Math 1191
Mathematica Introduction

Lab 3

Fall, 2005

Loading Packages

Mathematica has many packages—collections of definitions of commands and constants grouped around a common theme—that are not ordinarily loaded into memory when you start the program. One example is the `Graphics` package. The definitions in this package are loaded with the command `<<Graphics‘`.

At times you may only want a particular subset of a package rather than all of the commands/definitions. For example, to be able to use color names for colors, you need to load `<<Graphics‘Colors‘`.

If you use a package in your notebook, you should load it at the top. Then when you run the notebook again, you will know what packages you need.

Other packages can be found in the Help browser for Mathematica (under ‘Add-Ons’). Some examples of available packages include:

- `Algebra‘InequalitySolve‘` — for solving linear inequalities
- `Calculus‘Limit‘` — improves on the built-in `Limit` function
- `NumericalMath‘NLimit‘` — computes limits numerically
- `DiscreteMath‘Combinatorica‘` — many functions for use in discrete mathematics (combinatorics and number theory)
- `NumericalMath‘PolynomialFit‘` — construct a polynomial fit to given data
- `Statistics‘` — lots of statistical functions (includes many sub-packages that could be loaded independently)
- `LinearAlgebra‘` — linear algebra functions

Plotting Functions

The following command will plot the function $y = 3x + 2$ from $x = -1$ to $x = 4$:

```
In[1]:= Plot[3x + 2,{x,-1,4}]
```

The `Plot[]` function has two required arguments: the first is the function that is to be plotted and the second is a list containing the variable and its range over which the function should be plotted.

By default, Mathematica will try to set up the plot range (the y-values) so that the entire function is visible over the variable range (the x-values). However, Mathematica does not always set up tick marks on the vertical axis in a way that you might expect. Consider these examples:

```
In[2]:= Plot[ArcSin[x]-Exp[x],{x,-1,1}]
In[3]:= Plot[ArcSin[x]-Exp[3x],{x,-1,1}]
```

In both cases, the placement of the horizontal axis may mislead you about the range on the vertical axis.

To control the range, use the `PlotRange->` option:

```
In[4]:= Plot[ArcSin[x]-Exp[3x],{x,-1,1},PlotRange->{-4,4}]
```

Plots are *graphics objects*. As such, they can be named and stored for later use. To display a plot that has already been constructed, use `Show[]`. Note the difference between using immediate and delayed assignments!

```
In[5]:= f[x_]:= 4x+2
In[6]:= plot1 = Plot[f[x],{x,-2,2}]
         plot2 := Plot[f[x],{x,-2,2}] In[7]:= Show[plot1]
In[8]:= Show[plot2]
```

What will happen if you change the definition of f and then re-run the `Show[plot1]` and `Show[plot2]` commands?

In general: when defining plots for later display, you will probably want to use *immediate* assignment so that the display will not change without you knowing it.

Plotting Options

There are several options that you can use with either the `Plot[]` or `Show[]` functions. Each is listed as an argument in the function, placed after the variable range, and has the form *Option*`->` *option value*. Some options and examples:

- `AspectRatio->n` (where n is a number) specifies height to width of the graph. The default is to make the graph almost twice as wide as it is high, which can make circles a little skewed. Set the aspect ratio to 1 to make the width and height of the graph equal.
- `AxesLabel->{"x axis label","y axis label"}` sets the axes labels. The labels need to be given in quotation marks.
- `Frame->True` draws a box around the plot
- `FrameLabel->{"bottom label","left label", "top label", "right label"}` labels outside of the frame. The quotation marks are required. The last two labels are optional.
- `PlotLabel->{"title"}` places the title of the plot above the plot
- `DisplayFunction->`*`option`* suppresses or activates display of the plot; use option `Identity` to suppress display and option `$DisplayFunction` to reset the display.
- `PlotStyle->{...}` changes plot styles for the curves, such as the thickness of the lines, if the lines are dashed or connected (and how the dashes are displayed), and the color of the lines. This option is only available with the `Plot` function. Some examples:

```
In[9]:= Plot[Cos[x],{x,-Pi,Pi},PlotStyle->{Thickness[0.02]}]
In[10]:= Plot[4x - 2,{x,-2,2},PlotStyle->{Dashing[{.05,.05}],Blue}]
```

  (Note that you need to have the `Graphics‘Colors‘` package loaded to use the color name.)

