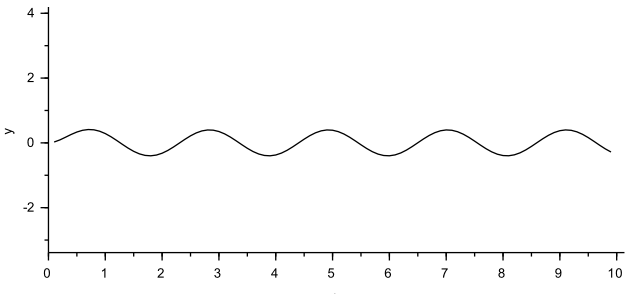


Figure 1.42: The Xcos model solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$.

Direction Fields

One can produce direction fields in MATLAB. For the differential equation

$$\frac{dy}{dx} = f(x, y),$$

we note that $f(x, y)$ is the slope of the solution curve passing through the point in the xy=plane. Thus, the direction field is a collection of tangent vectors at points (x, y) indication the slope, $f(x, y)$, at that point.

A sample code for drawing direction fields in MATLAB is given by

```
[x,y]=meshgrid(0:.1:2,0:.1:1.5);
dy=1-y;
dx=ones(size(dy));
quiver(x,y,dx,dy)
axis([0,2,0,1.5])
xlabel('x')
ylabel('y')
```

The mesh command sets up the xy-grid. In this case x is in $[0, 2]$ and y is in $[0, 1.5]$. In each case the grid spacing is 0.1.

We let dy = 1-y and dx =1. Thus,

$$\frac{dy}{dx} = \frac{1-y}{1} = 1 - y.$$

The **quiver** command produces a vector (dx,dy) at (x,y). The slope of each vector is dy/dx. The other commands label the axes and provides a window with xmin=0, xmax=2, ymin=0, ymax=1.5. The result of using the above code is shown in Figure 1.46.

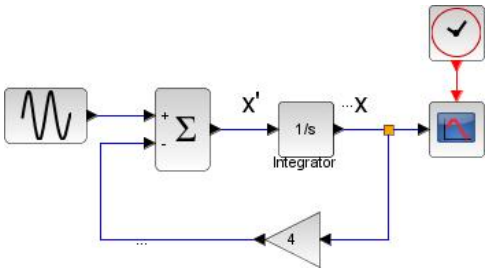


Figure 1.43: The Xcos model with annotation added.

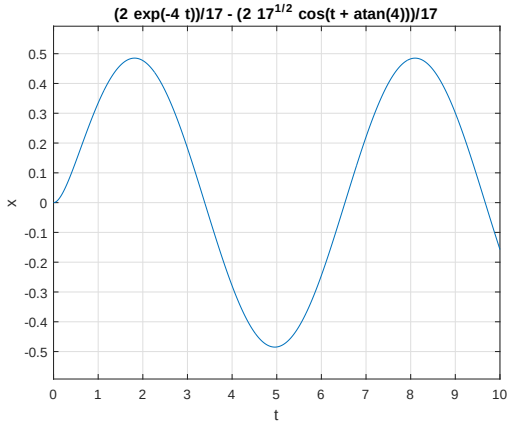


Figure 1.44: The solution of Equation (1.1) with $x(0) = 0$ found using MATLAB's **dsolve** command.

One can add solution, or integral, curves to the direction field for different initial conditions to further aid in seeing the connection between direction fields and integral curves. One needs to add to the direction field code the following lines:

```
hold on
[t,y] = ode45(@(t,y) 1-y, [0 2], .5);
plot(t,y,'k','LineWidth',2)
[t,y] = ode45(@(t,y) 1-y, [0 2], 1.5);
plot(t,y,'k','LineWidth',2)
hold off
```

Here the function $f(t, y) = 1 - y$ is entered this time using MATLAB's anonymous function, **@(t,y) 1-y**. Before plotting, the **hold** command is invoked to allow plotting several plots on the same figure. The result is shown in Figure 1.47

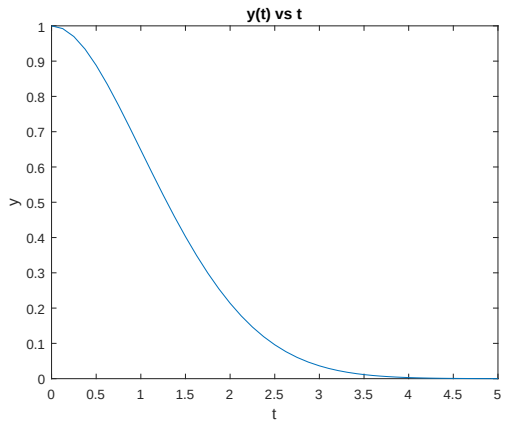


Figure 1.45: A plot of the solution of $\frac{dy}{dt} = -\frac{yt}{\sqrt{2-y^2}}$, $y(0) = 1$, found using MATLAB's **ode45** command.

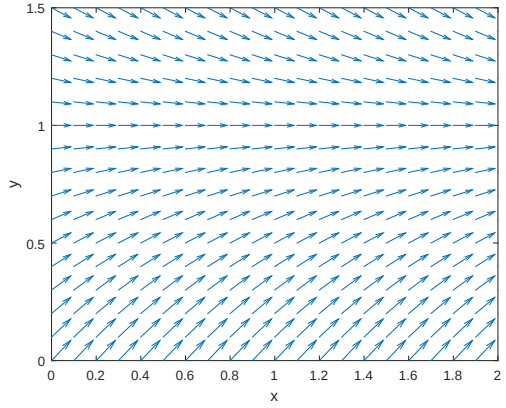


Figure 1.46: A direction field produced using MATLAB's **quiver** function for $y' = 1 - y$.

1.7 Exercises

1. Construct the model in Figure 1.6 for solving the initial value problem $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$, and produce a plot of the solution.

2. Modify the model in the Problem 1. to solve $\dfrac{dx}{dt} = f(t) - 2x$ for a different function, $f(t)$ and initial condition.

3. Solve the following initial value problems using MATLAB's **dsolve** command (See Section 1.6) and Simulink. Provide plots of the solutions for both cases. How do the solutions compare?

   a. $y' = xy$, $y(0) = 1$.
   b. $y' = 2y(3 - y)$, for different initial conditions, $y(0) = 4$, $y(0) = 2$, and $y(0) = -1$.
   c. $y' = 1 + x + y$, $y(0) = 1$.
   d. $y' = (y^2 - 4)(y - 4)$ for different initial conditions, $y(0) = 5$, $y(0) = 3$, $y(0) = 1$, $y(0) = -1$, and $y(0) = -3$.

4. Use MATLAB to plot direction fields for the following:

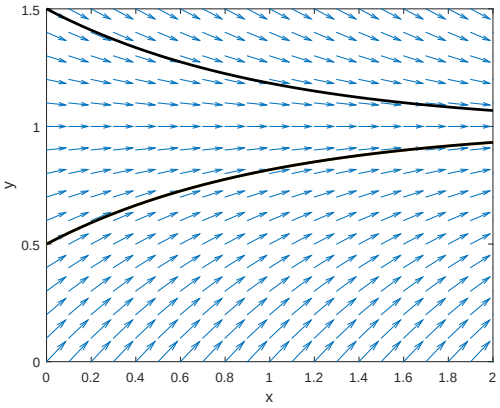


Figure 1.47: A direction field produced using MATLAB's **quiver** function for $y' = 1 - y$ with solution curves added.

a. $y' = xy$.
b. $y' = 2y(3 - y)$.
c. $y' = 1 + x + y$.
d. $y' = (y^2 - 4)(y - 4)$.

5. Solve the following initial value problems using one of MATLAB's numerical ODE solvers like **ode45**. Plot the solutions and compare with the corresponding solutions in Problem 3.

a. $y' = xy,\ y(0) = 1$.
b. $y' = 2y(3 - y)$, for different initial conditions, $y(0) = 4$, $y(0) = 2$, and $y(0) = -1$.
c. $y' = 1 + x + y,\ y(0) = 1$.
d. $y' = (y^2 - 4)(y - 4)$ for different initial conditions, $y(0) = 5$, $y(0) = 3$, $y(0) = 1$, $y(0) = -1$, and $y(0) = -3$.

2

First Order Differential Equations

We have seen how to solve simple first order differential equations using Simulink. In particular we have solved initial value problems for the equations

$$\frac{dy}{dt} = \frac{2}{t}y, \quad y(1) = 1, \tag{2.1}$$

$$\frac{dy}{dt} = \frac{2}{t}y + t^2, \quad y(1) = 1, \tag{2.2}$$

$$\frac{dx}{dt} = 2\sin 3t - 4x, \quad x(0) = 0. \tag{2.3}$$

The Simulink models were provided in Figures 1.18, 1.22, and 1.6, respectively.

In this chapter we solve a few more first order equations in the form of applications. These will include growth and decay, Newton's Law of Cooling, pursuit curves, free fall and terminal velocity, the logistic equation, and the logistic equation with delay.

2.1 Exponential Growth and Decay

The simplest differential equations are those governing growth and decay. As an example, we will discuss population models.

Let $P(t)$ be the population at time t. We seek an expression for the rate of change of the population, $\frac{dP}{dt}$. Assuming that there is no migration of population, the only way the population can change is by adding or subtracting individuals in the population. The equation would take the form

$$\frac{dP}{dt} = \text{Rate In} - \text{Rate Out}.$$

The *Rate In* could be due to the number of births per unit time and the *Rate Out* by the number of deaths per unit time. The simplest forms for these rates would be given by terms proportional to the population:

$$\text{Rate In} = bP \quad \text{and} \quad \text{Rate Out} = mP.$$

Here we have denoted the birth rate as b and the mortality rate as m. This gives the total rate of change of population as

$$\frac{dP}{dt} = bP - mP \equiv kP, \tag{2.4}$$

where $k = b - m$.

Equation (2.4) is easily modeled in Simulink. All of the needed blocks are under the Commonly Used Blocks group. We need an **Integrator**, **Constant**, **Gain**, and a **Scope** block. The output from the **Integrator** can be feed into a **Gain** control, which represents k, and the output from the **Gain**, kP, can then be used as an input to the **Integrator**. We add the **Scope** in order to plot the solution. The model is shown in Figure 2.1. Note that a **Constant** block was added to provide an external input of the initial condition.

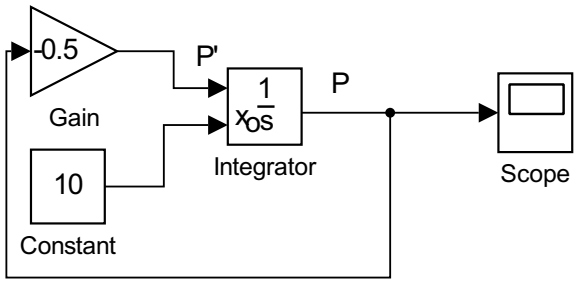


Figure 2.1: Simulink model for exponential growth and decay. The initial value, $P(0) = 10$, is set in the **Constant** block and $k = -0.5$ is set in the **Gain**.

The solution for exponential decay with $P(0) = 10$ and $k = -0.5$ is shown in Figure 2.2. The simulation time was set at 10s.

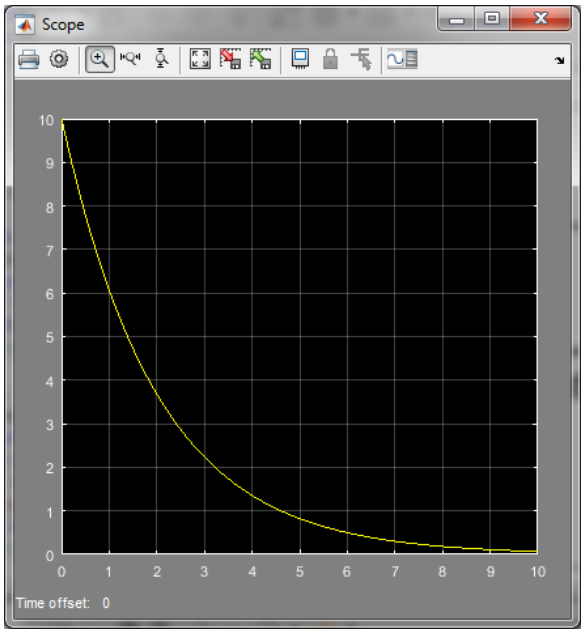


Figure 2.2: Solution for the exponential decay with $P(0) = 10$ and $k = -0.5$. The simulation time was set at 10.

The exact solution is easily found noting that Equation (2.4) is a separable equation. Rearranging the equation, its differential form is

$$\frac{dP}{P} = k\,dt.$$

Integrating, we have

$$\int \frac{dP}{P} = \int k\,dt$$

$$\ln|P| = kt + C. \tag{2.5}$$

Next, we solve for $P(t)$ through exponentiation,

$$\begin{aligned} |P(t)| &= e^{kt+C} \\ P(t) &= \pm e^{kt+C} \\ &= Ae^{kt}. \end{aligned} \tag{2.6}$$

Here we have defined the arbitrary constant, $A = \pm e^C$.

If the population at $t = 0$ is P_0, i.e., $P(0) = P_0$, then the solution gives $P(0) = Ae^0 = A = P_0$. So, the solution of the initial value problem is

$$P(t) = P_0 e^{kt}.$$

In the Simulink model, the initial value was given as $P(0) = 10$ and the decay constant by $k = -0.5$. Therefore, the solution in Figure 2.2 is of the function $P(t) = 10e^{-0.5t}$.

Equation (2.4) is the familiar exponential model of population growth:

Malthusian population growth.

$$\frac{dP}{dt} = kP.$$

We obtained solutions exhibiting exponential growth $(k > 0)$ or decay $(k < 0)$. This Malthusian growth model has been named after Thomas Robert Malthus (1766-1834), a clergyman who used this model to warn of the impending doom of the human race if its reproductive practices continued. Later we modify this model to account for competition for resources, leading to the logistic differential equation.

2.2 Newton's Law of Cooling

If you take your hot cup of tea, and let it sit in a cold room, the tea will cool off and reach room temperature after a period of time. The law of cooling is attributed to Isaac Newton (1642-1727) who was probably the first to state results on how bodies cool.[1] The main idea is that a body at temperature $T(t)$ is initially at temperature $T(0) = T_0$. It is placed in an environment at an ambient temperature of T_a. The goal is to find the temperature at a later time, $T(t)$.

[1] Newton's 1701 Law of Cooling is an approximation to how bodies cool for small temperature differences $(T - T_a \ll T)$ and does not take into account all of the cooling processes. One account is given by C. T. O'Sullivan, Am. J. Phys (1990) p 956-960.

We will assume that the rate of change of the temperature of the body is proportional to the temperature difference between the body and its surroundings. Thus, we have

$$\frac{dT}{dt} \propto T - T_a.$$

The proportionality is removed by introducing a cooling constant,

$$\frac{dT}{dt} = -k(T - T_a), \tag{2.7}$$

where $k > 0$.

This differential equation can be solved by first rewriting the equations as

$$\frac{d}{dt}(T - T_a) = -k(T - T_a).$$

This now takes the form of exponential decay of the function $T(t) - T_a$. The solution is easily found as

$$T(t) - T_a = (T_0 - T_a)e^{-kt},$$

or

$$T(t) = T_a + (T_0 - T_a)e^{-kt}.$$

Example 2.1. A cup of tea at 90^oC cools to 85^oC in ten minutes. If the room temperature is 22^oC, what is its temperature after 30 minutes?

Using the general solution with $T_0 = 90^o$C,

$$T(t) = 22 + (90 - 22)e^{-k} = 22 + 68e^{-kt},$$

we then find k using the given information, $T(10) = 85^o$C. We have

$$\begin{aligned} 85 &= T(10) \\ &= 22 + 68e^{-10k} \\ 63 &= 68e^{-10k} \\ e^{-10k} &= \frac{63}{68} \approx 0.926 \\ -10k &= \ln 0.926 \\ k &= -\frac{\ln 0.926}{10} \\ &\approx 0.00764 \text{ min}^{-1}. \end{aligned}$$

This gives the solution for this model as

$$T(t) = 22 + 68e^{-0.00764t}.$$

Now we can answer the question. What is $T(30)$?

$$T(30) = 22 + 68e^{-0.00764(30)} = 76^o\text{C}.$$

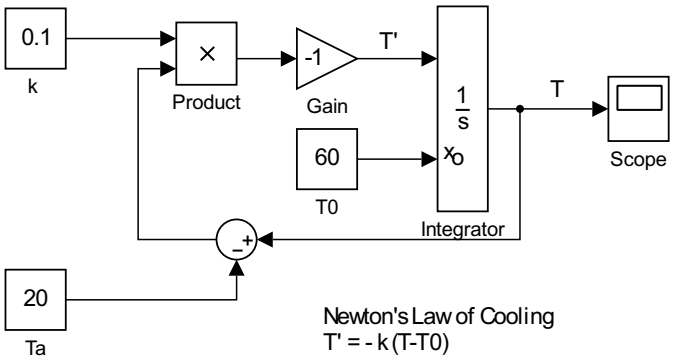


Figure 2.3: Simulation model for Newton's Law of Cooling, $T' = -k(T - T_a)$, $T(0) = T0$. Here we set $k = 0.1$ s^{-1}, $T_a = 20^o$C, and $T_0 = 60^o$C.

Next we model Equation (2.7) in Simulink. The input for the **Integrator** is simply $-k(T - T_a)$. We need to define the constants k and T_a. We will externally input the initial condition, $T(0) = T_0$ in the **Integrator** block. The simple model is shown in Figure 2.3. In this case we set $k = 0.1$ s^{-1}, $T_a = 20^o$C, and $T_0 = 60^o$C. Running the simulation for 100 s, we obtain the solution shown in Figure 2.4.

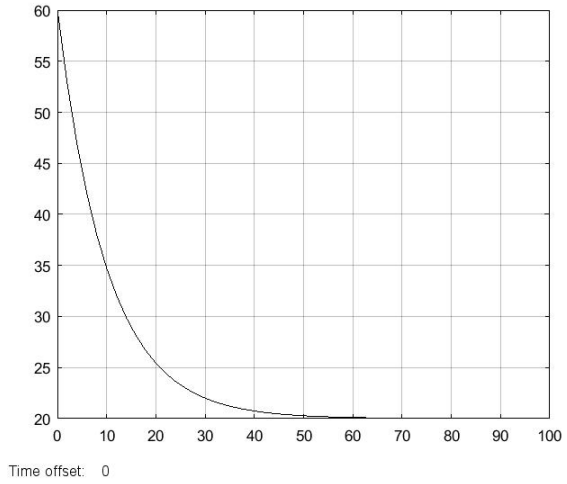


Figure 2.4: Solution of Newton's Law of Cooling example.

How good is the solution? We can solve the problem by hand for this set of parameters. However, we will take this opportunity to introduce the idea of a subsystem and set up a model in which we can interactively modify the constants and get Simulink to automatically provide the exact solution for comparison.

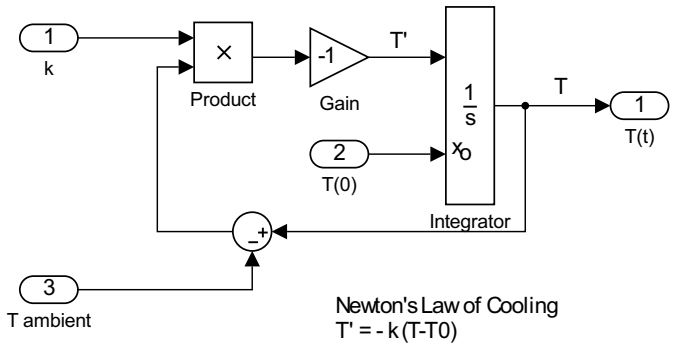


Figure 2.5: Creating a subsystem for the Newton's Law of Cooling model.

We begin by replacing the scope with an output block. The **Out1** block can be found in the Sink group. The input to the subsystem will be the three parameters, k, T_0, and T_a. Each of these constant blocks in Figure 2.3 will be replaced by an **In1** block, found in the Sources group. In Figure 2.5 the three inputs and one output are now oval blocks.

Creating a subsystem.

Double-click each of the three input blocks, one at a time, and set the Port Number of k, T_0, and T_a, to 1, 2, and 3, respectively. Finally, rename each of these controls using the labels that make sense, such as k for k. In

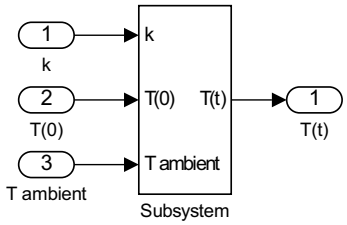


Figure 2.6: Subsystem for Newton's Law of Cooling, $T' = -k(T - T_a)$, $T(0) = T0$.

Figure 2.5 we show the subsystem that we have created.

Now highlight the entire subsystem using **CTRL-A**. In the menu system, look for **Create Subsystem from Selection**. This is under the menu item **Diagram** and subitem **Subsystem & Model Reference**. Rearranging the resulting subsystem, one has something like the subsystem block in Figure 2.6. This is the equivalent of a black box with three inputs and one output.

Next, we can make use of the subsystem just created. Replace the three input ports with constant blocks. Rename the **Constant** blocks with the parameter name and fill each block with a value. The output port can be replaced with a **Scope** block, or any other form of output desired. This can be seen in Figure 2.7.

