10 MATHEMATICA

10.1. Introduction

Mathematica is an immense computer software package for doing mathematical computation and exploration. It contains hundreds of mathematical functions, commands for producing graphics, and a complete programming language. The Mathematica system is a high-level computing environment including computer algebra, graphics and programming. At the basic level, it can be used as a scientific calculator; at more advanced levels, it incorporates all the features of classical programming languages such as PASCAL, LISP, MIRANDA, C, etc. It is particularly suitable for mathematics, as it incorporates symbolic manipulation and automates many mathematical operations.

This chapter introduces Mathematica by presenting several programs that investigate elementary, but interesting, mathematical problems.

10.2. Use of Mathematica

At UNIX level simply type Mathematica start the notebook version of Mathematica. Typing math starts the line version. In the Notebook version we can read in a file by pulling down File to Open and finding the desired file. Another convenient way to read a file, say file1, into the current session of Mathematica is
<< file1

When working on programs in Mathematica we prefer to write it to a file (using an editor) and then read it into a Mathematica session. This way when mistakes occur (and they do!), then I can edit the file in another window and re-read the file into Mathematica. This is particularly easy using the Notebook version.

To exit Mathematica at the Notebook level pull down File and go to Quit. At the line level type Exit[].

At the most basic level, Mathematica may be used interactively, like a calculator. Commands to be evaluated are entered into input cells, which are displayed in bold input in this text. They are evaluated by pressing the' ENTER' key. Results are displayed in output cells.

Illustration

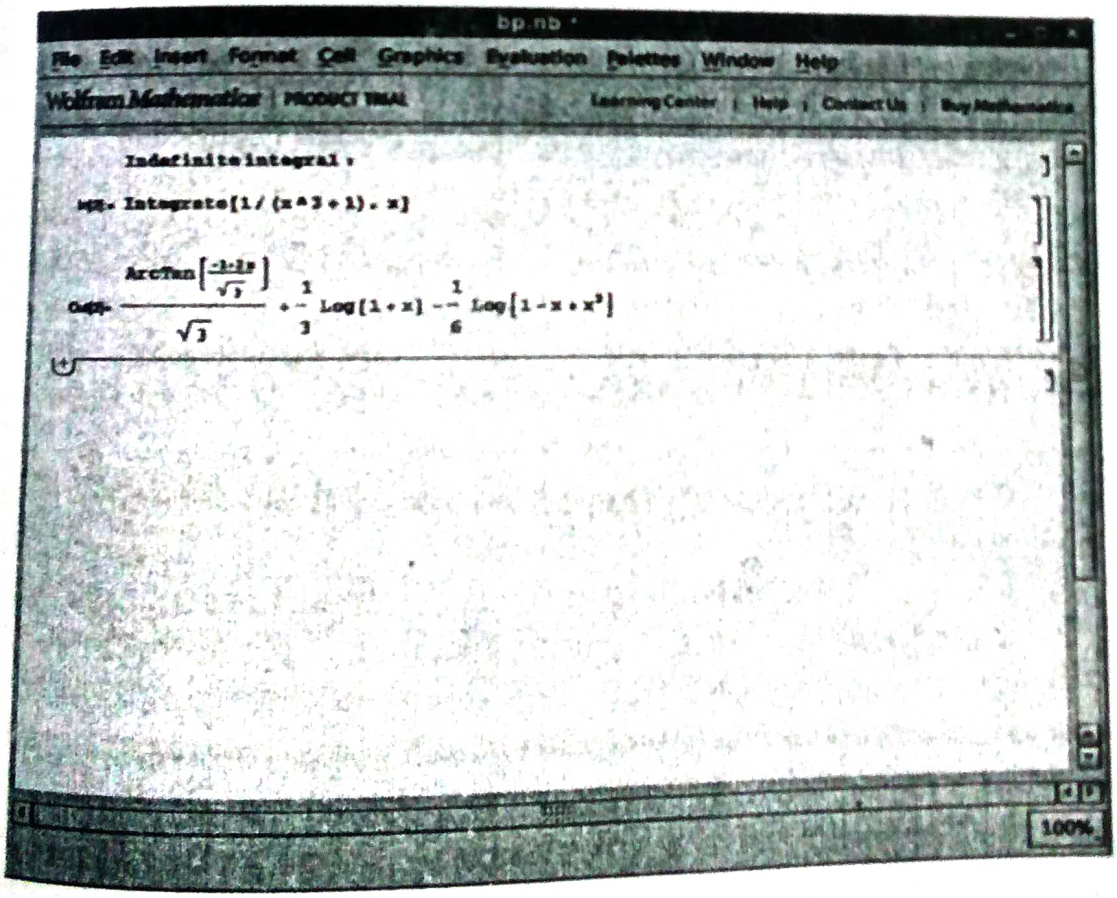
Here is a simple example:

```
2  2
4
```

