12.010 Computational Methods of Scientific Programming

Lecturers
Thomas A Herring
Chris Hill

Mathematica

- Look in more detail at some of the programming features in Mathematica
- There are many of these features and in all Mathematica expressions there are Function names and “short-hand” symbols
- The + usage is actually a function Plus, * is Times
- Use of FullForm shows full form of expressions
- Examples in http://geoweb.mit.edu/~tah/12.010/12.010.Lec13.nb

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 + i^2 + i^3; Sqrt[s]]

In main program or another subroutine/function:

sum = sub1[j]

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 j

Real*8 sum

Result = eval(j,sum)

Mathematica

func[list of variables]

- Invoked with

  result = func[same list of variables]

- Example

  eval[i_,value_] := i*value

  OR

  eval[i_Integer,value_Real] := i*value

In main program or subroutine or function

result = eval[j,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])

Flow control

- If statement form:
  If[condition, t, f] gives t if condition evaluates to True, and f if it evaluates to False.
  If[condition, t, f, u] gives u if condition evaluates to neither True nor False.
- The standard conditions tests are ==, !=, <, <=, >, >=
- Multiple test are && (and) || (or)
- It also possible combine:
  if[7 > 6 > 5, ..] rather than if[7 > 6 && 6 > 5, …]
- Which allows a range of actions:
  Which[test1, value1, test2, value2, test2, value2]
- Switch allows action based on result of expression:
  Switch[expr, form1, value1, form2, value2]

Loop structures

- Do structure: Most general structure
  Do[*expr*, {i, imin, imax,di}, {j, jmin, jmax,dj}, …]
  This would loop through values of j from jmin to jmax in increments of dj, for each value of i which would loop from imin to imax in increment of di.
- If the increment is not given 1 is assumed, if imax is not given, then loops from 1 to imin. If only 1 argument is given, expr is evaluated that many times.
- While[test, *body*] executes code in body (statements are separated by ;) while ever test is true.
  Return[val] can be used to return a value from the *body* code;
  Break[]can be used to exit body
- For[start, test, incr, *body*] executes start, then repeatedly evaluates body and incr until test fails to give True
- Mathematica does have a Goto[tag] statement using Label[tag]

Functions

- Function[body] or body& is a pure function. The formal parameters are # (or #1), #2, etc.
- Function[x, *body*] is a pure function with a single formal parameter x. Body can have mulitple statements separated by ;
- Function[{x1,x2,… }, *body*] is a pure function with a list of formal parameters.
- If the body is more than one statement, normally there would be a Return[..] call to set the quantity returned form the call.
- Map[f, *expr*] or f /@ expr applies f to each element on the first level in expr.
- Apply[f, expr] or f @@ expr replaces the head of expr by f. This is basically a way of changing what something is in Mathematica e.g., if expr is a list {…}, it can be changed to Times (multiply)

Pattern Matching

- _ or Blank[ ] is a pattern object that can stand for any Mathematica expression.
- _h or Blank[h] can stand for any expression with head h. We used this in an earlier lecture to make x_Integer for an integer argument.
- __h or BlankSequence[h] can stand for any sequence of one or more expressions, all of which have head h.
- g[x_, y__] := x + y; g[a, b, c] yield a+b+c
- Replace and Rules: -> (arrow on Palette) applies a rule for to convert lhs to rhs, /. is the replace all e.g.
  1 + x /. x -> a yields 1+a (same as ReplaceAll[1 + x, x -> a])
- There are many more forms of rules and replacements that are given in the Pattern Matching and Rule applications in the Programming section of the Mathematica help.

Format types

- Mathematica offers many different types of ways to display results and convert to different formats
- These are given in the Format Types under Input Output sections of the Built in Functions
- Some examples are
  TableForm, MatrixForm, TreeForm
- N[expr] gives the numerical value of expr.
- N[expr, n] attempts to give a result with n-digit precision.

Files and directories

- Directory[ ] - give your current working directory
- SetDirectory["dir"] - set your current working directory
- FileNames[ ] - list the files in your current working directory
- FileNames["form"] - list the files whose names match a certain form
- <<name - read in a file with the specified name (Get)
- <<context` - read in a file corresponding to the specified context
- CopyFile[“file1”,”file2”] - copies file1 to file2
- DeleteFile[“file1”] - deletes the file.
- Input[“prompt”] is used to read information from the keyboard

Graphics

- Mathematica supports a variety of graphics plots through is basic plot command.
- Simple plots can be modified with options given in the plot command.
- Mathematic 6.0 and above has a new Manipulate command
  Syntax of command: The variable a here is the one that can be manipulated between values of 0 and 2.
  Manipulate[Plot[Sin[x (1 + a x)], {x, 0, 6}], {a, 0, 2}]

Final Comments

- Users of Mathematica need to understand the basics of the syntax of the program. The online help however provides the details of the capabilities of the program
- Built-in Functions is grouped by
  Numerical Computation
  Algebraic Computation
  Mathematical Functions
  Lists and Matrices
  Graphics and Sounds
- Program development should be knowing what you want to do and then finding the Functions that, in combination, will do the task.
- With Notebooks, you can keep track and comment on the way the program works.
- Homework #4 will be due Thursday Nov 17, 2011.

12.010 Computational Methods of Scientific Programming
Fall 2011