

2. *Mathematica*

■ 2.1 Introduction

Mathematica is a general purpose computer algebra system. That means it can do algebraic manipulations (including calculus and matrix manipulation) and it can also be used to draw graphs. See Anton, Bivens & Davis (Calculus, 8th ed) section 1.2 for an overview of what computer algebra systems can do and see the rest of the book for many specific exercises intended to be done using computer algebra.

■ 2.2 Starting Mathematica

Starting Mathematica from an X-terminal (Maths)

Type the command "mathematica" and press RETURN or ENTER (in a command window, xterm or kconsole) and a *Mathematica* notebook interface to the system open. (If a window pops up complaining about fonts not properly installed, just click 'continue' as they are actually installed.)

All the maths servers (boole, bell, gossett, graham, graves, hamilton, jbell, salmon, stokes, synge, turing, walton) have mathematica on them but the number of simultaneous users of *Mathematica* is limited. If the number of users is exceeded on your server, type the name of another to get a window acting on the new server. Try again there.

The notebook interface is the usual one and is quite sophisticated. (It was used to create these notes, for instance.)

When the system starts you get 3 windows immediately. One is the main window where you do your work (called a notebook window) and another is a palette of symbols that you can use to input some (but not all) instructions. A third one is mainly a nice picture and you can stop it coming up again. The "Ten minute Tutorial" in this 3rd window may be useful. You can get to the tutorial any time you want by an item under "Help". We will not manage all the topics in the 10 minutes!

By using the palette, you can have your input look somewhat like standard mathematical notation (with superscript for powers, fractions that look like fractions, and a few other nice things). However, everything can be done without using the palette and the palette does not provide access to the full range of *Mathematica* features. For that reason, these notes often ignore the palette options in favour of text-style input.

Using Mathematica outside the Mathematics department

Mathematica is available on PC and Macintosh computers around College. Notebooks created on these computers should (in principle) be readable by the *Mathematica* on the mathematics computer system, provided you can transfer the notebook file (the .nb file) from one system to the other. ftp is one way to do that. Email attachments is another way.

■ 2.3 Mathematica commands

Mathematica has a large collection of Mathematical functions it knows about. There is a whole book about them, in fact, and all we can hope for is to learn enough to do for our purposes.

There is a method in the way *Mathematica* chooses its names for things – generally it uses one of the commonly used names with the *first letter capitalised*. Also arguments of functions have to be in *square brackets*, rather than round brackets as is more common in mathematical usage like $f(x)$. Following this logic, Cos[x], Sin[x], Tan[x], etc, are the trig functions in *Mathematica*–speak. The command for drawing a graph is Plot[] and the command for drawing parametric graphs is ParametricPlot[] (inside the square brackets, you must specify at least the thing to be plotted and the range for the plot, separated by a comma – we will come back to this later).

After this explanation of the way *Mathematica* does things, and with some experience, you may be able to guess the right *Mathematica* name for a function you want.

In the notebook interface, you type instructions and press **Shift–Return** at the end (hold down the Shift button and press the return key). Another way that should work is to press the Enter key on the numeric keypad (extreme right of most keyboards).

■ 2.4 Online help (text mode)

If you think you remember, or just guess, the name for a function, you can get a cryptic summary of what the system knows about the name with the questionmark command. For example, suppose I wanted to find out the factors of a number. I can type in ?Factor (and press Shift–Return) and here is what I get (it comes out just below).

? Factor

```
Factor[poly] factors a polynomial over the integers. Factor[poly, Modulus->p] factors a
polynomial modulo a prime p. Factor[poly, Extension->{a1, a2, ...}] factors a polynomial
allowing coefficients that are rational combinations of the algebraic numbers ai.
```

This is clearly designed to factor polynomials (such as perhaps $x^2 + 2x + 3$, for example). It might be useful, but is not what I wanted. There is another form of the question mark command with a * at the end which will tell me about anything *Mathematica* knows about beginning with Factor.

? Factor*

| | |
|----------------|----------------------|
| Factor | FactorList |
| FactorComplete | FactorSquareFree |
| Factorial | FactorSquareFreeList |
| Factorial2 | FactorTerms |
| FactorInteger | FactorTermsList |

We can then look for information on FactorInteger to see if it is what we want.

? FactorInteger

FactorInteger[n] gives a list of the
prime factors of the integer n, together with their exponents.

There is a slightly longer (usually more technical) description available using a double question mark. Thus ??FactorInteger would tell you some more intricacies about the FactorInteger function.