Before finishing with this model, we will build in the exact solution. Recall that the general solution can be written in terms of the parameters as

$$T(t) = T_a + (T_0 - T_a)e^{-kt}.$$

So, we can feed the values of the parameters in the model into a **Fcn** block and output the exact solution for comparison. We will also need a time value. So, we will need the **Clock** block as well.

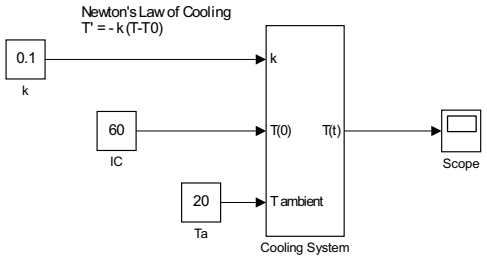


Figure 2.7: Using a user-created subsystem for Newton's Law of Cooling.

The entire model is shown in Figure 2.8. The subsystem is labeled **Cooling System** The top portion is a repetition of the Newton's Law of Cooling model implemented previously.

We have added a **Fcn** block from the User-Defined Functions group. The input will be a vector containing all of the variables in the exact solution. This is accomplished by adding a **Mux** (or Multiplex) block. Double-click the **Mux** block and set the number of inputs to 4.

Now, double-click the **Fcn** block and enter the exact solution in the form

```
u(1)+u(2)*exp(-u(3)*u(4))
```

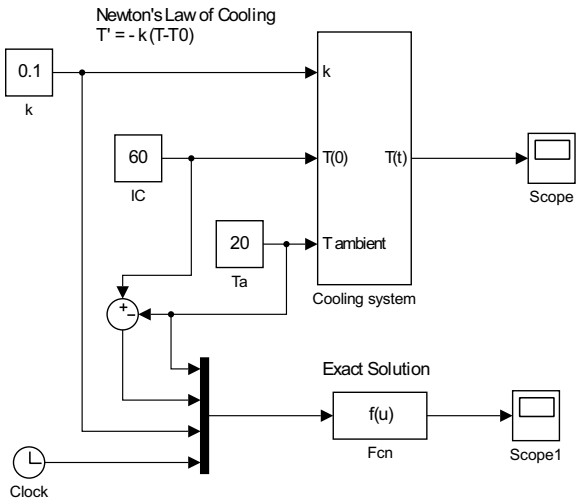


Figure 2.8: Model of Newton's Law of Cooling, $T' = -k(T - T_a)$, $T(0) = T0$, using the subsystem feature.

Here we have assumed that the variables are fed into the **Mux** block in the order T_a, $T_0 - T_a$, k, and t. In Figure 2.8 one can see how the values are routed into the **Mux** block.

The output can be attached to a second scope, as shown, or can be subtracted from the output of the **Cooling System** block to show the closeness of the two solutions. One can also send the output to MATLAB using the **To Workspace** block.

2.3 Free Fall with Drag

Consider an object falling to the ground with air resistance? Free fall is the vertical motion of an object solely under the force of gravity. It has been experimentally determined that an object near the surface of the Earth falls at a constant acceleration in the absence of other forces, such as air resistance. This constant acceleration is denoted by $-g$, where g is called the acceleration due to gravity. The negative sign is an indication that we have chosen a coordinate system in which "up" is positive.

We are interested in determining the position, $y(t)$, of a falling body as a function of time. The differential equation governing free fall is have

$$\ddot{y}(t) = -g. \tag{2.8}$$

Note that we will occasionally use a dot to indicate time differentiation.

We need to model the air resistance. As an object falls faster and faster, the resistive force becomes greater. This drag force is a function of the velocity. The idea is to write Newton's Second Law of Motion $F = ma$ in the form

$$m\ddot{y} = -mg + f(v), \tag{2.9}$$

where $f(v)$ gives the resistive force and mg is the weight. Note that this applies to free fall near the Earth's surface. Also, for $f(v)$ to be a resis-

tive force, $f(v)$ should oppose the motion. If the body is falling, then $f(v)$ should be positive. If the body is rising, then $f(v)$ would have to be negative to indicate the opposition to the motion.

We will model the drag as quadratic in the speed, $f(v) = bv^2$.

Example 2.2. Solve the free fall problem with $f(v) = bv^2$.

The differential equation that we need to solve is

$$\dot{v} = kv^2 - g, \tag{2.10}$$

where $k = b/m$. Note that this is a first order equation for $v(t)$.

Formally, we can separate the variables and integrate over time to obtain

$$t + C = \int^v \frac{dz}{kz^2 - g}. \tag{2.11}$$

If we can do the integral, then we have a solution for v. We evaluate this integral using Partial Fraction Decomposition.

In order to factor the denominator in the current problem, we first have to rewrite the constants. We let $\alpha^2 = g/k$ and write the integrand as

$$\frac{1}{kz^2 - g} = \frac{1}{k}\frac{1}{z^2 - \alpha^2}. \tag{2.12}$$

Noting that

$$\frac{1}{kz^2 - g} = \frac{1}{2\alpha k}\left[\frac{1}{z - \alpha} - \frac{1}{z + \alpha}\right], \tag{2.13}$$

the integrand can be easily integrated to find

$$t + C = \frac{1}{2\alpha k} \ln\left|\frac{v - \alpha}{v + \alpha}\right|. \tag{2.14}$$

Solving for v, we have

$$v(t) = \frac{1 - Ae^{2\alpha kt}}{1 + Ae^{2\alpha kt}}\alpha, \tag{2.15}$$

where $A \equiv e^C$. A can be determined using the initial velocity by inserting $t = 0$,

$$v(0) = \frac{1 - A}{1 + A}\alpha.$$

Then,

$$A = \frac{\alpha - v_0}{\alpha + v_0}.$$

There are other forms for the solution in terms of a tanh function, which the reader can determine as an exercise. One important conclusion is that for large times, the ratio in the solution approaches -1. Thus, $v \to -\alpha = -\sqrt{\frac{g}{k}}$ as $t \to \infty$. This means that the falling object will reach a constant terminal velocity.

Equation (2.10) can be modeled in Simulink. The model is shown in Figure 2.9. The solution for $k = 0.00159\text{m}^{-1}$, which is found for the above sample computation, is shown in Figure 2.10. We see that terminal velocity is obtained and matches the predicted value, $-\sqrt{\frac{g}{k}} = -78$ m/s.

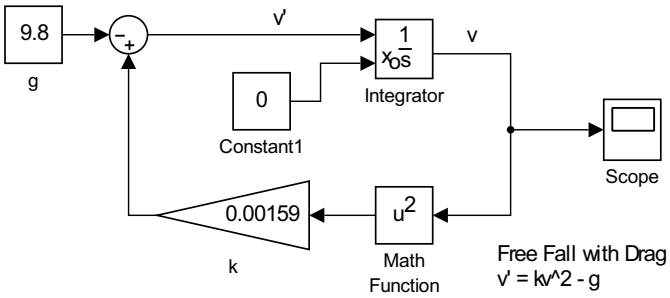


Figure 2.9: Model for free fall with drag as described by $\dot{v} = kv^2 - g$.

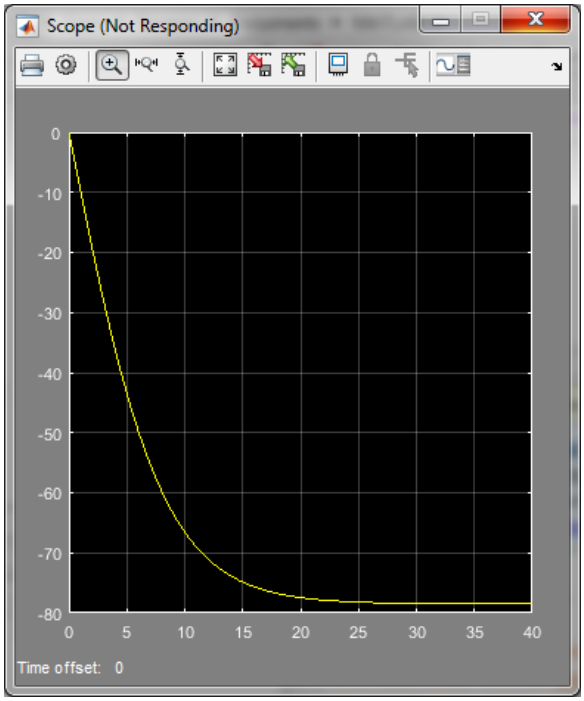


Figure 2.10: Solution for free fall with drag with $k = 0.00159$ starting from rest.

2.4 Pursuit Curves

Another application that is interesting is to find the path that a body traces out as it moves towards a fixed point or another moving body. Such curves are know as pursuit curves. These could model aircraft or submarines following targets, or predators following prey. For example, a hawk follows a sparrow, a large fish chases a small fish, or a fox chases a rabbit.

Example 2.3. A dog at point (x, y) sees a cat traveling at speed v along a straight line. The dog runs towards the cat at constant speed w but always in a direction along line of sight between their positions. If the dog starts out at the point $(0, 0)$ at $t = 0$, when the cat is at $(a, 0)$, then what is the path the dog needs to follow? Will the dog catch the cat?

We show the path in Figure 2.12. Let the cat's path be along the

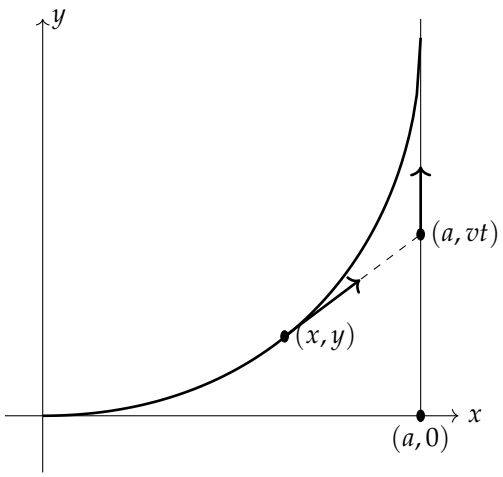


Figure 2.11: A dog at point (x, y) sees a cat at point (a, vt) and always follows the straight line between these points.

line $x = a$. Therefore, the cat is at position (a, vt) at time t. The goal is to find the dog's path, $(x(t), y(t))$, or $y = y(x)$.

First we consider the equation of the line of sight between the points (x, y) and (a, vt). Considering that the slope of this line is the same as the slope of the tangent to the path, $y = y(x)$, we have

$$y' = \frac{vt - y}{a - x}.$$

The dog is moving at a constant speed, w and the distance the dog to travels s given by $L = wt$, where t is the running time from the origin. The distance the dog travels is also given by the arclength of the path between $(0, 0)$ and (x, y) :

$$L = \int_0^x \sqrt{1 + [y'(x)]^2}\, dx.$$

Eliminating the time using $y' = \frac{vt-y}{a-x}$, we have

$$\int_0^x \sqrt{1 + [y'(x)]^2}\, dx = \frac{w}{v}(y + (a - x)y').$$

Furthermore, we can differentiate this result with respect to x to get rid of the integral,

$$\sqrt{1 + [y'(x)]^2} = \frac{w}{v}(a - x)y''. \tag{2.16}$$

This is the differential equation governing the dog's pursuit. A Simulink model of this problem is shown in Figure 2.12.

The full solution for the path is given by

$$y(x) = \frac{a}{2}\left[\frac{\left(\frac{x}{a}\right)^{1+\frac{v}{w}}}{1 + \frac{v}{w}} - \frac{\left(\frac{x}{a}\right)^{1-\frac{v}{w}}}{1 - \frac{v}{w}}\right] + \frac{avw}{w^2 - v^2}.$$

Can the dog catch the cat? This would happen if there is a time when $y(0) = vt$. Inserting $x = 0$ into the solution, we have $y(0) = \frac{avw}{w^2-v^2} = vt$. This is possible if $w > v$.

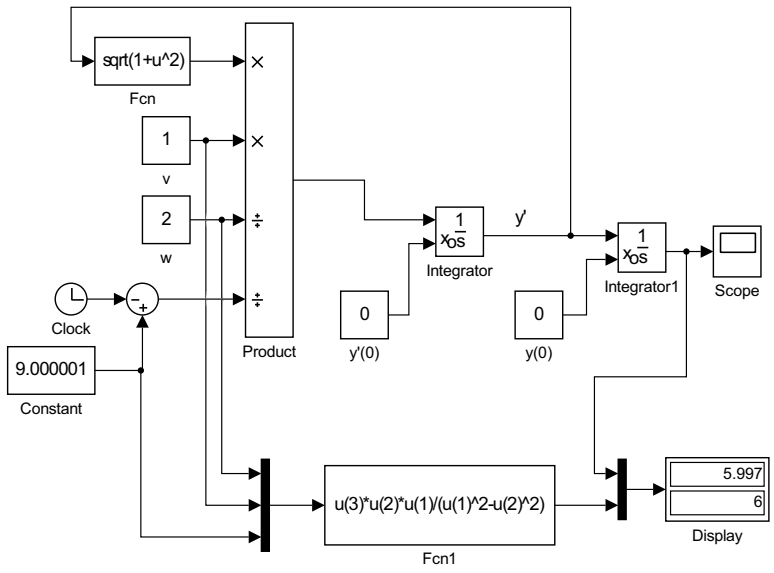


Figure 2.12: Model for the pursuit curve, $(a-x)y'' = \frac{v}{w}\sqrt{1+[y'(x)]^2}$, $y(0) = 0$, $y'(0) = 0$, for $w = 2$ and $v = 1$.

Analytic Solution

For the interested reader, we complete the solution of the problem by noting that Equation (2.16) can be rewritten as a first order separable equation in the slope function $z(x) = y'(x)$. Namely,

$$\frac{w}{v}(a-x)z' = \frac{v}{x}\sqrt{1+z^2}.$$

Separating variables, we find

$$\frac{w}{v}\int \frac{dz}{\sqrt{1+z^2}} = \ln(z+\sqrt{1+z^2})\int \frac{dx}{a-x}.$$

The integrals can be computed using standard methods from calculus.

We can easily integrate the right hand side,

$$\int \frac{dx}{a-x} = -\ln|a-x| + c_1.$$

The left hand side takes a little extra work,[2] or looking the integral to find

$$\int \frac{dz}{\sqrt{1+z^2}} = \ln(z+\sqrt{1+z^2}) + c_2.$$

Putting these results together, we have for $x > 0$,

$$\ln(z+\sqrt{1+z^2}) = \frac{v}{w}\ln x + C.$$

Using the initial condition $z = y' = 0$ and $x = a$ at $t = 0$,

$$0 = \frac{v}{w}\ln a + C,$$

or $C = -\frac{v}{w}\ln a$.

[2] One can use trigonometric substitution. Let $z = \tan\theta$ and $dz = \sec^2\theta\, d\theta$. Then, the method proceeds as follows:

$$\begin{aligned}\int \frac{dz}{\sqrt{1+z^2}} &= \int \frac{\sec^2\theta}{\sqrt{1+\tan^2\theta}}\,d\theta \\ &= \int \sec\theta\, d\theta \\ &= \ln(\tan\theta+\sec\theta)+c_2 \\ &= \ln(z+\sqrt{1+z^2})+c_2.\end{aligned}$$

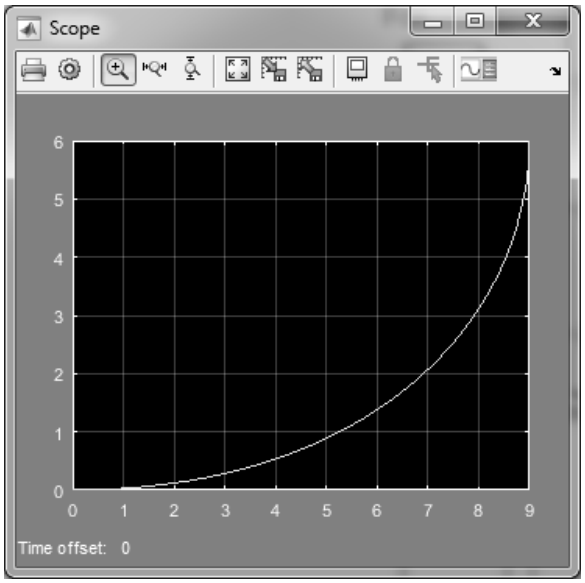


Figure 2.13: Solution for the pursuit curve.

Using this value for c, we find

$$\begin{aligned} \ln(z+\sqrt{1+z^2}) &= \frac{v}{w}\ln x - \frac{v}{w}\ln a \\ &= \ln\left(\frac{x}{a}\right)^{\frac{v}{w}} \\ z+\sqrt{1+z^2} &= \left(\frac{x}{a}\right)^{\frac{v}{w}}. \end{aligned} \tag{2.17}$$

We can solve for $z = y'$, to find

$$y' = \frac{1}{2}\left[\left(\frac{x}{a}\right)^{\frac{v}{w}} - \left(\frac{x}{a}\right)^{-\frac{v}{w}}\right]$$

Integrating,

$$y(x) = \frac{a}{2}\left[\frac{\left(\frac{x}{a}\right)^{1+\frac{v}{w}}}{1+\frac{v}{w}} - \frac{\left(\frac{x}{a}\right)^{1-\frac{v}{w}}}{1-\frac{v}{w}}\right] + k.$$

Since $y(a) = 0$, we can solve for the integration constant, k,

$$k = \frac{a}{2}\left[\frac{1}{1-\frac{v}{w}} - \frac{1}{1+\frac{v}{w}}\right] = \frac{avw}{w^2-v^2}.$$

2.5 The Logistic Equation

IN THIS SECTION WE WILL EXPLORE a nonlinear population model. Typically, we want to model the growth of a given population, $y(t)$, and the differential equation governing the growth behavior of this population is developed in a manner similar to that done in the section on growth and decay. Recall the simple population model from Section 2.1,

$$\frac{dy}{dt} = by - my, \tag{2.18}$$

where we had defined the birth rate as b and the mortality rate as m. If these rates are constant, then we can define $k = b - m$ and obtain the familiar exponential model of population growth.

When more realistic populations get large enough, there is competition for resources, such as space and food, which can lead to a higher mortality rate. Thus, the mortality rate may be a function of the population size, $m = m(y)$. The simplest model would be a linear dependence, $m = \tilde{m} + cy$. Then, the previous exponential model would take the form

$$\frac{dy}{dt} = ky - cy^2, \tag{2.19}$$

where $k = b - \tilde{m}$. This is known as the *logistic model* of population growth. Typically, c is small and the added nonlinear term does not kick in until the population gets large enough.

The logistic model was first published in 1838 by Pierre François Verhulst (1804-1849) in the form

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right),$$

where N is the population at time t, r is the growth rate, and K is what is called the carrying capacity. Note that in this model $r = k = Kc$.

Example 2.4. Show that Equation (2.19) can be written in the form

$$z' = kz(1 - z)$$

which has only one parameter.

We carry this out by rescaling the population, $y(t) = \alpha z(t)$, where α is to be determined. Inserting this transformation, we have

$$\begin{aligned} y' &= ky - cy^2 \\ \alpha z' &= \alpha kz - c\alpha^2 z^2, \end{aligned}$$

or

$$z' = kz\left(1 - \alpha\frac{c}{k}z\right).$$

Thus, we obtain the result, $z' = kz(1 - z)$, if we pick $\alpha = \frac{k}{c}$.

The point of this derivation is to show that there is only one free parameter, k, and that many combinations of c and k in the original problem lead to essentially the same solution up to rescaling.

We can model the logistic equation, $y' = ry(1 - y)$, with $r = 1$ and $y(0) = 0.1$ in Simulink. The model is shown in Figure 2.14. Running the model gives the solution in Figure 2.15. It shows the typical sigmoidal curve bounded by the solutions $y = 0$ and $y = 1$.

2.6 The Logistic Equation with Delay

SOMETIMES THE RATE OF CHANGE does not immediately take place when the system changes. This can be modeled using differential-delay equations. For example, when the resources are being depleted, the effects might be delayed. So, a possible model would be the logistic equation with delay,

$$y' = ry(t)(1 - y(t - \tau)),$$

where τ is a fixed delay time.

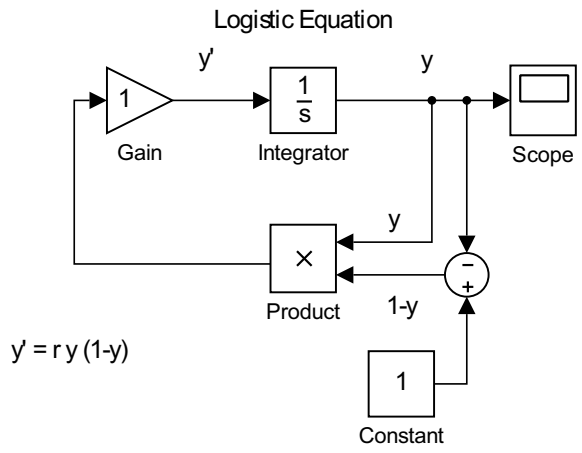


Figure 2.14: Simulink model for the logistic equation, $y' = ry(1-y)$.

