MATHEMATICA

What it can do for you.

Overview

- Uses of Mathematica
- How the program works
- Language rules
- **EXAMPLES!**

Background

- Created by Stephen Wolfram and his team Wolfram Research.
- Version 1.0 was released in 1988.
- Latest version is Mathematica 6.0 – released last year.

Stephen Wolfram: creator of Mathematica

Q: What is Mathematica?

A: An **interactive** program with a vast range of uses:

- **Numerical calculations to required precision**
- **Symbolic calculations/ simplification of algebraic expressions**
- **Matrices and linear algebra**
- **Graphics and data visualisation**
- **Calculus**
- **Equation solving (numeric and symbolic)**
- **Optimization (?)**
- Statistics
- Polynomial algebra
- Discrete mathematics
- Number theory
- Logic and Boolean algebra
- Computational systems e.g. cellular automata

Structure

Composed of two parts:

- Kernel: interprets code, returns results, stores definitions (be careful)
- Front end:

  - provides an interface for inputting Mathematica code and viewing output (including graphics and sound) called a **notebook**

  - contains a library of over one thousand functions

  - has tools such as a debugger and automatic syntax colouring

More on notebooks

- Notebooks are made up of cells.
- There are different cell types e.g. “Title”, “Input”, “Output” with associated properties
- To evaluate a cell, highlight it and then press **shift-enter**
- To stop evaluation of code, in the tool bar click on Kernel, then Quit Kernel

Language rules

- ; is used at the end of the line from which no output is required
- Built-in functions begin with a *capital* letter
- [ ] are used to enclose function arguments
- { } are used to enclose list elements
- ( ) are used to indicate grouping of terms
- *expr/ .x* $\longrightarrow$ *y* means “replace *x* by *y* in *expr*”
- *expr/ .rules* means “apply *rules* to transform each subpart of *expr*” (also see `Replace`)
- = assigns a value to a variable
- == expresses equality
- := defines a function
- x_ denotes an arbitrary expression named x

Language rules (2)

- Any part of the code can be commented out by enclosing it in `(* *)`.
- Variable names can be almost anything, BUT

  - must not begin with a number or contain whitespace, as this means multiply (see later)

  - must not be protected e.g. the name of an internal function
- BE CAREFUL - variable definitions remain until you reassign them or `Clear` them or quit the kernel (or end the session).

Mathematica as a calculator

- Contains mathematical and physical constants e.g. i (`I`), e (`E`) and π (`Pi`)
- Addition `+`

  Subtraction `-`

  Multiplication `*` or blank space

  Division `/`

  Exponentiation `^`
- Can carry out calculations to any precision - see `N`.
- Can do symbolic calculations and simplification of complicated algebraic expressions –see `Simplify` and `FullSimplify`.

Creating your own functions

Use an underscore for the dummy variable and :=

e.g. `f[x_]:=N[Log[Abs[x]]+x^3]`

`Do` and `If`

- `Do[`*expr*, {*i*, i_{min}, i_{max}, *di*}`]` evaluates *expr* with *i* successively taking the values i_{min} through i_{max} in steps of *di*.

- `If[`*condition*, *t*, *f*, *u*`]` evaluates *t* if condition evaluates to True, *f* if it evaluates to False and *u* if it evaluates to neither.

Calculus

- See `D` to differentiate.
- Can do both definite and indefinite integrals – see `Integrate`
- For a numeric approximation to an integral use `NIntegrate`.

Creating tensors

- There are many different ways to handle tensors in Mathematica.
- Lists are enclosed in braces { }, with the elements separated by commas.
- They can have symbolic or numeric entries.
- `Table` is most appropriate for creating 1D lists, where the entries are calculated according to a specified rule.
- Nested lists can be used to create tensors
  - use `Array` (or `SparseArray`) to do this
  - elements may be specified when the array is created by using `Function` or later on

Tensor operations

- To extract elements use `Part` or `[[ ]]`
- To append elements to lists, delete elements etc., see `Append`, `Delete`, `ReplacePart`
- Can change the number of levels in a list using `Flatten` or `Partition`
- Vector specific operations: `Dot`, `Cross`, `Norm`
- Matrix specific operations: `Inverse`, `Det`, `Eigensystem`, `RowReduce`
- Even more impressive: `SingularValueDecomposition`, `JordanDecomposition`

Equation solving

- Use `Solve` to solve an equation with an exact solution, including a symbolic solution.
- Use `NSolve` or `FindRoot` to obtain a numerical approximation to the solution.
- Use `DSolve` or `NDSolve` for differential equations.
- To use solutions need to use *expr*/ $.x \longrightarrow y$.

Importing/exporting data

- Need to set your working directory – see `SetDirectory`.
- To import data use `Get`, `OpenRead`, `ReadList` or `Import`.
- To export data use `Put` or `Export`.

Graphics

- Mathematica allows the representation of data in many different formats:
- 1D list plots, parametric plots
- 3D scatter plots
- 3D data reconstruction
- Contour plots
- Matrix plots
- *Pie charts, bar charts, histograms, statistical plots, vector fields (need to use special packages)*

- Numerous options are available to change the appearance of the graph.

- Use Show to display combined graphics objects

Using packages

- Sometimes you may want to use specialist packages that are not automatically loaded when you start a session.

- Use `Needs`.

Optimisation

- Facilities for numeric and symbolic, global and local, constrained and unconstrained optimisation.
- Numeric:
  - local – `FindMinimum`, `FindMaximum`
  - fitting - `FindFit`
  - global - `NMinimize`, `NMaximize`
- Symbolic: `Minimize`, `Maximize`
- The above functions have been updated for Mathematica 6.0.

Taking it further

- Mathematica has an excellent help menu (shift-F1)
- Can get help within a notebook by typing ?FunctionName
- Website:

  http://www.wolfram.com/products/mathematica/index.html
- To use Mathematica for parallel programming, look up gridMathematica.