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 examples of solutions of first and second order differential equations usually encountered in a differential equations course using Simulink. We will then look at examples of more complicated systems.

1.1 Solving an ODE

Simulink is a graphical environment for designing simulations of systems. When you have access to Simulink and MATLAB, you can start MATLAB and on the icon bar there is an icon that you can click to launch Simulink. Alternatively, you can type **simulink** to bring up the Simulink Library Browser as shown in Figure 1.1. 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.

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 1.2.

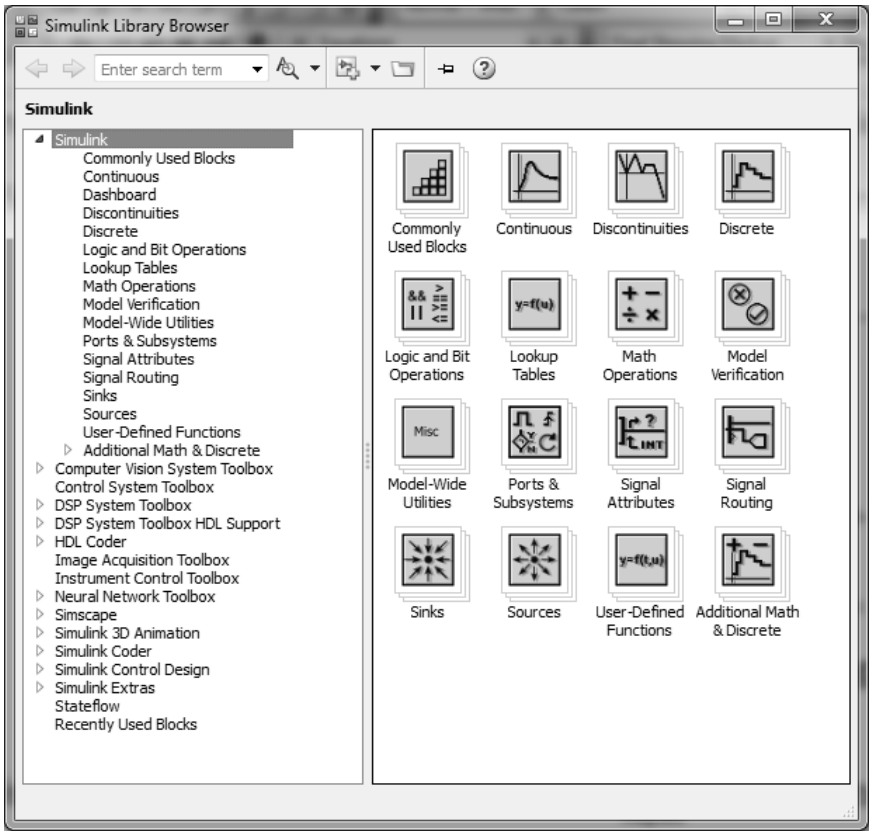

Figure 1.1: The Simulink Library Browser. This is where various blocks can be found for constructing models. [As seen in MATLAB 2015a.]

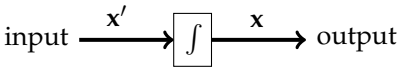

Figure 1.2: Schematic for a general system in which the block takes the input and produces an output.

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 1.3.

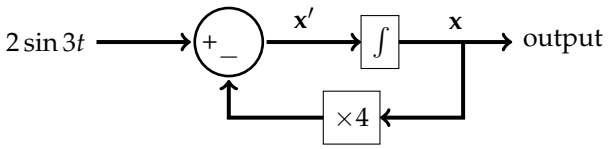

Figure 1.3: Schematic for solving $x' = 2\sin 3t - 4x$. The terms $2\sin 3t$ and $4x$ are fed into the integrator and x is output.

The simulation in Simulink carries out this procedure and takes the form shown in Figure 1.4. 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 

to integrate $\dfrac{dx}{dt}$, producing $x(t)$. The **Scope** is used to plot the output of the **Integrator** block, $x(t)$.

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 component. 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. That is the main idea behind solving this system.

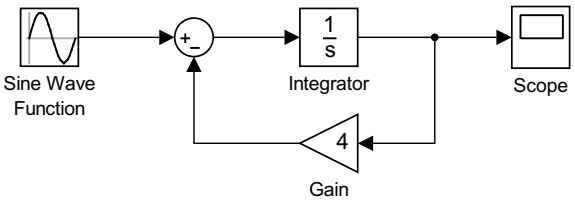

Figure 1.4: System for solving first order ODE $\frac{dx}{dt} = 2\sin 3t - 4x$ as a Simulink simulation.

For this example, we implement the following steps:

- Drag needed blocks into the model region [Figure 1.5.]:
  - **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.

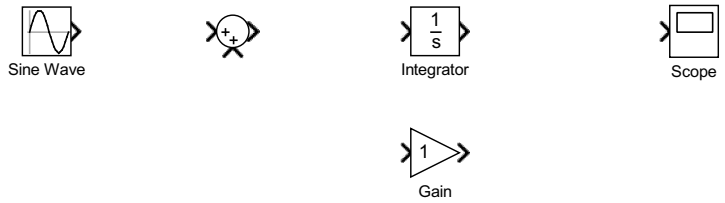

Figure 1.5: Add needed components to the model window.

- Connect the output of the **Sum** block to the input of the **Integrator** block. [Figure 1.6.]

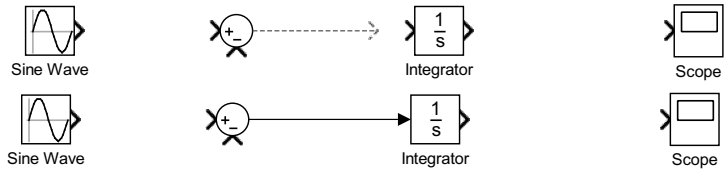

Figure 1.6: Example of connecting two components: Align the components, Click on output of one and drag to another. Then, release to finalize connection

- Connect the **Integrator** to the **Scope** by clicking on the **Integrator** output and dragging to the **Scope** until they are connected.
- 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.