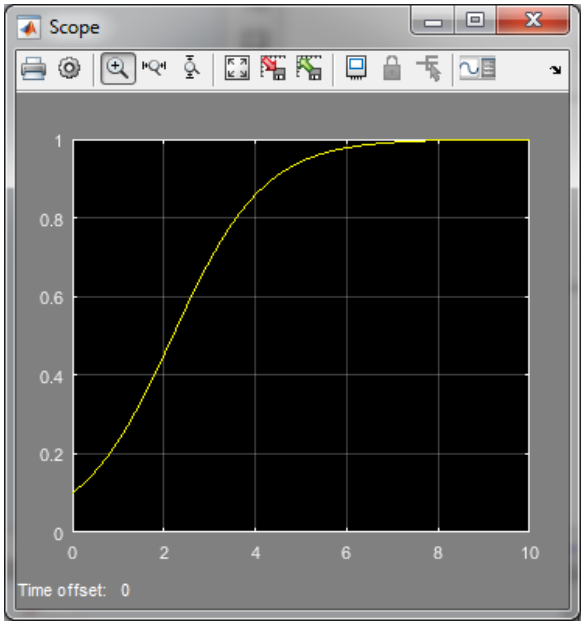


Figure 2.15: Solution of the logistic equation, $y' = ry(1-y)$, with $r = 1$ and $y(0) = 0.1$.

The problem with trying to solve this model at time t is that we need to know something about the solution for earlier times, $y(t-\tau)$. One way to tackle the problem is to specify the solution for times $[0, \tau]$ and then to solve the equation with delay using this starting value. So, if $y = 2$ initially, we could let $y = 2$ for $[0, \tau]$.

The Simulink model is shown in Figure 2.16. A Switch block is used to specify the starting values for times up to $\tau = 1$. Then, the differential equation solver takes over with a Delay block used to enter the delay term. This model produces the solution in Figure 2.17.

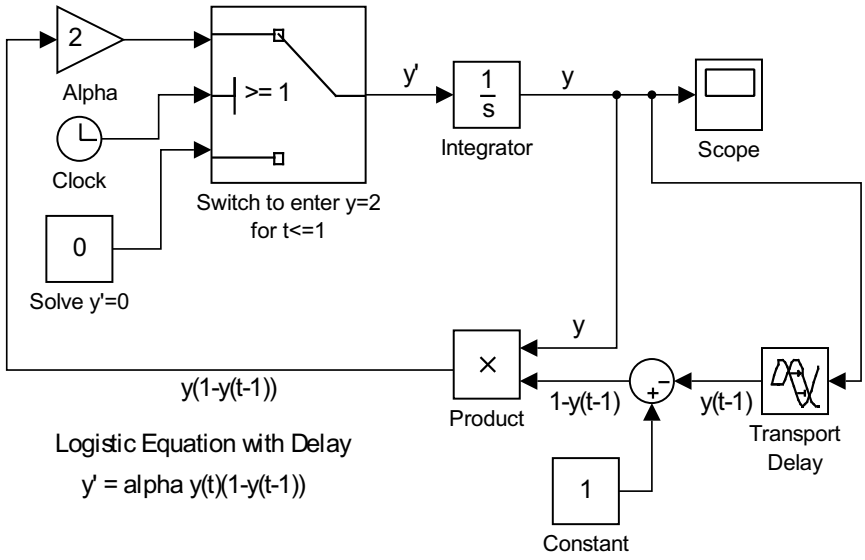


Figure 2.16: Model for the logistic equation with delay, $y' = ry(t)(1 - y(t-\tau))$.

2.7 Exercises

1. Model the following first order differential equations in Simulink and find the solutions for different initial conditions.
   a. $\dfrac{dy}{dx} = \dfrac{e^x}{2y}$.
   b. $\dfrac{dy}{dt} = y^2(1+t^2)$.
   c. $\dfrac{dy}{dx} = \dfrac{\sqrt{1-y^2}}{x}$.
   d. $xy' = y(1-2y)$.
   e. $y' - (\sin x)y = \sin x$.
   f. $xy' - 2y = x^2$.
   g. $\dfrac{ds}{dt} + 2s = st^2$.
   h. $x' - 2x = te^{2t}$.
   i. $\dfrac{dy}{dx} + y = \sin x,$.
   j. $\dfrac{dy}{dx} - \dfrac{3}{x}y = x^3$.
2. Consider the case of free fall with a damping force proportional to the velocity, $f_D = \pm kv$ with $k = 0.1$ kg/s.
   a. Using the correct sign, consider a 50 kg mass falling from rest at a height of 100m. Find the velocity as a function of time. Does the mass reach terminal velocity?
   b. Let the mass be thrown upward from the ground with an initial speed of 50 m/s. Find the velocity as a function of time as it travels upward and then falls to the ground. How high does the mass get? What is its speed when it returns to the ground?

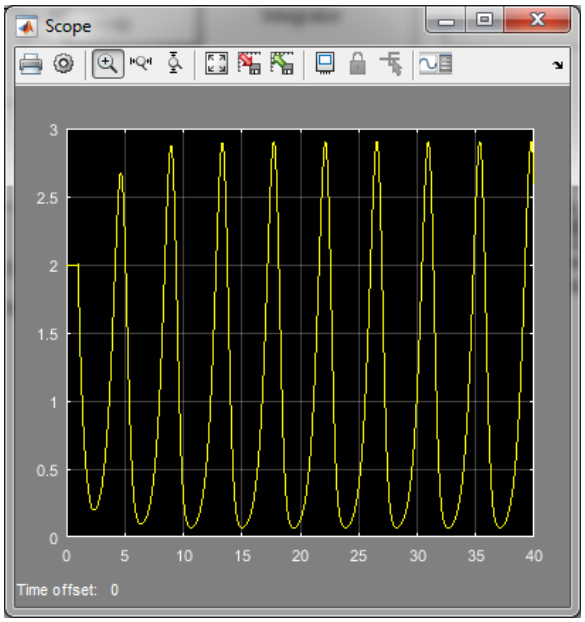


Figure 2.17: Solution of the logistic equation with delay, $y' = ry(t)(1 - y(t-1))$ for $y = 2$, $t \in [0,1]$.

3. A paratrooper, 322 lbs including munitions, jumps from 10,000 ft. Model this free fall with air resistance $f(v) = 15v^2$ in Simulink. First, write down the free fall equation. Use the model to solve for $v(t)$. Is there a terminal velocity? Find the time to land and the impact velocity.

4. Model the following problem in Simulink: The temperature inside your house is 70^oF and it is 30^oF outside. At 1:00 A.M. the furnace breaks down. At 3:00 A.M. the temperature in the house has dropped to 50^oF. Assuming the outside temperature is constant and that Newton's Law of Cooling applies, determine when the temperature inside your house reaches 40^oF.

5. Model the following problem in Simulink: A body is discovered during a murder investigation at 8:00 P.M. and the temperature of the body is 70^oF. Two hours later the body temperature has dropped to 60^oF in a room that is at 50^oF. Assuming that Newton's Law of Cooling applies and the body temperature of the person was 98.6^oF at the time of death, determine when the murder occurred.

3

Second Order Differential Equations

We now turn to second order differential equations. Such equations involve the second derivative, $y''(x)$. Let's assume that we can write the equation as

$$y''(x) = F(x, y(x), y'(x)).$$

We would like to solve this equation using Simulink. This is accomplished using two integrators in order to output $y'(x)$ and $y(x)$.

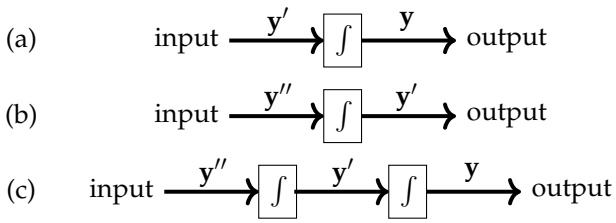


Figure 3.1: Basic schemes for using Integrator blocks for solving second order differential equations.

As shown in Figure 3.1(b), sending $y''(x)$ into the **Integrator** block, we get out $y'(x)$. This is similar to using $y'(x)$ to get $y(x)$ in Figure 3.1(a). As shown in Figure 3.1(c), combining two **Integrator** blocks, we can input $y''(x) = F(x, y, y')$ and get out y and y'. Feeding this output into $F(x, y, y')$, we then obtain a model for solving the second order differential equation. The general schematic for solving an initial value problem of the form $y'' = F(x, y, y')$, $y(0) = y_0$, $y'(0) = v_0$, is shown in Figure 3.2.

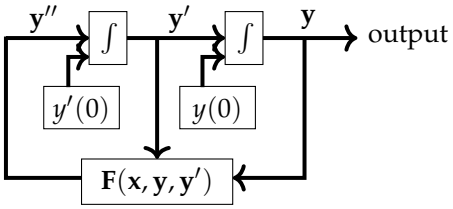


Figure 3.2: This is a general schematic for solving an initial value problem of the form $y'' = F(x, y, y')$, $y(0) = y_0$, $y'(0) = v_0$.

In this chapter we will demonstrate the modeling of second order constant coefficient differential equations and show some simple applications.

3.1 Constant Coefficient Equations

We can solve second order constant coefficient differential equations using a pair of integrators. An example is displayed in Figure 3.3. Here we solve the constant coefficient differential equation

$$ay'' + by' + cy = 0$$

by first rewriting the equation as

$$y'' = F(y, y') = -\frac{b}{a}y' - \frac{c}{a}y.$$

Example 3.1. Model the initial value problem

$$y'' + 5y' + 6y = 0, \quad y(0) = 0, y'(0) = 1,$$

in Simulink.

The simulation in Figure 3.3 solves the equation

$$y'' + 5y' + 6y = 0$$

with appropriate initial conditions. There are two integrators. One integrates the first input, y'', and the other integrates the output of the first integrator, y', giving an output of y. Each **Integrator** block needs an initial condition. The first takes $y'(0) = 1$ and the second needs $y(0) = 0$.

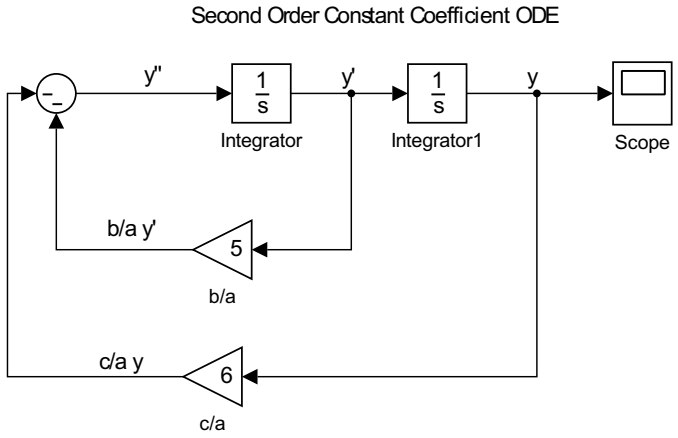


Figure 3.3: Model for the second order constant coefficient ODE $y'' + 5y' + 6y = 0$.

The outputs, y and y' are multiplied by the appropriate constants using a **Gain** block. They are then combined to form the input, $F(y, y') = -5y' - 6y$, to the system of integrators. Running the simulation for 5 units of time, the Scope gives the solution shown in Figure 3.4.

Analytical Solution

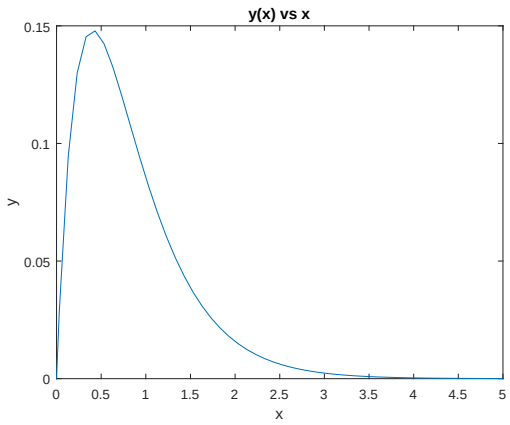


Figure 3.4: Solution plot for the initial value problem $y'' + 5y' + 6y = 0$, $y(0) = 0$, $y'(0) = 1$ using Simulink.

Recall the solution of this problem is found by first seeking the two linearly independent solutions. Assuming solutions of the form $y(x) = e^{rx}$, the characteristic equation is

$$r^2 + 5r + 6 = 0.$$

The roots of the equation are $r = -2, -3$. Therefore, the two linearly independent solutions are $y_1(x) = e^{-2x}$ and $y_2(x) = e^{-3x}$. The general solution is

$$y(x) = c_1 e^{-2x} + c_2 e^{-3x}.$$

The initial conditions hold if

$$0 = c_1 + c_2, \quad 1 = -2c_1 - 3c_2.$$

So, $c_1 = 1$ and $c_2 = -1$. The solution to the initial value problem is

$$y(x) = e^{-2x} - e^{-3x}.$$

The plot of this solution is shown in Figure 3.5. It is seen to agree with the solution shown in Figure 3.4.

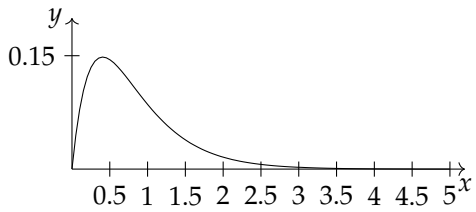


Figure 3.5: Plot of the exact solution of the initial value problem $y'' + 5y' + 6y = 0$, $y(0) = 0$, $y'(0) = 1$.

Harmonic Oscillation

A typical application of second order, constant coefficient differential equations is the simple harmonic oscillator as shown in Figure 3.6. Consider a mass, m, attached to a spring with spring constant, k. According to

In the following we will suppress units. In SI units the mass is in kilograms (kg), displacement x is in meters (m), and force is in Newtons (N). Then, k has units of N/m. One could also use CGS units of g, cm, dynes, and dynes/cm, respectively. Time units will generally be in seconds, leaving frequencies in s^{-1}, or Hertz (Hz).

Hooke's law, a stretched spring will react with a force $F = -kx$, where x is the displacement of the spring from its unstretched equilibrium. The mass experiences a net for and will accelerate according to Newton's Second Law of Motion, $F = ma$. Setting these forces equal and noting that $a = \ddot{x}$, we have

$$m\ddot{x} + kx = 0.$$

Here we assume that $x = x(t)$ and let the derivatives be time derivatives. The characteristic equation is given by $mr^2 + k = 0$, or

$$r = \pm i\sqrt{\frac{k}{m}} \equiv \pm i\omega_0.$$

Then, the general solution is given as

$$x(t) = A\cos\omega_0 t + B\sin\omega_0 t.$$

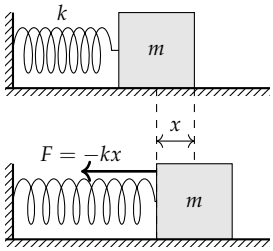


Figure 3.6: A simple harmonic oscillator consists of a mass, m, attached to a spring with spring constant, k.

We will model the equation for simple harmonic motion and it variations in the next examples. Namely, we will look at Simulink examples of simple harmonic motion, damped harmonic motion, and forced harmonic motion.

Example 3.2. Simple Harmonic Motion

A Simulink model for simple harmonic motion is shown in Figure 3.7. We write the differential equation in the form

$$\ddot{x} = -\frac{1}{m}(kx).$$

For this example we set $k = 5$ and $m = 2$. We also specify the initial conditions $x(0) = 1$ and $\dot{x}(0) = 0$ in the two integrators.

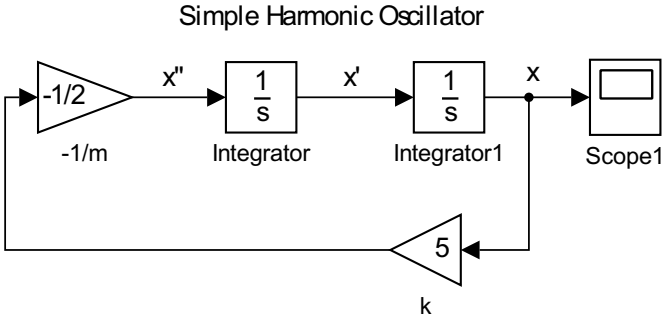


Figure 3.7: A model for simple harmonic motion, $m\ddot{x} + kx = 0$.

The output on the scope is shown in Figure 3.8 for $t \in [0, 10]$. Solving the initial value problem we find that $x(t) = \cos\omega_0 t$, where

$$\omega_0 = \sqrt{\frac{k}{m}} = \sqrt{\frac{5}{2}}.$$

Thus, the period is

$$T = \frac{2\pi}{\omega_0} \approx 3.9738\text{s}.$$

From Figure 3.8 we might have estimated the period as 4 s.

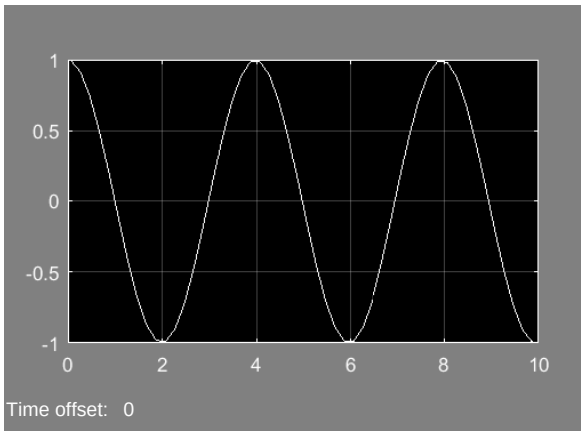


Figure 3.8: Output for the solution of the simple harmonic oscillator model.

Example 3.3. Damped Simple Harmonic Motion

A simple modification of the harmonic oscillator is obtained by adding a damping term proportional to the velocity, $\dot{x}$. This results in the differential equation

$$m\ddot{x} + b\dot{x} + kx = 0,$$

where $b > 0$ is the damping constant.

We can verify the damping behavior in the solution by studying the characteristic equation,

$$mr^2 + br + k = 0,$$

where $x(t) = e^{rt}$ is a guess for form of the the linearly independent solutions. The solutions of the characteristic equation are found using the quadratic formula,

$$r = \frac{-b \pm \sqrt{b^2 - 4km}}{2m}.$$

If $b^2 - 4km < 0$, then the roots of the characteristic equation are complex conjugate roots and the solution takes the form

$$x(t) = e^{-bt/2m} \left[A \cos \omega_0 t + B \sin \omega_0 t \right],$$

where

$$\omega_0 = \frac{\sqrt{4km - b^2}}{2m}.$$

In this case one has oscillatroy solutions with an exponentially decaying amplitude.

A Simulink model for the damped harmonic oscillator can be created using the differential equation in the form $\ddot{x} = -\frac{1}{m}(b\dot{x} + kx)$. This leads to a modification of the model in Figure 3.7. We simply add a term $b\dot{x}$. The model is shown in Figure 3.9.

We consider a specific example using $k = 5$, $m = 2$, and $b = 0.1$. The initial conditions $x(0) = 1$ and $\dot{x}(0) = 0$ are used in the two

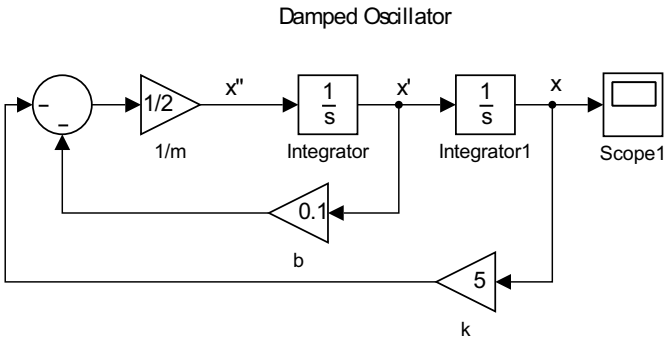


Figure 3.9: A model for damped simple harmonic motion, $m\ddot{x} + b\dot{x} + kx = 0$.

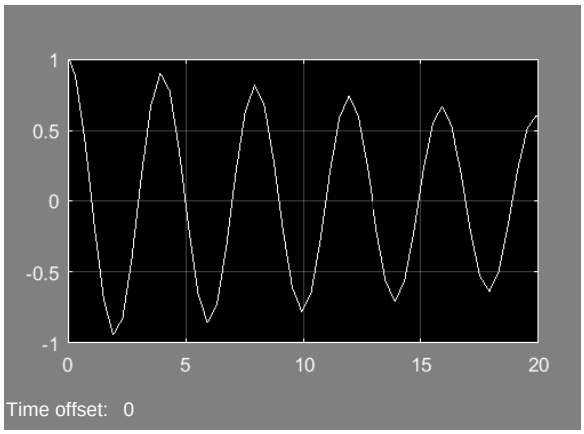


Figure 3.10: Output for the solution of the damped harmonic oscillator model.

integrators. Running the model for $t \in [0, 20]$, the solution seen in the **Scope** block is shown in Figure 3.10. We note that $\omega_0 = 1.5809$ Hz, or the period of oscillation is $T = 3.9743$s. This is consistent with the Simulink solution.

Applying the initial conditions, $x(0) = 1$ and $\dot{x}(0) = 0$, to the general solution, we find that $A = 1$ and

$$\begin{aligned} 0 &= -\frac{b}{2m}A + \omega_0 B, \text{or} \\ B &= \frac{b}{2m\omega_0}. \end{aligned} \tag{3.1}$$

Therefore, the particular solution of the initial value problem can be written as

$$x(t) = e^{-bt/2m}\left[\cos\omega_0 t + \frac{b}{2m\omega_0}\sin\omega_0 t\right].$$

For the parameter values in the problem a plot of this oslution is shown in Figure 3.11 and agrees with Figure 3.10 for this example.