■ 2.5 The help menu

There is a more convenient (and more informative) help system available through the "Help" menu button at the top of the notebook window. Choose the item "Master Index ..." in that menu and you will get a new window which provides online access to the whole *Mathematica* book. The book has a "Practical Introduction" that gives a brief account of what each *Mathematica* function does and provides references to the section of the book where the function is explained. Also there are sometimes examples in the summary. If you click on the links (highlighted) you should call up the page of the book referred to.

■ 2.6 Number Theoretic Examples

We can try out some of these commands a little, now we know about them.

```
FactorInteger[7953]
```

```
{{3, 1}, {11, 1}, {241, 1}}
```

This means that 7953 has three prime factors, 3, 11 and 241 and that each occurs just once. So if we multiply these 3 primes, we should get 7953.

```
3 * 11 * 241
```

```
7953
```

■ 2.7 Algebra

Here are some simple examples of *Mathematica* doing algebraic calculations. Basically, although *Mathematica* can do arithmetic calculations, its main strength is that it can do algebra. Note the use of Expand[] and Factor[] to tell *Mathematica* what we want it to do. (Remember the Shift–Return.)

```

x^2 + 2 x + 3

3 + 2 x + x^2

Factor[x^2 + 2 x + 3]

3 + 2 x + x^2

Expand[(x + 1) (x + 2) (x + 3)]

6 + 11 x + 6 x^2 + x^3

Factor[x^3 + 2 x - 3]

(-1 + x) (3 + x + x^2)

```

Here we have input the power using the exponentiation symbol ^ (and we could need parenthesis if the exponent is a complicated expression like a+1). Using the palette we could instead input the same thing with a nicer superscript layout in the input side.

```

Factor[x3 + 2 x - 3]

(-1 + x) (3 + x + x2)

```

Or, there is a way to do this without using a palette, where we use the key combination Control ^ to start the superscript (power) and the combination Control space to finish the superscript. (Hold down the Control key and then press the other key.)

Using either of these techniques you can make your input look more like the way you are used to writing mathematical formulae, though of course it is an extra complication to have two or three ways to do things.

```

(x + 1)^10

(1 + x)10

Expand[(x + 1)^10]

1 + 10 x + 45 x^2 + 120 x^3 + 210 x^4 +
252 x^5 + 210 x^6 + 120 x^7 + 45 x^8 + 10 x^9 + x^10

```

An important facility in *Mathematica* is the ability to solve equations, both exactly (symbolically) with Solve[] and numerically with NSolve[].

```

Solve[x^2 + 5 x + 2 == 0, x]

{{x -> 1/2 (-5 - Sqrt[17])}, {x -> 1/2 (-5 + Sqrt[17])}}

NSolve[x^2 + 5 x + 2 == 0, x]

{{x -> -4.56155}, {x -> -0.438447}}

```

You might want to note that we need to specify the equation to be solved with a double equals sign == and also we need to say what unknown (in this case x) to solve for. The

reason for the `==` as opposed to `=` is that *Mathematica* uses a single `=` to mean an assignment, to make the left hand side of the `=` have the value of the right hand side. We will come to using this assignment facility later in the course.

You might like to know that there is a general purpose `N[]` command in *Mathematica* to get the numerical value of any expression. So a nested

```
N[Solve[x^2 + 5 x + 2 == 0, x]]
{{x -> -4.56155}, {x -> -0.438447}}
```

will also work in this case instead of using `NSolve`. In fact `NSolve[]` is identical with `N[Solve[]]`. There is another command `FindRoot[]` which can succeed in finding a numerical value for a solution where `Solve[]` and `NSolve[]` may fail. `FindRoot` gives one solution and you have to tell it a starting value of the unknown where to look for a solution.

```
FindRoot[Cos[x] == x, {x, 0}]
{x -> 0.739085}
```

■ 2.8 Notation and Rules

The remarks above about the difference between `==` and `=` indicate that it is time to spell out the basic rules and notation used by *Mathematica* for interpreting what you type.

