12.010 Computational Methods of Scientific Programming

Lecturers
Thomas A Herring
Chris Hill

Mathematica

• History
  – Developed between 1986-1988 at Wolfram Research
  – Mathematica 1.0 released in 1988
  – Mathematica 2.0 released in 1991
  – Mathematica 3.0 released in 1996 (typesetting)
  – Mathematica 4.0 released in 1999 (performance)
  – Mathematica 5.0 current version (5.2 most recent release)
Basics of Mathematica

- Code developed for Mathematica can be generated while working in Mathematica.
- The Mathematica Note books (.nb extent to name) can be used to save this development
- When working in Mathematica, help files are available to guide usage and there can be instant feedback if there is a problem in the code.
- We will use a Mathematica Notebook in this class to demonstrate the ideas in the notes.

Mathematica Features

- Code (numerics, and control)
- Numerical calculations to arbitrary precision
- Symbolic calculations (algebra and calculus)
- Graphics
- Notebooks
- Several useful formats
  - command line
  - typeset equations
  - tabular data, and many more
- These features are demonstrated in the 12.010.Lec12.nb
Mathematica:

- Consists of two programs
  - "kernel" (does all the computations)
    - evaluates expressions by applying rules
  - "front end" (user interface and formatting)
  - Mathematica itself is written mostly in C
- Syntax follows rules, but errors are usually forgiving
- Basic Structure:
  - File types:
    - Mathematica code (end in ".m" by convention)
    - Mathematica notebook (end in ".nb" by convention)
  - Mathematica evaluates expressions by applying rules, both those that have been defined internally and those defined by the user, until no more rules can be applied.

Mathematica: Context of Use

- Mathematica notebooks can be used in research groups
  - beginning students need a place to start
  - graduating students leave a legacy
  - some alumni still contribute to Mathematica "packages"
- Upside
  - extremely powerful (integrated work environment)
  - dramatically decreases development time
- Downsides
  - slower number crunching (compile or link to C). Improves with each version.
  - memory (this has vastly improved)
  - single supporter of the language (Wolfram Research)
Mathematica Features

- Notebooks
  - easy to document work as you produce it
- State of the art numerical and symbolic evaluation
- Variable names usually say exactly what the variable is
  - not a problem, since a lot can be packed into a symbol
- Contexts
- Packages
- Link to C code for number crunching
- Typesetting (TeX)
- Conversion to Fortran and C-code
- Function arguments pass by value
  - more like mathematical notation

Conventions

- system symbols begin with upper case letter
- user symbols begin with lower case letter
- Function arguments are enclosed in [ ] (square brackets)
- Parentheses are used to assign precedence (normal use)
- { } are used to enclose lists (each item in list can be then acted on).
Basic Structure 02

– Variable types*
  • Integer (machine size or larger)
  • Rational (ratio of integers with no common divisors)
  • Real (machine double precision or larger)
  • Complex (machine double precision or larger)
  • String (can arbitrarily long)
  • Symbol
  • List (set of anything -- used more than Array)
    • virtually any other type can be defined
– Variable types tend to naturally get set by Mathematica and user does not need to be explicit.
The Head[ variable ] tells type of entity (see nb).

Basic Structure 03

– Constants: Numerical or strings, as defined by user; E, I, Pi, and others defined by the system
– I/O
  • Open and Close
  • Read (various forms of this command)
  • Write (again various forms)
  • Print (useful for debug output)
  • Can define how results are read and written.
– Math symbols: * / + - ^ (power) = ( immediate assignment) ::= (delayed assignment).
  Operations in parentheses are executed first, then ^, /, and *.
  + - equal precedence.*
Basic Structure 04

– Control
  • If statement (various forms)
  • Do statement (looping control, various forms)
  • Goto (you will not use in this course)
– Termination
  • Nothing special, just the last statement
– Communication between modules
  • Variables passed in module calls. One form:
    – Pass by value (actual value passed)
  • Global variables
  • Return from functions
  • Contexts isolate variables of the same name (see NB). Contexts define areas where variables are separated. Useful way to avoid “clobbering” values in rest of program.

Syntax

• Free form
  – Case is not ignored in symbols and strings
  – Spaces are interpreted as multiplies
  – ; at end of a line suppresses echoing of a result
    • must use at end of statements in Module, except for the last
  – Comments are enclosed in (* .... *)
Compiling and Linking

- Source code is created in Mathematica or a text editor.
- To compile and link: (not necessary)
- Mathematica code needs to run within Matheamatica. There is MathReader that allows notebooks to be read without the need to buy Mathematica. (These notebooks can not be changed).
- It is possible to convert Mathematica expressions into C and Fortran code (declarations need to be added).

Details on commands

- Functions can be defined with the structure (see NB):
  \[ h[x_] := f(x) + g(x) \]
  would define a new function \( h \) that is equal to function \( f(x) + \) function \( g(x) \). These functions are symbolically manipulated.
- Modules are invoked by defining Module and assignment statements for functions.
Subroutines (declaration)

name[v1_Type, …] := Module[{(local variables), body}
Type is optional for the arguments (passed by value)

• Invoked with
  name[same list of variable types]

• Example:
  sub1[i] := Module[{s}, s = i^2 + i^3; Sqrt[s]]

In main program or another subroutine/function:
  sum = sub1[i]

Note: Names of arguments do not need to match those
used to declare the function, just the types (if declared)
needs to match, otherwise the function is not defined. *

Functions: Comparison

Fortran
Real*8 function func(list of variables)

• Invoked with
  Result = func(same list of variable types)

• Example
  Real*8 function eval(i, value)
  Integer*4 i
  Real*8 value
  eval = i*value

In main program or subroutine or function
  Real*8 result, eval
  Integer*4 i
  Real*8 sum
  Result = eval(i,sum)

Mathematica
func(list of variables)

• Invoked with
  result = func(same list of variable types)

• Example
  eval[i,_,value_] := i*value
  OR
  eval[i,Integer,_,_,Real] := i*value

In main program or subroutine or function
  result = eval[i,sum]
Functions 02

• Functions can return any of the variable types
• The function name is a symbol
• The function must always appear with the same name, but other names can be defined in desired.

Intrinsic functions

• These functions are embedded in the language and often go by "generic names." Mathematica has MANY of these (check out the Help under "Built in Functions")!
• Examples include Sin, Cos, Tan, ArcTan. Precisely which functions are available are machine independent.
• If a function is not available: function called is returned unchanged (i.e. function[x])
Using Mathematica

• On the MIT server (X-window interface)
  – MIT server% add math; mathematica &
  – On a machine with Mathematica installed this should be fine but if windows are displayed on a generic X-windows system, the fonts often to not appear correctly
• On the MIT server (tty interface)
  – add math; math
  – Graphics and “neat” looking symbols do not appear (π will appear as Pi rather than π).

Summary

• Introduction to Mathematica and use of notebooks.
• Since Mathematica is a self contained environment, help is readily available.
• Use of the Mathematica Help:
  – When looking at functions etc; look of examples at the bottom this is often a good way to get an idea of how to use the function. Eg., under numerical computations, equation solving, NDSolve examples of solving differential equations (hint: Question 3 of the homeworks, is the solution to an ordinary differential equation)