The plot in Figure 3.11 was obtained using MATLAB's **ezplot** function and it symbolic capability. The code is given below for this example.

```
syms t
```

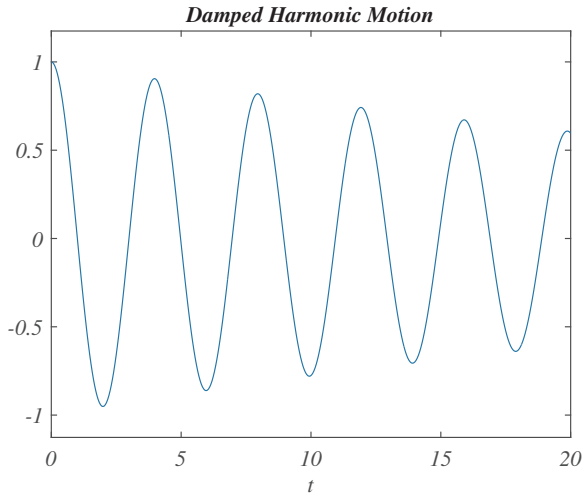


Figure 3.11: The analytic solution for the damped harmonic oscillator example.

```
b=.1; m=2; k=5;
omega=sqrt(4*k*m-b^2)/2/m;
alpha=b/2/m;
A=1;
B=b/(2*m*omega);
x=exp(-alpha*t)*(A*cos(omega*t)+B*sin(omega*t));

ezplot(x,[0,20]);
title('Damped Harmonic Motion')
```

Another modification of the problem is to introduce forcing. In general, the corresponding nonhomogeneous equation is $m\ddot{x} + b\dot{x} + kx = f(t)$. One need only add $f(t)$ to the sum that is sent into the first **Integrator** block. This also requires the **Clock** block and some function blocks. We show this in the next examples.

Example 3.4. Forced Simple Harmonic Motion

We consider a simple sinusoidal forcing and no damping given by

$$m\ddot{x} + kx = F_0 \sin \omega t.$$

The Simulink model in Figure 3.9 is modified to produce the model in Figure 3.12 by adding a **Sine Wave Function** and a **Clock**. We left the damping **Gain** block but set the multiplier to zero. We also note that the **Sum** block shape was changed to rectangular to accommodate more inputs and to direct a consistent flow of the processes.

Using the constants $m = 2$, $k = 10$, we set $F_0 = 1$ and $\omega = 2$ in the **Sine Wave Function**. This results in the output shown in Figure 3.13. Note that the solution is a modulated oscillation. This is understood from looking at the analytic form of the solution.

Recall that we can obtain the analytic solution to this problem using the Method of Undetermined Coefficients. The general solution

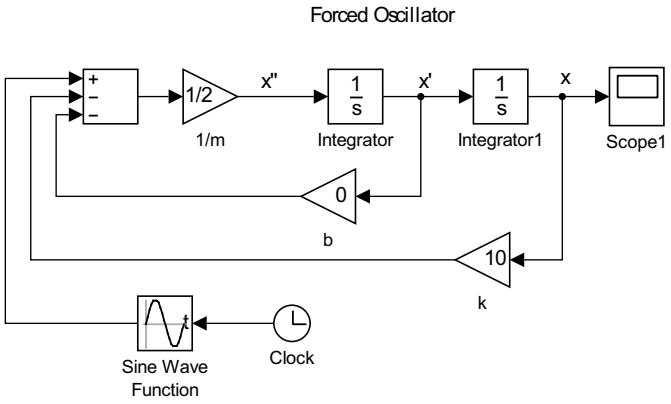


Figure 3.12: A model for forced simple harmonic motion, $m\ddot{x} + kx = \sin \omega t$.

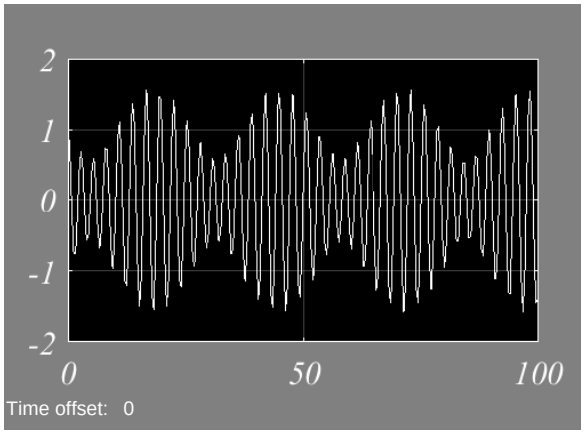


Figure 3.13: Output for the solution of the forced simple harmonic oscillator model.

is a solution of the homogeneous problem plus a particular solution, or guess, to the nonhomogeneous problem. Thus, we have

$$x(t) = A \cos \omega_0 t + B \sin \omega_0 t + x_p(t).$$

We make an educated guess for a function $x_p(t)$ satisfying

$$m\ddot{x}_p + kx_p = F_0 \sin \omega t.$$

Knowing that two derivatives of a sine function returns a constant times the sine function, we assume that $x_p(t) = a \sin \omega t$, providing that this is not a solution of the homogeneous problem. Namely, $\omega \neq \omega_0$.

Inserting this guess into the differential equation, we have

$$-m\omega^2 a \sin \omega t + ka \sin \omega t = F_0 \sin \omega t.$$

Since this is true for all t, $-m\omega^2 a + ka = F_0$. Noting that $k = m\omega_0^2$, we can solve for a,

$$a = \frac{F_0}{m(\omega_0^2 - \omega^2)}.$$

Then, the general solution is given by

$$x(t) = A \cos \omega_0 t + B \sin \omega_0 t + \frac{F_0}{m(\omega_0^2 - \omega^2)} \sin \omega t, \quad \omega \neq \omega_0.$$

The initial conditions, $x(0) = 1$ and $\dot{x}(0) = 0$, were again used in the two integrators. The first condition gives $A = 1$. The second condition can be written as

$$0 = \omega_0 B + \frac{F_0 \omega}{m(\omega_0^2 - \omega^2)}.$$

Solving for B, we obtain

$$B = -\frac{F_0 \omega}{m\omega_0(\omega_0^2 - \omega^2)}.$$

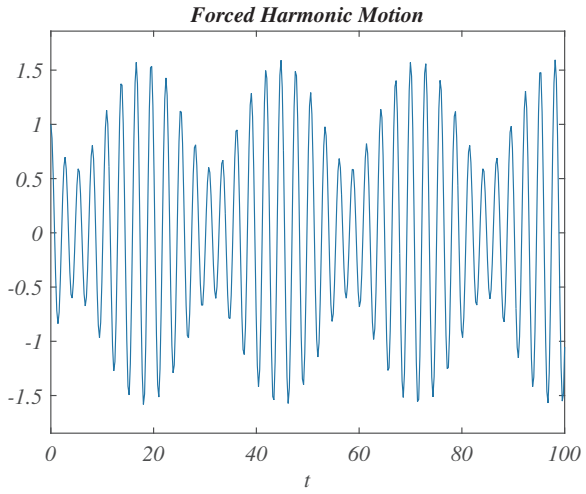


Figure 3.14: The analytic solution for the forced harmonic oscillator example.

Inserting the constants[1] in this problem, the exact solution to the initial value problem is found as

$$x(t) = \frac{1}{2}\sin 2t + \cos\sqrt{5}t - \frac{1}{\sqrt{5}}\sin\sqrt{5}t.$$

[1] Recall that $m = 2$, $k = 10$, $F_0 = 1$, $\omega = 2$. Therefore, $\omega_0 = \sqrt{k/m} = \sqrt{5}$.

The plot of this solution is in Figure 3.14. It agrees with that given by the Simulink model in Figure 3.13.

Example 3.5. Derive a modulation form of the solution from Example 3.4.

Details showing how to derive a modulated form, $x(t) = C(\psi(t))\sin(\theta(t) + \delta)$.

The solution,

$$x(t) = \frac{1}{2}\sin 2t + \cos\sqrt{5}t - \frac{1}{\sqrt{5}}\sin\sqrt{5}t, \tag{3.2}$$

in Figure 3.14 looks like what one would get when adding sinusoidal functions with frequencies that are close. It is the principle used by piano tuners when using a tuning fork to tune a piano key. If the piano key note is slightly different from that of a tuning fork, then when both are sounded at the same time, one hears a beat pattern. This is heard as the low frequency of the envelope similar to that in Figure 3.14. In the last example we had two frequencies, $\omega = 2$ and $\omega_0 = \sqrt{5} \approx 2.2361$, which were close together.

We will combine the the trigonometric functions in Equation (3.2) and show the root of this modulation. We seek a solution in the form

$$x(t) = C(\psi(t)) \sin(\theta(t) + \delta),$$

where $C(\psi(t))$ is the modulation amplitude for a higher frequency sinusoidal function and δ is a phase shift. This is accomplished using trigonometric identities.

In the following we will need the result that

$$\begin{aligned} y &= \alpha \cos\theta + \beta \sin\theta \\ &= a \sin(\theta + \varphi). \end{aligned} \tag{3.3}$$

Expanding the second expression, we have

$$a \sin(\theta + \varphi) = a \sin\varphi \cos\theta + a \cos\varphi \sin\theta.$$

Equating coefficients of $\cos\theta$ and $\sin\theta$, we have

$$\alpha = a \sin\varphi, \quad \beta = a \cos\varphi.$$

Adding the squares of these equations,

$$a^2 = \alpha^2 + \beta^2,$$

and taking the ratio of the equations yield

$$\tan\varphi = \frac{\alpha}{\beta}.$$

We now use this result to combine the terms in $x(t)$ into a single sine function with a varying amplitude. We begin by combining the last two terms of Equation (3.2) as

$$\cos\sqrt{5}t - \frac{1}{\sqrt{5}} \sin\sqrt{5}t = a \sin(\sqrt{5}t + \varphi)$$

and set $\theta = \sqrt{5}t$, $\alpha = 1$ and $\beta = -\frac{1}{\sqrt{5}}$. Then, we have that

$$a^2 = 1 + \frac{1}{5} = \frac{6}{5}$$

and

$$\tan\varphi = -\sqrt{5}.$$

Since the angle is in the second quadrant of the $\beta\alpha$-plane, $\varphi = \pi - \tan^{-1}(\sqrt{5})$. This gives the solution in the new form

$$x(t) = \frac{1}{2} \sin 2t + \sqrt{\frac{6}{5}} \sin(\sqrt{5}t + \varphi). \tag{3.4}$$

We now combine the terms in Equation (3.4). Assume that the solution is the sum of the two sine functions

$$x(t) = A \sin(\theta + \psi) + B \sin(\theta - \psi), \tag{3.5}$$

where the variables A, B, θ and ψ are to be determined. It is easy to see that $A = \frac{1}{2}$, $B = a = \sqrt{\frac{6}{5}}$, and

$$\theta + \psi = 2t, \qquad \theta - \psi = \sqrt{5}t + \varphi.$$

Solving this system,

$$\theta = \frac{(2+\sqrt{5})t + \varphi}{2}, \qquad \psi = \frac{(2-\sqrt{5})t - \varphi}{2}.$$

Expanding the sine functions in Equation (3.5), we have

$$\begin{aligned} x(t) &= (A+B)\sin\theta\cos\psi + (A-B)\cos\theta\sin\psi \\ &= [(A-B)\sin\psi]\cos\theta + [(A+B)\cos\psi]\sin\theta \\ &= \alpha\cos\theta + \beta\sin\theta, \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} \alpha &= (A-B)\sin\psi \\ \beta &= (A+B)\cos\psi. \end{aligned}$$

We can combine the terms in $\alpha\cos\theta + \beta\sin\theta$ in the form

$$x(t) = \alpha\cos\theta + \beta\sin\theta = C(\psi(t))\sin(\theta(t) + \delta)$$

using the previous derivation, leading to

$$\begin{aligned} C^2 &= \alpha^2 + \beta^2 \\ &= (A-B)^2\sin^2\psi + (A+B)^2\cos^2\psi \\ &= A^2 + B^2 + 2AB\cos 2\psi \\ &= \frac{29}{20} + \sqrt{\frac{6}{5}}\cos 2\psi \\ \tan\delta &= \frac{\alpha}{\beta}. \\ &= \left(\frac{A-B}{A+B}\right)\tan\psi. \\ &= \left(\frac{\frac{1}{2} - \sqrt{\frac{6}{5}}}{\frac{1}{2} + \sqrt{\frac{6}{5}}}\right)\tan\psi \\ &= \left(\frac{\sqrt{5} - 2\sqrt{6}}{\sqrt{5} + 2\sqrt{6}}\right)\tan\psi. \end{aligned}$$

Thus, we have $x(t) = C\sin(\theta(t) + \delta)$ for C and δ defined by the above relations,

$$\theta = \frac{(2+\sqrt{5})t + \varphi}{2}, \qquad \psi = \frac{(2-\sqrt{5})t - \varphi}{2},$$

and $\tan\varphi = -\sqrt{5}$. This gives a modulated solution by an amplitude envelope with a slowly varying frequency and high freqency oscillations given by the function $\sin(\theta(t) + \delta)$, whose period is $T = \frac{2\pi}{\omega_\theta} = \frac{4\pi}{2+\sqrt{5}} = 2.9665$s as compared to $\frac{\pi}{\omega_\psi} = \frac{2\pi}{|2-\sqrt{5}|} = 26.6160$s for the envelope. This function is shown in Figure 3.15.

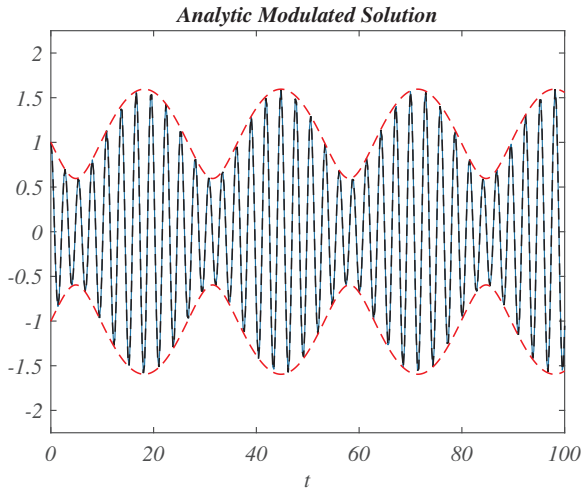


Figure 3.15: The solution, $x(t) = C\sin(\theta(t) + \delta)$, for the forced harmonic oscillator example.

Forced damped oscillator

Example 3.6. Model the forced, damped harmonic oscillator.

A simple application is the forced, damped harmonic oscillator. Recall that this is modeled using a second order, constant coefficient equation,

$$mx'' + cx' + kx = F(t)$$

for some driving force $F(t)$. Rewriting the equation, we have

$$x'' = \frac{1}{m}F(t) - \frac{c}{m}x' - \frac{k}{m}x.$$

This suggests a model like that shown in Figure 3.16. In this example the forcing term was taken as a step function.

$$F(t) = \begin{cases} 0, & t < 1, \\ 1, & t \geq 1. \end{cases}$$

The **Step** function block is set to start at $F = 0$ and increases to a constant value of $F = 1$ after $t = 1$. The other constants are given as $m = 1.0$ kg, $c = 0.5$ kg/s, and $k = 2.0$ N/m.

In Figure 3.17 is shown the solution plot for the forced, damped, harmonic oscillator model with initial values of $x(0) = 1$ and $x'(0) = 0$. In this model there is also an **XY Graph** block. The position and velocity data is fed into this block and the output is a plot of the solution in the phase plane. This is shown in Figure 3.18.

3.2 Projectile Motion

ANOTHER EXAMPLE IS THAT OF PROJECTILE MOTION. This is a system of equations or a single equation for a vector function. Let the position vector for the projectile be given by $\mathbf{r} = [x, y]$. Then, the projectile satisfies

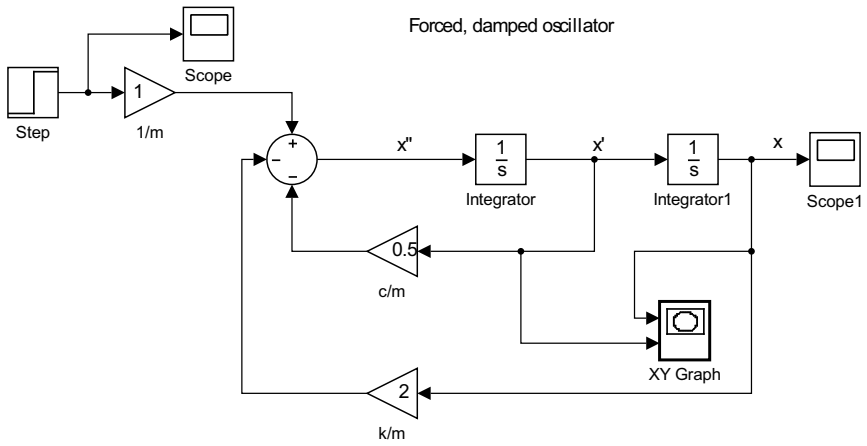


Figure 3.16: Forced, damped oscillator.

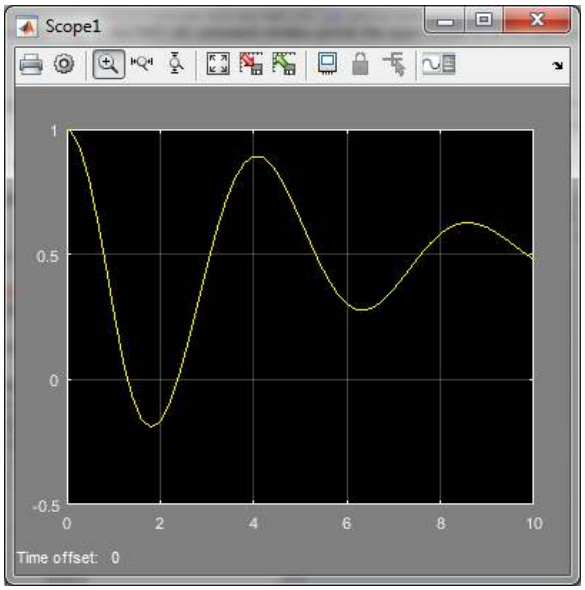


Figure 3.17: **Scope plot** of the solution of the forced, damped, harmonic oscillator model.

the second order equation $\mathbf{r}'' = -\mathbf{g}$. We solve this equation using two integrators and setting up the system with a two component vector. [For a simpler model solving for a two component vector, look at the Bouncing Ball Problem in the next section.]

To make things more interesting, we can add a drag force. Thus, we solve the system

$$\mathbf{r}'' = -\mathbf{g} - kv\mathbf{v}.$$

The magnitude of the drag is proportional to v^2. If the projectile is moving directly upward, the drag is negative, opposing the motion. The model will need functions to compute the speed, v, and will need two integrators with the appropriate initial position and velocity. The gravitational force will also be provided with a constant block. This model is shown in Figure 3.19

For a change, we set up the model in British units (foot-pound-second). The initial position is $[0, 4]$ ft and the initial velocity is $[80, 80]$ ft/s. The

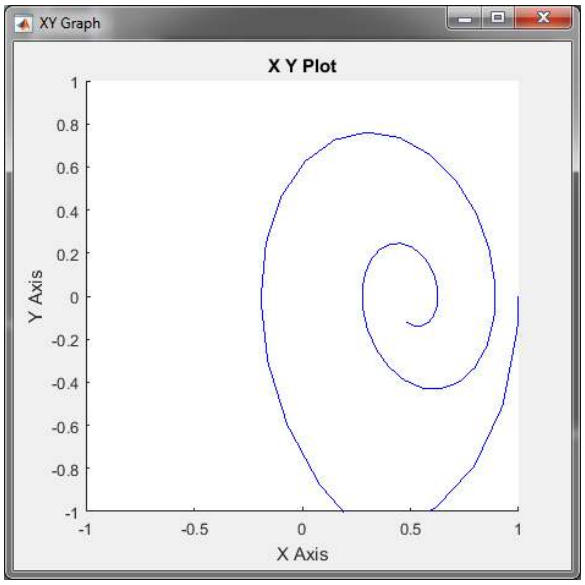


Figure 3.18: **XY Graph** output for the solution of the forced, damped, harmonic oscillator model.

gravitational constant is $-\mathbf{g} = [0, -32]$ ft/s^2. The value of the drag coefficient does not show in the figure. [The value shows when the Gain block is resized.] The position and speed vs time plots are shown in Figure 3.20. Note that changing the simulation time is one way to only display the time that the mass is above $y = 0$. Also, the plot of speed shows that the speed is always positive.

Also shown in this model is the use of the **XY Graph** block. It takes two inputs in order to plot the path y vs x. XY Graphs automatically plot when the simulation is run, as opposed to the **Scope** plots, which need to be double-clicked to display the plots. One also needs to double-click the **XY Graph** block to change the scale shown. For this model the output is shown in Figure 3.21. This plot is useful for determining the maximum height and range of the projectile.

