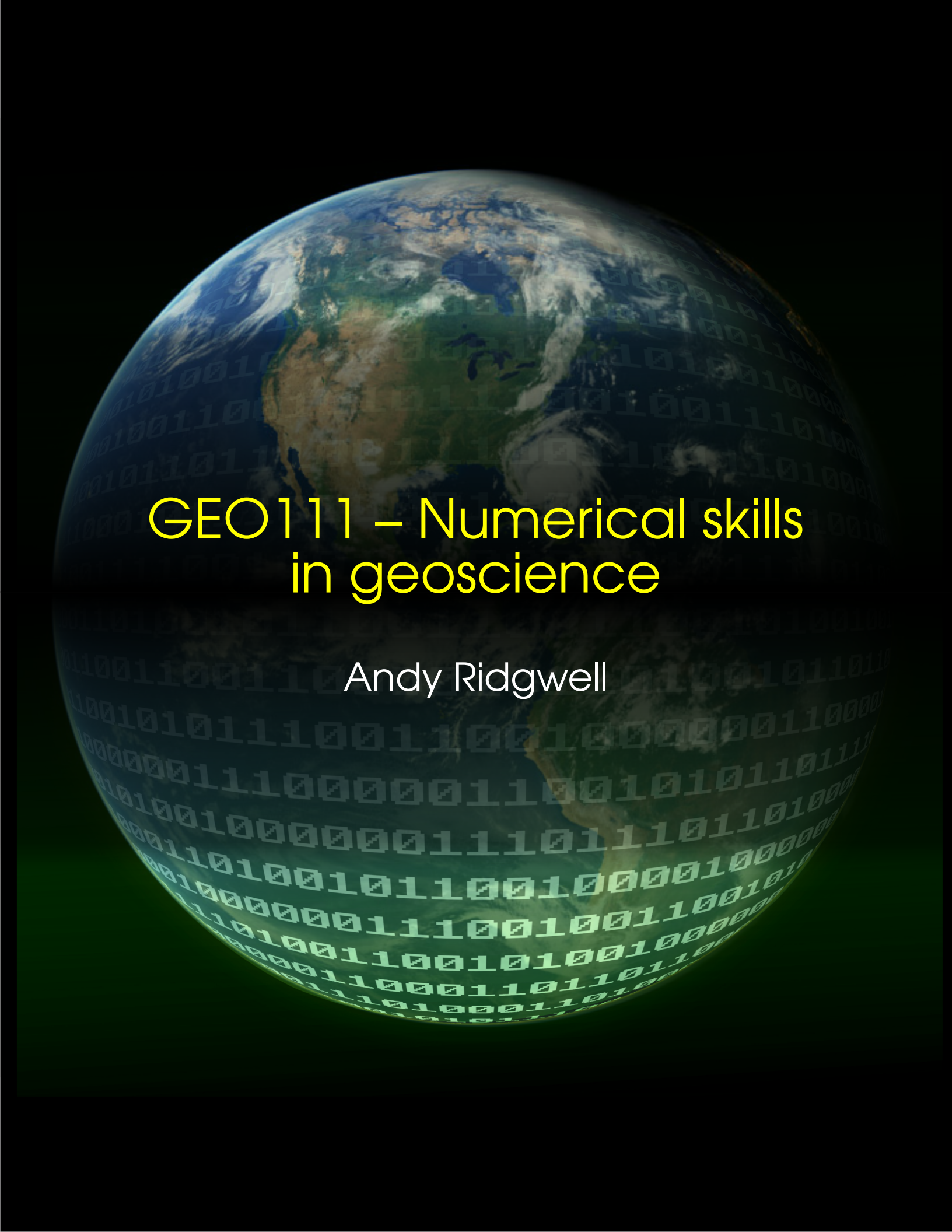


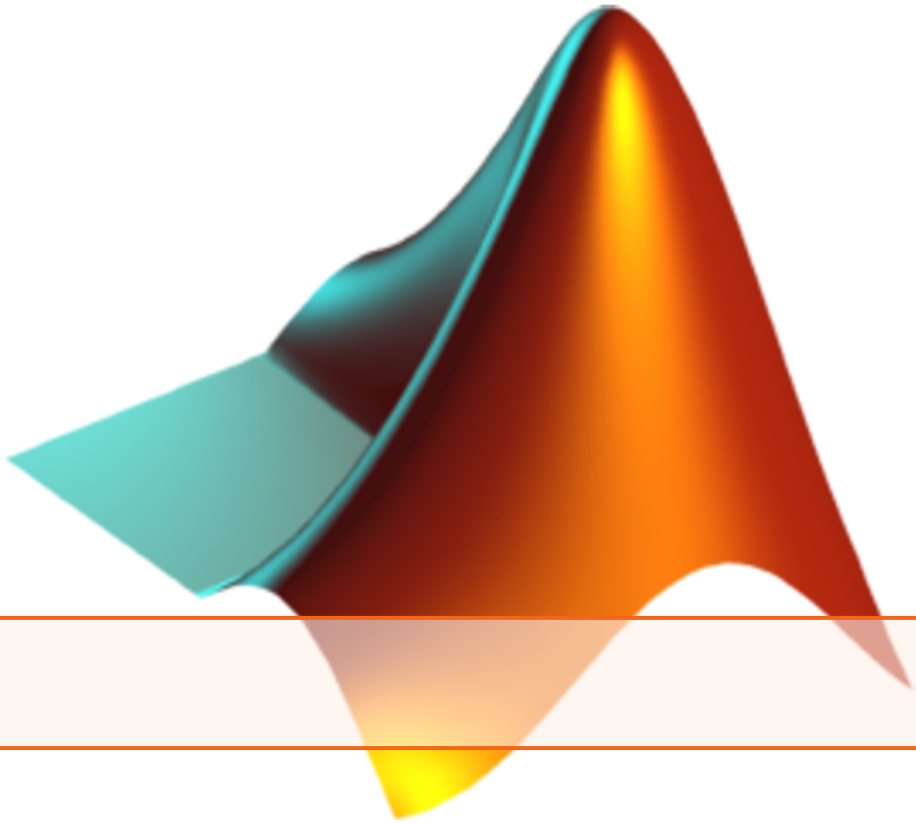
GEO111 – Numerical skills in geoscience

Andy Ridgwell

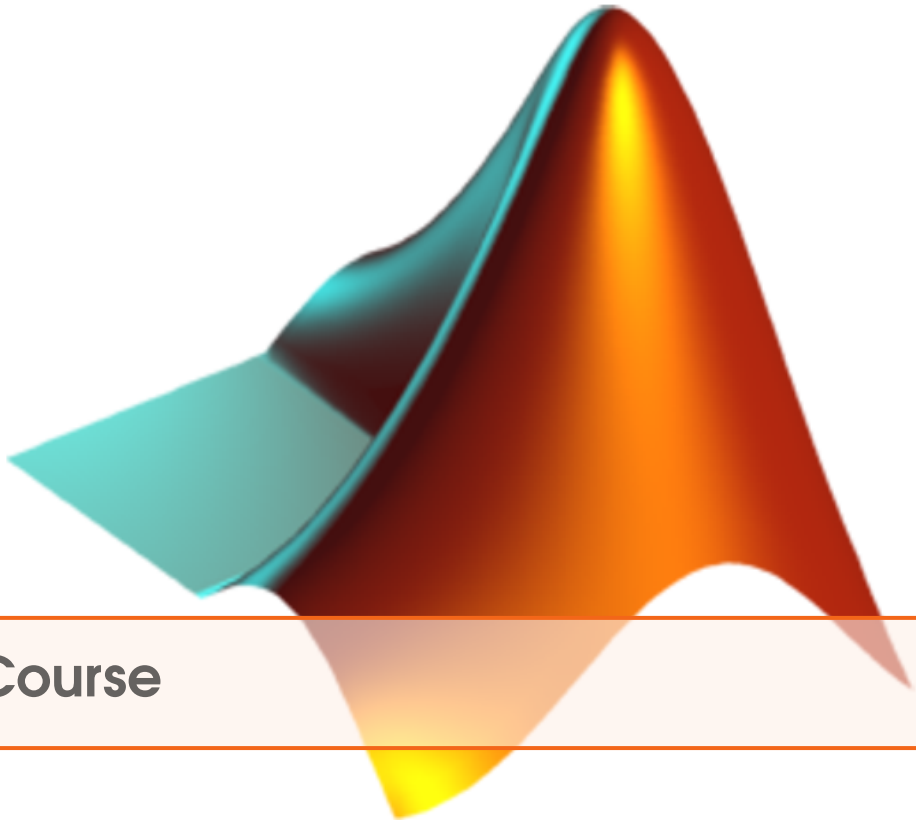
PUBLISHED BY DERPY-MUFFINS INC.

