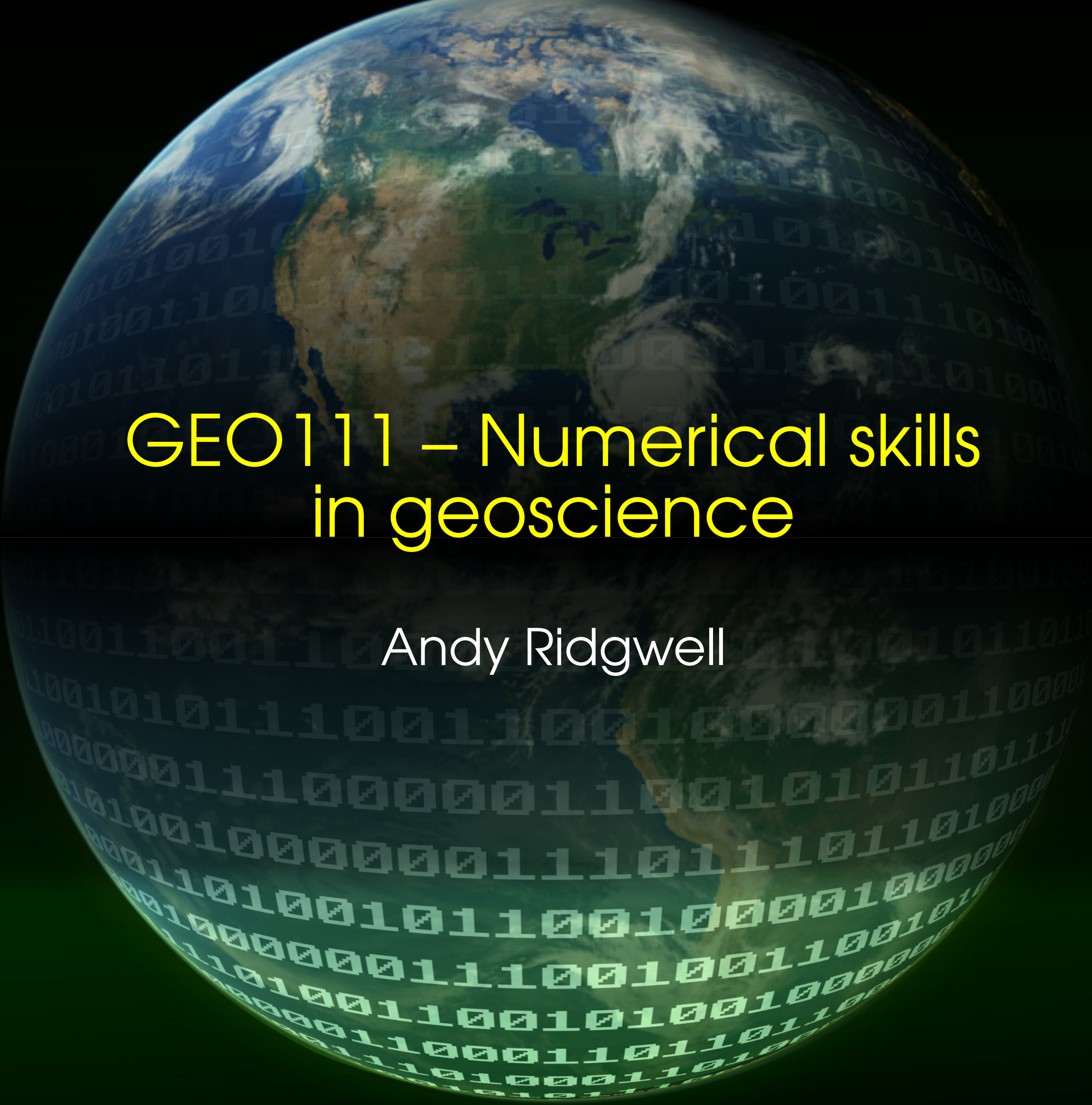
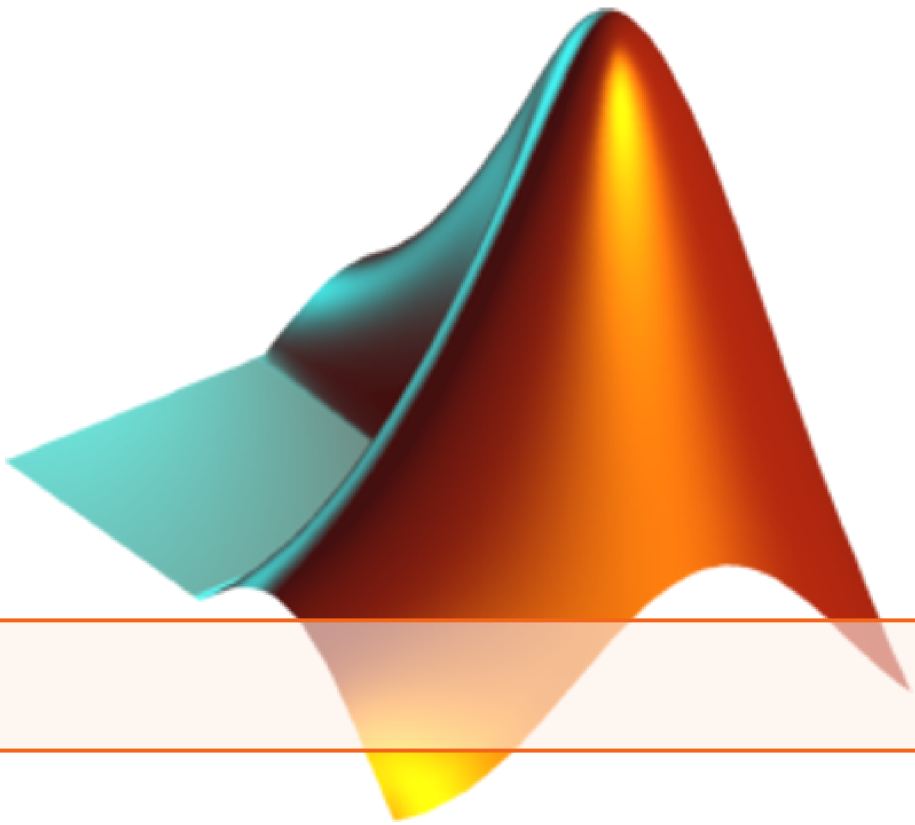


