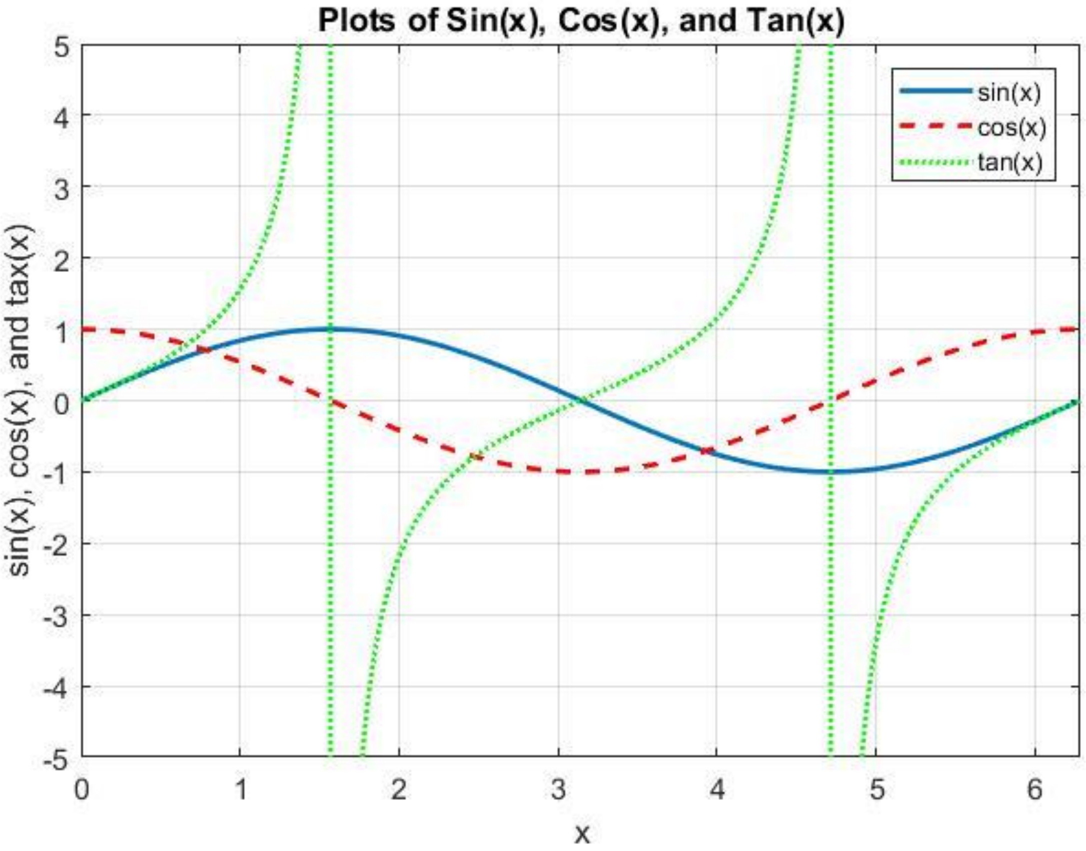
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
First Year Program – Core 8 and TrackOne

FIRST YEAR PROGRAM
ENGINEERING PROBLEM SOLVING LABS

MAT188: Laboratory #3
Two-Dimensional Plots



TWO-DIMENSIONAL PLOTS

In this lab, you will learn more about two-dimensional plotting and how to manipulate them to clearly present and interpret data. In particular, you will work through a problem focused on the motion of projectiles and how to present this motion for different initial conditions.

Learning Outcomes

By the end of this lab you should…

1) Understand how to create 2D figures with multiple plots,
2) Adjust the basic properties and labels of 2D figures, and
3) Use logical indexing to redefine specific values of a vector.

Preparation *(Required doing* ***before*** *you come to the laboratory session)*

1) If you have not already done so in a previous term, complete Sections 12 through 14 of **MATLAB Onramp** online exercise at https://matlabacademy.mathworks.com (i.e., *Section 12: Logical Arrays* to *Section 14: Programming*). This should take you less than an hour. Remember your login information for the Mathworks account you create to complete the Onramp, since you will be asked to log in during the lab to show your Lab TA your Progress Report at the start of the lab session. **Make sure you are completing the MATLAB Onramp using version R2017a.**
2) Read through the entire lab handout.