HTTP://WWW.SEAO2.INFO/TEACHING.HTML

First printing, October 2017



Contents



About the Course

GEO111 will provide an introduction to computer programming and numerical modelling (with a focus on Earth and Environmental Science problems). It will provide a chance to learn a computer programming language and all the elements that constitute it, including concepts in number bases and types, logical constructs, debugging, etc. The course will develop programming skills step-wise, intertwining them with practical questions and outcomes, such as in data processing and visualization. How complex environmental processes can be encapsulated and approximated, and numerical models thereby constructed, will be illustrated.

The cumulating objectives of the course are to:

1. Provide hands-on training in how computer programs are written and numerical models constructed.
2. Develop both general (transferable) as well as specific numerical and analytical skills applicable to the Earth and Environmental Sciences.
3. Develop an understanding of how computers and the internet work, and how programs are constructed and interfaced, and hence foster a critical understanding of modern technology.

The associated learning goals are firstly; to provide, through hands-on practical exploration, factual knowledge and an understanding of:

- The basic building blocks of computers and computer programs, and how they overall 'work'. (Learning Outcome 2).
- How numerical models are constructed and applied. (Learning Outcomes 1 and 2).
- Basic climate processes. (Learning Outcome 1).
- The use of numerical models in addressing scientific questions and testing hypotheses as well as the limitations of numerical models. (Learning Outcomes 2 and 4).

and provide transferable skills in

- Written communication and presentation. (Learning Outcome 3).
- Problem solving and logical analysis, fault-finding. (Learning Outcomes 4 and 5).
- Computer programming. (Learning Outcomes 2 and 4).

Course logistics

Format

The weekly format of GEO111 is: 1 $\times$ 1-hour lecture together with 1 $\times$ 3-hour computer practical session (both in Watkins 2101), plus 1 $\times$ 2-hour interactive lecture/discussion session of worked problems and examples (also in Watkins 2101). The computer practical class is the central element, and will consist of structured exercises leading step-by-step through the components of computer programming and numerical model construction, debugging, and testing, plus applications to common geosciences problems. The lecture starting each week will outline the basics and introduce the key concepts of the week. The purpose of the 2-hour lecture/discussion session ending the week is to ensure all the concepts are understood and misconceptions resolved and will be a mix of presentation and worked-through examples, plus questions and discussion.

In between the Monday and Friday classes, there will be some homework :(This will be assessed and needs to be completed in a timely fashion (i.e. there is a deadline). It will not be unduely time-consuming or difficult. You will be earning valuable genuine marks (towards the final grade).

Finally, to ensure that we keep to schedule and cover all the planned material, please aim to complete work not finished in the Monday lab session prior to Friday, if possible. If stuck, please make use of Office Hours (see below). Friday is scheduled as time available to address problems etc., but you should not count on it as time solely to finish up uncompleted Monday work as additional work will be set and further topics introduced each Friday.

Timetable

The timetabling and overall course structure of GEO111 are given in Table 1.

Assessment

The course will be assessed as follows:

- 8 $\times$ weekly micro assessments @ 2.5% each = 20%
- Midterm paper – 30%
- Finals paper – 50%

In the (8) weeks with no UCR Friday holiday, a short (micro) assessment will be set. The hand-in date will be Friday at noon (PST) the same week as the assessment is set. Each micro assessment will be worth 2.5% of the total marks available for the course. The assessment will be set by noon (PST) on the Monday of that week. The idea is to help reinforce what you should have been learning in the Monday lab. By putting in a modicum of effort, it also enables you to accumulate some marks towards the course and hopefully take a little stress off of the Examinations. Or alternatively, utterly amplifying the stress if you don't bother completing the micro assessment ...

The Mid-term paper will be a written exam, consisting of a mixture of multiple choice and short-answer format questions. Its purpose is to test basic knowledge of computers and programming, plus general concepts and basic commands you have come across in MATLAB. The testable content will comprise the material covered in the lectures and lab sessions (in weeks #1-5). The exam will be 2 hours long. No study aids of any sort will be allowed. The mid-term paper will constitute 30% of the total assessment for the course.

The Finals paper will be an on-line / at computer exam. Answering the questions will require a mixture of: writing short (1 or 2 line) code fragments, completing or debugging provided program code, and writing complete programs. Study aids (course text and handouts, textbooks, lecture

notes etc.) plus MATLAB 'help' and on-line documentation are allowed (but no internet!). This will constitute the remaining 50% of the total assessment of the course.

Office Hours

There are no specific Office Hours, but rather an open invitation to drop by[1] (excluding Thursdays) and/or email[2] questions. Wednesdays (almost any time) would be your best chance of finding me (and not busy).

Note that a large part of the purpose of the lecture/discussion/lab session on Fridays is to provide an opportunity for further clarification of the course material and to go through worked examples. So please feel free to treat the Friday class as a kind of group tutorial session.

Please note that programming may be completely new to almost everypony in the class. It may well be something unlike anything you have taken classes in before and hence how to go about 'learning' it may not be obvious. So please do not hold back on questions – no question is too stupid[3]! Or rather, given the likely newness to you and total weirdness of programming, you are not stupid and so no question you could ask can be stupid.[4]

Course text

There is no one (or even two, between them) commercial (published) course texts that covers both basic computer programming and numerical modelling at a suitable introductory level, and certainly not in the context of MATLAB. Hence the reason for this *e*-book – to provide a 1-stop shop for a range of information and practical tutorials in useful and commonly used data manipulation and visualization, numerical techniques, and programming methodologies.

Note that I will be revising the text as we go, and the revised chapters will only be posted immediately prior to each class. These will appear (PDF format) on my teaching webpage[5]. However, a copy of the complete course text used in 2016/17 will be made available and will serve you as a fair guide to the 2017/18 year content.

In conjunction with the course text (this document), if you were to work through any commercial textbook, I would recommended (but remember it is **not** required): *Matlab (Third Edition): A Practical Introduction to Programming and Problem Solving*[**Attaway2013**], which provides a good general introduction to MATLAB and covers a similar range of material to much of the course.

For additional reading (on both MATLAB and numerical modelling), you might also try:

- *The Climate Modelling Primer (4th Edition)*, by Kendal McGuffie and Ann Henderson-Sellers. Wiley-Blackwell (2014). ISBN: 978-1-119-94336-5.
- *Introduction to MATLAB (3rd Edition)*, by Delores M. Etter. Prentice Hall (2014). ISBN: 978-0133770018.
- *Mathematical Modelling of Earth's Dynamical Systems, A Primer*, by Rudy Slingerland and Lee Kump. Princeton (2011). ISBN: 978-0-691-14514-3.

[1] My office is in the Geology building, room 464 (basement floor).

[2] andy@seao2.org

[3] In the context of MATLAB and programming that is. The current political situation gives rise to questions at a level of stupidity completely off the scale.

[4] Instead, some of the programming syntax in MATLAB is genuinely stupid.

[5] http://www.seao2.info/teaching.html

Mid-term

Format of exam

What might be examined on?

1. Anything covered in the course text.
2. Anything covered in lectures.
3. Anything covered on white board or discussed during the lab.

With the exception of (i.e. things that will *not* be tested):

- Logic gates (but note that logic and its implementation in MATLAB *is* included).
- The details of algorithms.
- Detailed usage of MATLAB commands and syntax.

You will not be expected to write complete and/or working programs on paper, although you might be lightly tested on the MATLAB commands that you have seen and their general/conceptual usage.

Logistics

The Midterm is nominally scheduled from 2.10 through 3 pm on Friday 3rd November, 2017. However, you are allowed up until the official end of class – 4 pm, if you wish.