We can enter longer expressions over several lines and also include comments in input cells by enclosing them in

```
(* These *)
 5 7 32 6       4
(* Here is a comment*)
89
12
```

Notice that we get a fraction. Mathematica is capable of exact computations, rather than decimal approximations. We can always get numerical approximations using the N function (for numerical). In the following line, % refers to the previous result and N passes that result to the N function




```
7.41667
```

Mathematica's ability to do exact computations is truly impressive.

For example, the value of 2 ^ 1000 is

10 715 086 071 862 673 209 484 250 490 600 018 105
614 048 117 055 336 074 437 503 883 703 510 511 249 361
224 931 983 788 156 958 581 275 946 729 175 531 468 251
871 452 856 923 140 435 984 577 574 698 574 803 934 567
774 824 230 985 421 074 605 062 371 141 877 954 182 153
046 474 983 581 941 267 398 767 559 165 543 946 077 062
914 571 196 477 686 542 167 660 429 831 652 624 386 837
205 668 069 376

10.3. Some Mathematica calculations

In our notebook, type

(1 + 2) * 3

and press **Enter**. We'll get the result:

9

We can omit the multiplication sign, and Mathematica is smart enough to know that it is implied:

(1 + 2) 3

9

But Mathematica can do a lot more than simple integer math. Try

102 / 9

We get the result

34 / 3

We can see that Mathematica reduced the fraction to its simplest form by cancelling out common factors in the numerator and denominator. Also, notice that the result is still expressed as a fraction, not as a decimal number. This is because the fraction is more exact than any decimal approximation to it, and Mathematica tries to maintain as much accuracy as possible unless we specify otherwise.

To get a numerical approximation to this result, type

N[102 / 9]

We'll get the result

11.3333

10.4. Arithmetic Operations

The arithmetic operations + and – are as usual addition and subtraction. Multiplication of x times y is x*y and the division of x by y is x/y. The operation of x to the power y, x^y, is $x \wedge y$. A common beginner's error is to think x/y+2 is $\frac{x}{y+2}$ when in fact it is $\frac{x}{y}+2$. To get the desired result use parentheses: x/(y+2). Similarly, $a \wedge b * c$ is $a^b c$ whereas $a \wedge (b * c)$ is a^{bc}.

10.5. User Defined Functions.

In Mathematica the function $f(x) = x^2$ is defined by f[x]:=x^2

On the left-hand side the x appears as x to denote that x is a variable. After f has been so defined, typing f[4] on a command line will result in the output 16.

Typing f[a+b] will result in the output $(a + b)^2$.

To define, say, the function $g(x, y) = x^2 + 6y^4$ of two variables x and y:

$$g[x,y] := x \wedge 2 + 6 * y \wedge 4 ;$$

Observe this example ended with a semicolon whereas the first example did not. Ending with a semicolon tells Mathematica not to show the output. This rule applies to all Mathematica inputs. (Sometimes the output is quite lengthy and one does not want to see it printed on the screen.)

10.6. Built-In Functions

Mathematica has many built-in functions that are familiar from calculus: sin x, cos x, exp x, log x, to name but a few. They are all called with the same syntax, e.g. Sin[x], Log[x], etc.

Mathematica has many functions which carry out algebraic operations. For example, if we apply the Mathematica function Expand to our function f[a+b] defined above, e.g.

Expand [f[a+b]]

the output will be $a^2 + 2ab+b^2$.

If we input Log[2], the output is log 2. If we want a numerical approximation to log 2 we use the Mathematica function N, e.g.

N[Log[2]]

results in the output 0.693147. If we want accuracy to, say, 15 decimal places we input

N[Log[2],15]

and we get the output 0.693147180559945.

