Departments of Civil Engineering and Mathematics
CE 109: Computing for Engineering

Mathematica Session 1: Introduction to the system

Mathematica is a piece of software described by its manufacturers as "A System for Doing Mathematics by Computer". If you haven't seen it, or something like it, before, you may be amazed at what it can do. Algebra, calculus, curve plotting: all the traditional skills of mathematics, *Mathematica* can handle automatically.

This session, and the two following, are designed as a brief introduction to *Mathematica*. It's a huge program, and we'll only be able to scratch the surface in the time we have.

1. How *Mathematica* works

Start by *loading Mathematica*. Information for this is given on the separate *Start-up Sheet*: ask for it if you don't have it.

To start with, it may be helpful to think of *Mathematica* as an exceptionally powerful calculator. Like a calculator, it does arithmetic. For example, try typing

```
2+2
```

To get *Mathematica* to perform this calculation (technically, to **evaluate** this **expression**) you have to *hold down* the "shift" key and press "return" (which on some keyboards looks like "↵").

The first calculation will appear to take a long time: it always does. That's not because *Mathematica* finds this particular calculation hard: it's because the part of the program that actually does the calculating doesn't get loaded until the first time it's needed, and it's the loading process that takes the time. All subsequent calculations should be much quicker: here are a few you might like to try. It is best to evaluate each of these calculations separately: do "shift-return" after typing each line.

```
6+2
6-2
6*2
6 2
6/2
6^2
100!
1000!
Sin[Pi/3]
Sqrt[9]
Sqrt[50]
2^(1+4)
```

Notice the use of the symbols `*`, `/` and `^`. Notice, too, the way *Mathematica* interprets the **space** between the 6 and the 2.

(i) The last few examples all involve the use of ***brackets*** *of some kind. But sometimes square brackets have been used, and sometimes round ones (**parentheses**). Does this matter? What are the rules governing what kind of brackets to use when? Experiment, and find out the answers to these questions.*

(ii) Would it matter if we'd typed

```
sin[pi/3]
```

instead? What are Mathematica's rules about capital and lower-case letters?

(iii) Was the output from

```
Sqrt[50]
```

what you expected? What's happening here? What other examples can you find?

Mathematica allows all the standard arithmetical commands to be input using a special format that looks like normal written mathematics. You can access it using the **palettes** (arrays of special buttons): go to the "Palettes" entry on the "File" menu; "Basic Input" is what its name suggests (it may already be up on the screen, at the right-hand side). Then, you can do inputs like:

6^2
$\mathrm{Sin}[\pi/3]$
$\sqrt{50}$

We'll mostly use this format from now on, but if you prefer to stick with "`^`", "`Sqrt`", etc. that's exactly equivalent.

All the above are examples of what's known as **exact arithmetic**: calculations with whole numbers, surds, and special constants like π. This is something your calculator *doesn't* do — it works with numbers in the form of decimals or in standard scientific from. This latter kind of calculation, known technically as **floating-point arithmetic**, can also be done by *Mathematica*. Try the following pairs of commands, and note the difference in the output:

```
2/6
2.0/6.0

Cos[Pi/6]
Cos[3.1415927/6]

Sin[2]
Sin[2.0]

100!
100.0!
```

$\sqrt{50}$
$\sqrt{50.0}$

(i) Experiment with the Mathematica command `N`*, as in* `N[Pi]` *or* `N[Pi, 100]`.

(ii) Experiment with the Mathematica commands `Floor`, `Ceiling` *and* `Round`*, as in* `Floor[3.77]` *or* `Round[25]`*. Use whole number and floating-point inputs, as well as exact non-integer constants like* `Pi` *and* `E`*. Describe precisely what these commands do.*

Where *Mathematica* really begins to differ from a calculator—at least, from an ordinary one—is in its ability to perform *symbolic* calculations, for example:

$\mathrm{Expand}[(x+y)^3]$
$\mathrm{Factor}[x^3 + 3x^2\ y + 3x\ y^2 + y^3]$
$\mathrm{Apart}[(x)/(\ (1+x)\ (1+x^2)\ )]$
$\mathrm{Together}[\%]$

(Notice the use of the `%` sign to mean "last output".)