Addition, subtraction and division are `+`, `-` and `/`. For multiplication there are quite a few options (3 altogether). Looking back over the previous examples, there are examples where we used `*` for multiplication (as in `3 * 11 * 241`) but there are other examples where we did not put in any `*` (for example the `5 x + 2` in the last equation could be `5*x + 2`). So *Mathematica* will interpret a space between two quantities as meaning multiplication in cases standard mathematical notation would take that meaning, except that square brackets are used as indicating arguments of functions and curly brackets `{ }` have another special purpose (for lists of items). Ordinary round brackets `()` have a grouping affect as in ordinary algebra notation, but other types of brackets cannot be used for grouping terms. Thus we were able to write `(x + 1)(x + 2)` where we might have been more explicit about the multiplication between the factors with `(x + 1)*(x + 2)`. So we can indicate multiplication with a star `*`, or a space, or sometimes by juxtaposition without any space needed. This latter possibility (leaving out the space) is available for examples like `(x + 1)(x + 2)` where we mean to multiply grouped things together and also in the case of numerical quantities times variables, such as `2x`. For two times `x`, we can write `2*x`, `2 x` (with a space) or just `2x` (no space between). Note again that

```
[x + 1] [x + 2]
Syntax::sntxb : Expression cannot begin with "[x + 1] [x + 2]".
[x + 1] [x + 2]
```

produces an error. **Only round brackets** (parentheses) can be used for grouping.

All this looks good, but there is a difference between *Mathematica* notation and ordinary algebra notation when it comes to multiplication. If we have two quantities called x and t , and we want to multiply them, we must put in the $*$ or else a space ($x*t$ or $x t$ are both ok). The catch is that *Mathematica* will treat xt (no space in between) as a new name for a new quantity. You can see this if we divide $x*t$ by x and see what we get with the $*$, with a space and with no space.

```
x * t / x
```

```
t
```

```
x t / x
```

```
t
```

```
xt / x
```

```

xt
--
x

```

This may seem to be a silly way for *Mathematica* to do things, but it has the advantage that you can use names for quantities that are longer than one letter. So, for example, we could use "mass" for mass and "time" for time, and that might be clearer than having to abbreviate them as something like m and t .

```
mass / (mass * time)
```

```

  1
--
time

```

```
m a s s / (mass * time)
```

```

  a m s2
-----
mass time

```

Note that the spaces in the "m a s s" mean that *Mathematica* treats it as $m*a*s*s$

Observe that we have been using the hat notation for "raised to the power of".

The $/$ notation for division is the standard way to indicate division or fractions, but as with Control ^ for powers it is possible to input fractions so that they look like fractions. You can do that with the palette or by typing Control / to enter the denominator and Control space to finish the denominator. One advantage of these methods, apart from making it easier to read the input, is that you don't have to put parentheses around your numerators and denominators.

```

      s
m a s ---
      mass * time

```

```

  a m s2
-----
mass time

```

■ 2.9 Differentiation

$$D[a x \cos[x^2] - \frac{x}{x+1}, x]$$

$$\frac{x}{(1+x)^2} - \frac{1}{1+x} + a \cos[x^2] - 2 a x^2 \sin[x^2]$$

■ 2.10 Integration

```
Integrate[x Cos[x], x]
Cos[x] + x Sin[x]
Integrate[x Cos[x], {x, 0, Pi}]
-2
```

These were examples of indefinite integration (no limits) and definite integration (with limits). Pi (with a capital P) is the text way of inputting the number pi. With the palette you could enter the usual symbol for Pi and you could also make the inetgral look like an integral.

$$\int_0^{\pi} x \cos[x] dx$$

-2

■ 2.11 Solving differential equations

```
DSolve[D[y[x], x] - a y[x] == 0, y[x], x]
{{Y[x] -> E^{a x} C[1]}}
```

We needed to tell DSolve the differential equation, the unknown function y[x] and the independent variable x.

■ 2.12 Plotting

As mentioned a long while back, the basic commands for ordinary graphs is Plot[] and the command for parametric plots is ParametricPlot[]. We can use the double question mark help facility to see that there are a huge variety of ways of controlling the way a Plot comes out, depending on what aspect of the graphs you think needs to be shown.