10.7. Mathematical Constants

Mathematica has symbols for some common mathematical constants:

Pi $(=\pi \approx 3.14159)$, E $(=e \approx 2.71828)$, I $\left(i=\sqrt{(-1)}\right)$ and Infinity (∞).

For example,

Sin[Pi]

evaluates to 0 and

E^(2*Pi*I)

evaluates to 1.

10.8. Incrementing Variables

Mathematica has commands that increment variables which follow the programming language C. For example, i++ increments the value of i by 1 and i+=di adds di to the value of i. These constructions are useful in the programming part of Mathematica.

10.9. Conditionals

Mathematica provides various ways to set up conditionals, which specify that particular expressions should be evaluated only if certain conditions hold. The most common such construction is the If statement which has the general form If[test, then, else] which evaluates then if test is True, and evaluates else if it is False. For example the input

If[7<8,x,y]

returns the output x.

The expression test is a Boolean variable that evaluates to either True or False. Note that in Boolean expressions when testing whether x equals y one writes x==y

(note double equal sign).

A more interesting example is

```
If[Random[]<.5,1,-1]
```

Here the function Random is first called and if it returns a number less that 0.5 then the If statement returns the number 1 but if Random returns a number greater than or equal to 0.5 the If statement returns the number -1. This construction is useful for simulating the toss of a fair coin.

10.10. Loops

There are programming constructs that allow an expression to be evaluated several times. The three most common are Do, While, and For. Here are some typical examples

```
Do [Print[f[j]],{j,1,3}]
```

If f is the function defined earlier ($f(x) = x^2$), then the ouput of the above statement is 1, 4, 9. The general form is

```
Do[expr,{i,imin,imax}]
```

where expr is repetitively evaluated with i varying from imin to imax in steps of 1. Writing {i, imin, imax, di} instead gives steps of di.

10.11. Modules

In an earlier example we showed how the user defines functions. The construction used above is good for single line statements. Many times we wish to define a function which requires several Mathematica statements. Also, in the process we use expressions which are only necessary for the evaluation of the function. It is a good habit to keep these internal expressions local to the evaluation of the function we are defining. This can all be done in the Module command. The general form is

```
Module[{a,b,. . . }, proc]
```

that is, a procedure with local variables a, b, ...

As an example, suppose we toss a coin n times and we want to know the proportion of heads to the total number of tosses. Also, we want this proportion in decimal form. Here is a simple procedure to do this:

```
coin[n]:= Module[{headcounter = 0}, For[i=1,i<=n,i++,
If[Random[]<.5,headcounter++]]; N[headcounter/n,10]]
```

The expression headcounter is a local variable that is initialized to the value 0. We ran this for 1000 tosses by inputing coin[1000] with the result 0.492. For 10,000 tosses we got the result 0.499.

10.12.Graphics

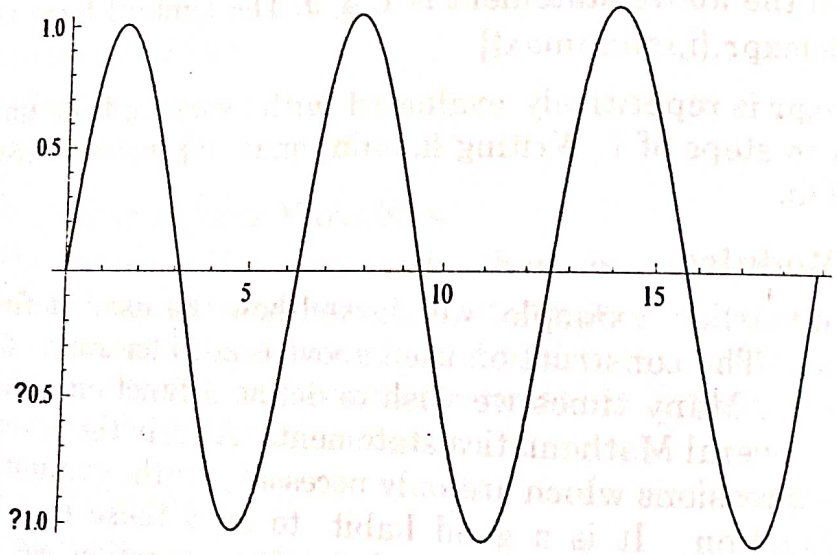
Here are two graphics commands frequently used: Plot and List Plot.

For Example.