$\mathrm{D}[x^3,\ x]$
$\mathrm{Integrate}[3x^2,\ x]$

```
Solve[x² + 4x + 3 == 0, x]
Solve[{x + 2y == 5, x y == 2}, {x, y}]
```

Notice the use of the double equals sign, `==`, in equations. The single equals sign is used only for **assignment**: assigning a value to a variable, as in:

```
z = 3
```

The value can be many more things than a simple number; it could be the result of an integration, or the set of solutions to an equation:

```
int1 = Integrate[3x², x]
D[int1, x]

solns = Solve[x² + 4x + 3 == 0, x]
x² + 4x + 3 /. solns
```

One can even assign an alternative name for one of those long, capital-lettered *Mathematica* function names—though this is not a practice we recommend:

```
int = Integrate
int[int1, x]
```

To undo assignments, use the `Clear` command:

```
Clear[z, int1, solns, int]
```

Any assignments you make will apply till the end of the current *Mathematica* session. It's a good idea to get into the habit of `Clear`-ing any variables (especially common ones like `x` and `y`) before starting to use them again.

A related idea to assignment is that of **substitution**.

(i) Type the following to substitute $x = 3$ *into the expression* $x^2 + 2x - 5$*:*

```
x = 3
x² + 2x - 5
```

What's the problem with this form of substitution?

(ii) Try the following, alternative way of substituting $x = 3$ *into the expression* $x^2 + 2x - 5$*:*

```
Clear[x]
x² + 2x - 5 /. x -> 3
```

What happens if you type

```
x
```

now? How does the "`->`*" operation differ from assignment ("*`=`*")?*

(Note: "`/.`" is a shorthand form for the command `ReplaceAll`.)

2. Graph plotting

As well as its various numerical and symbolic capabilities, *Mathematica* incorporates a set of advanced graph plotting capabilities.

To plot the graph of the function $f(x) = 2x^2 + x + 3$, for example, type

```
Plot[2x² + x + 3, {x, -3, 3}]
```

Or, to plot the function and the graph of its derivative on the same pair of axes, type

```
Plot[{2x² + x + 3, 4x + 1}, {x, -3, 3}]
```

The **option setting** `PlotRange->{-5, 10}` has the effect of altering the *y*-range of the plot. You implement option settings like this:

```
Plot[{2x^2 + x + 3, 4x + 1}, {x, -3, 3}, PlotRange->{0, 6}]
```

(i) Investigate the effects of the following option settings:

```
PlotRange->{{-1, 1}, {-2, 4}}
PlotRange->All
AspectRatio->1
AspectRatio->Automatic
PlotStyle->RGBColor[1,0,0]
PlotStyle->{RGBColor[1,0,0], RGBColor[0,0,1]}
AxesOrigin->{-1/4, 23/8}
```

Try using several option settings at once (use commas to separate them inside the `Plot` *command).*

(ii) Produce, using Mathematica, a plot showing the relationship between the graphs of $y = x^2$ *and* $y = \sqrt{x}$*. Your graphs should be in different colours and the scales should be the same on both axes. Repeat for some other pairs of functions related in this way.*

(iii) Produce graphs of the functions `Floor[x]`, `Ceiling[x]` *and* `Round[x]`.

The `Plot` function is the simplest graphical function in *Mathematica*, and one of the most useful, but it is far from being the only one. Here are some examples of others:

```
ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2Pi}]
ParametricPlot[{t^2, t^3}, {t, -3, 3}]

Plot3D[(x^2 - y^2) Exp[-x^2 - y^2], {x, -2, 2}, {y, -2, 2}]
ContourPlot[(x^2 - y^2) Exp[-x^2 - y^2], {x, -2, 2},
  {y, -2, 2}]

ParametricPlot3D[{Cos[t], Sin[t], t}, {t, 0, 2Pi}]
ParametricPlot3D[{Cos[s] Cos[t], Sin[s] Cos[t], Sin[t]},
  {s, 0, 2Pi}, {t, -Pi/2, Pi/2}]
```

(i) Using a suitable option setting, produce a parametric plot of the circle that really is circular on the screen. Produce, too, a parametric plot of the ellipse

$$x = 2\cos\theta, \quad y = \sin\theta.$$

(ii) Produce 3-D parametric plots of a helix that spirals outwards as it rises, and of a ***prolate spheroid*** *(rugby ball or American football shape).*

3. Using *Mathematica* to look at beam deflections

In many engineering situations, **piecewise-defined** functions have an important role: functions which are defined by joining together pieces of different functions on different parts of the *x*-domain. For example, here is a function that has a constant value of 1 for all values of $x < 0$, and takes the value $x^2 + 1$ for $x \geq 0$:

$$g(x) = \begin{cases} 1, & x < 0 \\ x^2 + 1, & x \geq 0 \end{cases}$$

Piecewise-defined functions don't have to be continuous; the basic "step function" is a key example:

$$s(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

One way of setting up piecewise-defined functions in *Mathematica* is to use the `If` command. `If` is a basic "conditional operator" which you may have met in programming languages (or in a programmable calculator). *Mathematica*'s version takes three inputs: a condition which is true or false, an output if the condition is true, and an output if the condition is false, for example:

```
If[-2.35 < 0, 0, 1]
If[1.78 < 0, 0, 1]
```