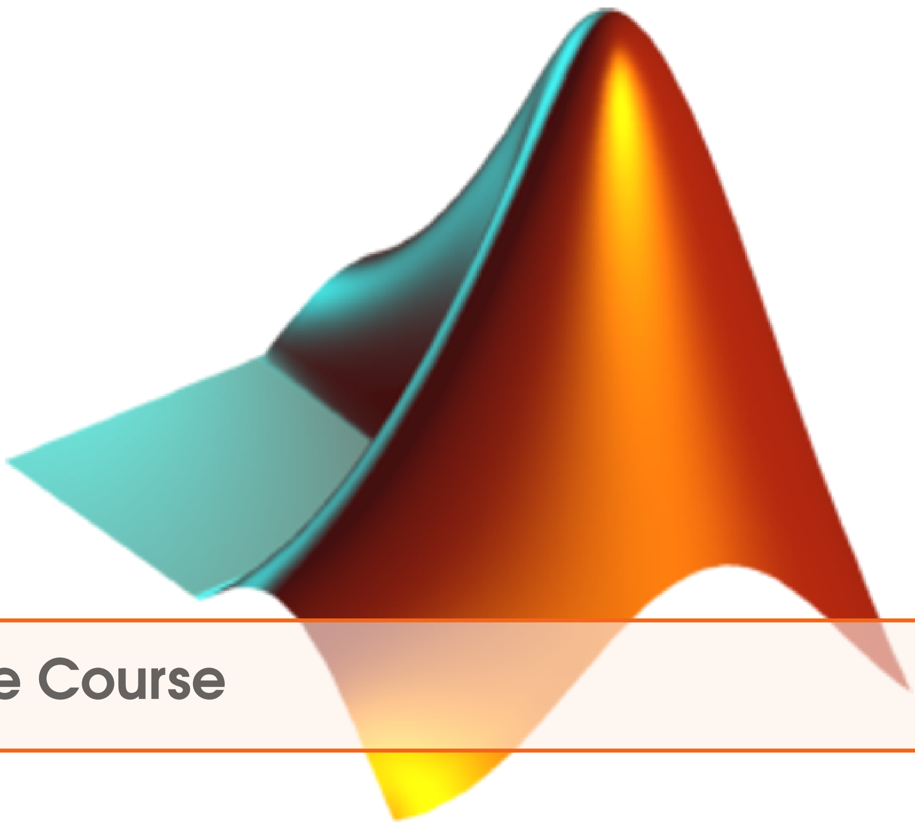
GEO111 – Numerical skills in geoscience

Andy Ridgwell

First printing, October 2017



Contents



0. 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

- Problem solving and logical analysis, fault-finding. (Learning Outcomes 4 and 5).
- Computer programming. (Learning Outcomes 2 and 4).