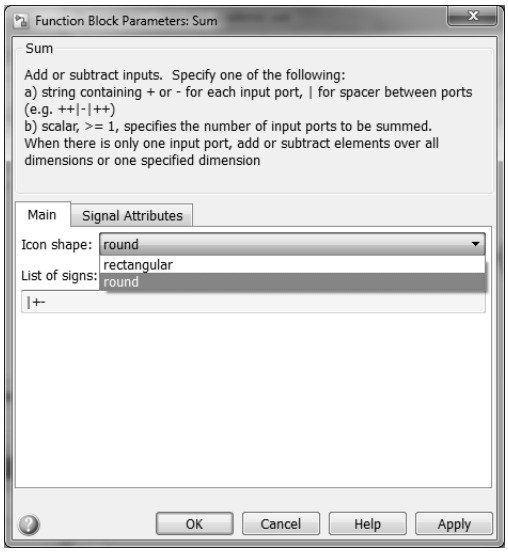

Figure 1.7: Block Parameters for the **Sum** control.

- Double-click the **Sum** control to bring up **Block Parameters** as shown in Figure 1.7 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 sometimes useful in displaying an overall flow direction to the model.]

- Double-click the **Sine Wave** block and change the frequency to 3 rad/s and the amplitude to 2. [See Figure 1.8] Set the time dropdown menu to **Use Simulation Time**.

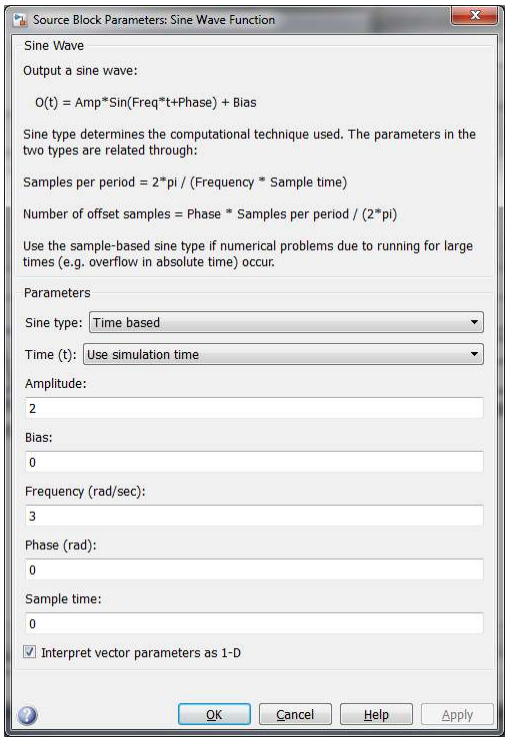

Figure 1.8: Parameters for the **Sine Wave** block. Select the amplitude and frequency desired.

- 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 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 1.9.

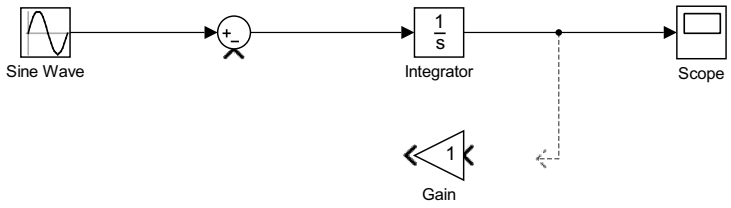

Figure 1.9: Add a node by right-clicking one the line and dragging to the input of a block.

- 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 1.10.

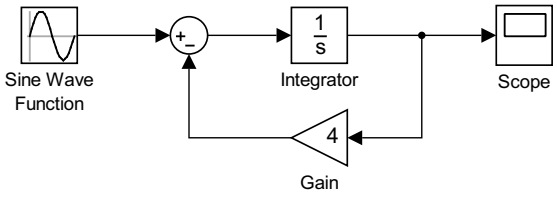

Figure 1.10: Connections for First Order ODE model for $\frac{dx}{dt} = 2\sin 3t - 4x$.

- 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.

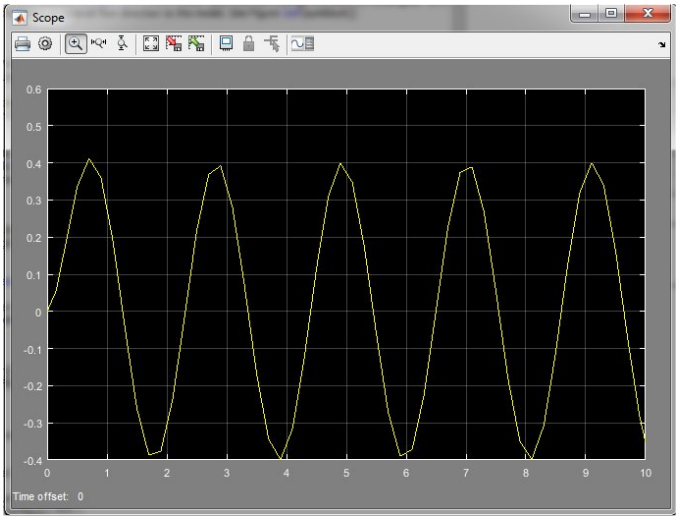

Figure 1.11: **Scope** plot of the solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$, with **Refine Factor**$= 1$.

- Double-click the **Scope** to see the solution. Figure 1.11 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 1.12-1.13. In particular, changing the **Refine Factor** can lead to smoother solutions. The solution shown in Figure 1.11 had a setting of 1 and that in Figure 1.14 is the result of setting the **Refine Factor** to 10.