Here, then, is a plot of $s(x)$:

```
Plot[If[x < 0, 0, 1], {x, -5, 5}]
```

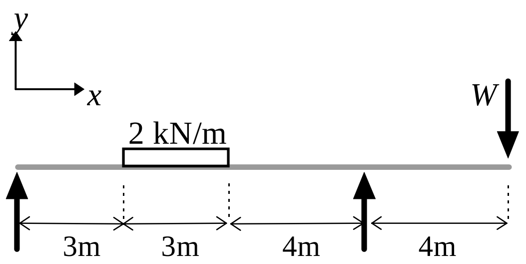
In your course on structural mechanics, you will be meeting Macaulay's method for solving beam deflection problems. In that, one ends up with terms in the displacement formula of the form

$$(x-3)^4,$$

where it is understood (and is sometimes indicated explicitly by the use of square brackets) that the term becomes zero when $x < 3$.

(i) Can you think of a way to represent a Macaulay-type term in Mathematica by using the `If` *command?*

Consider the following situation: a beam of length 14 m and flexural stiffness 10 MN m^2 is simply-supported at the points $x = 0$ and $x = 10$, and a distributed load of intensity 2 kN / m is applied between $x = 3$ and $x = 6$:



(Note: positive deflections are taken upwards here, opposite to convention, since this simplifies the plotting later on.)

A variable concentrated load *W* is applied downwards at the right-hand end of the beam. Then, the bending moment analysis yields the following formulae for *y*, *y*' and *y*'' as functions of *x* and *W*:

```
Clear[x, y, W]
EI = 10^7

y = (1/EI)*(5(-22275 + 4W)x/3 + (33000 - 4W)x^3/60 -
  (250/3)*If[x<3, 0, (x-3)^4] + (250/3)*If[x<6, 0, (x-6)^4] +
  (27000 + 14W)*If[x<10, 0, (x-10)^3]/60)

ydash = (1/EI)*(5(-22275 + 4W)/3 + (33000 - 4W)x^2/20 -
  (1000/3)*If[x<3, 0, (x-3)^3] + (1000/3)*If[x<6, 0, (x-6)^3] +
  (27000 + 14W)*If[x<10, 0, (x-10)^2]/20)

yddash = (1/EI)*((33000 - 4W)x/10 -
  1000*If[x<3, 0, (x-3)^2] + 1000*If[x<6, 0, (x-6)^2] +
  (27000 + 14W)*If[x<10, 0, (x-10)]/10)
```

The "`/.`" operation introduced in Section 1 can be used to substitute a value for W in these expressions, for example, if W = 1.87 kN:

```
y1870 = y /. W->1870
```

and then to make a plot:

```
Plot[y1870, {x, 0, 14}]
```

(ii) Plot graphs of y against x for values of W = 0, 1, 2, 3, 4 kN.

You can plot a set of graphs on the same axes by defining, say, the expressions `y0`, `y1000`, ..., `y4000` for each of the values of W and then doing:

```
Plot[{y0, y1000, …, y4000}, {x, 0, 14}]
```

Alternatively, you can make use of the `Table` command, for example:

```
Plot[Evaluate[Table[y /. W->Wval, {Wval, 1000, 2000, 250}]],
  {x, 0, 14}]
```

Here, the range of W values goes from 1 to 2 kN in steps of 0.25 kN.

(iii) For each of the values of W in (ii), *estimate* *using graphs:*

- *the maximum absolute displacement of the beam, and the point at which it occurs;*
- *the gradient of the beam,* y'*, at the point of support* $x = 10$*;*
- *the position of the (interior) point of contraflexure (where the curvature along the beam changes sign and* $y''(x) = 0$*).*

W / kN	Maximum absolute displacement / mm	Point of maximum displacement / m	Gradient at $x = 10$	Point of contraflexure / m
0				
1				
2				
3				
4				

There are several ways that *Mathematica* can assist in the estimation process:

- enlarge the graphic output: click on the graphic, move the mouse to the "resize" blob at the bottom right-hand corner, click (and hold) on the blob and drag the mouse to the right;
- use the `PlotRange` option in `Plot` to zoom in on the part of the graph that you need to examine;
- use the option `GridLines->Automatic` in `Plot`;
- click on the plot graphic and hold down the "Ctrl" key; the cursor's shape will change to a "+"; as you move the cursor around, its coordinate position can be read off in the bottom left corner of the *Mathematica* window.