?? Plot

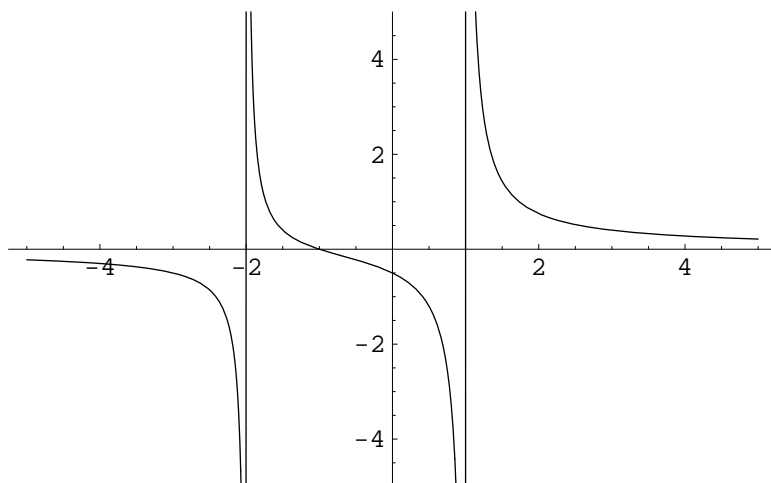
`Plot[f, {x, xmin, xmax}]` generates a plot of f as a function of x from x_{\min} to x_{\max} . `Plot[{f1, f2, ...}, {x, xmin, xmax}]` plots several functions f_i .

`Attributes[Plot] = {HoldAll, Protected}`

`Options[Plot] = {AspectRatio -> GoldenRatio^(-1), Axes -> Automatic, AxesLabel -> None, AxesOrigin -> Automatic, AxesStyle -> Automatic, Background -> Automatic, ColorOutput -> Automatic, Compiled -> True, DefaultColor -> Automatic, Epilog -> {}, Frame -> False, FrameLabel -> None, FrameStyle -> Automatic, FrameTicks -> Automatic, GridLines -> None, ImageSize -> Automatic, MaxBend -> 10., PlotDivision -> 30., PlotLabel -> None, PlotPoints -> 25, PlotRange -> Automatic, PlotRegion -> Automatic, PlotStyle -> Automatic, Prolog -> {}, RotateLabel -> True, Ticks -> Automatic, DefaultFont -> $DefaultFont, DisplayFunction -> $DisplayFunction, FormatType -> $FormatType, TextStyle -> $TextStyle}`

If we wanted to plot $y = (x + 1)/(x^2 + x - 2)$ over the range $x = -5$ to $x = 5$, the command `Plot[(x + 1)/(x^2 + x - 2), {x, -5, 5}]` would do a reasonable job. In the following example, I have added a `PlotRange` option to control the range on the y -axis. (Without that I think *Mathematica* chooses too big a range.)

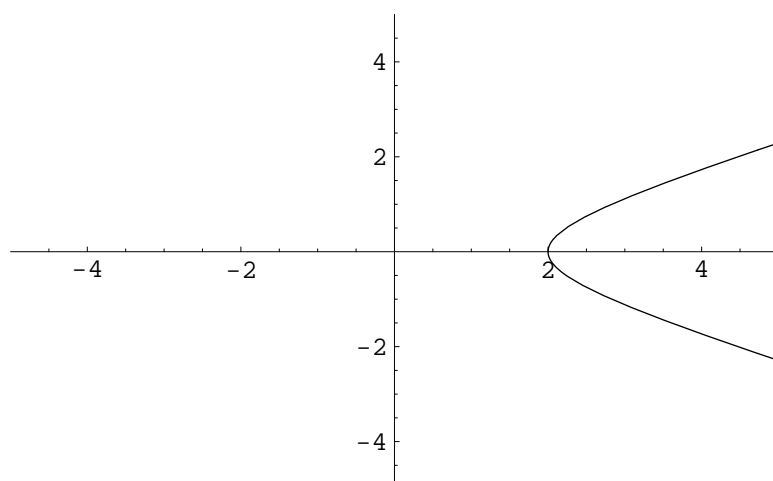
`Plot[(x + 1)/(x^2 + x - 2), {x, -5, 5}, PlotRange -> {-5, 5}]`



- Graphics -

For an example of a parametric plot, I use hyperbolic cosine and hyperbolic sine as a parametrization of the hyperbola $x^2/4 - y^2 = 1$. (You will learn about hyperbolic functions later in the year.)


```
ParametricPlot[{2 Cosh[t], Sinh[t]}, {t, -2, 2}, PlotRange -> {{-5, 5}, {-5, 5}}]
```



- Graphics -

This graph actually shows only the half of the hyperbola where $x > 0$. There is a mirror image half for $x < 0$.

■ 2.13 Printing

The easiest way to print out graphs and other work you do in a *Mathematica* notebook is to use the "Print ..." menu item you will find under the "File" menu. You get a little menu to fill in. You should select "A4 letter" paper size, untick the File box and tick the Print box (just click with the left button of your mouse). We don't really support printing in maths now and so you need to use this in a College computer room.