No-one will be allowed to start if arriving later than 2.30 pm. The earliest anyone is allowed to leave is 2.30 pm.

No study aids or reference materials of any sort are allowed, including, but not limited to:

- Notes.
- Textbooks.
- Laptops or desktop computers (and help / internet search thereon).

All computers, 'phones, tablets, and other computer devices must be switched off and left off for the duration of the exam.

The exam will be paper-based, and the format will be of short (single word or number to up to 1 sentence) answers, and multiple choice questions.

The Midterm will account for 30% of the total marks for the course.

Finals

Format of exam

What might be examined on?

1. Anything covered in the course text.
2. Anything covered in lectures.
3. Anything covered on white board or discussed during the lab.

The exam will be computer-based, and the format will be of single or multiple written lines or code and m-files.

Logistics

The Finals is nominally scheduled from 2.10 through 4 pm on Friday 8th December, 2017. However, you are allowed up until the end of the room booking – 5 pm, if you wish.

No-one will be allowed to start if arriving later than 2.30 pm. The earliest anyone is allowed to leave is 2.30 pm.

Study aids and reference materials are allowed, including:

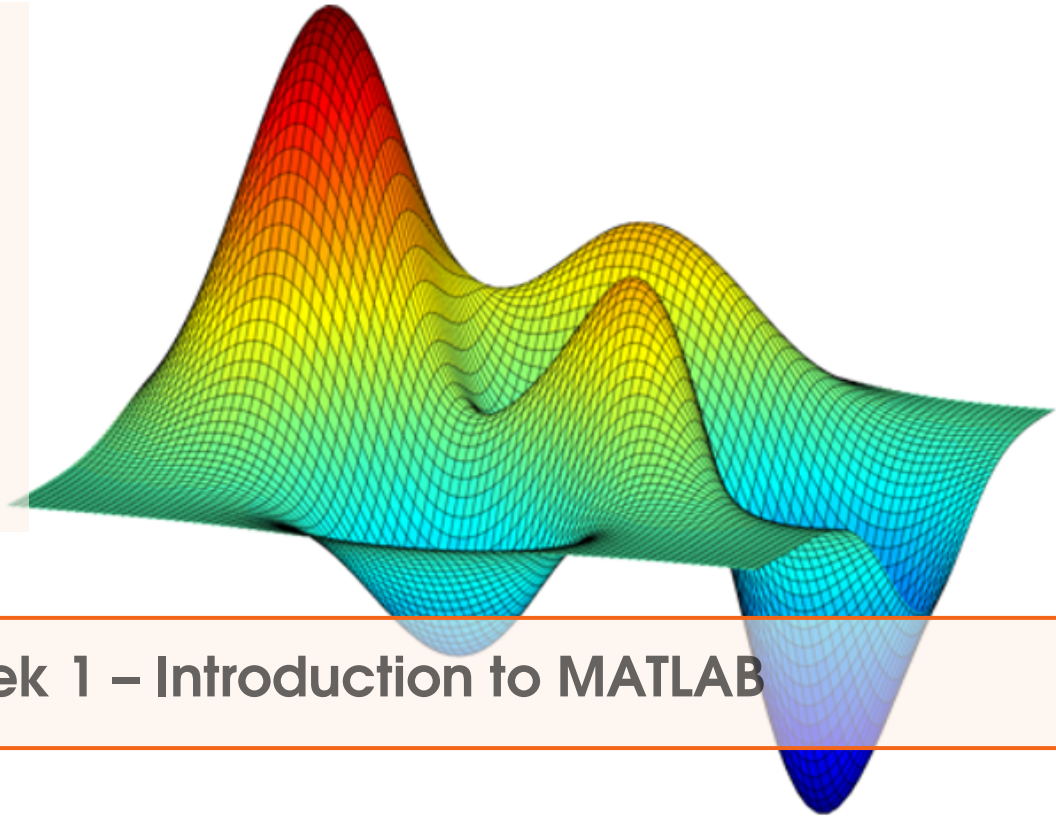
- Notes.
- Textbooks (including the GEO111 course text).
- **MATLAB** Help.
- Anything on the course webpage.

However, internet searches outside of the Mathworks domain are not allowed.

The Finals will account for 50% of the total marks for the course.

Table 1: Schedule of GEO111

Week #	**Monday lecture**	**Monday LAB**	***micro assessment***	**Friday LAB**
01 (10/02)	Introduction to the course & MATLAB.	Elements of MATLAB and data visualization.	1. Practice in math and using arrays in MATLAB.	(CONTINUED)
02 (10/09)	Fundamentals of computer programming I.	Scripts and functions in MATLAB. Loops and conditionals.	2. Writing a simple text-based program.	Further programming / elements and structures.
03 (10/16)	Fundamentals of computer programming II.	Further MATLAB and data visualization.	3. Plotting exercise.	(CONTINUED)
04 (10/23)	Algorithms and problem-solving.	Further programming tricks and techniques in MATLAB.	4. Data processing exercise.	(CONTINUED)
05 (10/30)	How computers work.	Statistics in MATLAB.	5. Fun with number bases.	**Mid-term**
06 (11/06)	Introduction to numerical modelling.	Basic (zero-D) numerical modelling.	NONE	UCR HOLIDAY
07 (11/13)	Further numerical modelling – techniques and applications.	'Daisy World'.	6. Experiments with the Fate of the Earth.	'Daisy World'.
08 (11/20)	Computer program Graphical User Interfaces(GUIs) and GUIing in MATLAB.	Basic GUI creation in MATLAB.	NONE	UCR HOLIDAY
09 (11/27)	Time-stepping in numerical models.	Dynamic (time-stepping) modelling – ballistics 101.	7. The 'boiling kettle' model(!)	Dynamic modelling CONTINUED – climate change.
10 (12/02)	Computer networks and the internet; web pages and basic html scripting.	Putting it all together – GUI-based programs.	8. Creating a webpage.	**Finals**



1. Week 1 – Introduction to MATLAB

1.1 Work plan

Work through Chapter #1 of the GEO111 course text – aim to complete Sections 1.1 through 1.4 during the Monday am lab, and Sections 1.5 through 1.8 during the Friday pm class. If you complete 1.1-1.4 before the end of the Monday lab, start on the Micro Assessment. (Note that Sections 1.5-1.8 are being updated and will not be ready much before Friday pm.)

There is a lot of stuff crammed in here and it would be easy to get lost in a mire of commands and instructions. So here is a brief guide to what you will ne doing/seeing as you go through Chapter #1:

1. **Section 1.1.** Firstly ... just get familiar with the software window(s) that appear. (HINT: make the MATLAB program window full screen so that you can see properly what is going on. PDF instructions etc. could be opened the monitor of a 2nd PC, or your laptop (or vice versa).)
2. **Section 1.2.** Some important basic stuff about what a variable is and the different types of variables. Also how you assign some information to a variable name (and read it back out again). There are some lists of expressions and operators ... some of these will be familiar, and some not. For now: simply note the existence of the non-familiar ones (we'll come back to them when we need them).
3. **Section 1.3.** Vectors ... there is a steeper learning curve here. It is important to understand quite what they are and how to select ('address') specific elements from them. Conceptually, this is the biggest step to take in all of MATLAB. The colon operator is key here.
4. **Section 1.4.** Some light relief and basic (line) plotting.
5. **Section 1.5.** Another big step and matrices (2D arrays). Again – how you select elements and entire rows of columns, is key understanding. Arrays (matrices and vectors etc), how they are represented and used in MATLAB is the most single difficult thing. It is all easier

after this!

6. **Section 1.6.** Basic loading and saving of data from/to files. Useful, and not too difficult. There are many ways of doing this – here is data input/output in its most simple incarnation. We'll see other ways later on.
7. **Section 1.7.** A few useful MATLAB commands will be introduced here (and some more later on in the course) that greatly help in data processing and later on, in programming. These techniques (sorting data, scaling data) are buried in a couple of 'real world' data examples.
8. **Section 1.8.** A little more on plotting – how to make your graphs nice! Also buried in here is some more practice in basic vector and array usage.