.

3.3 The Bouncing Ball

As seen in the projectile motion model output in Figure 3.20, the projectile may not stop when it reaches the ground. One needs a way to determine when this has happened and reverse the direction of the motion. In this section we will look at a simple model in which a ball goes through free fall and bounces when it reaches the ground.

The ball satisfies the second order equation $x'' = -g$. Noting that the velocity is $v = x'$, this can be written as two first order equations,

$$\begin{aligned} x' &= v, \\ v' &= -g. \end{aligned} \tag{3.7}$$

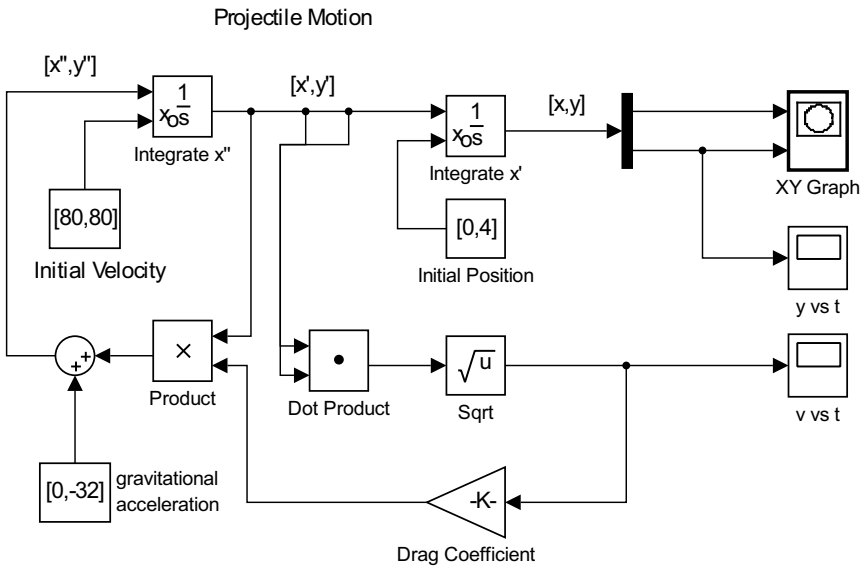


Figure 3.19: Projectile motion model.

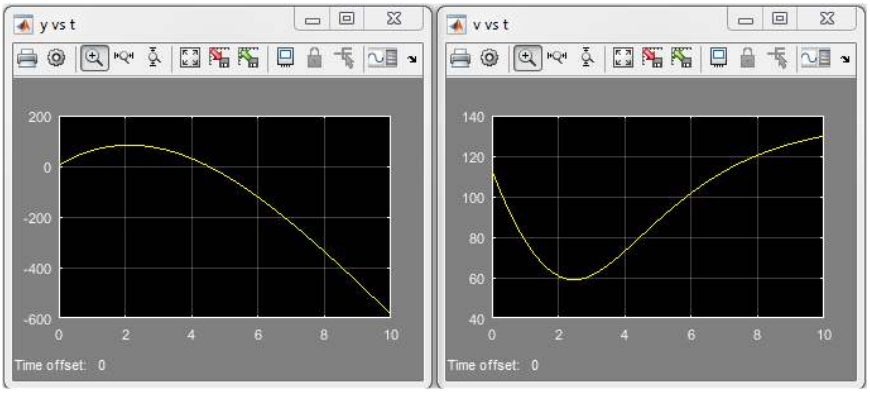


Figure 3.20: Output of the **Scope** Blocks for the projectile motion model for position and velocity vs time.

This system of equations can be then be put into matrix form,

$$\frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

This system can be used to produce the Simulink model in Figure 3.22, where we have introduced initial conditions $x(0) = 3$ and $v(0) = 0$ in the form of an initial vector, $[3;0]$.

In the model we use matrix multiplication to set up the right hand side of the equation. A 2×2 matrix is entered in the gain and the acceleration term is added separately. In order to plot the position vs time, we put a **Demux** block to separate out the components of the "state" vector and added a **Terminator** block to terminate the unused v-branch.

The output of the simulation, which was run for a time of 1 second, is shown in Figure 3.23. Note that the ball has fallen below ground level. We would like for the ball to bounce from the ground. In order to do so, we will test to see when $x \leq 0$ and $v \leq 0$. This is accomplished by adding some test conditions to the **Integrator** block.

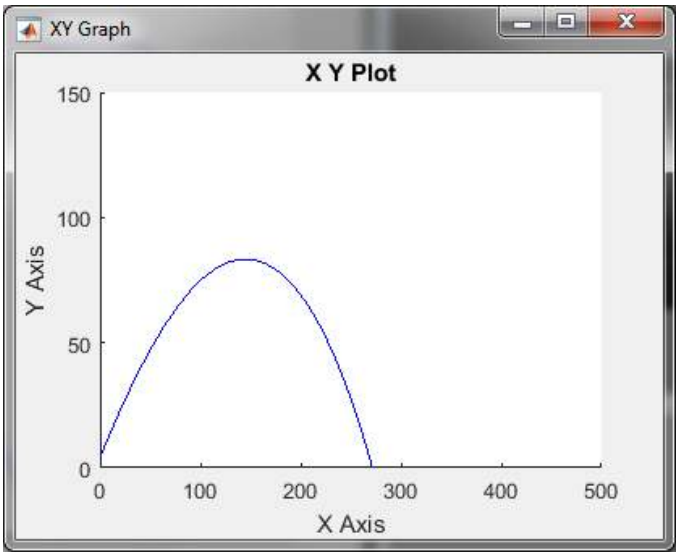


Figure 3.21: Output of the **XY Graph** block for the projectile motion model showing vertical position vs horizontal position.

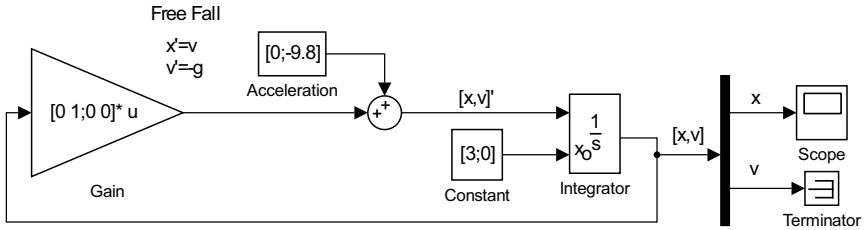


Figure 3.22: Free fall model.

Double-click the **Integrator** block and set the **External reset** to **rising**. This will add a third input as shown in Figure 3.24. Then, replace the initial condition **Constant** block with an **IC** block. This is found in the Signal Attributes group. It looks like the **IC** block in Figure 3.24.

Next, we enter the conditions determining when the block hits the ground and change the block velocity. The input to the condition consist of the Boolean condition, **(u[1]<=0)&&(u[2]<0)**, and the new position and velocity. Here u[1] and u[2] are the position and velocity components. We set the position as **u[1]** and the velocity as **-0.8*u[2]**. These expressions are entered using **Fcn** blocks from the User-Defined Function group. This model is shown in Figure 3.25 with the needed connections to the **Fcn** blocks and the **Integrator** block. This output is shown in Figure 3.26.

3.4 Nonlinear Pendulum Animation

PLOTTING AND ANIMATING SOLUTIONS FROM A MODEL can be done by sending the output of a model to MATLAB. In this section we will solve a nonlinear pendulum problem and show how one sends the output to create a simple animation of the pendulum motion.

A simple pendulum consists of a point mass m attached to a string of length L as shown in Figure 3.27. It is released from an angle θ_0. Newton's

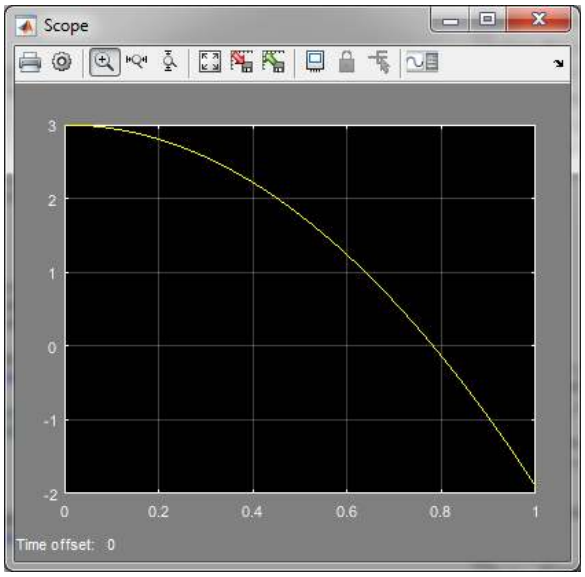


Figure 3.23: Height vs time Scope plot for the free fall model.

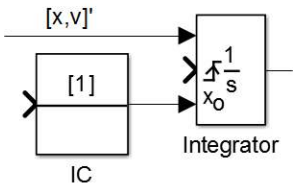


Figure 3.24: Modified **Integrator** and **IC** blocks.

Second Law of Motion tells us that the net force is the mass times the acceleration. So, we can write

$$m\ddot{x} = -mg\sin\theta.$$

Next, we need to relate x and θ. x is the distance traveled, which is the length of the arc traced out by the point mass. The arclength is related to the angle, provided the angle is measure in radians. Namely, $x = r\theta$ for $r = L$. Thus, we can write

$$mL\ddot{\theta} = -mg\sin\theta.$$

Canceling the masses, this then gives us the nonlinear pendulum equation

Nonlinear pendulum equation.

$$L\ddot{\theta} + g\sin\theta = 0. \tag{3.8}$$

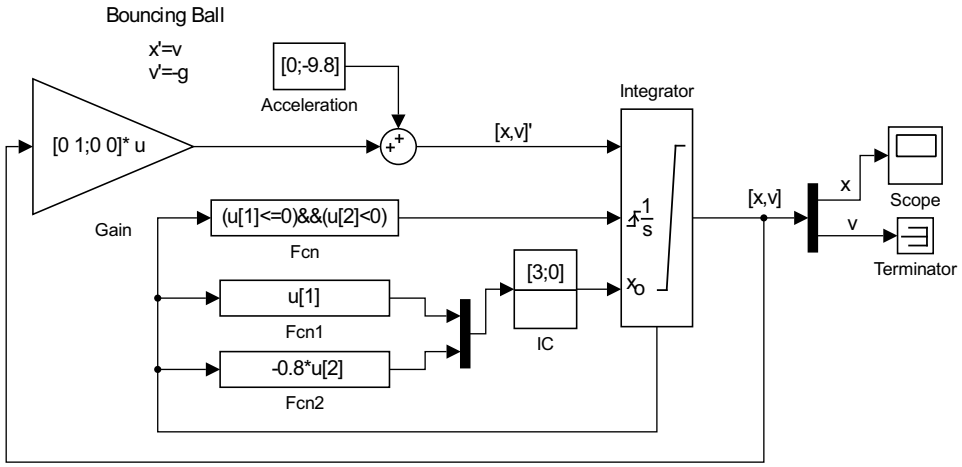


Figure 3.25: The bouncing ball model.

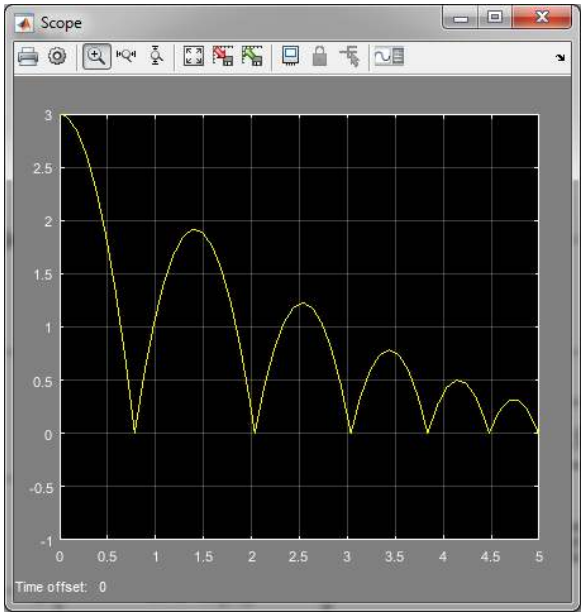


Figure 3.26: Output from the bouncing ball model showing plot of height vs time.

We can use Simulink to model this equation. Such a model is shown in Figure 3.28.It is set up to solve the model in the form

$$\ddot{\theta} = -\frac{g}{L}\sin\theta.$$

The constants are entered using **Constant** blocks and two **Integrator** blocks are used.

We enter the parameters in the system using variables instead of particular constants. These parameters are introduced in a MATLAB m-file. The constants are **L**, **g**, and initial conditions **thetao** and **vo** in the **Integrator** blocks. Save this model as **pend.mdl**.

Now, one creates an m-file, **pendulum.m** with the following code. Here we define the constants first. Then enter the initial conditions followed by the simulation.

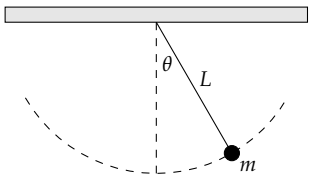


Figure 3.27: A simple pendulum consists of a point mass m attached to a string of length L. It is released from an angle θ_0.

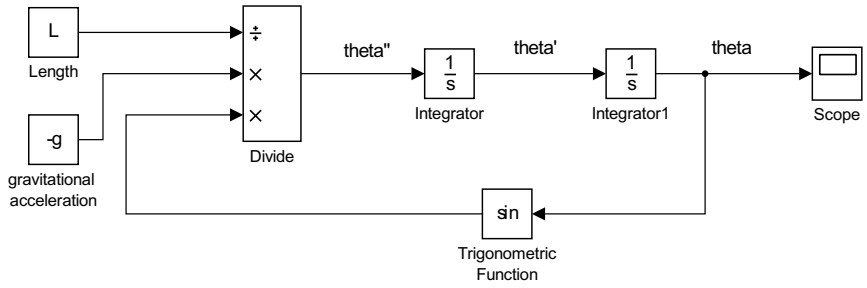


Figure 3.28: Nonlinear pendulum model.

```
m=1.0;
L=1.0;
g=9.8;

v0=0;
theta0=pi/6;
t0=0;
tf=15;

myopts = simset('MaxStep', 0.01);
sim('pend', [t0 tf],myopts)
```

Typing **pendulum** in the command window, assuming that this file and the model are save and run from the same folder, will produce a **Scope** plot for $t \in [0, 15]$. The function **simset** will make the plot smoother.

In order to plot the solution in MATLAB, the solution needs to be output to the MATLAB workspace. This is accomplished by adding a To Workspace block for the **theta** output variable and one for time, using a **Clock**. Double-clicking each block, one can change the output variable names to **theta** and **time**, respectively. The resulting model is shown in Figure 3.29

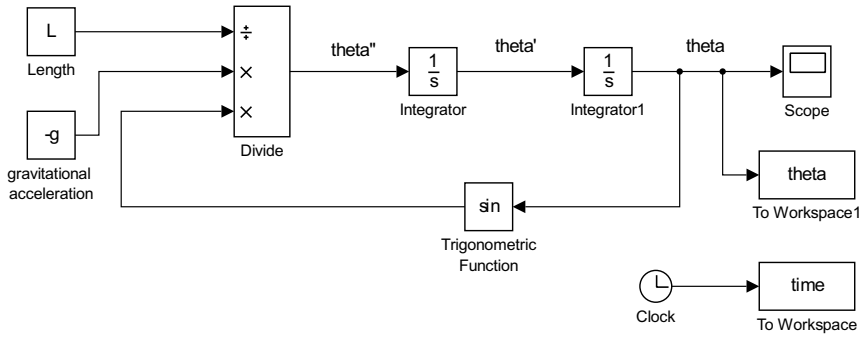


Figure 3.29: Nonlinear pendulum model with **To Workspace** blocks added to output $\theta(t)$ and t.

To see a plot of the solution, add the following lines to **pendulum.m**:

```
figure(1)
plot(time,theta)
xlabel('t')
ylabel('\theta')
```

Running the new **pendulum.m** m-file produces the plot in Figure 3.30.

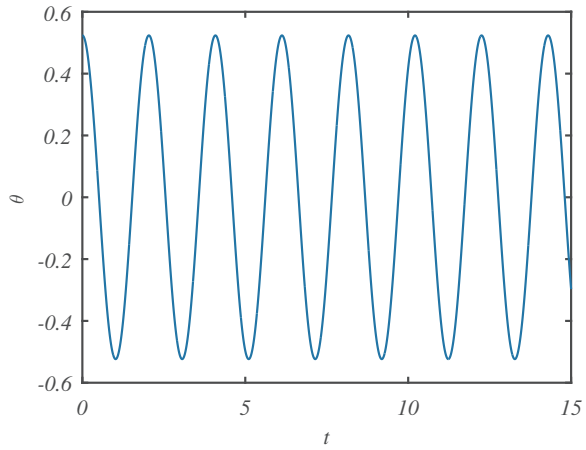


Figure 3.30: Plot of solution, $\theta(t)$ vs t, to the nonlinear pendulum model.

One can also animate the motion of the pendulum mass on the string. We use the data produces from Simulink to locate the position of the mass (as a ball) and the end of the string. For each time the mass and string are redrawn as we loop through time. The code to be added to **pendulum.m** is given as

```
rball=.05;              % mass radius
x=L*sin(theta);
y=-L*cos(theta);

    posx=x(1); posy=y(1);     % Mass's initial position

%Initialize figure, mass, and string
fig=figure(2);
axs=axes('Parent',fig);
ball=rectangle('Position',[posx-rball,posy-rball,2*rball,2*rball],...
               'Curvature',[1,1],...
               'FaceColor','b',...
               'Parent',axs);
rod=line([0 posx],[0 posy],'Marker','.','LineStyle','-')
axis(axs,[-L,L,-L-rball,L]);

for j=2:length(time)
    set(ball,'Position',[x(j)-rball,y(j)-rball,2*rball,2*rball]);
    set(rod,'XData',[0 x(j)],'YData',[0 y(j)]);
    axis([-L,L,-L-rball,L])
    pause(0.1);
end
```

In Figure 3.31 we show the starting location of the pendulum simulation.

It is interesting to compare the linear and nonlinear pendulum solutions

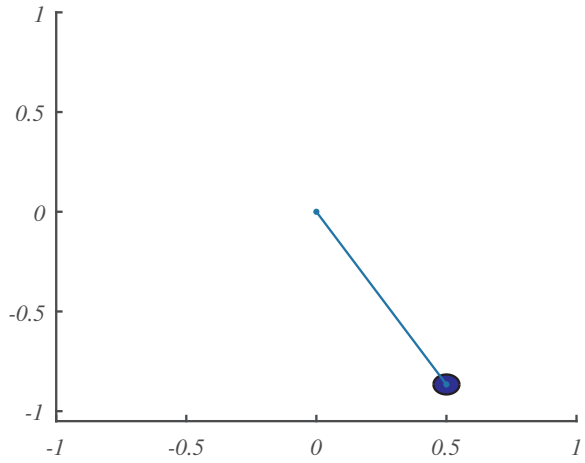


Figure 3.31: Simulation of the nonlinear pendulum in MATLAB.

to see when the small angle approximation holds. This can be done by combining the two models and comparing the solutions in a scope plot. Figure 3.32 shows such a model. The equations being solved are

$$\begin{aligned} \theta'' + 4\theta &= 0 \\ theta'' + 4\sin\theta &= 0. \end{aligned} \tag{3.9}$$

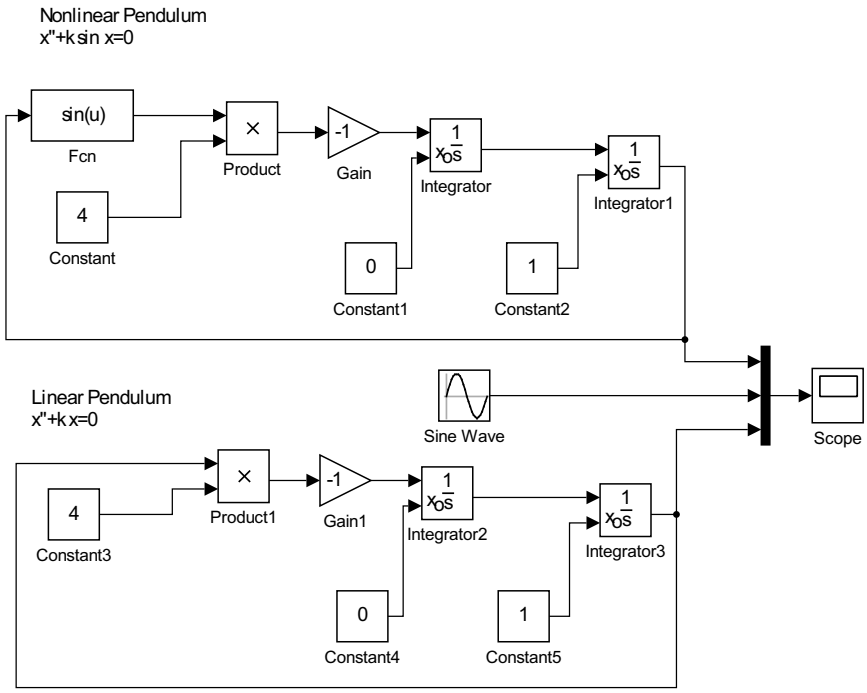


Figure 3.32: Linear and nonlinear pendulum.