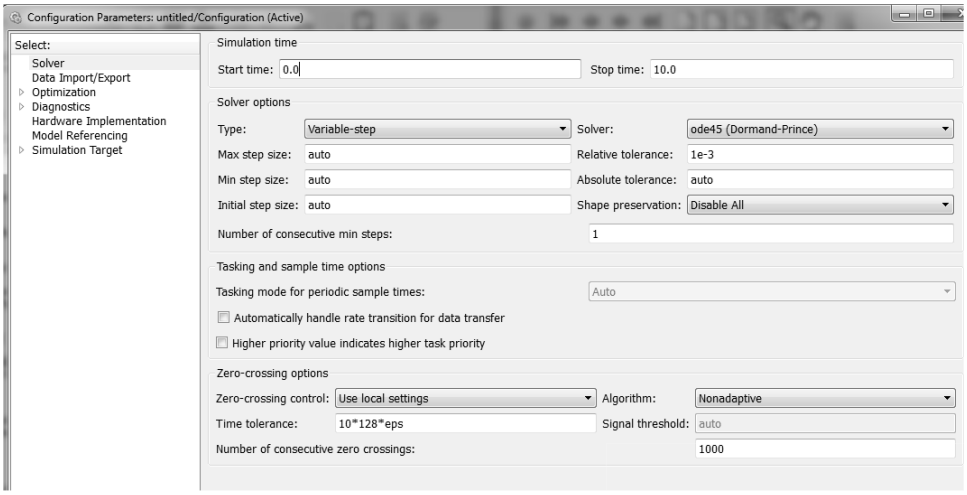

Figure 1.12: System Configuration Parameters.

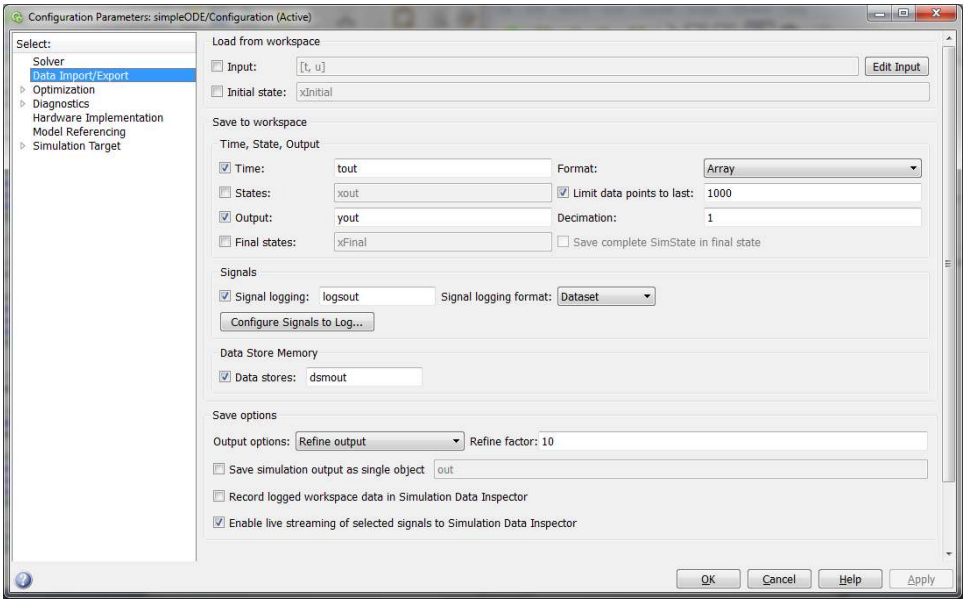

Figure 1.13: Configure Data Import/Export Parameters. Changing the**Refine Factor** can lead to smoother solutions.

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 1.15.

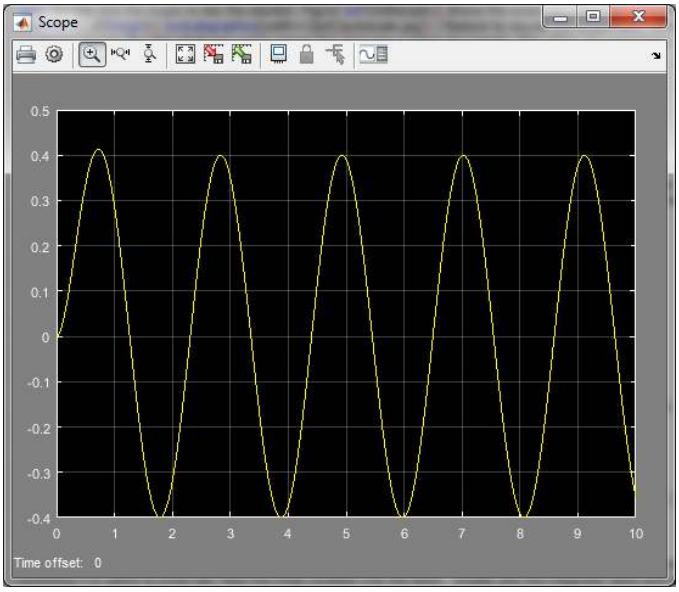

Figure 1.14: **Scope** plot of the solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$, with **Refine Factor**= 10.

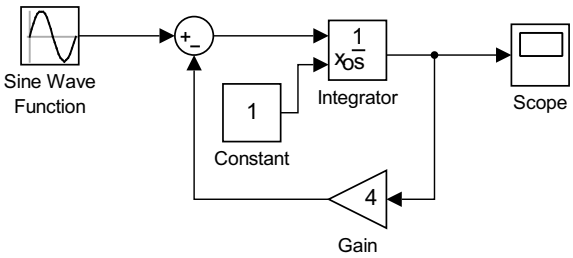

Figure 1.15: Connections for the First Order ODE model for $\frac{dx}{dt} = 2\sin 3t - 4x$ showing how to provide an external initial value.