For additional/background reading:

The published textbook – **MATLAB®7 – A Practical Introduction to Programming and Problem Solving [*Attaway*, 2013]**

- Chapter 1 – *Introduction to Programming using MATLAB* (all)
- Chapter 2 – *Vectors and Matrices* (all)

1.2 Learning goals

Topics and methodologies you should be familiar with:

- variables and variable types
- vectors and matrices
- addressing (elements in) vectors and matrices
- basic transformations of vectors and matrices
- basic loading and saving of data and graphics
- basic data plotting

specific MATLAB commands you should be familiar with:

- numerical expressions (add, subtract, multiply, etc.)
- array addressing: the **colon operator**, `end`
- array related functions: `length`, `size`, `transpose` (or `.'`), `flipud`, `fliplr`, `sortrows`
- data related functions: `load`, `save`, `cd`, `addpath`
- plotting functions: `plot`, `scatter`, `pcolor`
- plotting related functions: `axis`, `title`, `xlabel`, `ylabel`
- misc: `sum`

1.3 Micro Assessment #1

Your assessment for this week ... is to work through the start of the **MATLAB®7 – Getting Started Guide**. The PDF version of this document can be found on the Mathworks website on the as well as on the course webpage.

Specifically, work through:

- Chapter 1 – *Introduction* (all).
- Chapter 2 – *Matrices and Arrays* (but you can skip the section *More About Matrices and Arrays*).
- Chapter 3 – *Graphics* (up to and including page 3-63).

Some of this repeats similar material to that already covered in the Monday am lab, and some is similar to material that will be covered in the Friday pm lab. This will all be helpful in reinforcing

the basic MATLAB concepts. In addition, Chapter 3 gives an alternative GUI-view of creating and editing scientific figures.

There may, or may not, be some sort of brief verbal 'test' on this material ...

```
function [a,b] = callKalmanFilter(position)

numPts = size(position,2);

a = zeros(2,numPts,'double');
b = zeros(2,numPts,'double');
y = zeros(2,1,'double');

% Main loop
for idx = 1: numPts
    z = position(:,idx);      % Get the input

    % Call the initialize function
    coder.ceval('kalmanfilter_initialize');

    % Call the C function
```

2. Week #02: Computer programming

2.1 Work plan

Work through Chapter #2 of the GEO111 course text – aim to complete Sections 2.1 through 2.3 during the Monday am lab, and Sections 2.4 through 2.6 during the Friday pm class. If you complete 2.1-2.3 before the end of the Monday lab, firstly ensure that you have completed everything in Chapter #1, then start on Section 2.4. (Note that Sections 2.4-2.6 are currently being updated ready for Friday ... but will not change out of all recognition.)

A brief guide as to what you will be doing/seeing as you go through Chapter #2 is as follows:

1. **Section 2.1. Introduction to scripting (programming!) in MATLAB**
Basic information about '*m-files*' – (plain text) code files used in **MATLAB**, and *script* files. Also some pointers to programming good practice and debugging code.
2. **Section 2.2. Functions**
What *functions* are in MATLAB and how they are used.
3. **Section 2.3. Conditionals '101'**
What the the *conditional* structure is, how it is used, and what the different forms this can take in **MATLAB**. Many many examples ...
4. **Section 2.4. Loops '101'**
What the the *loop* structure is, how it is used, and what the different forms this can take in **MATLAB**. Many many examples ...
5. **Section 2.5. Loops and conditionals ... together(!)**
Combining conditional and loop structures in the same program code.
6. **Section 2.6. Even more (and loopier) loops**
Further examples.

2.2 Learning goals

Topics and methodologies you should be familiar with:

- scripts and functions
- good programming practices
- debugging strategies
- conditionals
- loops

specific MATLAB commands you should be familiar with:

- the `function` definition
- conditional structures: (1a) `if ... end`, (1b) `if ... else ... end`, (1c) `if ... elseif ... else ... end`, (2) `switch ... case ...`
- loop structures: `for ...`, `while ...`
- misc: `disp`, `input`, `strcmp`

2.3 Micro Assessment #2

Your assessment for this week is firstly, to write 2 short programs:

1. Write a script program that asks for the current month (a string) at the command line, and displays at the command line, the number of days in that month. If the month given as input does not exist, a message should state this in some way.
2. Debug (i.e. get working) the following code, adding any comments you feel appropriate and making any 'improvements' you feel justified. The code is designed as a *function* ... it takes 2 inputs, `ip1` and `ip2` (numbers), and returns 2 outputs: `ans1` and `ans2` (also numbers). The first output (`ans1`) should be equal to `ip1` divided by `ip2`, and the second output (`ans2`) should be equal to `ip2` divided by `ip1`. If either input is equal to zero (a divide-by-zero situation), both outputs are meant to be set equal to zero.

```
function [ans1 ans1] = divide_both_ways(ip1,ip2)
% function to plot a sine wave
ans1 = ip1/ip2
ans2 = ip2/ip1
if ((ip1 == 0) && (ip2 == 0))
ip1=0
ip2=0
end
end
```

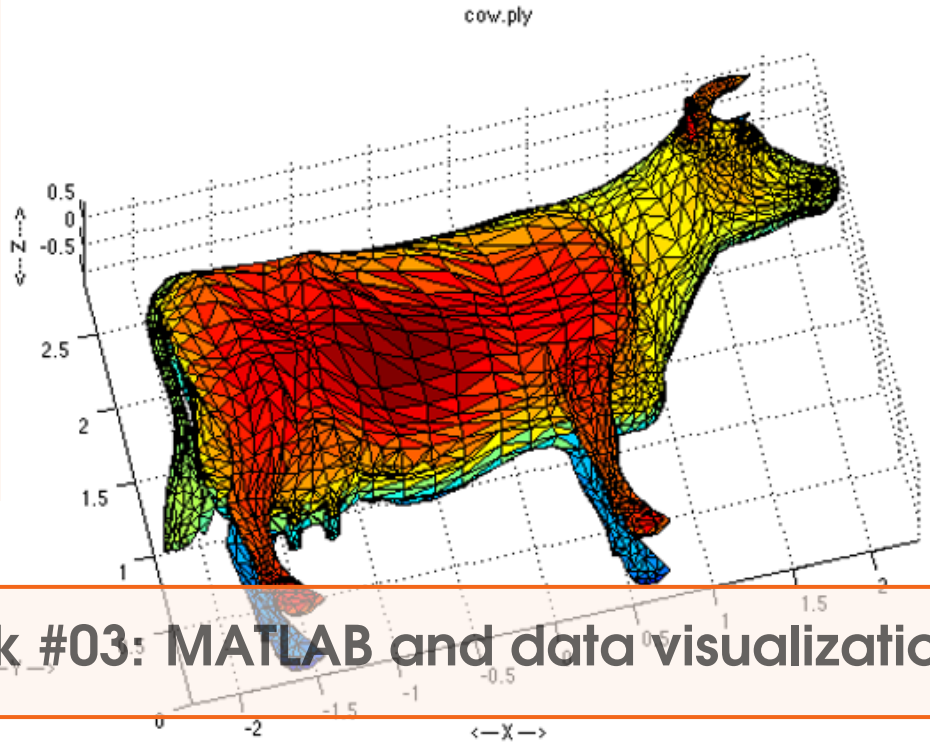
Submit by email (andy@seao2.org) as separate `.m` files. You will be marked for basic program structure and commenting of the code, in addition to the code correctly working! 80% of the total marks is available simply for working code that has adequate commenting. Maximum marks (100%) will be awarded for neat and compact code and/or clever solutions. Please include somewhere a comment line with your name in it to ensure there can be no file mix-up.

In addition,

3. Provide a figure (in any common graphics format you like) of past relative sealevel (see Section 1.7). Your graph should have a title that includes your name. It should be in the form of a scatter plot, with filled circles as the data markers. For maximum marks, the data points

should be color-coded by oxygen isotope value (not reconstructed sealevel). This should also be submitted by email.

The deadline (for the emails to appear in my in-box) is noon (12 pm) on Friday 13th October.



3. Week #03: MATLAB and data visualization

3.1 Work plan

Work through Chapter #3 of the GEO111 course text – aim to complete Sections 3.1 and 3.2 during the Monday am lab, and Section 3.3 during the Friday pm class. (Note that Section 3.3 is currently being updated ready for Friday.)