3.5 Second Order ODEs in MATLAB

We can also use **ode45** to solve second and higher order differential equations. The key is to rewrite the single differential equation as a system of first order equations. Consider the simple harmonic oscillator equation, $\ddot{x} + \omega^2 x = 0$. Defining $y_1 = x$ and $y_2 = \dot{x}$, and noting that

$$\ddot{x} + \omega^2 x = \dot{y}_2 + \omega^2 y_1,$$

we have

$$\begin{aligned} \dot{y}_1 &= y_2, \\ \dot{y}_2 &= -\omega^2 y_1. \end{aligned}$$

Furthermore, we can view this system in the form $\dot{\mathbf{y}} = \mathbf{y}$. In particular, we have

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ -\omega^2 y_2 \end{bmatrix}$$

Now, we can use **ode45**. We modify the code slightly from Chapter 1.

```
[t y]=ode45('func',[0 5],[1 0]);
```

Here [0 5] gives the time interval and [1 0] gives the initial conditions

$$y_1(0) = x(0) = 1, \quad y_2(0) = \dot{x}(0) = 0.$$

The function **func** is a set of commands saved to the file **func.m** for computing the righthand side of the system of differential equations. For the simple harmonic oscillator, we enter the function as

```
function dy=func(t,y)
omega=1.0;
dy(1,1) = y(2);
dy(2,1) = -omega^2*y(1);
```

There are a variety of ways to introduce the parameter ω. Here we simply defined it within the function. Furthermore, the output **dy** should be a column vector.

After running the solver, we then need to display the solution. The output should be a column vector with the position as the first element and the velocity as the second element. So, in order to plot the solution as a function of time, we can plot the first column of the solution, y(:,1), vs t:

```
plot(t,y(:,1))
xlabel('t'),ylabel('y')
title('y(t) vs t')
```

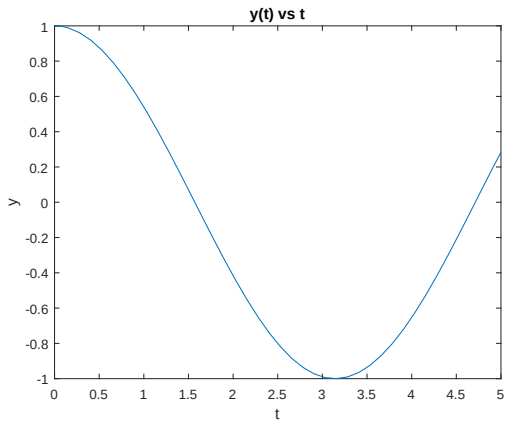


Figure 3.33: Solution plot for the simple harmonic oscillator.

The resulting solution is shown in Figure 3.33.

We can also do a phase plot of velocity vs position. In this case, one can plot the second column, y(:,2), vs the first column, y(:,1):

```
plot(y(:,1),y(:,2))
xlabel('y'),ylabel('v')
title('v(t) vs y(t)')
```

The resulting solution is shown in Figure 3.34.

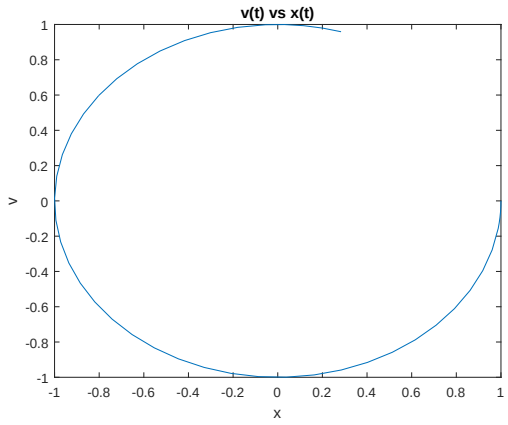


Figure 3.34: Phase plot for the simple harmonic oscillator.

Finally, we can plot a direction field using a quiver plot and add solution curves using **ode45**. The direction field is given for $\omega = 1$ by dx=y and dy=-x.

```
clear
[x,y]=meshgrid(-2:.2:2,-2:.2:2);
dx=y;
dy=-x;
quiver(x,y,dx,dy)
axis([-2,2,-2,2])
xlabel('x')
ylabel('y')
```

```
hold on
[t y]=ode45('func',[0 6.28],[1 0]);
plot(y(:,1),y(:,2))
hold off
```

The resulting plot is given in Figure 3.35.

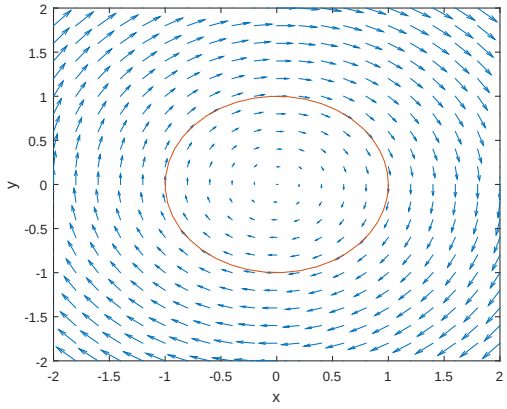


Figure 3.35: Phase plot for the simple harmonic oscillator.

3.6 Exercises

1. Model the following initial value problems in Simulink and compare solutions to those using **o**de45.

   a. $y'' - 9y' + 20y = 0,, \quad y(0) = 0, \quad y'(0) = 1.$
   b. $y'' - 3y' + 4y = 0, \quad y(0) = 0, \quad y'(0) = 1.$
   c. $8y'' + 4y' + y = 0, \quad y(0) = 1, \quad y'(0) = 0.$
   d. $x'' - x' - 6x = 0$ for $x = x(t), \quad x(0) = 0, \quad x'(0) = 1.$

2. Model the given equation in Simulink for an appropriate initial condition and plot the solution. Analytically determine and plot the solution and compare to the model solution.

   a. $y'' - 3y' + 2y = 10.$
   b. $y'' + 2y' + y = 5 + 10\sin 2x.$
   c. $y'' - 5y' + 6y = 3e^x.$
   d. $y'' + 5y' - 6y = 3e^x.$
   e. $y'' + y = \sec^3 x.$
   f. $y'' + y' = 3x^2.$
   g. $y'' - y = e^x + 1.$

3. Consider the model in Figure 3.36. Fill in the question marks with the correct expression at that point in the computation. What differential equation is solved by this simulation? [Put the equation in the simplest, recognizable form.]

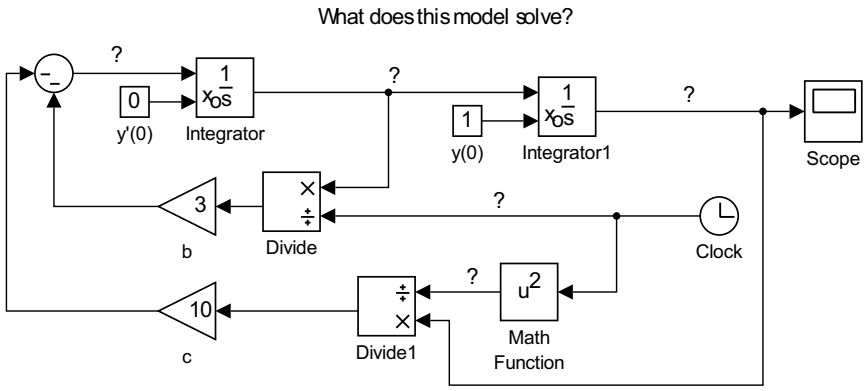


Figure 3.36: Mystery model for Problem 4.

4. Model the given equation in Simulink for an appropriate initial condition and plot the solution. Analytically determine and plot the solution and compare to the model solution.

   a. $x^2y'' + 3xy' + 2y = 0.$
   b. $x^2y'' - 3xy' + 3y = 0, \quad y(1) = 1, y'(1) = 0.$
   c. $x^2y'' + 5xy' + 4y = 0.$
   d. $x^2y'' - 2xy' + 3y = 0, \quad y(1) = 3, y'(1) = 0.$

e. $x^2y'' + 3xy' - 3y = x^2$.
f. $x^2y'' + 3xy' - 3y = x^2$.
g. $2x^2y'' + 5xy' + y = x^2 + x$.
h. $x^2y'' + 5xy' + 4y = 0$.
i. $x^2y'' - 2xy' + 3y = 0$.

5. Consider an LRC circuit with $L = 1.00$ H, $R = 1.00 \times 10^2$ Ω, $C = 1.00 \times 10^{-4}$ f, and $V = 1.00 \times 10^3$ V. Suppose that no charge is present and no current is flowing at time $t = 0$ when a battery of voltage V is inserted. Use a Simulink model to find the current and the charge on the capacitor as functions of time. Describe how the system behaves over time.

6. A certain model of the motion light plastic ball tossed into the air is given by

$$mx'' + cx' + mg = 0, \quad x(0) = 0, \quad x'(0) = v_0.$$

Here m is the mass of the ball, g=9.8 m/s^2 is the acceleration due to gravity and c is a measure of the damping. Since there is no x term, we can write this as a first order equation for the velocity $v(t) = x'(t)$:

$$mv' + cv + mg = 0.$$

a. Model this problem using Simulink.
b. Determine how long it takes for the ball to reach it's maximum height?
c. Assume that $c/m = 5$ s^{-1}. For $v_0 = 5, 10, 15, 20$ m/s, plot the solution, $x(t)$, versus the time.
d. From your plots and the expression in part b., determine the rise time. Do these answers agree?
e. What can you say about the time it takes for the ball to fall as compared to the rise time?

4

Transfer Functions and State Space Blocks

4.1 State Space Formulation

There are other more elegant approaches to solving a differential equation in Simullink. Take for example the differential equation for a forced, damped harmonic oscillator,

$$mx'' + bx' + kx = u(t). \tag{4.1}$$

Note that we changed the driving force to $u(t)$.

Defining $x_1 = x'$ and $x_2 = x'$, this second order differential equation can be written as a system of two first order differential equations,

$$\begin{aligned} x_1' &= -\frac{b}{m}x_1 - \frac{k}{m}x_2 + \frac{1}{m}u(t) \\ x_2' &= x_1 \end{aligned} \tag{4.2}$$

Note that $x_2'' = x_1'$ gives the second order equation with $x = x_2$. Also, this is not the typical order of equations usually encountered when studying systems of differential equations. This order is chosen to be consistent with the **State Space Block** which we will use later.

This system can be written in matrix form: $\mathbf{x}' = A\mathbf{x} + B\mathbf{u}$, where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$A = \begin{bmatrix} -\frac{b}{m} & -\frac{k}{m} \\ 1 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix}.$$

We now think of $\mathbf{x}' = A\mathbf{x} + B\mathbf{u}$, as a system whose input is given by the forcing term $u(t)$ and we need to integrate the right hand side for a given input function. The output of this system is the solution vector, $\mathbf{x}$. Also, we might want to output a plot of the forcing function. Thus, the complete output from the system can be written as

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u},$$

where C is a row vector and $D = 0$ or 1. In particular, we might only want to output the solution component x_2. So, we let $C = [0, 1]$ and $D = 0$. The block diagram for this process is shown in Figure 4.1.

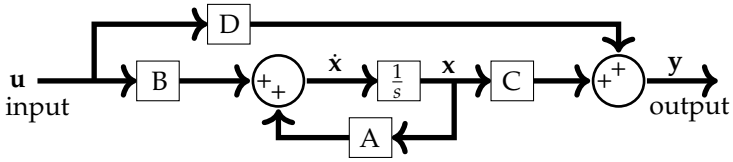


Figure 4.1: State space representation of the system $\mathbf{x}' = A\mathbf{x} + B\mathbf{u}$, $\mathbf{y} = C\mathbf{x} + D\mathbf{u}$,

The whole process is captured in the **State Space Block**. This block is found in the Continuous group. The implementation of this system with a sinusoidal forcing term is depicted in Figure 4.2. This shows the pair of equations

$$\begin{aligned} \mathbf{x}' &= A\mathbf{x} + B\mathbf{u}, \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u}. \end{aligned} \tag{4.3}$$

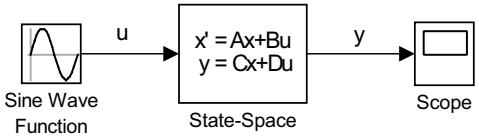


Figure 4.2: The use of the**State Space Block** displaying a **Sine Wave** input and output to a **Scope**.

As an example, we consider the case where $m = 2$ kg, $b = 0.2$ kg/s, and $k = 1.0$ N/m. We also let $u(t) = \sin t$. Then, we have

$$A = \begin{bmatrix} -0.1 & -0.5 \\ 1 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix},$$

$C = [0, 1]$, and $D = 0$. These values are put into the block by going into the **Function Block Parameters** dialog box for the **State Space** block as shown in Figure 4.3. Note that there is a place to enter the initial condition, such as $[x_1(0), x_2(0)]^T = [0, 1]^T$. In this case one would type **[0; 1]**. A comparison of outputs from using this initial condition to zero initial conditions is shown in Figure 4.4. Figure 4.5 shows the system needed to produce this plot.

4.2 Transfer Functions

ANOTHER METHOD FOR SOLVING THE DIFFERENTIAL EQUATION compactly is to use the **Transfer Fcn** block. This is shown in Figure 4.6. One needs to enter the transfer function numerator and denominator in the **Function Block Parameter** box, shown in Figure 4.7. Essentially, this is

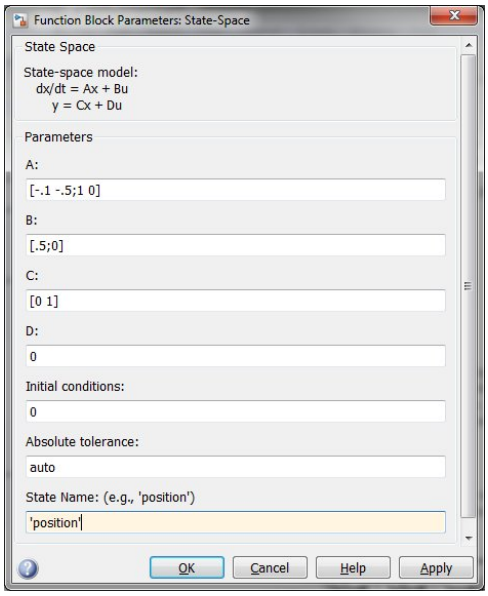


Figure 4.3: State space block parameters.

recognized as the Laplace transform of the differential equation with zero initial conditions.

Recall, the Laplace transform of a function $f(t)$ is defined as

$$X(s) = \mathcal{L}[x](s) = \int_0^\infty x(t)e^{-st}\,dt, \quad s > 0. \tag{4.4}$$

Also, we have the properties

$$\begin{aligned} \mathcal{L}\left[\frac{dx}{dt}\right] &= sX(s) - x(0) \\ \mathcal{L}\left[\frac{d^2x}{dt^2}\right] &= s^2X(s) - sx(0) - x'(0). \end{aligned} \tag{4.5}$$

Taking the Lplace transform of the differential equation, $2x'' + .2x' + x = \sin t$, with $x(0) = x'(0) = 0$,

$$(2s^2 + .2s + 1)X(s) = \frac{1}{s^2+1}.$$

Solving for $X(s)$,

$$X(s) = \frac{1}{2s^2 + .2s + 1}\mathcal{L}[u(t)].$$

Therefore, the transfer function comes from the factor multiplying $\mathcal{L}[u(t)]$. In this case, one enters the coefficients of the second order differential equation into the denominator as **[2 .2 1]**. Unfortunately, one can only solve problems with zero initial conditions.

If one knows the transfer function, then one can use it to create an equivalent **State Space block**. This is done using the MATLAB function **tf2ss(1,[2,.2,1])**. We note that this produces the parameters A, B, C, D, but what it gives is

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

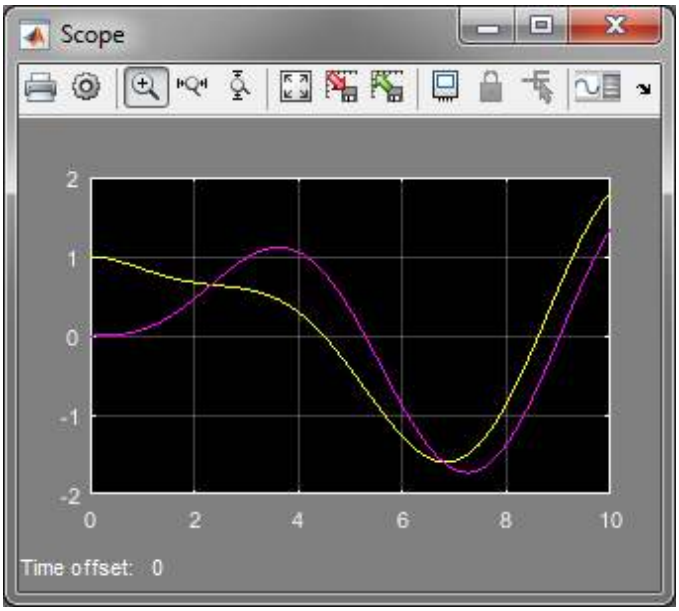


Figure 4.4: Solution to the forced, damped harmonic oscillator problem with initial conditions set to 0 or [0;1].

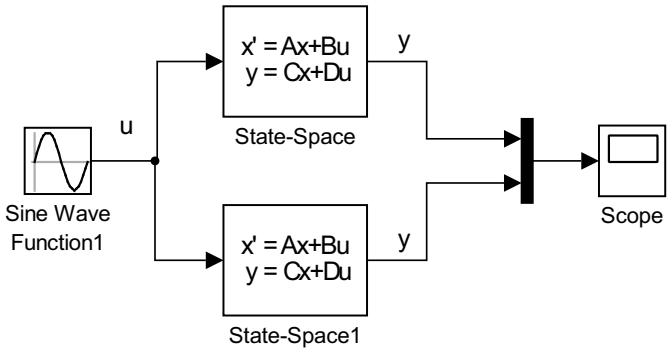


Figure 4.5: The use of the**State Space Block** dispaying a Sine Wave input and output to a **Scope**. The **Mux** block (from Signal Routing) is used to feed solutions from two systems using different initial conditions.

and $C = [0, 0.5]$, This differs from what we derived above. The difference lies in the fact that we can multiply B by any constant and divide C by that constant and not affect the solution of the problem. In this case, the constant in question is the mass, $m = 2.0$.

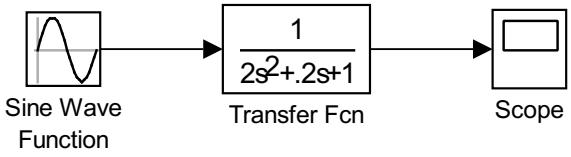


Figure 4.6: The use of the **Transfer Fcn** Block with a **Sine Wave** input and output to a **Scope**.

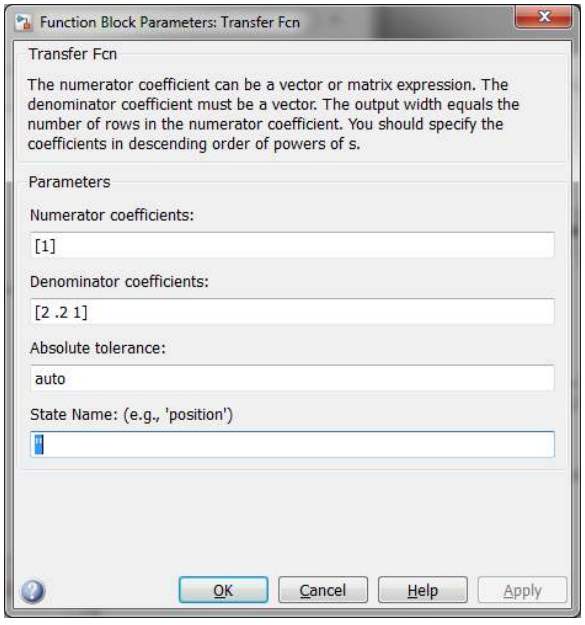


Figure 4.7: Block parameter display for the Transfer Fcn block.

5 Systems of Differential Equations

5.1 Linear Systems

We consider the linear system

$$\begin{aligned} x' &= ax + by \\ y' &= cx + dy. \end{aligned} \tag{5.1}$$

This can be modeled using two integrators, one for each equation. Due to the coupling, we have to connect the outputs from the integrators to the inputs.

As an example, we show in Figure 5.1 the case $a = 0$, $b = 1$, $c = -1$, $d = 0$. This is the linear system of first order equations for $x'' + x = 0$, and $y = x'$. We also insert the initial conditions $x(0) = 1$, $y(0) = 2$. Running the model, results in the plots in Figures 5.2 and 5.3.