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.

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

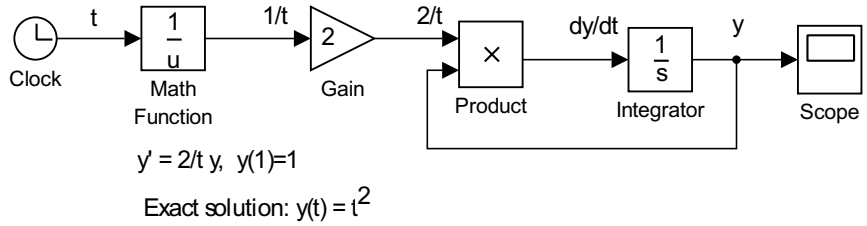

Figure 1.16: First order separable differential equation model.

We can set up the problem in Simulink as shown in Figure 1.16 for the initial value problem

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

where $y(1) = 1$. Running the simulation, we obtain the solution shown in Figure 1.17.

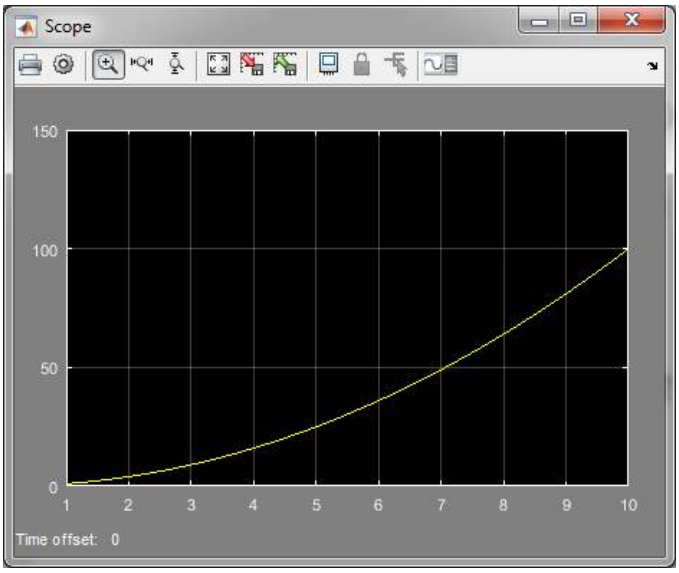

Figure 1.17: **Scope** plot of the solution of initial value problem (1.2), $\frac{dy}{dt} = \frac{2}{t}y$, where $y(1) = 1$.

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 1.18.

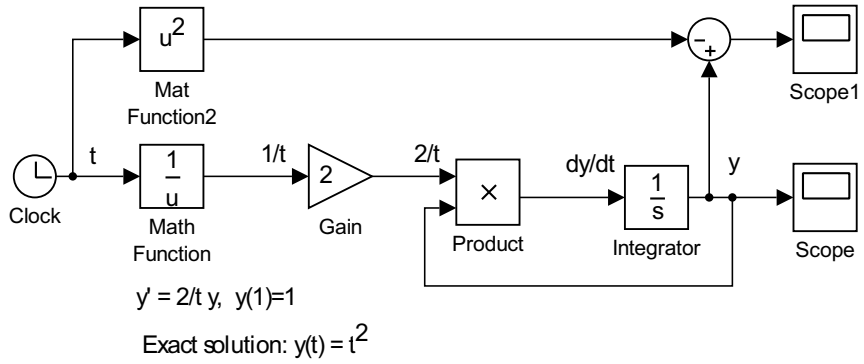

Figure 1.18: First order separable differential equation model with extra blocks to plot the difference between the numerical and exact solution, $y(t) - t^2$, for Equation (1.2).

The result of the simulation is shown in Figure 1.19. 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.

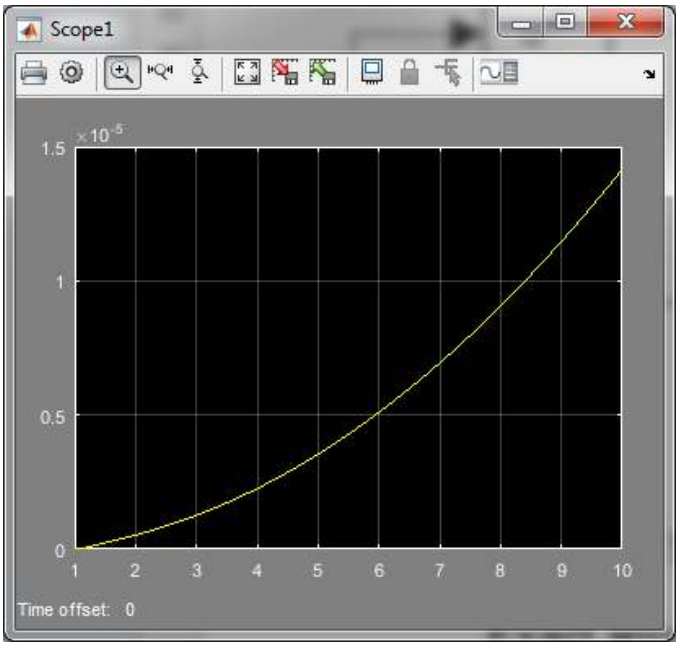

Figure 1.19: **Scope** plot of the difference between the numerical and exact solution, $y(t) - t^2$, for Equation (1.2)

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

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

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 $y(t) = t^3$.