A brief guide as to what you will be doing/seeing as you go through Chapter #3 is as follows:

1. **Section 3.1. Further data input**

   Basically, a tour through different ways of importing data into **MATLAB** (other than the simple function `load` for pre-formatted numerical-only (or character-only) ASCII format data. Included are: mixed (numerical and text) ASCII data, **Excel** sheets, and a common scientific spatial format – *netCDF*.

2. **Further (spatial / (x,y,z)) plotting**

   This primarily comprises are series of Examples, taking you through more advanced 2D and contour plotting, including setting color scales and labelling contours.

3. **Further data processing**

   Additional data processing techniques including interpolation.

4. **Even nicer graphing and graphics**

   Drawing lines and shapes. Placing text in figures.

3.2 Learning goals

Topics and methodologies you should be familiar with:

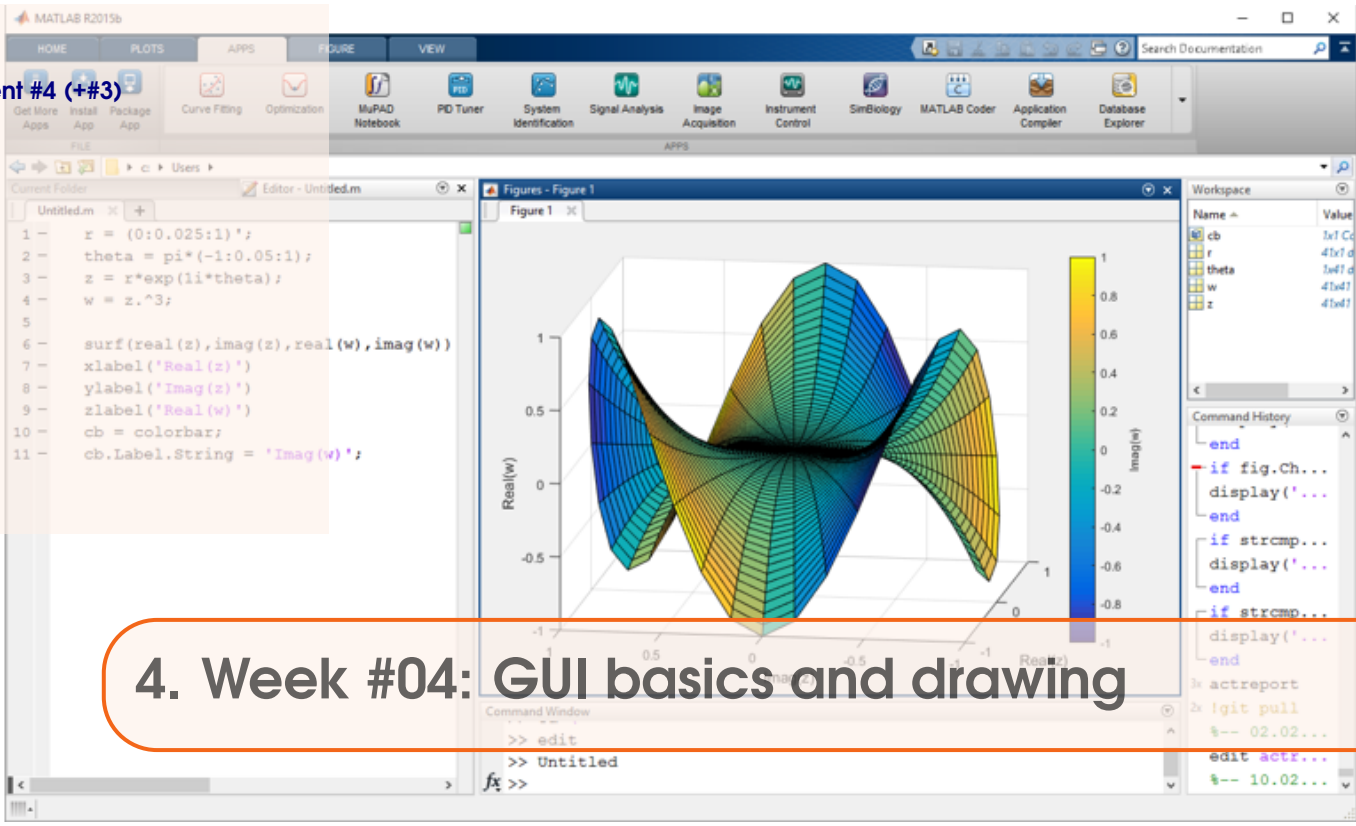
- how to create 2D plots, including the x- and y-axis information (via `meshgrid`)
- how to set color scales, and change scale limits
- setting the number and value, and also labelling, of coutours

specific MATLAB commands you should be familiar with:

- additional gridded plotting: `imagesc`
- 2D plotting: `contour` and `contourf`
- plotting controls: `clabel`, `colorbar`, `colormap`
- `meshgrid`

3.3 Micro Assessment #3

[enfolded into week 4 assessment]



4. Week #04: GUI basics and drawing

4.1 Work plan

Monday: Work through Section 5.1 of the GEO111 course text.
Friday: Work through Section 5.2 and Section 3.4. (Both these Sections are currently being updated, ready for Friday.)

A brief guide as to what you will be doing/seeing:

1. **Section 5.1. MATLAB GUI basics**
   How to design a simple GUI in **MATLAB**. Basic interfacing (linking) of the GUI with a program.
2. **Section 5.2. Further helpings of GUI**
   More advanced GUI design and program interfacing.
3. **Section 3.4. Even nicer graphing and graphics**
   Drawing lines and shapes. Placing text in figures.

4.2 Learning goals

Topics and methodologies you should be familiar with:

- Using the MATLAB GUIDE GUI editor – creating, positioning, and initializing GUI objects.
- Adding code to respond to events generated when actions are performed in the GUI (e.g. mouse button clicks).

Specific **MATLAB** commands you should be familiar with:

- `global`

4.3 Micro Assessment #4 (+#3)

Your assessment for this week is to write 3 short programs, and provide graphical output:

1. Write a script program that imports and creates a line plot, from a pair of **Excel** files.
   The files are accessed from the Week #4 data section of the course webpage. One is of annual solar insolation received at the Earth's surface at 65 degrees North, on 21st March, and how this varies with time (in the recent geological past) [65n000l.xls]. The other is for 21st June [65n090l.xls].
   Look up **MATLAB** `help` on `xlsread` and load in only the first sheet of each file.
   Plot both insolation data-sets together on a single (labelled) plot, with time on the x-axis and insolation on the y-axis, and with the 21st March data in red, and 21st June data in blue.
   Export as a normal graphics format (any), not a MATLAB figure (.fig).

2. Write a script program that imports, plots, and then creates an animation, from a *netCDF* file.
   The file is accessed from the Week #4 data section of the course webpage – SOMB.month.nc, and is a series of monthly climate model projections of the rate of dust deposition (units of kg/m2/s) to the Earth's surface.
   There is only one variable in the *netCDF* file, called dustdep. It has 3 dimensions – month (length 12), latitude (length 64), and longitude (length 128).
   Load in the dustdep – if successful, it should appear in Workspace as a single (Class), $128 \times 64 \times 12$ variable array.
   Start by doing a basic 2D plot (e.g. `pcolor`) of the first (or any single) month (no need to hand in). The data ranges over a number of orders of magnitude, meaning that you should see just a few spots of (non blue) color and nothing much else. Try converting the data to log_{10} by, e.g. `data2=log10(data);` (and re-plot). If you do not have 128 cells along the x-axis ... transpose the array. There is a related figure in:
   http://science.sciencemag.org/content/308/5718/67/tab-pdf
   that might help you see what you are looking at and be happier that the orientation is correct.
   There are 2 options here (bonus marks for the second), depending on how confident you feel:
   (a) Use `pcolor` to do the plotting.
   (b) Use `contourf` to do the plotting. If you take this path, you'll need to create a lon-lat grid using `meshgrid`. I'll give you for free, the grid vectors:
   longitude: `[360/128/2:360/128:(360-360/128/2)]`
   latitude: `[(-90+180/64/2):180/64:(90-180/64/2)]`
   (we can discuss later, exactly where these came from)
   Either way, you should produce a single .avi animation file with 12 frames corresponding to the 12 months in the data, to hand in. Do the plotting using the hot color scale in **MATLAB**. The axes should be labelled and some sort of title given.

