Mathematica and You

December 1, 1994

Why Mathematica?

Mathematica is a high-level...

- interactive calculator
- 1D and 2D plotting package
- symbolic calculator
- programming language

$\Rightarrow$ a very effective tool

Goal of tutorial is to:

- introduce syntax
- basic concepts and programming
- ideas on how to **best** use it

Basic Syntax

Basic syntax in mathematica is "statement" oriented

- Interactively, a carriage return delimits a command, e.g.

```
In[5]:=  Plot[Sin[x], {x, 0, 2Pi}]
```

```
...output goes to X-window
```

```
In[6]:=  Integrate[Sin[x] Cos[x], x]

                  2
          -Cos[x]
Out[6]=  --------
             2
```

- When programming, use ";" to delimit statements
  - Suppresses any "output" from command
  - Clearly delimits statements

More on syntax

Other points on syntax:

- Square brackets ("[. . .]") delimit arguments to commands
- Curly parentheses ("{. . .}") delimit lists
- Mathematica is case-sensitive!
- **ALL** built-in commands are capitalized

```
In[4]:= Integrate[ x^2, {x, -5, 5}]

        250
Out[4]= ---
         3
```

Mathematica remembers **ALL** symbols created

- Any string it doesn't recognize is a "new" symbol
- Doesn't like same symbols being used inconsistently
- Good idea to clear any previous definitions first. . .

```
Clear[Hist,x]
Hist[x_] := .....
```

And some more syntax...

You can make assignments and definitions in **2** ways:

1. Use "=" symbol

```
In[4]:= Clear[f,x]
In[5]:= f[x_] = Sin[x] Cos[x]

Out[5]= Sin[x] Cos[x]
```

- immediately evaluates right-hand side
- Best for assignments

2. Better yet, use ":=" symbol

```
In[4]:= Clear[f,x]
In[5]:= f[x_] := Sin[x] Cos[x]
```

- Suppresses evaluation until f[x] is used
- Best to use for functions and other definitions

Usually "safer" to use ":=" . . .

Mathematica as a Calculator

One can use mathematica as an intelligent calculator

- Averaging...

```
In[2]:= <<Statistics`DescriptiveStatistics`

In[3]:= datapoints = {10.1, 12.0, 13.1, 9.3, 10.2};

In[4]:= Mean[datapoints]
Out[4]= 10.94

In[5]:= Variance[datapoints]
Out[5]= 2.433
```

- Solving roots...

```
Clear[f]
f[x_] := x^3 -3x
FindRoot[f[x] == 0, {x, 0}]
```

- Fourier decomposition...

```
     data = {-1, -1, -1, -1, 1, 1, 1, 1};
Fourier[data]
```

generates Fourier coefficients

$$b_s = \frac{1}{n}\sum_{i=1}^{n} a_i \exp(2\pi i(k-1)(s-1)/n)$$

Defining Your Own Functions

In many cases, you want to perform a set of commands

- best to group them into a single definition
- better to save them away for later use

Mathematica provides many ways of doing this

- Usual way is to create a **Module**

```
mycommand[x_, y_, ...] := Module[
                   {list of local variables},
                   command_1;
                   command_2;
                       .
                       .
                   final_result
                   ]
```

Can also use **Block** construction

- Differs from **Module** in subtle ways
- Usually, safest to use **Module** construct

Histogramming Function

As a concrete example, consider a "histogramming" function that:

- takes a list of data
- Histograms it into 10 bins between 0 and 100.
- displays overflows and underflows

It should be used as follows:

```
In[3]:= data = Table[Random[]*120-10, {i,1,100}]

Out[3]= {42.59, 15.9368, 83.7368, 53.7483, 43.5542, 29.6103,
>       31.2515, 39.5079, 34.8375, 68.3888, 48.2199, 88.3216,
>      -6.91025, 7.55149, -7.29719, 49.8615, -6.51958, .....

In[4]= Hist[data]
```

An Even More Complex Example

Look at a more complex example

- A function that numerically calculates solution to Laplace's equation in 2 dimensions

$$\vec{\nabla}^2\phi(x,y) = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = 0$$

- Uses Relaxation Method

- Regions defined by rectilinear boundaries

```
         BBBBBBBBBBBBBBBBBBBBB
         B                   B
         B                   B
         B                   B
         B                   B
         B                   B
         B                   B
         B                   B
Y        B                   B
         B                   B
         B                   B
         BBBBBBBBBBB         B
                   B         B
                   B         B
                   B         B
                   BBBBBBBBBBB


            X
```

Relax . . . continued

Problem can be specified by

- 2D table describing boundary
- Value of function on boundary
- size of overall grid

Define the following function:

```
RelaxSolve[
        phi_,
        boundary_,
        nx_, ny_,
        maxiterations_,
        tolerance_]
```

For example,

```
nx = 20; ny=30;
fcurrent = Table[0, {i,nx}, {j,ny}];

Do[
    fcurrent[[1,j]]     = N[ Sin[ (4 j Pi)/ny ] ];
    fcurrent[[nx,j]]    = N[ Sin[ (2 j Pi)/ny ] ],
    {j,ny}
    ]

(*  Now set up boundary function  *)
boundary = Table[0, {i,nx}, {j,ny}];

Do[
    boundary[[1,j]]=1;
    boundary[[nx,j]]=1,
    {j,ny}
    ]
Do[
    boundary[[i,1]]=1;
    boundary[[i,ny]]=1,
    {i,nx}
    ]
```

To generate solution, run the program:

```
In[3]:= g = RelaxSolve[fcurrent, boundary, nx, ny, 30, 0.05]

Out[3]= {{{0.406737, 0.743145, 0.951057, 0.994522, 0.866025,
>      0.207912, -0.207912, -0.587785, -0.866025, -0.994522,
>      -0.743145, -0.406737, 0, 0.406737, 0.743145, 0.951057,
   .....}}

In[4]:= ListPlot3D[g]

   .....X window graphic generated...
```

Summary

Mathematica can be a powerful tool

- Start by "playing" with it
- Use it for simple tasks initially
- Write down what works!
  - Syntax is a pain
  - Lots of ways of solving each problem

Start collecting a batch of useful tools

- Save them away and improve them
- Talk to your friends about problems

References are available

- Home-grown documentation ("How it All Works")
- *Mathematica, Second Edition*, S. Wolfram (Addison-Wesley, Don Mills, 1991).
- *Mathematica for the Sciences*, R. Crandall (Addison-Wesley, Don Mills, 1991).
- *Mathematica by Example*, M. Abell and J. Braselton, (Academic Press, Toronto, 1992).
- *Programming in Mathematica, Second Edition*, R. Maeder (Addison-Wesley, Toronto, 1991).