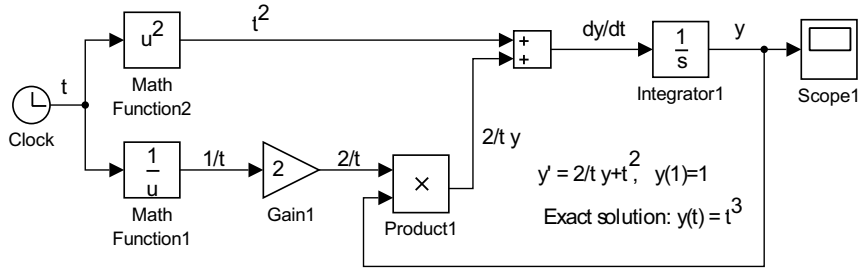

Figure 1.20: Linear first order differential equation model.

The model for this problem is shown in Figure 1.20. Running the simulation, we obtain the numerical solution, $y(t) = t^3$, as shown in Figure 1.21. 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. The integrating factor is found to be

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

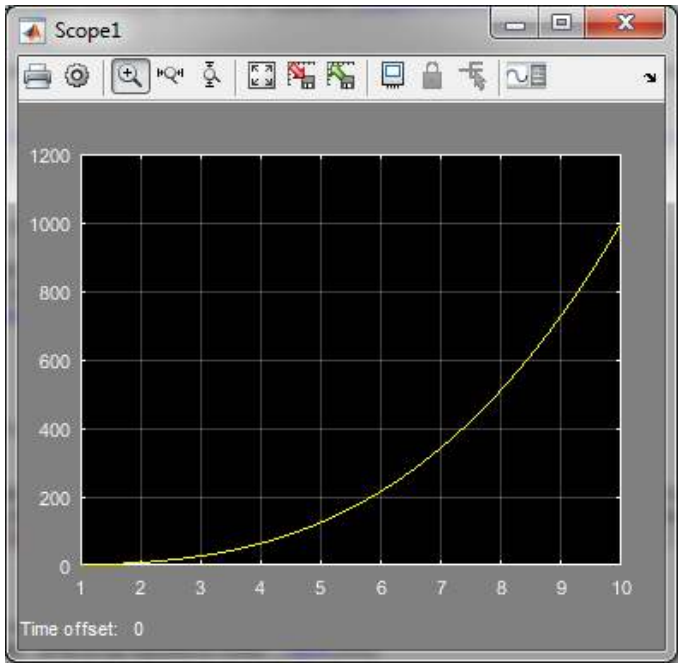

Figure 1.21: **Scope** plot of the difference between the numerical and exact solution, $y(t) - t^2$.

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

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

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

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

The solution can be found using Simulink. The model for this is shown in Figure 1.22. The plot on the scope matches that in Figure 1.11.

1.3 Working with Simulink Output

Often we might want to access the solutions in MATLAB. Using the model in Figure 1.16, add the **To Workspace** block. Double-click and rename the variable as y and change the output type to array. Run the simulation. 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

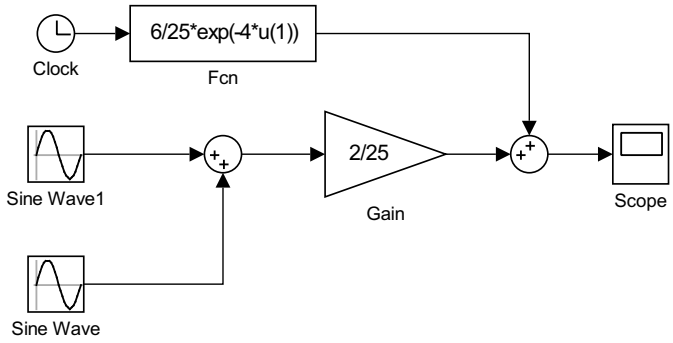

Figure 1.22: Model for plotting the exact solution (1.8 of the initial value problem (1.6.

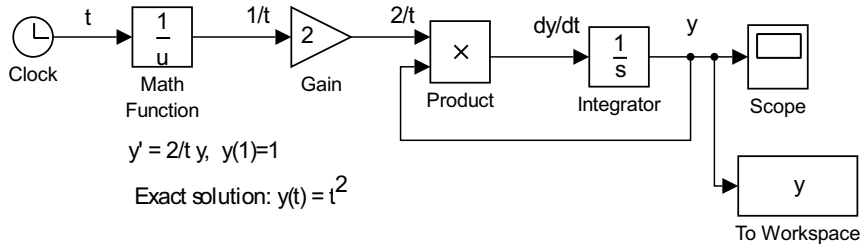

Figure 1.23: Adding To Workspace block for sending output to MATLAB.

set(gcf,'Color',[1,1,1]) makes the plot background white. The result is shown in Figure 1.24.

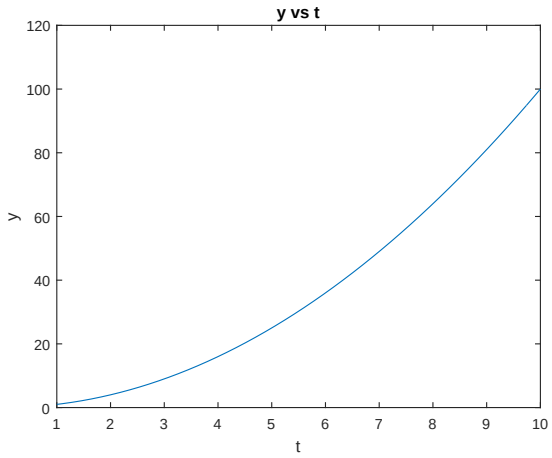

Figure 1.24: Plot of model solution in MATLAB.

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 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.

Method 1:

Select the **Scope** figure window in Figure 1.25, then hit **ALT+PrintScrn** to copy the figure to a clipboard and paste the figure into your application.