3. Write a script program that draws the outlines of the continents. (Effectively, just ensuring that a class exercise was completed successfully.)
   Submit fully labelled plots (in any normal graphics format, not a MATLAB figure (.fig)) for:
   (a) The entire Earth. The line color should be black.
   (b) For the region: -180:0 longitude, 0:90N latitude.
   For bonus marks and fun (and (c)) ... see if you can plot the entire Earth with the Northern hemisphere continental outline in red, and the Southern hemisphere continental outline in blue. (Don't stress over accomplishing this.) Note that doing it *exactly* right might be really hard, so 'approximately' right is fine.

Submit by email (andy@seao2.org) as separate .m files. You will be marked for basic program structure and commenting of the code, in addition to the code correctly working! 80% of

the total marks is available simply for working code that has adequate commenting. Maximum marks (100%) will be awarded for neat and compact code and/or clever solutions. Please include somewhere a comment line with your name in it to ensure there can be no file mix-up.

The deadline (for the emails to appear in my in-box) is noon (12 pm) on Friday 27th October.

5. Week 5 – Coding and algorithms

5.1 Work plan

For Monday 30th October – work through Section 4.1 – '*Algorithms and problem-solving*'.

There are no specific key words or techniques *per se*, to learn here, but instead some worked-through examples of how to go about solving problems in **MATLAB** (or any programming language).

5.2 Learning goals

(Solving problems in code.)

5.3 Micro Assessment #5

There will be 2 tasks in the Micro Assessment for week #5:

1. Based on the class exercise in which the edges of the 'continents' in the GENIE model grid were identified and highlighted (by thicker black lines in the plot), color-code the lines according to:
   - North coast (North edge of land) – RED.
   - South coast – BLUE.
   - West coast – GREEN.
   - East coast – YELLOW.

   Use the same greater thickness of lines as in the worked example.
   Hand in both the .m file as well as an image (and 'normal', non-.fig format) of the final graphical output.

2. Write an algorithm as an .m file *function*, that takes in a string as input at the command line, and 'encrypts' it, with a simple cypher of moving one letter of each word to the next word (or alternatively: shifting the spaces by one place). The aim is to leave the length of each word the same. Remember to deal with the 'ends' of the sentence as a special case.
   For now: ignore punctuation – either don't input any in the first place (at the command line), or find and remove it from the input string.
   The original sentence, as well as the encrypted sentence, should be displayed in a figure window – the original in black, above the new version below in red text.
   Hand in both the .m file as well as an image (and 'normal', non-.fig format) of the final graphical output.

The second problem is much more challenging and will take a lot of thinking, so more (marking) weight will be given to the first one. Do what you can!

Submit by email (andy@seao2.org) as separate .m or graphics files (depending on the exercise). For .m files – you will be marked for basic program structure and commenting of the code, in addition to the code correctly working! 80% of the total marks is available simply for working code that has adequate commenting. Maximum marks (100%) will be awarded for neat and compact code and/or clever solutions. Please include somewhere a comment line with your name in it to ensure there can be no file mix-up. For graphics files – plots should be appropriately labelled (if relevant).

The deadline (for the emails to appear in my in-box) is noon (12 pm) on Friday 3rd November.

6. Week 6 – Computer programs ... & games

6.1 Work plan

For Monday 6th November, we'll adopt he following plan:

1. Firstly, we'll go through the Midterm, and address any prevailing problems and/or programming and/or **MATLAB** issues.
2. We'll go through the string cypher (encryption) micro assessment example.
3. Then, if you would like a little more info and worked examples on nested loops – go through Section 4.1 (in the current, revised version of the course text).
4. Finally – onto Chapter 6 in the course text.

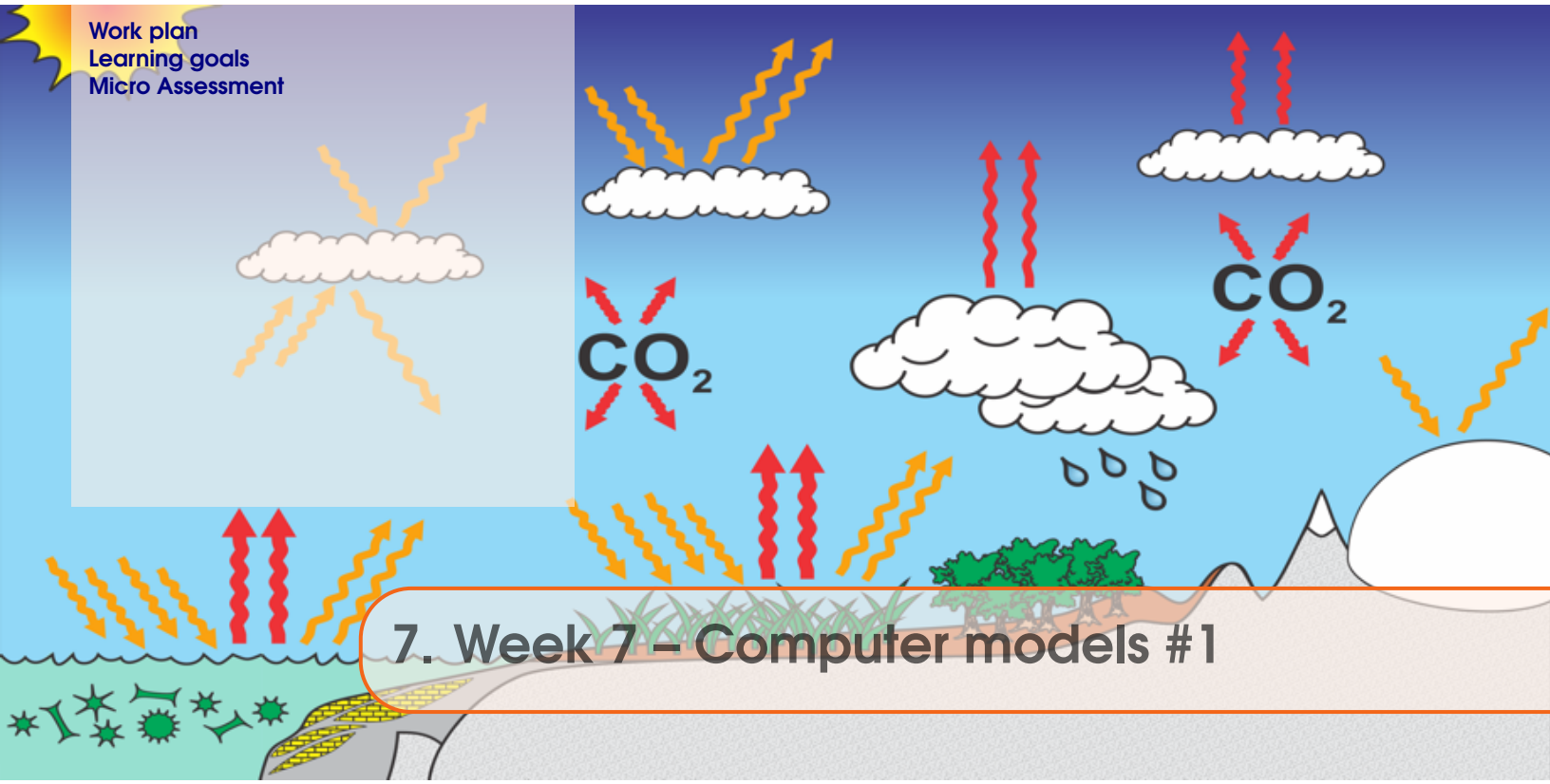
Chapter 6 takes the example of some simple games to work through further examples of program structure and nested loops, and further uses of functions.

6.2 Learning goals

Nothing particular ... more familiarity with loops, particularly nested loops, and use of functions etc.

6.3 Micro Assessment

(There is no scheduled micro-assessment for week #6.)