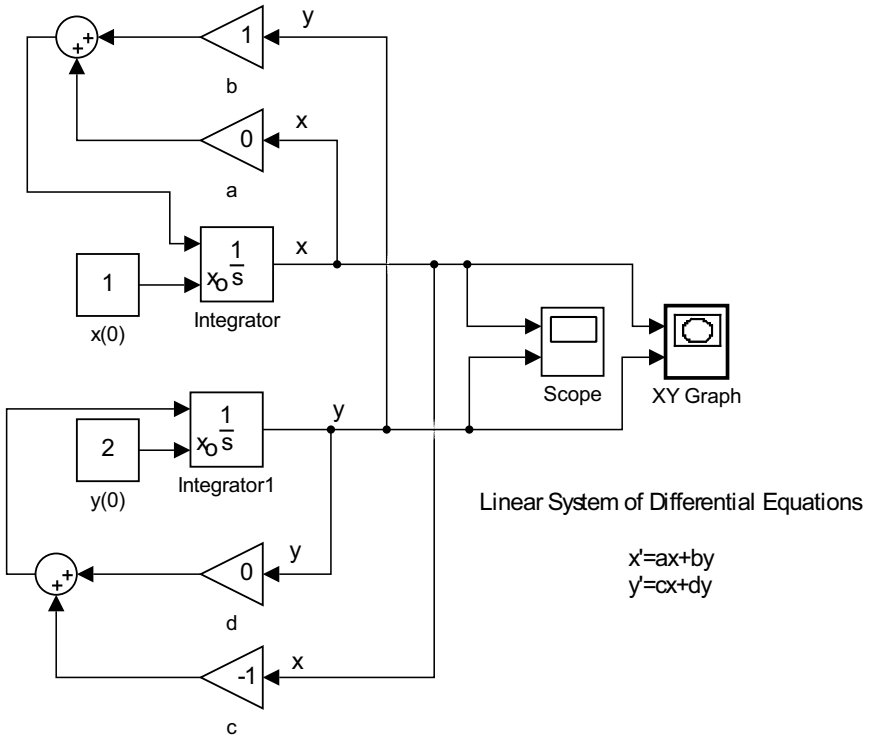


Figure 5.1: Linear system using two integrators.

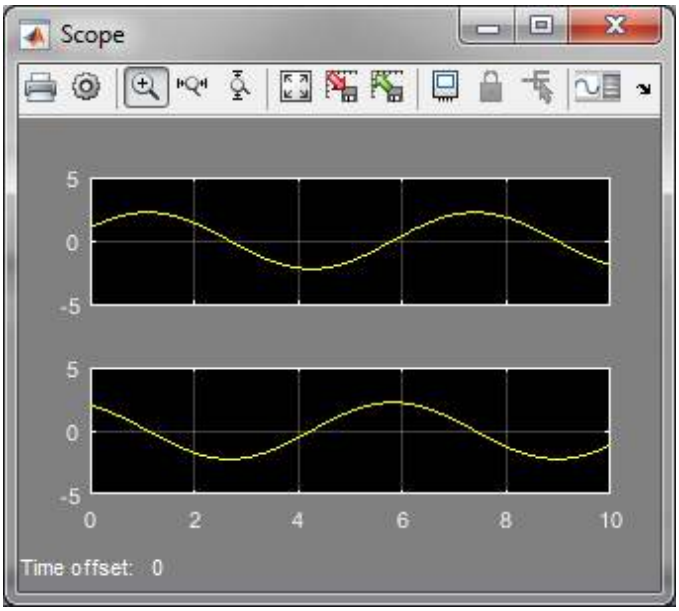


Figure 5.2: Linear system using two integrators.

This system can by put in matrix form,

$$\begin{bmatrix} x \\ y \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

This can be modeled by introducing matrix multiplication in a gain block as shown in Figure 5.4. The input and output to the **Integrator block** are vectors. The output is split using a **Demux** block to plot x and y separately. The **Scope** block plots the two signals separately as functions of t. The **XY Graph** block is used to plat the phase plane, y vs x..

We can also use a **State Space** block to solve this system. This is shown in Figure 5.5. We set the input as $u = 0$. In order to output both x and y, we set $A = [01; 0 - 1]$, $B = [0; 0]$, $C = [10; 01]$, and $D = [0; 0]$. We also set the initial conditions to $[1; 2]$. The solution plots are the same as shown in Figures 5.2 and 5.3.

5.2 Nonlinear Models

The Jerk Equation

IN THIS SECTION we consider modeling a few common nonlinear systems with interesting behaviors in Simulink. These examples stem from a variety of applications such as biological systems, predator-prey models, chemical reactions, such as Michaelis-Menten kinetics, circuits, and other dynamical systems. We begin with the jerk model.

If one denotes $x(t)$ as the position as a function of time, t, then we are familiar with the idea that $x'(t)$ would be the velocity and $x''(t)$ the acceleration. However, you might not be as familiar with the *jerk*. This is the

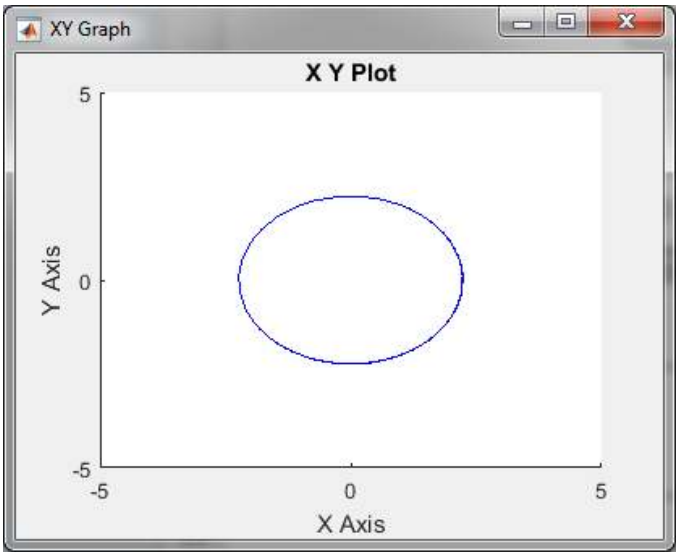


Figure 5.3: Linear system using two integrators.

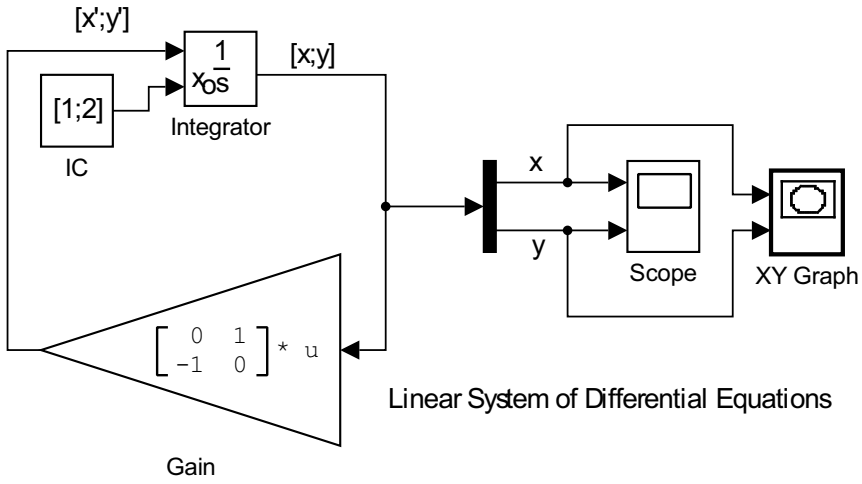


Figure 5.4: Linear system using matrix operation.

third derivative, $x'''(t)$. The jerk equation modeled in Figure 5.6 is

$$x''' + cx'' + bx' + ax + x^2 = 0.$$

As a third order equation, one needs initial values for x, x', and x''.

Van der Pol Equation

SOLUTIONS, KNOWN AS LIMIT CYCLES, are common in nature. Rayleigh investigated the problem

$$x'' + c\left(\frac{1}{3}(x')^2 - 1\right)x' + x = 0 \tag{5.2}$$

in the study of the vibrations of a violin string. Balthasar van der Pol (1889-1959) studied an electrical circuit, modeling this behavior. Limit

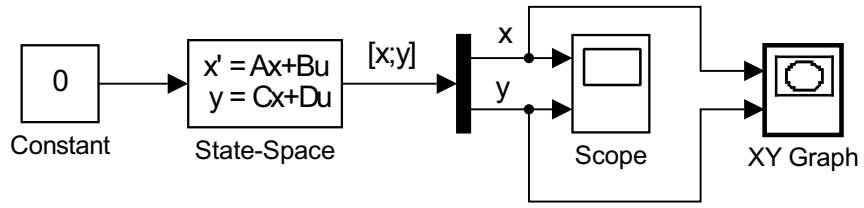


Figure 5.5: Linear system using matrix operation.

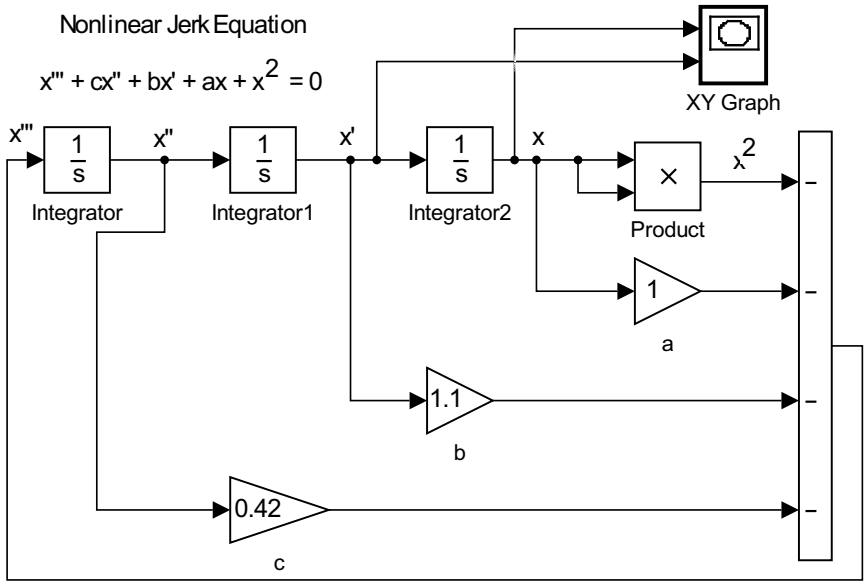


Figure 5.6: Nonlinear jerk model.

cycles are isolated periodic solutions towards which neighboring states might tend when stable. A slight change of the Rayleigh system leads to the van der Pol equation:

$$x'' + c(x^2 - 1)x' + x = 0 \tag{5.3}$$

The limit cycle is found in the model and solutions in Figures 5.7-5.9.

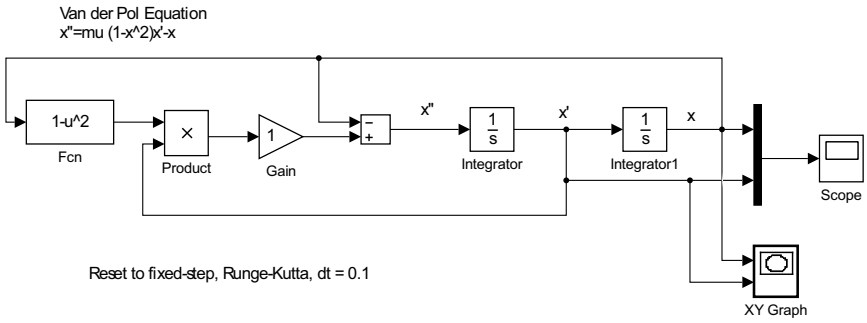


Figure 5.7: van der Pol equation.

Lorenz Equations

THE LORENZ MODEL IS ANOTHER TYPICAL MODEL used as an example of a nonlinear system. The Lorenz model is a simple model for atmospheric

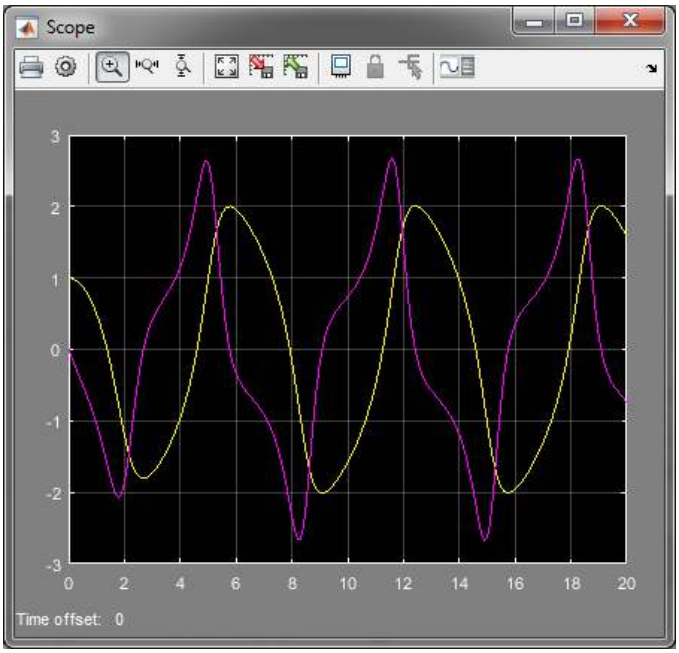


Figure 5.8: Solution plot for the van der Pol equation.

convection developed by Edward Lorenz in 1963. The system is given by the three equations

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x) \\ \frac{dy}{dt} &= x(\rho-z)-y \\ \frac{dz}{dt} &= xy-\beta z. \end{aligned}$$

Figures 5.10-5.12 show the models and a famous solution to the Lorenz equations.

Using the data sent to the MATLAB workspace, a three dimensional model can be constructed. The following produces an animation of the data resulting in a 3D plot.

```
Z=simout.data;
N=length(Z(:,1));

figure(3)
axHndl = gca;
figNumber = gcf;
hndlList = get(figNumber,'UserData');
set(axHndl, ...
    'XLim',[0 50],'YLim',[-20 20],'ZLim',[-30 30], ...
    'XTick',[],'YTick',[],'ZTick',[], ...
    'SortMethod','childorder', ...
    'Visible','on', ...
    'NextPlot','add', ...
```

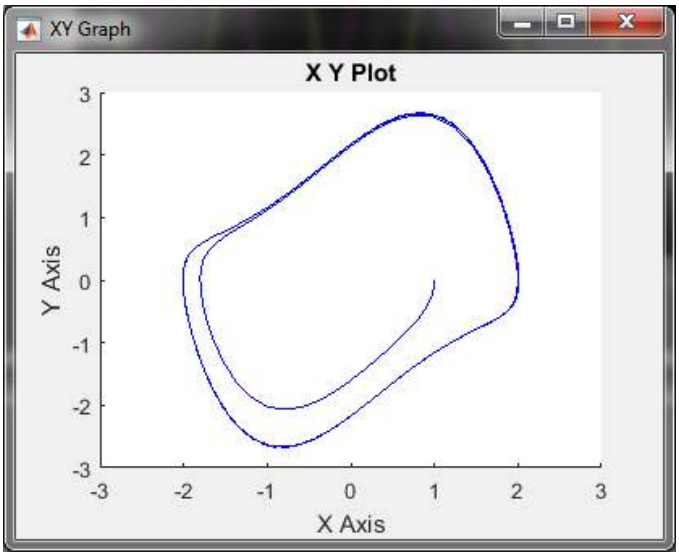


Figure 5.9: Phase plane plot for the van der Pol equation.

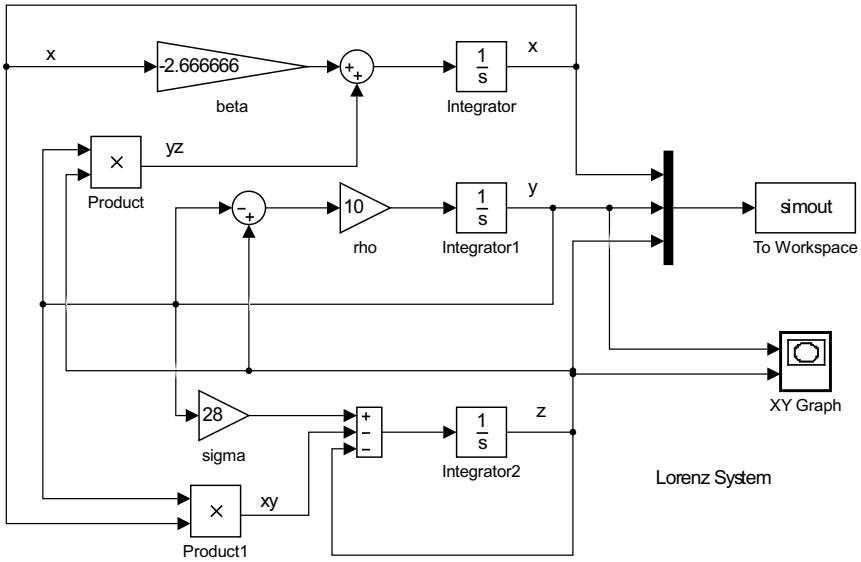


Figure 5.10: Model for Lorenz equations.

```
    'View',[-37.5,30], ...
    'Clipping','off');
xlabel('x');
ylabel('y');
zlabel('z');

y(1) = Z(1,1);
y(2) = Z(1,2);
y(3) = Z(1,3);

L = 5;
Y = y*ones(1,L);
```

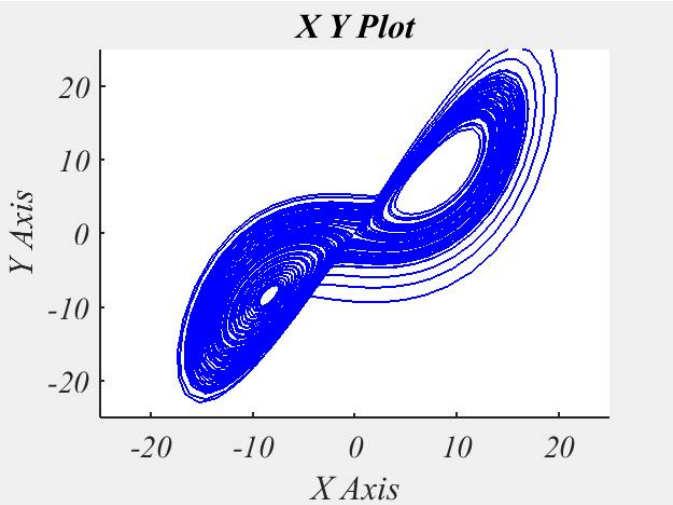


Figure 5.11: XY plot for the Lorenz model.

```
cla;
head = line('color','r', 'Marker','.','MarkerSize',10,'LineStyle','none', ...
            'XData',y(1),'YData',y(2),'ZData',y(3)) ;
body = animatedline('color','b', 'LineStyle','-') ;
tail = animatedline('color','b', 'LineStyle','-') ;

for j=2:N
    y(1) = Z(j,1);
    y(2) = Z(j,2);
    y(3) = Z(j,3);

    % Update the plot
    Y = [y Y(:,1:L-1)];
    set(head, 'XData', Y(1,1), 'YData', Y(2,1), 'ZData', Y(3,1));
    addpoints(body, Y(1,2), Y(2,2), Y(3,2));
    addpoints(tail, Y(1,L), Y(2,L), Y(3,L));
    pause(0.1)

    % Update the animation every ten steps
    if ~mod(j,10)
        drawnow;
    end
end
```

Lotka-Volterra Predator-Prey Model

Two well-known nonlinear population models are the predator-prey and competing species models. In the predator-prey model, one typically has one species, the predator, feeding on the other, the prey. We will

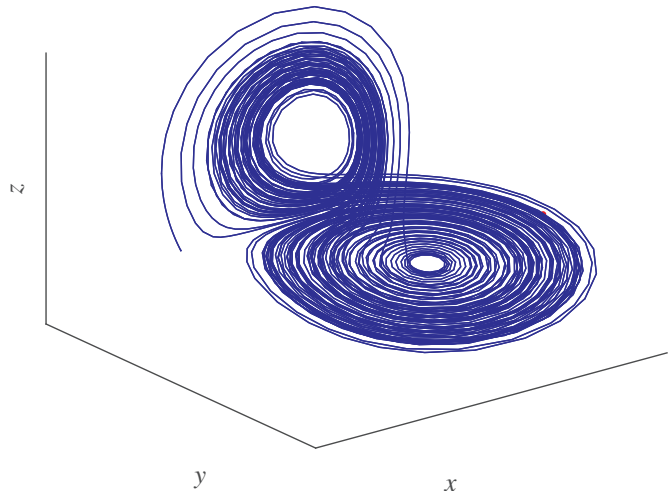


Figure 5.12: Three dimensional plot for the Lorenz model.

look at the standard Lotka-Volterra model in this section. The competing species model looks similar, except there are a few sign changes, since one species is not feeding on the other. Also, we can build in logistic terms into our model. We will save this latter type of model for the homework.

The Lotka-Volterra model is named after Alfred James Lotka (1880-1949) and Vito Volterra (1860-1940).

The Lotka-Volterra model of population dynamics.

The Lotka-Volterra model takes the form

$$\begin{aligned} \dot{x} &= ax - bxy, \\ \dot{y} &= -dy + cxy, \end{aligned} \tag{5.4}$$

where a, b, c, and d are positive constants. In this model, we can think of x as the population of rabbits (prey) and y is the population of foxes (predators). Choosing all constants to be positive, we can describe the terms.

- ax: When left alone, the rabbit population will grow. Thus a is the natural growth rate without predators.
- $-dy$: When there are no rabbits, the fox population should decay. Thus, the coefficient needs to be negative.
- $-bxy$: We add a nonlinear term corresponding to the depletion of the rabbits when the foxes are around.
- cxy: The more rabbits there are, the more food for the foxes. So, we add a nonlinear term giving rise to an increase in fox population.