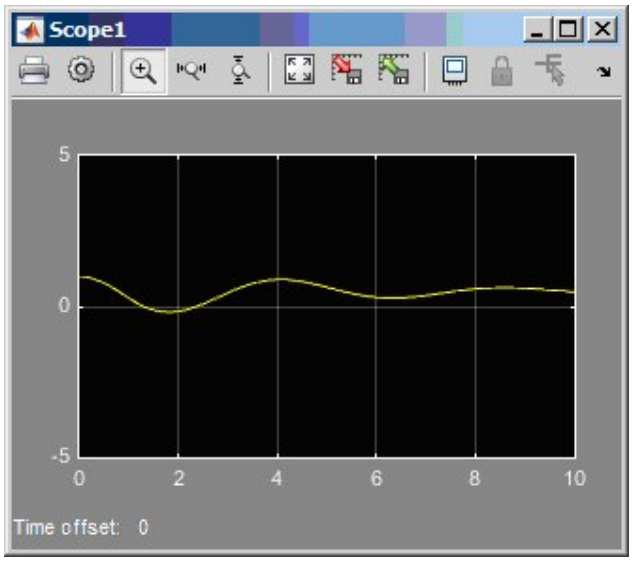

Figure 1.25: **Scope** plot. Note that the plots in this section are generated by the oscillator model in the next chapter.

You might want to change the colors before copying the scope image. Click the **Scope Parameters** icon (2nd icon) and go to the **Style** tab as seen in Figure 1.26. 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 1.26.

Now the **Scope** plot looks like Figure 1.27.

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 1.29 which can be manipulated

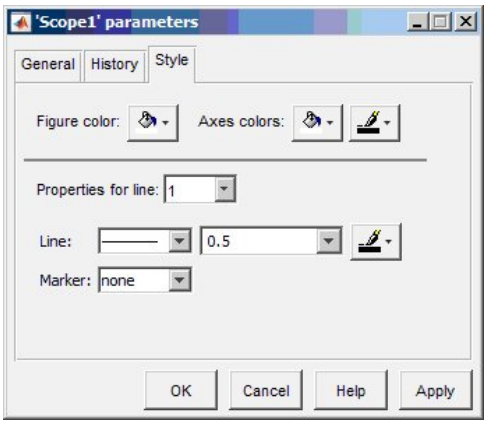

Figure 1.26: **Scope** color parameters.

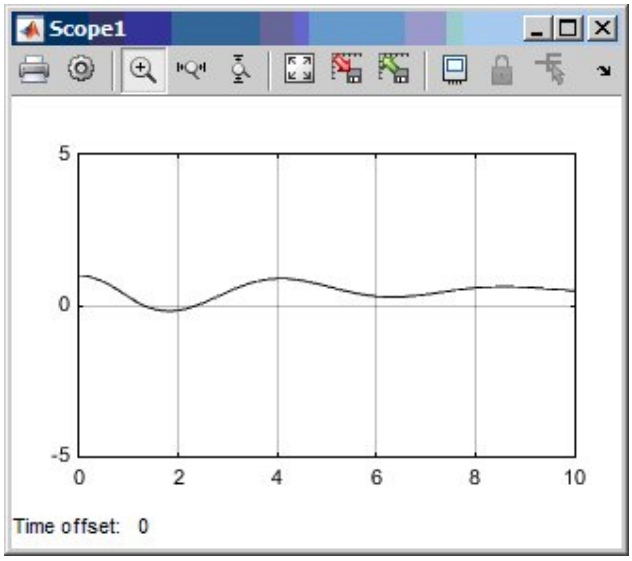

Figure 1.27: **Scope** plot with a white background.

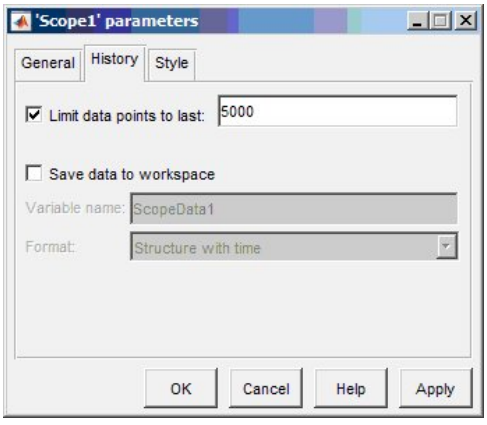

Figure 1.28: History tab in **Scope parameters**.

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 MAT-

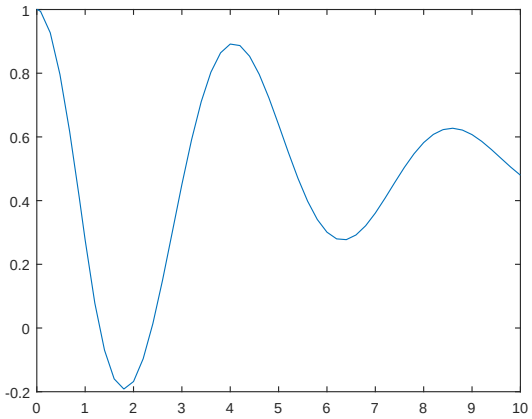

Figure 1.29: Plot generated by Method 2.