Plot[Sin[x],{x,0,6*Pi}]

is the command to plot $\sin x$ for x in the domain $0 \le x \le 6\pi$.

The output is



There are several options that allow the user to customize the resulting graph. Given a list $\{\{x_1,y_1\},\{x_2,y_2\},\cdots,\{x_n,y_n\}\}$ of coordinates of points—let's call this list PointList—the command

ListPlot[PointList]

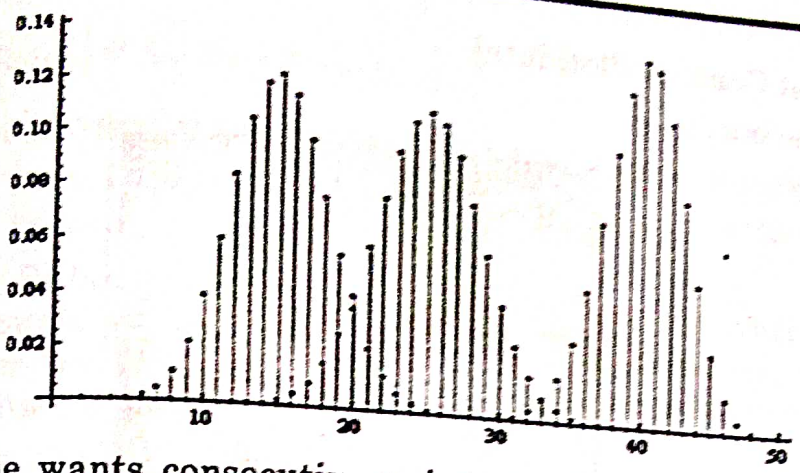
produces a plot of these points.

For Example.

Listplot[Table[{k,PDF[BinomialDistribution[50,p],k},

{p,{0.3,0.5,0.8}},{k,0,50}],Filling– > Axis]

gives the following results:



If one wants consecutive points in the list joined by a straightline (as we frequently do in probability), then one writes

```
ListPlot[PointList,PlotJoined->True]
```

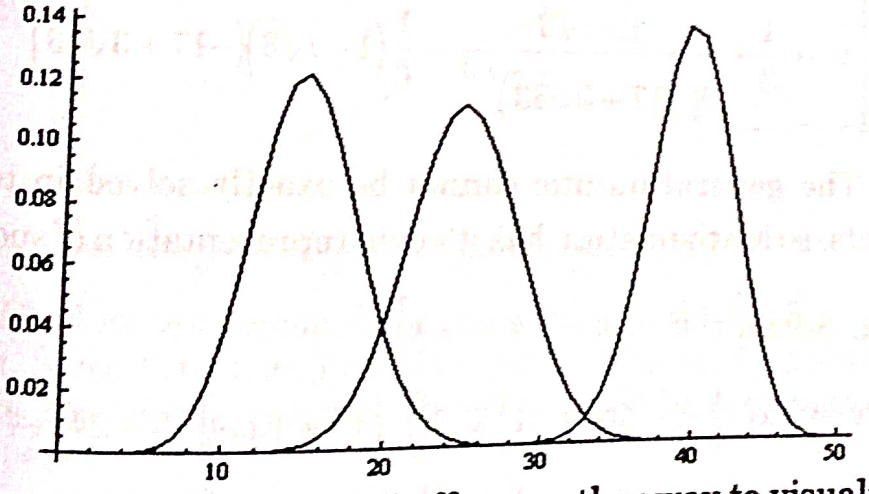
If instead of the list $\{\{x_1,y_1\},\{x_2,y_2\},\cdots,\{x_n,y_n\}\}$, one uses $\{y_1,y_2,\cdots,y_n\}$,

then the x-coordinates are defaulted to the values 1, 2, . . . , n

For Example.

```
Listplot[Table[{k,PDF[BinomialDistribution[50,p],k},
{p,{0.3,0.5,0.8}},{k,0,50}],Plotjoined → Axis]
```

gives the following output



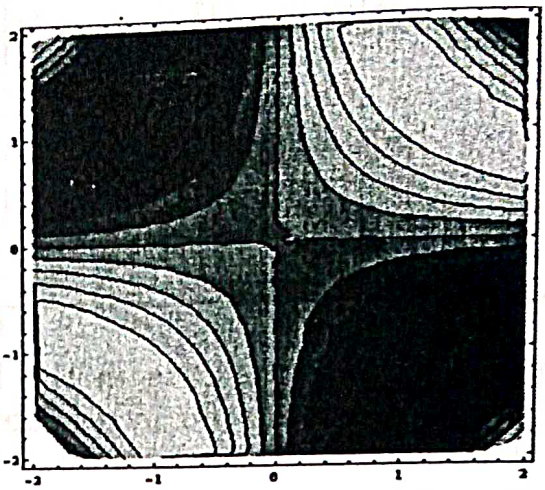
The Contour Plot command offers another way to visualize a function of two variables.