Multiple Plots

Multiple functions can be displayed on a single pair of axes in several ways:

```
In[11]:= plot1 = Plot[Sin[x],x,-Pi,Pi,PlotStyle->Blue];
         plot2 = Plot[Sin[2x],x,-Pi,Pi,PlotStyle->Red];
         plot3 = Plot[Sin[3x],x,-Pi,Pi,PlotStyle->Green];
In[12]:= Show[plot1,plot2,plot3]
```

or:

```
In[12]:= listoplots = {plot1,plot2,plot3}
In[13]:= Show[listoplots]
```

or:

```
In[14]:= Plot[ {Sin[x], Sin[2x], Sin[3x]}, {x,-Pi,Pi},
         PlotStyle->{Blue, Red, Green}]
```

Note that if you use this last option and want to use several different styles, you will need to put each plot's style in a separate list:

```
In[15]:= Plot[{Exp[-x],Exp[x]},{x,-2,2},
       PlotStyle->{Blue,{Red,Dashing[{.02,.02}]}}]
```

Another point: when defining the plots objects, the plot was displayed (even when we used a semicolon to end the command). To *not* display the plot, use the plot option `DisplayFunction->Identity`. To *redisplay* the plot in a `Show[]` command, use the option `DisplayFunction->$DisplayFunction`.

Plotting multiple functions can be useful if you are plotting discontinuous piecewise-defined functions. For example, to plot the function

$$f(x) = \begin{cases} x^2 - 6 & \text{if } x < 1 \\ -x^2 + 3 & \text{if } x \geq 1 \end{cases}$$

you could define first define f with `Which[]`:

```
In[16]:= f[x_]:= Which[x < 1, x^2 - 6, x >= 1, -x^2 + 3]
```

then define two plots:

```
In[17]:= leftplot = Plot[f[x], {x, -3,1},DisplayFunction->Identity]
       rightplot = Plot[f[x], {x,1,5},DisplayFunction->Identity]
       Show[leftplot,rightplot,DisplayFunction->$DisplayFunction]
```

You may also want to place plots in an array. Use `GraphicsArray[]` for this:

```
In[18]:= Show[GraphicsArray[{plot1,plot2,plot3}]]
In[19]:= Show[GraphicsArray[{{plot1,plot2}, {plot3}}]]
In[20]:= Show[GraphicsArray[{{plot1},{plot2},{plot3}}]
```

Parametric and Polar Plots

To plot the ellipse defined by the parametric equations $x(t) = 3\cos(t)$, $y(t) = 2\sin(t)$, use `ParametricPlot`:

```
In[21]:= ParametricPlot[{3 Cos[t], 2 Sin[t]},{t,0,2 Pi}]
```

And the polar equation $r(\theta) = 3\sin(5\theta)$ is plotted with

```
[In[22]:= PolarPlot[{3 Sin[5 t]}, {t,0,2 Pi}]
```

The plot options can be used with the parametric and polar plot commands as well.

Note that the `PolarPlot[]` function is only accessible through the `Graphics‘` package—you need to load it first before you can use the `PolarPlot[]` command.

Assignment

1. Create a plot of the function $y = 8x^5 - 4x^3 + 8x^2 - x - 2$. Set the ranges so that the important features of the graph are displayed.

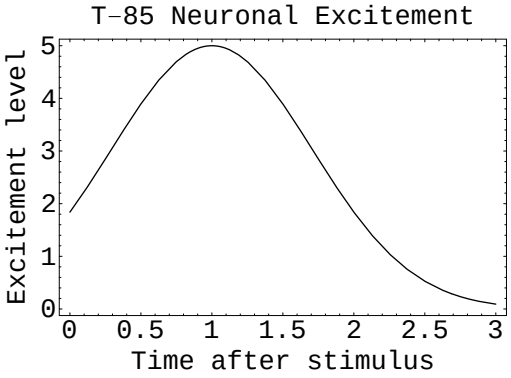
2. Create a single graph with the following functions on the interval $-2\pi < x < 2\pi$:

   $$y = \sin(2x) \qquad y = \cos(2x) \qquad y = 2\sin(x)$$

   Use different colors to distinguish the functions.

3. Redo the previous problem but this time distinguish the functions by changing the `Dashing` and `Thickness` values for each (make the first with dashing of {.04,.08} and thickness .02, the second with dashing of {.01,.01} and default thickness, and the third with no dashing but thickness of .01).

4. Recreate the following plot. (The function is $f(t) = 5e^{-(t-1)^2}$.)



5. Arrange the plots of the polar equations $r(\theta) = 2\cos(A\theta)$ for $A = 1, 2, 3$ and 4 in a 2×2 array.

6. Plot the function

   $$f(x) = \begin{cases} 4x - 2 & \text{if } x < 0 \\ x^2 & \text{if } 0 \le x \le 4 \\ -x + 2 & \text{if } x > 4 \end{cases}$$

   over the range of $x = -2$ to $x = 10$. (The graph should be discontinuous.)