LAB 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**.

```
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.

Now you can Insert the figure into your MS Word document as a Picture file.

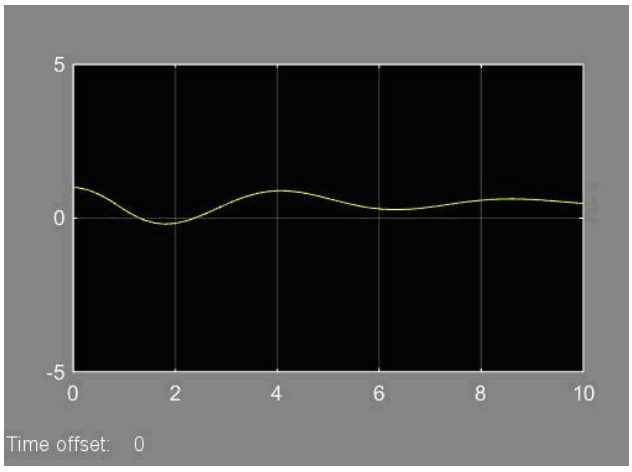

Figure 1.30: Scope plot from Method 3.

Method 4:

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. Connect this block to the input to 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 1.31.

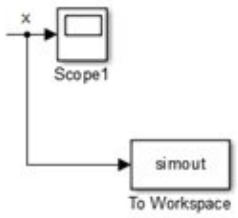

Figure 1.31: Use of a **To Workspace** block.

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.

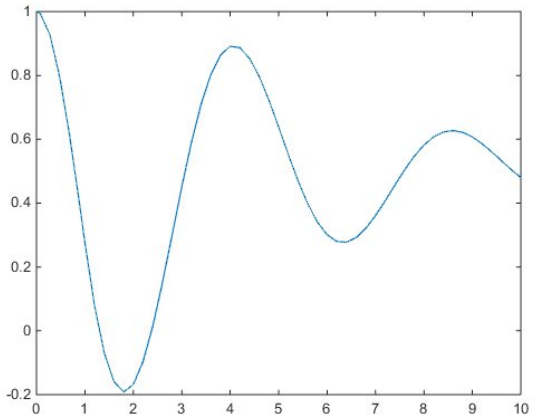

Figure 1.32: **Scope** plot from Method 4.

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.

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 computation similar to MATLAB. Xcos is a graphical design environment. The Xcos environment is shown in Figure 1.33.

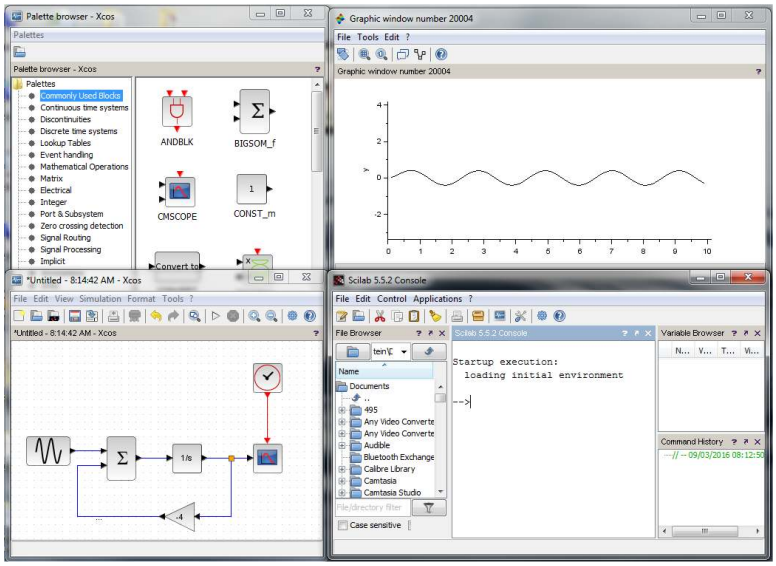

Figure 1.33: The Xcos workspace.

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 1.34 and 1.35. This looks similar to Simulink's Library Browser as shown in Figure 1.1.

In Figure 1.36 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 1.4. 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 to inputs, the first is positive and the second is negative.

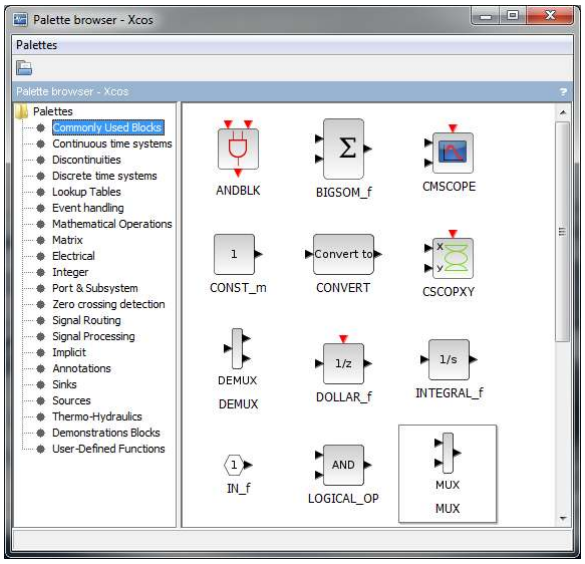

Figure 1.34: The Xcos Palette browser.

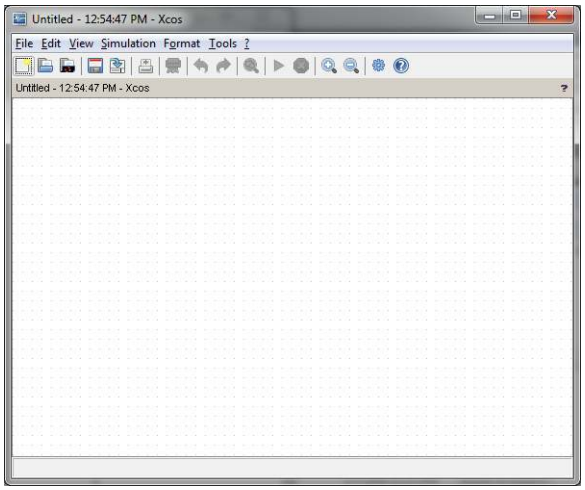

Figure 1.35: The Xcos workspace.

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.

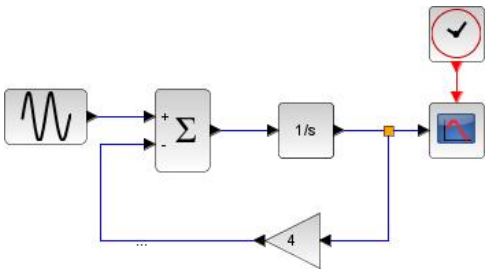

Figure 1.36: The Xcos model for solving the first order ODE $\frac{dx}{dt} = 2\sin 3t - 4x$.

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 1.37.

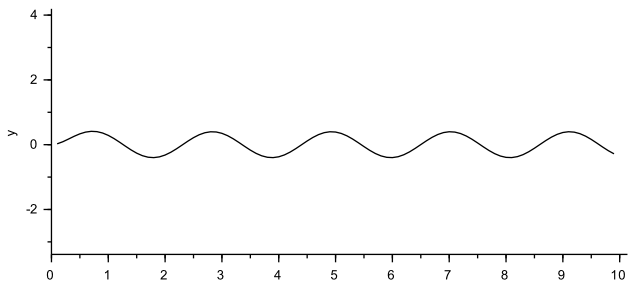

Figure 1.37: The Xcos model solution of $\frac{dx}{dt} = 2\sin 3t - 4x$, $x(0) = 0$.

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 1.38.

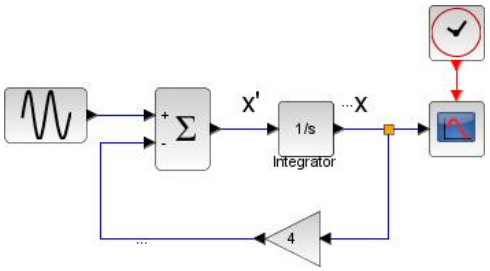

Figure 1.38: The Xcos model with annotation added.

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 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');
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 1.39 shows the solution plot.

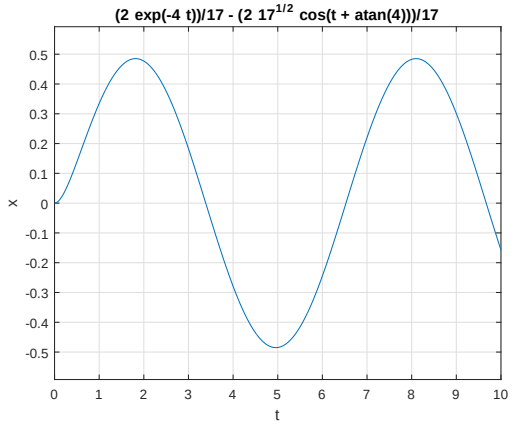

Figure 1.39: The solution of Equation (1.1) with $x(0) = 0$ found using MATLAB's **dsolve** command.

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')
```