For example

```
data=Table[{x=Random Real[{-2,2}],
        y=Random Real[{-2,2}],Sin[x y]},{1000}];
```

List Contour Plot[data]

The output is



10.13. Solving equations

Mathematica has powerful built in techniques for solving equations algebraically and numerically. The basic command is *Solve*.

Here is how to use Solve to find the roots of a polynomial.

Ex. Solve $\left[x\text{^}4+2x+1==0,x\right][[3]]$

The output is

$$\left\{x\to\frac{1}{3}+\frac{1+i\sqrt{3}}{3\left(-17+3\sqrt{33}\right)^{1/3}}-\frac{1}{6}\left(1-i\sqrt{3}\right)\left(-17+3\sqrt{33}\right)^{1/3}\right\}$$

The general quintic cannot be exactly solved in terms of roots, so Mathematica has it's own representation of such roots.

Ex. Solve $\left[x\text{^}5-2x-3==0,x\right]$

$$\left\{\left\{x\to \text{Root}\left[-3-2\#1+\#1^5\&,1\right]\right\},\left\{x\to \text{Root}\left[-3-2\#1+\#1^5\&,2\right]\right\},\right.$$
$$\left\{x\to \text{Root}\left[-3-2\#1+\#1^5\&,3\right]\right\},\left\{x\to \text{Root}\left[-3-2\#1+\#1^5\&,4\right]\right\}$$
$$\left.\left\{x\to \text{Root}\left[-3-2\#1+\#1^5\&,5\right]\right\}\right\}$$

Note that the **NSolve** commands is similar, but returns numerical approximations.

Ex. NSolve[x^5 − 2x − 3 == 0, x]

{{x → −0.958532−0.498428i}, {x → −0.958532+0.498428i},
{x → −0.246729−1.32082i}, {x → 0.246729 +1.32082i},
{x → −1.42361}}

The **Solve** and **NSolve** commands work well with polynomials and systems of polynomials. However, many equations involving transcendental functions are beyond their capabilities. Consider, for example, the simple equation $\cos(x) = x$. **NSolve** will not solve the equation, but returns a statement to let you know this.

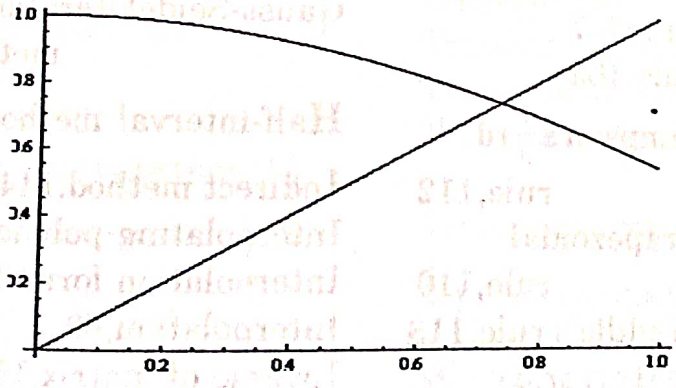
```
NSolve[Cos[x]== x, x]
```

Solve::tdep : The equations appear to involve the variables to be solved for in an essentially non-algebraic way.

```
NSolve[Cos[x]== x, x]
```

A simple plot shows that there is a solution in the unit interval.

Ex. Plot[{cos[x], x}, {x, 0, 1}]



The **Find Root** command uses Newton's method to find this solution. Since this is an iterative method, an initial guess is required. The graph indicates that 0.8 would be a reasonable initial guess.

Ex. FindRoot[cos[x] == x, {x, 0, 8}]

The result is

{x → 0.739085}

Note that we will use FindRoot to solve certain equations for the fractal dimension of a set. For $n = 3$