■ 2.14 Saving and reloading

With the notebook interface, you can save your work to date. Use the "Save" or "Save As" menu item in the "File" menu. You should save the notebook with a more useful name than Untitled-1.nb (but you should keep the extension nb for saved notebooks).

If you then start *Mathematica* at a later time, you can get back your earlier work. You can do this in two ways. Start *Mathematica* as already indicated and use the "Open ..." item in the "File" menu to open your earlier notebook. Another way is to start *Mathematica* by giving the command "mathematica mywork.nb" (replace mywork.nb with the actual name you saved it under).

When you reload a notebook this way, *Mathematica* will not recompute anything. If you had defined your own variables or functions (we will do this later), these definitions will not be active upon reloading the notebook. You must re-activate or recompute any such definitions by selecting them and pressing Shift-Return. There is also a menu item under 'Kernel' -> 'Evaluation' to recompute everything in the notebook in one fell swoop.

Bug: There was a small problem with the implementation we were using on the mathematics system. I am not sure if it is still there. It causes *Mathematica* to make a mistake about which directory it is working in (sometimes) and the effect is that you cannot save your work when you do "Save As". If you look carefully at where it is trying to save your work, you will see the problem. Most of you have home directories called /u4/science/2008/xyz/ (where xyz is replaced by your user name). *Mathematica* sometimes replaces the u4 at the beginning with something else. Click on it and fix it.

■ 2.15 Defining your own variables and constants in *Mathematica*

Mathematica treats everything it does not already know about as the name of a new unknown. We saw that an x or y or a can be introduced into a formula and *Mathematica* will manipulate with it as an unknown (numerical) quantity.

In most computer languages, you cannot do that, but you can say that x is going to be a number and calculate with x after you have given x a value. *Mathematica* makes it easy to calculate with x even when x has not yet got a value assigned to it. However, *Mathematica* also allows you to give a value to a "variable". For example, if we were going to have to refer several times to a number 321.4567 we might prefer to denote this number by the letter c (say), or some other letter that seemed appropriate in the context. We do this in *Mathematica* by the command

```
c = 321.4567
```

```
321.4567
```

The single = is an instruction to put c equal to the right hand side. (Remember the double == we had to use in equations for Solve and NSolve.)

Whenever we refer to c from now on (during this session of *Mathematica* calculations), *Mathematica* immediately replaces it by 321.4567. For example

```
2 c - 1
```

```
641.913
```

One advantage of this is that we can type in formulae involving c and they can look more digestable than formulae with 321.4567 occurring in several places. Another advantage is that we can calculate the same formula again with a different value of c

One thing to watch out for is that if you make something have a value in this way, it is no longer available as a variable. For example the sequence of steps

```
x = 3
```

```
3
```

```
D[x^2, x]
```

```
General::ivar : 3 is not a valid variable.
```

```
∂3 9
```

generates a complaint because x has become 3 and so it does not make sense to differentiate 3^2 with respect to a constant 3. To get out of this mess, we can tell *Mathematica* to forget that we put x equal to 3 with the `Clear[]` command.

```
Clear[x]
```

```
D[x^2, x]
```

```
2 x
```

In *Mathematica* we can combine the ability to deal with unknowns with the ability to assign values to things. We can make a letter or name stand for a whole formula.

```
y = (x + a) / (x - a + 1)^2
```

$$\frac{a + x}{(1 - a + x)^2}$$

```
D[y, x]
```

$$\frac{1}{(1 - a + x)^2} - \frac{2(a + x)}{(1 - a + x)^3}$$

■ 2.16 Defining your own functions in *Mathematica*

This method of using a letter to stand for a formula is handy, but it has its drawbacks. The function notation $f(x)$ we use in mathematics is frequently more convenient. For example, to evaluate the above y at a particular x (say $x = a$) we can do it with the following rather awkward notation.

```
y /. x -> a
```

```
2 a
```

Mathematica allows you to define your own functions with your own names. Thereafter these functions will behave like the built in functions such as `Cos[]` and `Exp[]`. As for the built in functions, we must use square brackets `[]` to surround what we are applying the function to, rather than the round brackets we are used to in mathematical notation $f(x)$. When defining a function, we must use a special underscore to denote things that are free or independent variables in the function. (*The underscore is used only on the left hand side of the definition.*) Here is an example of how to define a function called `f` that will have the value $f(x) = x - \cos(x)$ and a `g(x)`.

```
f[x_] = x - Cos[x]
```

```
x - Cos[x]
```

$$g[x_] = (x + a)/(x - a + 1)^2$$

$$\frac{a + x}{(1 - a + x)^2}$$

We can now evaluate these at particular values of x.

