

# A Brief Mathematica Tutorial

## Part 2: Functions of a Single Variable and Graphing

Michael Penna, Indiana University – Purdue University, Indianapolis

### Objective

In this project we discuss how to define and graph functions in Mathematica.

### Narrative

In Mathematica, the function  $f = f(x)$  is defined by the command:

`f[x_] := <expression in x>`

Mathematica uses square brackets [ and ] for the arguments of functions, and the underscore \_ after the  $x$  in the definition “`f[x_]`” of  $f$  (as we will see) indicates that the definition applies to any variable, not just  $x$ . Further:

<code>Plot[f[x], {x,a,b}]</code>	graphs $f$ over the interval $[a, b]$ ,
<code>Plot[f[x], {x,a,b}, PlotRange -&gt; {c,d}]</code>	graphs $f$ over the interval $[a, b]$ , restricting $y$ -values to the interval $[c, d]$ , and
<code>Plot[{f[x], g[x]}, {x,a,b}]</code>	graphs $f$ and $g$ over the interval $[a, b]$ .

In addition, there are various options to the **Plot** command (some of which are illustrated below).

### Task

1. Type the command lines in the left-hand column below into Mathematica in the order in which they are listed. They draw several versions of the graph of  $f(x) = \sin x$ . The effect of each command is described in the right-hand column for your reference.

<code>In[1] := (* Your name, today's date *)</code>	
<code>In[2] := (* Functions of a Single Variable and Graphing *)</code>	
<code>In[3] := f[x_] := Sin[x]</code>	Let $f(x) = \sin x$ .
<code>In[4] := f[Pi/2]</code>	Evaluate $f(\pi/2)$ .
<code>In[5] := Plot[f[x], {x,0,2Pi}]</code>	Graph $f$ over the interval $[0, 2\pi]$ .
<code>In[6] := Plot[f[x], {x,-500,500}]</code>	Graph $f$ over the interval $[-500, 500]$ .
<code>In[7] := Plot[f[x], {x,-1,1}]</code>	Graph $f$ over the interval $[-1, 1]$ .
<code>In[8] := Plot[f[x], {x,-0.1,0.1}]</code>	Graph $f$ over the interval $[-0.1, 0.1]$ .

Note that the units for the arguments of trig functions are radians in Mathematica, and that the size of the domain makes a big difference in what we see: graphing  $f$  over the interval  $[0, 2\pi]$  gives what we expect, but graphing over the intervals  $[-500, 500]$  and  $[-1, 1]$  might not. Note also that over the interval  $[-1, 1]$  the graph of  $f$  looks almost linear, and over the interval  $[-0.1, 0.1]$  it looks even more so.

2. Continue by typing the command line below into Mathematica. It draws the graph of  $f(x) = \tan x$ .

`In[9] := Plot[Tan[x], {x,-3,3}]`

Note the vertical lines at the odd multiples of  $\pi/2$  in this graphic. They aren't really part of the graph of  $f(x) = \tan x$ , but Mathematica draws them anyway. (Mathematica does this, in fact, with discontinuous functions in general.)

3. Continue by typing the command lines below into Mathematica. The first illustrates how we can plot more than one function in one graphic, and the second illustrates how we can introduce color.

```
In[10] := Plot[{Sin[x], Sin[2x], Sin[3x]}, {x,0,2Pi}]
In[11] := Plot[{Sin[x], Sin[2x], Sin[3x]}, {x,0,2Pi},
  PlotStyle -> {RGBColor[1,0,0], RGBColor[0,1,0], RGBColor[0,0,1]}]
```

4. Use the commands you've learned in this project to graph a function of our own choice. (Be creative!)

At this time, make a hard-copy of your typed input and Mathematica's responses. Then, in each of the graphics you created, label by hand the graph of each function. (In the first graphic you created, for example, label the graph of  $f$  by " $f(x) = \sin x$ ".)

Your lab report will be a hard copy of your typed input and Mathematica's responses (both text and hand-labeled graphics).

### Comments

1. A variable must be unassigned for it to be used in the definition of a function. The following example illustrates what happens when an assigned variable is used in the definition of a function:

```
In[1] := x=5
Out[1]= 5
In[2] := f[x_] := x^2
Out[2]= 25
```

2. The underscore after the  $x$  in the definition " $f[x_]$ " of  $f$  indicates that the definition applies to any variable, not just  $x$ . Thus:

```
In[1] := f[x_] := x^2
Out[1]= x^2
In[2] := f[y]
Out[2]= y^2
```

3. Note that when using a function such as  $\sin x$  in Mathematica you must type "**Sin[x]**" (being careful to include brackets) rather than "**Sin x**" (or even "**sin x**") which you might write by hand.
4. If you want to write  $\sin^2 x$  in Mathematica then you may write "**Sin[x]^2**" or "**(Sin[x])^2**", but you may not write "**Sin^2[x]**".
5. Mathematica allows you to control numerous aspects of the appearance of graphics via "options" such as color. While there are many advantages to using different colors when viewing a graphic, a potential disadvantage to using certain colors is that they don't always come out well when you print them. Since this issue is hardware dependent, try the code in the projects to follow as written, but if your graphics don't come out well when you print them, try different colors.
6. We illustrated one way to draw the graphs of several functions. Another way to do the same thing — and, in fact, to more generally combine several graphics into one — is illustrated by the following code:

```
In[1] := Plot1 = Plot[Sin[x], {x,0,2Pi}]
In[2] := Plot2 = Plot[Sin[2x], {x,0,2Pi}]
In[3] := Plot3 = Plot[Sin[3x], {x,0,2Pi}]
In[4] := Show[{Plot1, Plot2, Plot3}]
```

(This is sometimes referred to as *delayed* plotting.)

7. Even though the graphs of curves may appear to be smooth, the way Mathematica graphs functions is by plotting a finite number of points and “connecting-the-dots” with short line segments. Thus, rather than seeing a smooth curve when you plot a function such as  $\sin x$ , you are actually seeing a polygonal approximation to its graph.
8. In addition to the Mathematica functions we discussed in this and the previous project, which are always available when Mathematica is running, there are other functions that are defined in “Mathematica packages” and which are loaded and available for your use, only when you explicitly load them. For example, in addition to Mathematica’s **Plot** command, there is an **ImplicitPlot** command in Mathematica’s **Graphics** package. More will be said about implicitly defined functions later in this course, but for now, the following code illustrates how we load and use the **ImplicitPlot** command:

```
In[1] := <<Graphics`ImplicitPlot`           Load ImplicitPlot from the Graphics package.  
In[2] := ImplicitPlot[x2+y2=1, {x,-2,2}]   Graph the equation  $x^2 + y^2 = 1$  for  $x \in [-2, 2]$ 
```