MS6021 Scientific Computing

Fortran and Matlab for Mathematical Modelling.

Aimed at the absolute beginner.

william.lee@ul.ie

Hi

- William Lee.
- Email: `william.lee@ul.ie`
- Web: `www.ul.ie/wlee/ms6021.html`
- Room: A2016a

Timetable

- Mondays, 5pm, C2062.
- Tuesdays, 3pm, C2062.
- Thursdays, 5pm, SR2030.

Office hours ??? A2016a.

Textbooks

- Matlab Guide. Higham and Higham.
- Introduction to Fortran 90/95. Chapman.

Outline

- Introduction Course introduction. Matlab. Fortran. Datatypes and control structures. Functions, procedures and subroutines and modules. Inputs and output.
- Linear Algebra Linear algebra review. Simple linear system. Factorisations. Overdetermined and underdetermined systems. Singular value decomposition. Sparse systems. Eigensystems review. Solving eigensystems. Linear algebra case studies.
- Nonlinear Equations One dimensional root finding. Multidimensional root finding. Optimisation.

Outline

- Ordinary Differential Equations Ordinary differential equations review. Runge-Kutta algorithms. Boundary value problems. Singular perturbations and boundary layers. Stiff solvers. Richardson extrapolation.
- Partial Differential Equations Partial differential equations review. Parabolic and elliptic equations. Hyperbolic systems.

Outline

- Mathematical Modelling Introduction to mathematical modelling. Dimensional analysis and nondimensionalisation. Asymptotic simplification. Case study I. Case study II. Case study III.
- Toolboxes Matlab PDE toolbox. NAG library. Making your own toolbox.

Assessment

- 2 miniprojects each worth 50% of marks
- Set in weeks 5 and 11, two weeks to complete
- Marked on code and LaTeX writeup.

Plagiarism

- Discussing ideas and algorithms is fine.
- Discussing/copying code without citation isn't.
- If I can't reproduce your results from your code I may ask you to demonstrate it for me.

Anatomy of a Mathematical Modelling Project

- A new/poorly understood phenomenon/process
- Analysing/visualising data
- An idea about the science behind it
- Rough estimates: is it plausible?
- The equations describing the system
- Nondimensionalisation: which terms are important?
- Analysis: what do solutions look like?
- Numerical solution. Were we right? Figures?

Don't

We tend to model to gain insight into problems rather than for numerical accuracy.

- Don't carry out a simulation unless you know what the answer will be (roughly).
- Never do a simulation to just see what will happen.

I have stronger feelings on this than my colleagues, who will have their own ideas on this..

- Most of the time (for the kind of problems I work on) the question is: *what are the right ingredients to put into the model?*
- By the time you hit the `GO` button on the computer you've already chosen, or had chosen for you, the model ingredients.

Why Matlab and Fortran?

- Really because Profs O'Brien (Fortran) and Gleeson (Matlab) designed the course
- But Matlab and Fortran make a good combination.
- Matlab High level. Interpreted. Graphics. Prototyping
- Fortran Low level. Compiled. Type safe.

Other languages

- Low level. C.
- Between. C++. Java.
- High level. Unix shells. Octave (matlab clone). Python (everything!). R (statistics). Mathematica (expensive). Perl. Ruby. Scheme. Tcl. PHP.

Octave is worth looking into. Free fortran compilers (g95, gfortran) are easy to find. Matlab isn't free.

Matlab: what we won't cover

- Graphical user interfaces. I don't like them because they make it hard for programs to interact.
- Object oriented Matlab programming.
- Linking matlab to compiled fortran/c.

Style

- Put lots of comments in your code
- Clarity more important than compact code.
- Split problem up into components to be handled by separate (reusable) functions/subroutines

Clarity

$j = 0, 1, 2 \longrightarrow k = 1, 2, 0$

```
k=(2-j)*(1+3*j)/2  ! Too obscure

k=j+1              ! About right
IF(k=3)  k=0
```

- It is surprising how hard it is to understand code you've written a even few weeks ago.
- This is normally the timescale on which you start to become worried about the results your code is outputting.

Maybe too much, maybe not too.

```
IF(j.eq.0) THEN
  k=1
ELSE IF(j.eq.1) THEN
  k=2
ELSE IF(j.eq.2) THEN
  k=0
ELSE
  PAUSE "Never get here"
END IF
```

Write code a lot less complex than the most complex code you can understand. (And for this course write code it is easy for me to understand.)

The computer is not on your side

- Garbage in: garbage out
- Computers do what you tell them to do
  not what you want them to do
- Modern, expensive simulation packages will happily produce beautiful, compelling but totally wrong pictures and movies.

Test your programs on cases you can solve analytically. Never do a simulation unless you know (roughly) what the answer will be.

Course Style

- 2 ways to teach a course like this: (1) lots of details on algorithms, (2) how to do x using black boxes.
- Try to do a bit of both.
- Mostly worksheet based rather than lectures.

Why not just use black boxes?

- Sometimes you need to know how an algorithm works to understand what it is doing (wrong).
- Specialist mathematical modellers normally called for when the standard algorithm doesn't work or there is no black box.
- Sometimes you have to use or hack tools produced by non-reputable sources.

Matlab

- Interactive system for numerical computation
- Written by Cleve Moler late 1970s in FORTRAN (now in C)

Matlab Advantages

- High level language - can do a lot with a few statements
- Easy data structures - arrays created and extended automatically
- Interactive - easy to experiment with and debug code
- Graphics
- M files - platform/operating system agnostic
- Toolboxes (commercial, free or homemade) add domain specific capability to matlab.
- Lots of free m-files available over Internet

Matlab Disadvantages

- Interpreted and therefore slow
- Expensive: detailed knowledge of matlab useless if you work somewhere that can't afford it or prefers mathematica
- Closed source
- Only one function per M-file
- Can't protect constants: pi=4 accepted
- Encourages trust in black boxes

Worksheet 1: Hello Matlab!

Tomorrow!

william.lee@ul.ie