SIR Model of Disease

ANOTHER INTERESTING AREA OF APPLICATION of differential equation is in predicting the spread of disease. Typically, one has a population of susceptible people or animals. Several infected individuals are introduced into the population and one is interested in how the infection spreads and

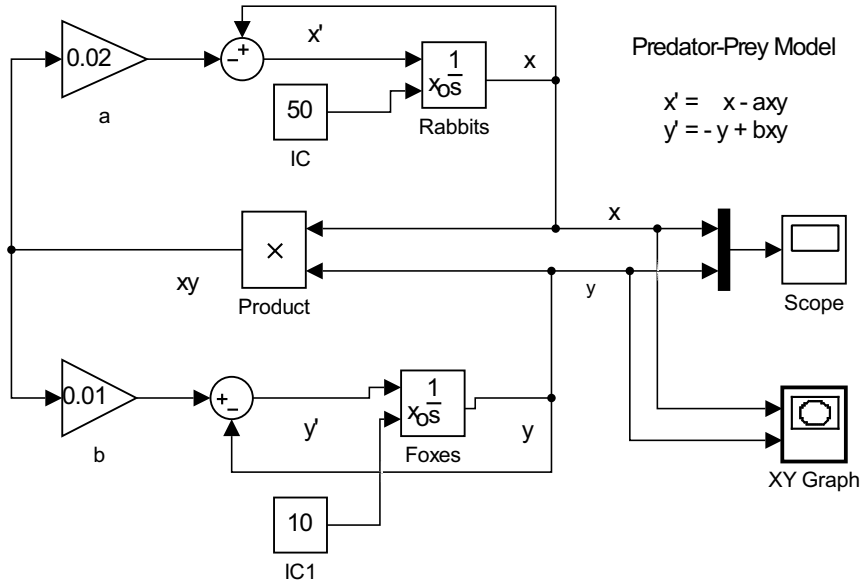


Figure 5.13: Predator-Prey model.

if the number of infected people drastically increases or dies off. In the SIR model one uses a compartmental analysis by breaking the population into three classes. First, we let $S(t)$ represent the healthy people, who are susceptible to infection. Let $I(t)$ be the number of infected people. Of these infected people, some will die from the infection and others could recover. We will consider the case that initially there is one infected person and the rest, say N, are healthy. Can we predict how many deaths have occurred by time t?

We can first look into a linear model. We assume that the rate of change of any population would be due to those entering the group less those leaving the group. For example, the number of healthy people decreases due infection and can increase when some of the infected group recovers. Let's assume that a) the rate of infection is proportional to the number of healthy people, aS, and b) the number who recover is proportional to the number of infected people, rI. Thus, the rate of change of healthy people is found as

$$\frac{dS}{dt} = -aS + rI.$$

Let the number of deaths be $D(t)$. Then, the death rate could be taken to be proportional to the number of infected people. So,

$$\frac{dD}{dt} = dI$$

Finally, the rate of change of infected people is due to healthy people getting infected and the infected people who either recover or die. Using the corresponding terms in the other equations, we can write the rate of change of infected people as

$$\frac{dI}{dt} = aS - rI - dI.$$

This linear system of differential equations can be written in matrix form.

$$\frac{d}{dt}\begin{pmatrix} S \\ I \\ D \end{pmatrix} = \begin{pmatrix} -a & r & 0 \\ a & -d-r & 0 \\ 0 & d & 0 \end{pmatrix}\begin{pmatrix} S \\ I \\ D \end{pmatrix}. \tag{5.5}$$

The commonly used nonlinear SIR model is given by

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{5.6}$$

where S is the number of susceptible individuals, I is the number of infected individuals, and R are the number who have been removed from the the other groups, either by recovering or dying. The Simulink model is given in Figure 5.14.

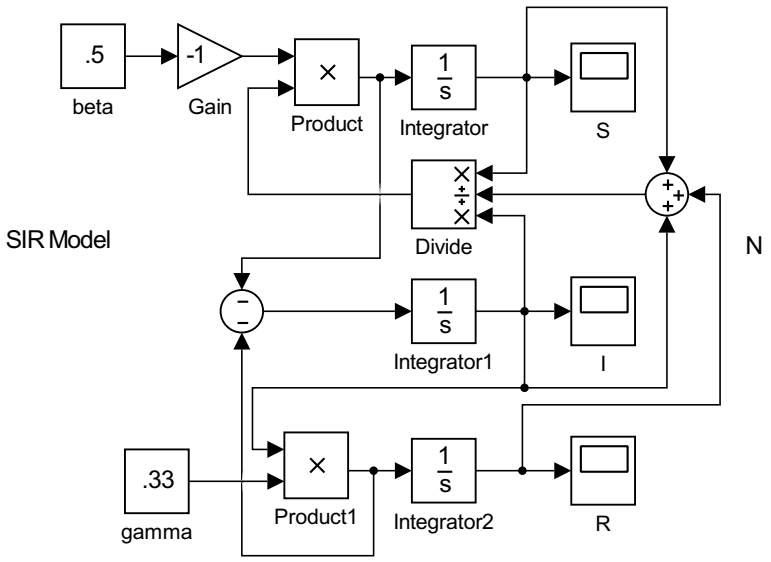


Figure 5.14: SIR epidemic model.

Michaelis-Menten Kinetics

THE MICHAELIS-MENTEN KINETICS REACTION is given by

$$E + S \underset{k_1}{\overset{k_3}{\leftrightarrows}} ES \xrightarrow[k_2]{} E + P.$$

This approximates the dynamics under the assumption that the concentration of the enzyme remains constant. The enzyme interacts with the substrate to form an enzyme–substrate complex, leading to a release of enzyme. E, S, and P are the enzyme, substrate, and product, respectively. The system of differential equations corresponding to the dynamics of these reactions is

$$\begin{aligned}
\frac{d[S]}{dt} &= -k_1[E][S] + k_3[ES], \\
\frac{d[E]}{dt} &= -k_1[E][S] + (k_2 + k_2)[ES], \\
\frac{d[ES]}{dt} &= k_1[E][S] - (k_2 + k_2)[ES], \\
\frac{d[P]}{dt} &= k_3[ES]. \qquad (5.7)
\end{aligned}$$

In chemical kinetics one seeks to determine the rate of product formation ($v = d[P]/dt = k_3[ES]$). Assuming that $[ES]$ is a constant, one seeks v as a function of $[S]$ and the total enzyme concentration $[E_T] = [E] + [ES]$.

The Chua Circuit

THE CHUA CIRCUIT, as shown in Figure 5.15, consists of an inductor, a resistor, two capacitors and a nonlinear resistor, or other nonlinear component. The system of differential equations is found using Kirchoff's circuit laws. There are two junctions, labeled as 1 and 2. The total current into each node equals the current leaving the node. There are three loops over which one sums the potential rises and drops.

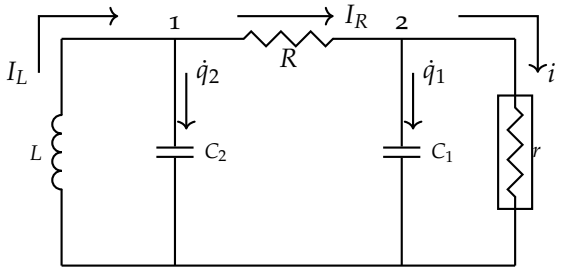


Figure 5.15: The Chua circuit used in this note.

Using junction rules, we have at nodes 1 and 2:

$$\begin{aligned}
I_L &= \dot{q}_2 + I_R, \qquad (5.8) \\
I_R &= \dot{q}_1 + i. \qquad (5.9)
\end{aligned}$$

Kirchoff's Loop rules for the three small loops are

$$\begin{aligned}
L\frac{dI_L}{dt} &= -V_2, \qquad (5.10) \\
I_R R &= V_2 - V_1, \qquad (5.11) \\
V_r &= \frac{q_1}{C_1}. \qquad (5.12)
\end{aligned}$$

We seek a system of differential equations for V_1, V_2, and I_L. Noting that $q_i = C_i V_i$, for $i = 1, 2$, we find from Equations (5.8) and (5.11):

$$C_2\dot{V}_2 = I_L - R^{-1}(V_2 - V_1).$$

From Equation (5.9) we have, using Equation (5.11),

$$C_1\dot{V}_1 = R^{-1}(V_2 - V_1) - g(V_1),$$

where $g(x)$ gives the characteristics of the nonlinear component in the circuit. This is typically of the form

$$g(x) = ax + \frac{1}{2}b\left(|x+1| - |x-1|\right).$$

This function is show in Figures 5.16-5.17 for $a = 0$ abd $a \neq 0$.

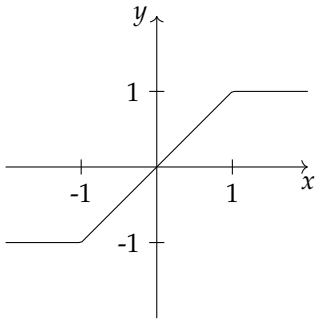


Figure 5.16: $g(x) = \frac{1}{2}\left(|x+1| - |x-1|\right)$.

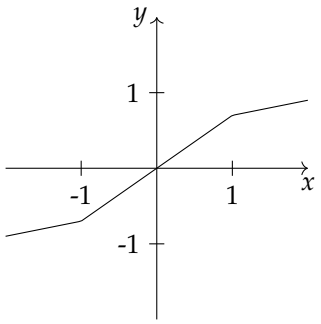


Figure 5.17: $g(x) = ax + \frac{1}{2}b\left(|x+1| - |x-1|\right)$.

The last equation comes from Equation (5.10) and often a term $-rI_L$ is added. So, we have

$$\begin{aligned} C_1\dot{V}_1 &= R^{-1}(V_2 - V_1) - g(V_1), && (5.13)\\ C_2\dot{V}_2 &= I_L - R^{-1}(V_2 - V_1), && (5.14)\\ L\dot{I}_L &= -V_2 - rI_L. && (5.15) \end{aligned}$$

These equations are made dimensionless by introducing some characteristic scales. Let C_1 and R_1 be characteristic scales of capacitance and resistance. We let $\alpha^{-1} = R/R_1$, $\bar{r} = r/R_1$, and define

$$x = \frac{V_1}{V_C}, \quad y = \frac{V_2}{V_C}, \quad z = \frac{I_L R_1}{V_C}.$$

This gives

$$\begin{aligned} R_1C_1\dot{x} &= \alpha(y - x) - g(V_1)/V_C, && (5.16)\\ R_1C_2\dot{y} &= z - \alpha(y - x), && (5.17)\\ \frac{L}{R_1}\dot{z} &= -y - \bar{r}z. && (5.18) \end{aligned}$$

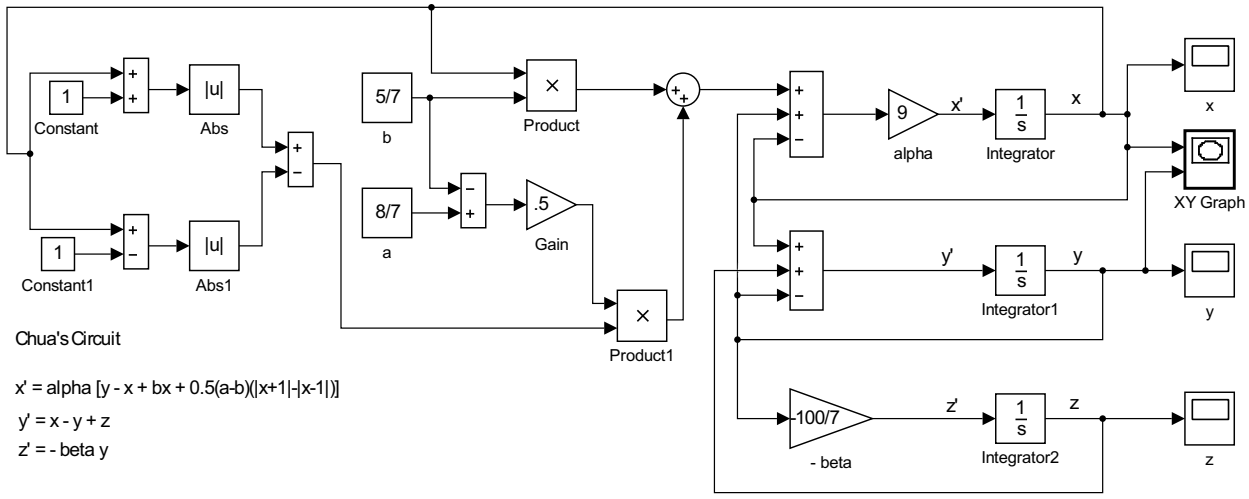


Figure 5.18: Nonlinear Chua model.

Finally, we can rescale the time as $\tau = t/R_1C_1$, where R_1C_1 is the characteristic time constant. Then,

$$\frac{d}{dt} = \frac{d\tau}{dt}\frac{d}{d\tau} = \frac{1}{R_1C_1}\frac{d}{d\tau}.$$

So,

$$\begin{aligned} \dot{x} &= \alpha(y-x) - g(V_1)/V_C, && (5.19)\\ \frac{C_2}{C_1}\dot{y} &= z - \alpha(y-x), && (5.20)\\ \frac{L}{R_1^2C_1}\dot{z} &= -y - \bar{r}z. && (5.21) \end{aligned}$$

So, we define $\sigma = \frac{C_1}{C_2}$, $\beta = \frac{R_1^2C_1}{L}$, $\gamma = \bar{r}$, and

$$f(x) = g(V_1)/aV_C.$$

Then,

$$\begin{aligned} \dot{x} &= \alpha(y - x - f(x)), && (5.22)\\ \dot{y} &= \sigma(z - \alpha(y-x)), && (5.23)\\ \dot{z} &= -\beta y - \gamma z. && (5.24) \end{aligned}$$

Finally, many models have no parameters in the second equation. So, we let $x = \mu X$, $y = \mu Y$ and $z = \nu Z$ to see if this is possible. Then,

$$\begin{aligned} \mu\dot{X} &= \alpha\mu(Y-X) - af(\mu X), && (5.25)\\ \mu\dot{Y} &= \sigma(\nu Z - \alpha\mu(Y-X)), && (5.26)\\ \nu\dot{Z} &= -\beta\mu Y - \gamma\nu Z. && (5.27) \end{aligned}$$

Simplifying, we have

$$\dot{X} = \alpha(Y-X) - \frac{\alpha}{\mu}f(\mu X), \quad (5.28)$$

$$\dot{Y} = \sigma(\frac{\nu}{\mu}Z - \alpha(Y - X)), \tag{5.29}$$

$$\dot{Z} = -\frac{\beta\mu}{\nu}Y - \gamma Z. \tag{5.30}$$

So, we need to chose $\sigma = \alpha^{-1}$ and $\frac{\mu}{\nu} = \sigma$.

$$\dot{X} = \alpha(Y - X - \bar{f}(X)), \tag{5.31}$$

$$\dot{Y} = Z - Y + X, \tag{5.32}$$

$$\dot{Z} = -\bar{\beta}Y - \gamma Z, \tag{5.33}$$

where $\bar{\beta} = \beta\sigma$ and $\bar{f}(X) = \mu f(\mu X)$. This is the version of the model we can explore.

We have obtained a dimensionless set of first order differential equations of the form

$$\dot{x} = \alpha(y - x - f(x)), \tag{5.34}$$

$$\dot{y} = z - y + x, \tag{5.35}$$

$$\dot{z} = -\beta y - \gamma z, \tag{5.36}$$

where

$$f(x) = ax + \frac{1}{2}b\left(|x+1| - |x-1|\right).$$

We can write this system in matrix form as

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} -\alpha & \alpha & 0 \\ 1 & -1 & 1 \\ 0 & -\beta & -\gamma \end{pmatrix}\mathbf{x} + \begin{pmatrix} -\alpha f(x) \\ 0 \\ 0 \end{pmatrix},$$

where

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

We can model this in Simulink as shown in Figure 5.19. The linear part of the system is encoded as a subsystem. The subsystem is shown in Figure 5.20.

The subsystem takes inputs of the variables α, β, and γ and outputs the matrix in the linear part of the system, L. Then, the nonlinear part of the system is added to $L\mathbf{x}$. This is integrated with given initial conditions to arrive at the solution. A sample of the solutions is given in Figures 5.21 and 5.22.

The plots in Figures 5.21 and 5.22 were created by using the **to Workspace** block. The variable name was changed to **chuaput** and the data was sent to the MALAB workspace. Then the following code was used to plot the data.

```
% Plot x, y, z vs t
figure(1)
```

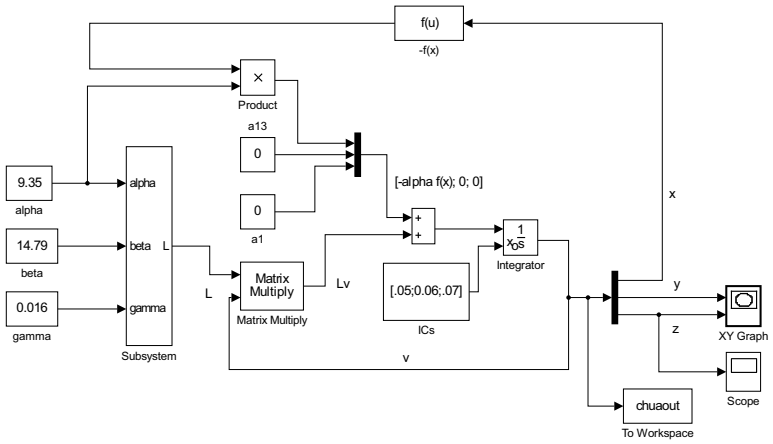


Figure 5.19: Chua circuit.

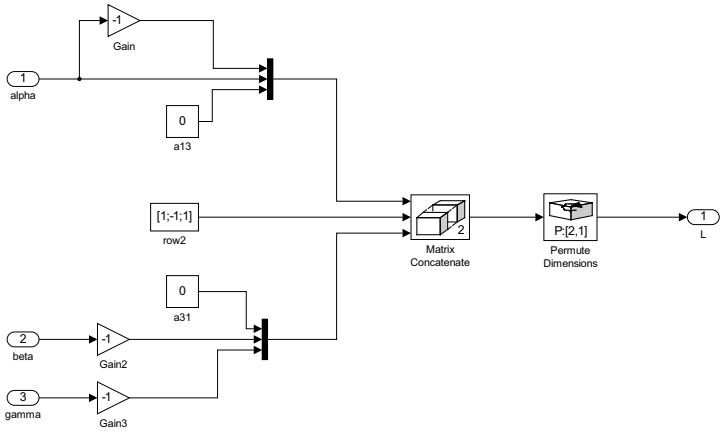


Figure 5.20: Linear subsystem of Chua model.

```
plot(chuaout.time,chuaout.Data);
xlabel('t')
legend('x(t)','y(t)','z(t)','Location','south','Orientation','horizontal')

% Plot spacecurve
figure(2)

x=chuaout.data(:,1);
y=chuaout.data(:,2);
z=chuaout.data(:,3);

plot3(x,y,z)
xlabel('x')
ylabel('y')
zlabel('z')
```

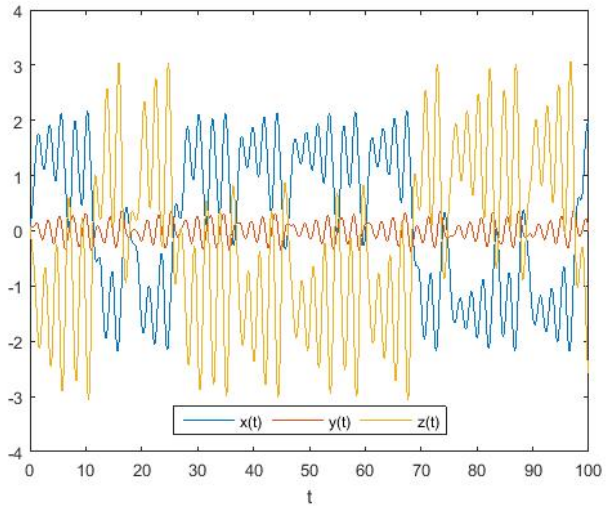


Figure 5.21: Solutions of Chua model as a function of time.

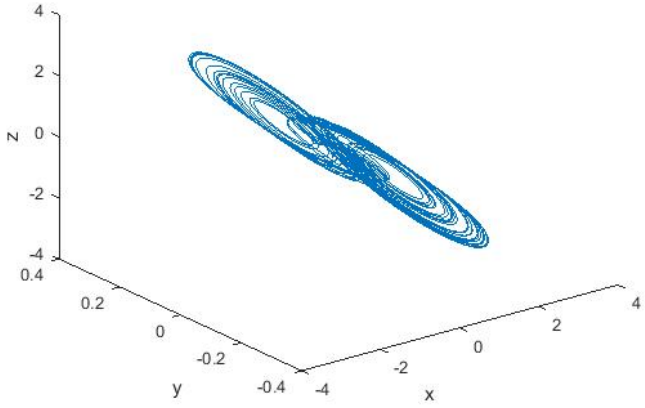


Figure 5.22: 3d plot of Chua solutions.

6

Index