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 code produces Figure 1.40.

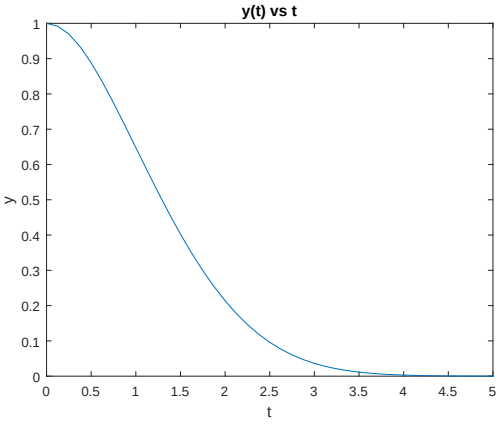

Figure 1.40: A plot of the solution of $\frac{dy}{dt} = -\frac{yt}{\sqrt{2-y^2}}, \quad y(0) = 1$, found using MATLAB's **ode45** command.

One can also use **ode45** to solve higher order differential equations. Second order differential equations are discussed in Section 4. See MATLAB help for other examples and other ODE solvers.

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 isdy/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 1.41.

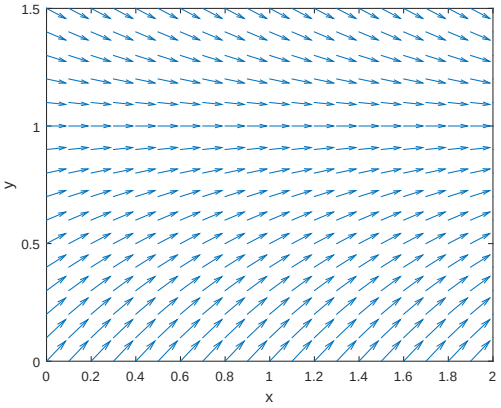

Figure 1.41: A direction field produced using MATLAB's **quiver** function for $y' = 1 - y$.

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 1.42

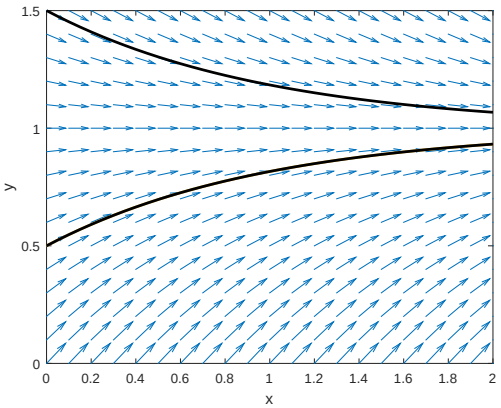

Figure 1.42: A direction field produced using MATLAB's **quiver** function for $y' = 1 - y$ with solution curves added.

1.7 Exercises

1. Construct the model in Figure 1.4 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 (See Section 1.6) and Simulink. Are the solutions the same? Provide plots of the solutions.

   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:

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