Related Reference Materials *(Not required, but may be a helpful resource)*

Video demonstration, *Using Basic Plotting Functions (5:52)*
http://www.mathworks.com/videos/using-basic-plotting-functions-69018.html

Review – Lab #1-2

In Labs 1-2, you learned how to define vectors and use MATLAB's built-in functions to calculate their values at points specified within a given vector. You also used the `plot` command to visualize functions.

At this point, you should be comfortable with using MATLAB as a "super" graphing scientific calculator. You should be able to evaluate expressions with the standard set of built-in functions. You should understand the basics of working with vectors, and how to create simple two-dimensional plots.

For example, to calculate and plot the expression $(e^x + 1)^2$, for the values of *x* ranging from -5 to 15:

```
>> x=linspace(-5,15,101) % x has 101 elements equally spaced from -5 to 15
>> y=(exp(x)+1).^2  % the .^ operator is used to square each element
>> plot(x,y,'b--') % This plots the function with a blue dashed line
>> grid on % We can create a grid so it is easier to interpret the figure
>> xlabel('x');ylabel('y(x)') % Labels can be added to the x and y axes
>> title('A Plot of the Function y(x)=(e^x+1)^2') % A title can also be added
```

MATLAB Skills and Knowledge: Creating and Manipulating 2D Plots

Plotting Multiple Functions in a Single Figure

As you read through this lab, type each of the example commands into the MATLAB command window to work along and see the results immediately.

Create a plot of the sine and cosine functions over a single period (2π radians) on the same figure, similar to Figure 1. ***Hint*:** use the `linspace` and `plot` functions.

You can create multiple plots with a single use of the `plot` function: `plot(x,y1,x,y2)`, where `x` is the shared domain and `y1` and `y2` are two vectors of the function values.

After the plot is created, add gridlines for better comparison of the two graphs.

```
>> grid on
```

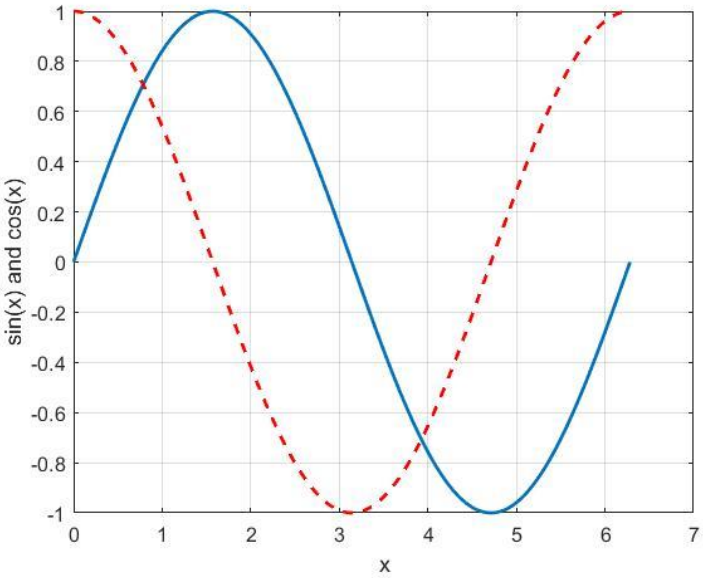


Figure 1. Plot of `sin(x)` and `cos(x)`

Adding Titles, Axis Labels, and Legends

We will now add the tangent function to the current plot of the sine and cosine functions. First, calculate the tangent of the values in the independent variable vector `x`:

```
>> y3 = tan(x);
```

Now let's add the plot of `y3` versus `x` to the previous figure. To prevent losing the previous graphs, tell MATLAB to keep them using `hold all`. Otherwise, the existing plots would be replaced by the new plot.

```
>> hold all
```

Now, any new plots will be superimposed on the existing plots in the figure. Add the new plot:

```
>> plot(x,y3)
```

If you ever need to erase existing plots and create a new one, you can use `hold off` to tell MATLAB not to preserve existing plots. Alternatively the `clf` function will clear the current figure window.