7. Week 7 – Computer models #1

7.1 Work plan

For week 7:

1. **Monday November 13th.**
   Work through Section 8.1 – there are 5 main sub-sub-sections and distinct pieces of work here. Get each working individually and make sure you understand it before moving on to the next ...
   (a) Create and 'play with' (i.e. change some parameter values and see what 'happens') a simple representation of the Earths climate system. (A very simple representation ...)
   (b) Turn this into a function and test it.
   (c) Explore and plot, how sensitive Earths surface temperature is to various parameter choices, using the function.
   Here is another opportunity for you to practice more nested loop examples.
   (d) Create a function to calculate the value of the solar constant, given a value for 'time' (since the Suns formation).
   (e) Put all the functions together – calculating how Earths surface temperature may have evolved through geological time.
2. **Friday November 17th.**
   Will continue on the theme of evolution of Earths climate through time (and make use of the function for calculating the solar constant), but create and play with a classic model of the interaction of life with climate – 'Daisy World'.

7.2 Learning goals

- Appreciate how simple physical models can be encoded in ... code (and **MATLAB**).
- Remind yourself how to re-write equations in terms of a different term.
- More practice with loops and counters, and creating arrays of data as the loop progresses.
- More practice with nested loops to ensure you are fully 100% with this concept.

7.3 Micro Assessment

This weeks micro-assessment has 3 parts, based on the simple 0D climate and 'solar evolution' models that you wrote.

1. Firstly – 'look up' (aka, Google) typical albedo (α) values for the following potential surfaces (/atmospheres) of a planet, and using a value of S_0 of $1368 W m^{-2}$, calculate what the equilibrium surface temperature would be:
   - i Cloud (any sort, but specify).
   - ii Fresh snow.
   - iii Trees (any sort, but specify).
   - iv Tarmac (asphalt).
   - v A perfectly reflecting surface.

   (Note that there will be no one answer for any of these, depending on the albedo value you find and assume.)
2. Secondly – run the EBM climate model as a function of time, as per Section 8.1.5, but with a modification ...

   The Earths continents (and thick continental crust) did not form all at once but some reconstructions have it mostly forming between about 3.5 and 2.5 Ga (billions of years ago). If we assumed that it formed all at once, at 3.0 Ga, and that ocean albedo is much lower than land surface albedo, there was presumably a substantial associated change in planetary surface temperature (maybe).

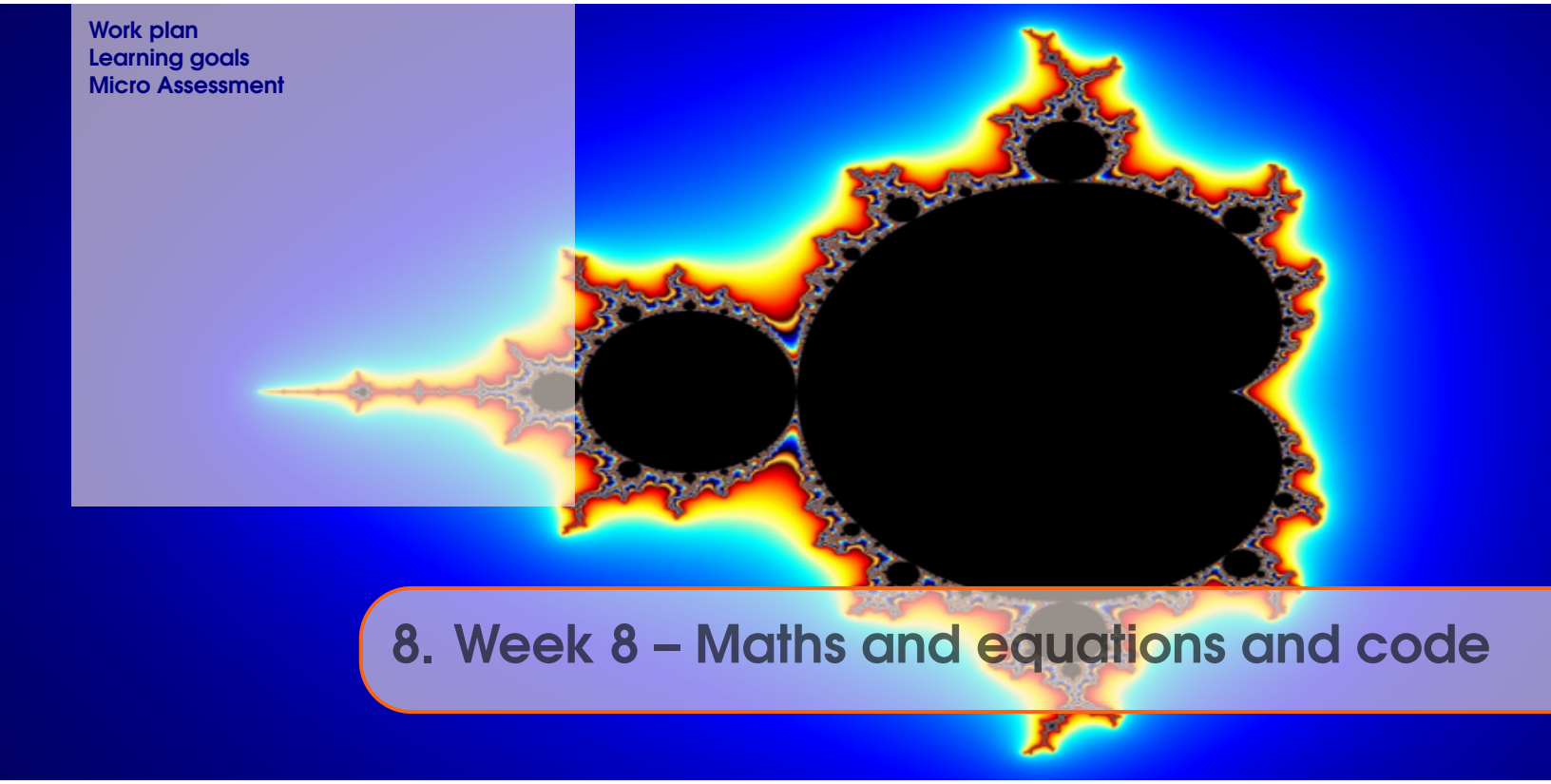
   In the simple EBM, we assume an average planetary albedo of 0.3. Very very crudely, and because the modern land fraction is about 0.3 and the numbers will be nice and easy ... lets assume that there are no clouds and the land is perfectly reflecting, so that the average planetary albedo of 0.3 is made up of 70% ocean ($\alpha = 0.0$) and 30% land ($\alpha = 0.0$), hence an area-weighted average of 0.3. So in this insanely simplified world, an ocean-only planet would have a mean global albedo value $\alpha = 0.0$.

   Your code modification hence needs to make the following happen:
   when the age of the Earth is less than 3.0 Ga, the value of α changes from 0.0 to 0.3.
   (Hint: you'll need to add a *conditional* in the function for S_0.)
3. Lastly – create a new function, based on your EBM function and the energy balance model equation but now: write the energy balance equation in terms of S_0 (i.e. S_0 equals something) and create a new function that takes as input, temperature, and tells you the value of the solar constant that would produce that temperature. Use this function to solve the following situations (for fixed, $\alpha = 0.3$):
   - vi What is the value of S_0 that would leave the planetary surface at freezing point?
   - vii What is the value of S_0 that would leave the planetary surface at boiling point?

   Also rearrange the equation for the solar constant S_0 against time, to instead give time as a function of S_0. Using both modified functions, answer the following:

viii If 'habitability' were defined as any surface temperature in the range of 0 to 40 degrees Centigrade – how long (in time) would the Earth be habitable for?
(Hint: To solve this – first determine the 2 values of S_0 that correspond to the 2 temperature limits. Then, taking your 2nd modified function (for time as a function of S_0), find the times (ages) at which these values of S_0 occur. The difference between them is the interval (in years) that you require.)

For (1) and (3), simply provide the answers to (i) through (viii) as simple text in an email (for (i) through (v), give the albedo value you assumed alongside the mean global surface temperature of the planet in units of degrees Centigrade). For (2), submit a nice plot showing the evolution of surface temperature with time (i.e. similar to that in Section 8.1.6, but with the change in albedo imposed at 3.0 Ga).