$$f[1.0] g[a]$$

$$0.919395 a$$

There is also an alternative way to work out derivatives, using more or less the usual $f'(x)$ notation.

$$D[g[x], x]$$

$$\frac{1}{(1 - a + x)^2} - \frac{2(a + x)}{(1 - a + x)^3}$$

$$g'[x]$$

$$\frac{1}{(1 - a + x)^2} - \frac{2(a + x)}{(1 - a + x)^3}$$

$$g[a] + g'[a] (x - a)$$

$$2a + (1 - 4a)(-a + x)$$

■ 2.17 Summations

Mathematica understands a version of the "Sigma" notation for sums. In case you don't know,

$$\sum_{n=1}^6 (n^2 + n + 1)$$

means the result you get by evaluating $n^2 + n + 1$ for each value of n in 1, 2, 3, etc up to 6 – and then add these values. Here we calculate the values.

$$\sum_{n=1}^6 (n^2 + n + 1)$$

$$118$$

$$(1^2 + 1 + 1) + (2^2 + 2 + 1) + (3^2 + 3 + 1) + (4^2 + 4 + 1) + (5^2 + 5 + 1) + (6^2 + 6 + 1)$$

$$118$$

Without using the palette, the way to input this is with the `Sum[]` command.

$$\text{Sum}[n^2 + n + 1, \{n, 1, 6\}]$$

$$118$$

■ 2.18 Lists, vectors and matrices

In *Mathematica*, a symbol like x does not have to stand for an unknown number. It can also be something of another type.

We have seen at least 3 types of objects in *Mathematica*: numerical constants (like 3 or Pi), numerical variables and function identifiers. When we define a function $f[x]$, then f becomes an identifier (or name) for a function. It is important that f is followed by square brackets $[]$ for the system to recognise it as a function. The predefined functions like $\text{Sin}[]$, $\text{Cos}[]$, etc are treated in the same way.

Another type or class of object in *Mathematica* is a list. Basically we can have lists of almost anything.

However, the simplest case is a list of numbers. These might be a list of data points from some experiment, or a list of values of some function. One simple case is a list of two numbers x and y which we take as the coordinates of a point. In *Mathematica* we can say something is a list by enclosing it in curly brackets $\{ \}$.

```
p = {2, 5}
{2, 5}
```

From the point of view of mathematics, we usually use round brackets $p = (2, 5)$, but curly brackets are what *Mathematica* needs.

We can get at the entries in a list with double square brackets. Let's ask for the second coordinate of p (or as far as *Mathematica* knows, the second entry in the list p)

```
p[[2]]
5
```

There is a repetition operator (in the same mould as $\text{Sum}[]$) for building lists, called $\text{Table}[]$. Here is a list of the first 5 primes numbers.

```
Table[ Prime[i], {i, 1, 5} ]
{2, 3, 5, 7, 11}
```

It might not seem natural, but *Mathematica* uses the same list idea to deal with matrices. We can look on a 2×2 matrix a list of two rows, where each row is a list of two numerical entries.

```
A = { {a, b}
      {c, d} }
{{a, b}, {c, d}}
```

You can see I made it look a bit like a matrix when I typed it in, but Mathematica echoed it as a list. If I calculated a matrix and wanted to printed it out to look like a matrix, I can use the `MatrixForm[]` call.

```
MatrixForm[A]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

The basic input palette is very useful for entering matrices. There is a 2 by 2 matrix in the standard palette. Click with the left button on the little matrix in the palette and a 2 times 2 matrix with blobs as entries will show up in your notebook. Fill in the blobs by selecting them with the mouse and typing or you can use tab to get to the next blob.

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
{ {a, b}, {c, d} }
```

You can see the lists are still visible in the echo (after pressing Shift–enter), but this is a convenient way to type in matrices. Use Control–comma to get a new column in the matrix after the one where you are typing and use Control–return to get an extra row.

$$\text{biggermatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

```
{ {1, 2, 3}, {4, 5, 6} }
```

Matrix multiplication needs a dot rather than a `*`. Other kinds of matrix operations are also available, but these will not make sense until you learn the mathematical concepts a little later.

```
MatrixForm[ B . biggermatrix ]
```

$$\begin{pmatrix} a + 4b & 2a + 5b & 3a + 6b \\ c + 4d & 2c + 5d & 3c + 6d \end{pmatrix}$$

R. Timoney (November 2006)