If you ever need to create a new figure, keeping existing plots and continuing work on a new canvas, you can use the `figure` command. This creates a white canvas ready for you to plot on, and anything you plot will go to this active plot window.

Look carefully at the figure. *Is there a problem with it? Does this figure clearly represent the differences between these three functions?*

The range of `tan(x)` is much larger than that of `sin(x)` and `cos(x)`, so MATLAB automatically adjusts the axis limits to show the entire graph. To fix this, let us manually define the limits of the axes to better represent the graphs.

Use `axis` (see `help` if needed) to change the axis limits so the `x`-axis ranges from $[0, 2\pi]$ and the `y`-axis ranges from $[-5, 5]$, as shown in Figure 2. This way you can compare all three functions together and observe the changes within a narrow range.

```
  >> axis([0 2*pi -5 5])
```

To increase readability of the plots, add titles, axis labels, and a legend. Note the use of a new data type called a ***string***, which is a piece of text indicated by enclosing quotation marks (`''`).

```
>> title('Sin(x), Cos(x), and Tan(x)')
>> xlabel('x')
>> ylabel('sin(x), cos(x), and tan(x)')
>> legend('Sin(x)','Cos(x)','Tan(x)')
```

Note how `legend` assigns labels in the same order as you plotted the functions in.

The results are illustrated in Figure 2. These parts of the plot can also be changed using the plot tools in the figure window. With the cursor in pointer mode (the white arrow) you can double click on the lines, title, axes labels, or the legend to bring up the plot tool editor. ***Try this now.***

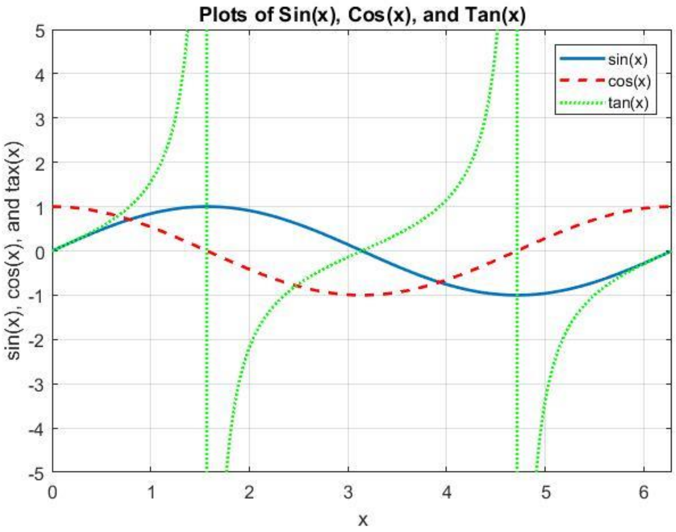


Figure 2. Adjusting the axes and adding a title and axis labels

Changing Line Properties

Controlling line properties helps you to better present figures. Type `help plot` in the command window (or search the documentation for `plot`) to familiarize yourself with the available line properties.

Reference: See the MATLAB Summary, Table 5 for a summary of key plotting commands.

Exercise:
Consider the equations for the x (distance) and y (height) positon of a projectile that is launched from a height $y(t=0) = y_0$, at an angle θ, with an initial velocity v_0:

$$x(t) = (v_0 \cos\theta)t \quad \text{and} \quad y(t) = -\frac{1}{2}gt^2 + (v_0 \sin\theta)t + y_0 \qquad g = 9.81 \text{ m/s}^2$$

Create a script that generates a single figure with three plots that represent the motion of a projectile launched from the ground ($y = 0$ m) with launch angles $\theta = 15^\circ$, 35°, and 50°. You can chose your own initial velocity and appropriate time range, but these should be the same for all three launch angles. Your figure should include the full arc of the projectile, so you need to include the point at which it returns to the ground. Use *logical indexing* (which you learned about in Section 12.3 of *MATLAB Onramp*), to ensure that values of $y(t)$ do not include negative numbers. In this way your plot will more closely represent the physical reality. Include a title, axis labels, and a legend, and make use of three different line styles and colours. Save and run your script.