8. Week 8 – Maths and equations and code

8.1 Work plan

For Monday 20th November, we'll adopt he following plan:

1. Firstly, work through Section 4.5 – a new chapter section on encoding maths problems in **MATLAB**.
2. Continue on with, and finish up the Daisy World model exercise frmm last week

8.2 Learning goals

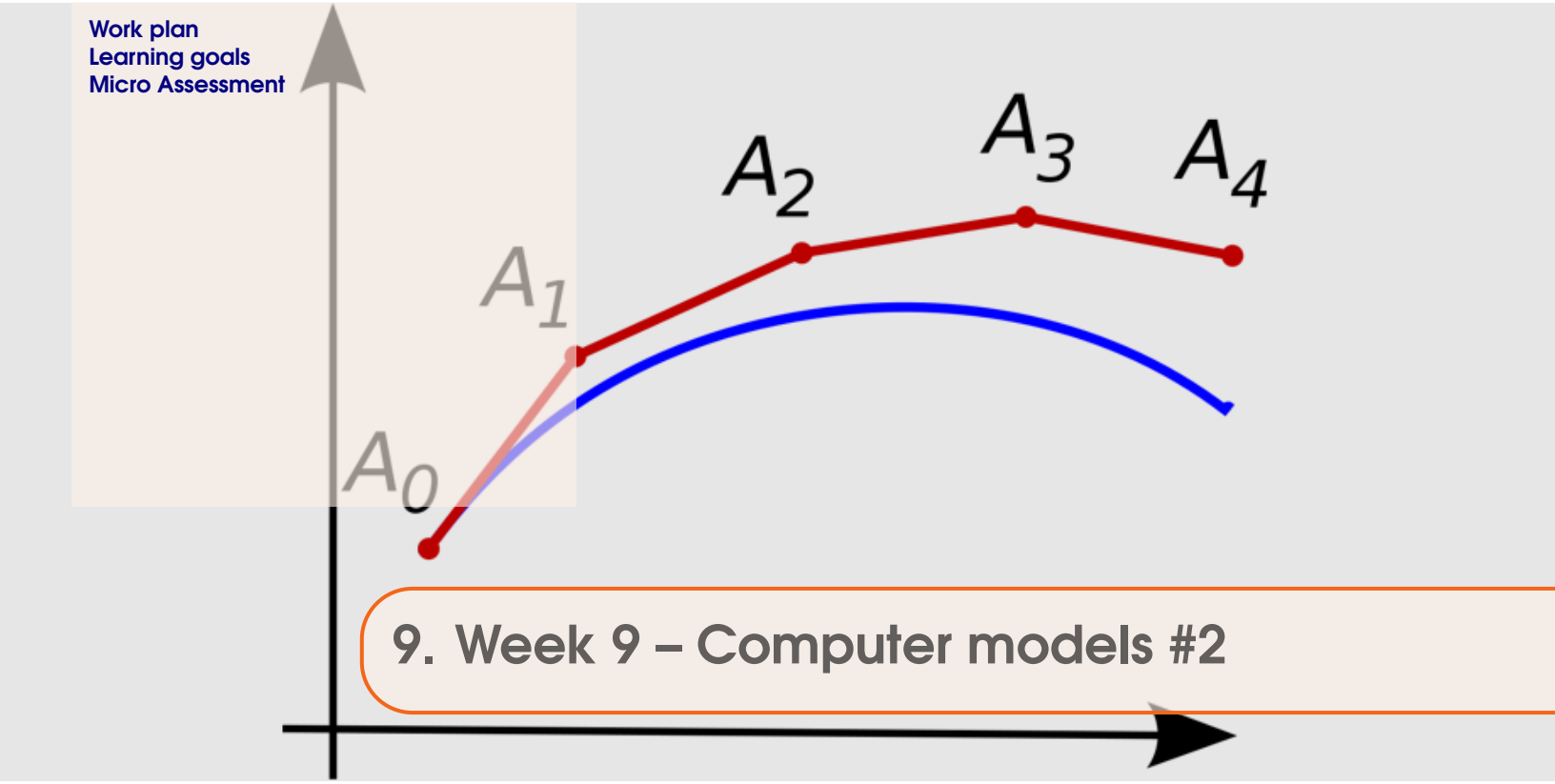
There are a few new functions introduced (but not entirely commonly used ones):

- `tic` and `toc` (for timing an application).
- `numel`, which returns the total number of elements in an array,.

In addition, there will be more practice with `meshgrid` and `reshape`.
Also, the creation of new color schemes for plotting.

8.3 Micro Assessment

(There is no scheduled micro-assessment for week #8.)



9.1 Work plan

For week 9:

1. **Monday November 27th.**
   Work through Sections 8.1 and 8.2 (in the revised book structure and chapter numbering scheme). These involve:
   (a) Going through a '101' on dynamic numerical modelling.
   (b) Creating a model of the trajectory of a thrown ball (or any object), experiencing the force of gravity (only). You'll piece this together in a series of steps.
   (c) Creating a time-dependent version of the 0D EBM. Then, adding greenhouse gas forcing and then driving with historical greenhouse gas concentration trends.
2. **Friday December 1st.**
   Continue on from Monday.
   (There are further, more advanced exercises in numerical modelling if you finish ahead of time.)
   Note ... that there is the EDGE Institute Fall Student Symposium starting at 3 pm.

9.2 Learning goals

- Integration in models and code.
- More practice with, and exposure to, numerical models.
- More practice with scripts and functions, loops and counters.

9.3 Micro Assessment

1. Find ... on the internet ... a future scenario for atmospheric CO_2 up to the year 2100 (or beyond if you like) – any scenario for which you can provide a reference for, will do. Combine this with the historical/observed trend into a single dataset of CO_2 concentration vs. time (i.e. so that the future scenario carries on from where the historical trend finished).
   Drive your greenhouse-gas-enabled time-dependent EBM with this scenario. Plot the projected trajectory of mean global surface temperature as a function of time, from year 1800 to 2100, and hand in the figure (graphic).
2. Simulate how the concentration of pollution in a lake would decay with time, given the following information about the lake:
   - The lake volume is $10,000,000\ m^3$.
   - The flow of clean (pollution-free) water into the lake, always exactly balances the flow out. The flow rate is $1.0\ m^3 s^{-1}$.
   - The pollution is well mixed throughout the lake (and remains well mixed).
   - The initial pollution concentration is $18\ ng\ g^{-1}$. (Assume that the density of water is $1\ g\ cm^{-3}$.)

   Plot, and hand in, a graph of the decay of pollution concentration ($ng\ g^{-1}$) with time, over the course of 1 year. Label the x-axis with the month name at the start of each month – rotate the month names so that they fit the axis. Start the x-axis at (1st) January. (Assume 12, 30 day equal length months, and a 360 day year.)

10. Week 10 – Putting it all together

10.1 Work plan

For week 10:

1. **Monday December 4th.**
   Work through Chapter 10. Tasks involve:
   (a) Some basic graphics/image stuff.
   (b) Adapting the ball/trajectory model to incorporate a picture (sprite) rather than a plot/scatter point.
   (c) Creating a GUI game based on the ball/trajectory model, in five steps:
      i. Create the GUI.
      ii. Set up the graphics.
      iii. Add in the ball/trajectory model.
      iv. Activate the Sliders.
      v. Determine when the ball 'hit' the Pokemon.
      vi. (Further refinements to the App.)
2. **Friday December 8th.**
   Will be the Finals exam ... (see Intro).

10.2 Learning goals

10.3 Micro Assessment

Take, your Pokémon App, and make the following modifications/improvements:

1. Make the initial values of the speed and angle, random (between reasonable limits).
2. In a new game, display a different Pokémon (at least one more different one, but can be as many as you like). The choice could be random, or part of a set sequence.
3. Keep score (how many 'catches') as well as how many tries total.
   Calculate the score as the number of successful catches, divided by the total number of tries.
   This would require a single new Static Text box object in the GUI.
4. You track of the high score from game-to-game ... saving this value when you close the App, and then loading it in when you start it up again.

Hand in the .m, .fig, and all associated graphics files (i.e. all files needed to run your App).

11. Bibliography

Books

Articles