0.1 Course Overview

0.1.1 Format

The weekly format of GEO111 is: 1 × 1-hour lecture together with a 1 × 3-hour computer practical session (both in Watkins 2101) on a Monday, plus 1 × 2-hour interactive lecture/discussion session of worked problems and examples (also in Watkins 2101) on a Friday. 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 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 may be set and further topics introduced each Friday.

0.1.2 Timetable

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

0.1.3 Assessment Summary

The course will be assessed as follows:

- 6 × weekly-ish micro assessments @ 5% each = 30% total
- Midterm paper – 20%
- Finals paper – 50%

In the (6) weeks with no Exam (weeks #5 and #10), and also excluding the week of Thanksgiving (#8) and the first, introductory week (#1) ... a short (micro) assessment will be set. The hand-in deadline will be the start of class (8.10 am) on the following Monday to the week in which it was set. Each micro assessment will be worth 5% of the total marks available for the course. The assessment will be set by noon (PST) on the Monday of that week (i.e. the end of the Monday class). The aim of the micro-assessments is to help reinforce what you should have been learning.

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 exam will be 2 hours long. No study aids of any sort will be allowed. The mid-term paper will constitute 20% of the total assessment for the course. (Further details given in a subsequent section.)

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. (Further details given in a subsequent section.)

0.1.4 TA and Office Hours

There is a designated TA for GEO111 – Pam Vervoort <pverv001@ucr.edu>. She will hold office hours on Wednesdays, 2-4 pm, in room 1324 of the Geology Building.

In addition, I am happy for people to drop by[1] with questions (or concerns). Wednesdays (almost any time) would be your best chance of finding me (and not busy), or Monday pm / Friday am ... avoid Thursdays like the plague ... and/or email[2] questions. (For general course / assessment clarifications and help, your first port of call should be the TA.)

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]

0.1.5 Communications and Course Material

Group (email) communications will be via iLearn. However, course materials will be uploaded to and made available from my website (rather than iLearn):

http://www.seao2.info/teaching.html

Assessments will be 'handed in' by email.

0.1.6 Course Text and Datafiles

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 creating a source *e*-book – to provide a single (and free!) source for a range of information plus 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 a new revision of the book will be posted immediately prior to each class. This (PDF format file) will appear on my teaching webpage[5]. Refer to the weekly work plan given in this document (GEO111 course guide) for which sections of the text to follow (as not all will be used in the class).

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.

Finally, periodically during the course, you will be directed to obtain a data file (or files), or perhaps a code fragment. These will be available from my website (same page as per for the course text), in a blue box on the left hand side of the page, headed '`got data?`'.

[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

0.1.7 Computers and Software

Watkins 2101 is equiped with PCs running the latest (or almost) version of **MATLAB**. You will need to bring your own USB pen-drive ('memory stick' / 'flash-drive') in order to save your work (or you can use cloud storage). Your work is not retained by the Watkins lab PCs when you log off.

You can also install **MATLAB** on your own laptop if you have one. UCR provides a free student license for running **MATLAB** under all of: Windoz, MacOS, and linux. Read and follow the instructions to install and obtain a license (the procedure is different from most UCR provided/licensed software).

Regardless of whether you work on a Watkins lab PC and store your work on a USB pen-drive, or work on your own laptop, please backup your work regularly (to a location different from the pen-drive or laptop).

0.2 Assessment

0.2.1 Homework

In the (6) weeks with no Exam (weeks #5 and #10), and also excluding the week of Thanksgiving (#8) and the first, introductory week (#1) ... a short (micro) assessment will be set. The hand-in deadline will be the start of class (8.10 am) on the following Monday to the week in which it was set. Each micro assessment will be worth 5% of the total marks available for the course. The assessment will be set by noon (PST) on the Monday of that week (i.e. the end of the Monday class). The aim of the micro-assessments is to help reinforce what you should have been learning. (Additional reading material and exercises may be provided in the other 4 weeks.)

0.2.2 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.
4. Anything covered in the weekly set exercises.

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 long and/or working programs on paper, although you might be tested on the MATLAB commands that you have seen and their general/conceptual usage.

Logistics

The Midterm is nominally scheduled from 2.10 through 4 pm on Friday 2nd November, 2018.

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 20% of the total marks for the course.

0.2.3 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.
4. Anything covered in the weekly set exercises.

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 5 pm on Friday 7th December, 2018.

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.

0.3 Medical Matters and Disablity

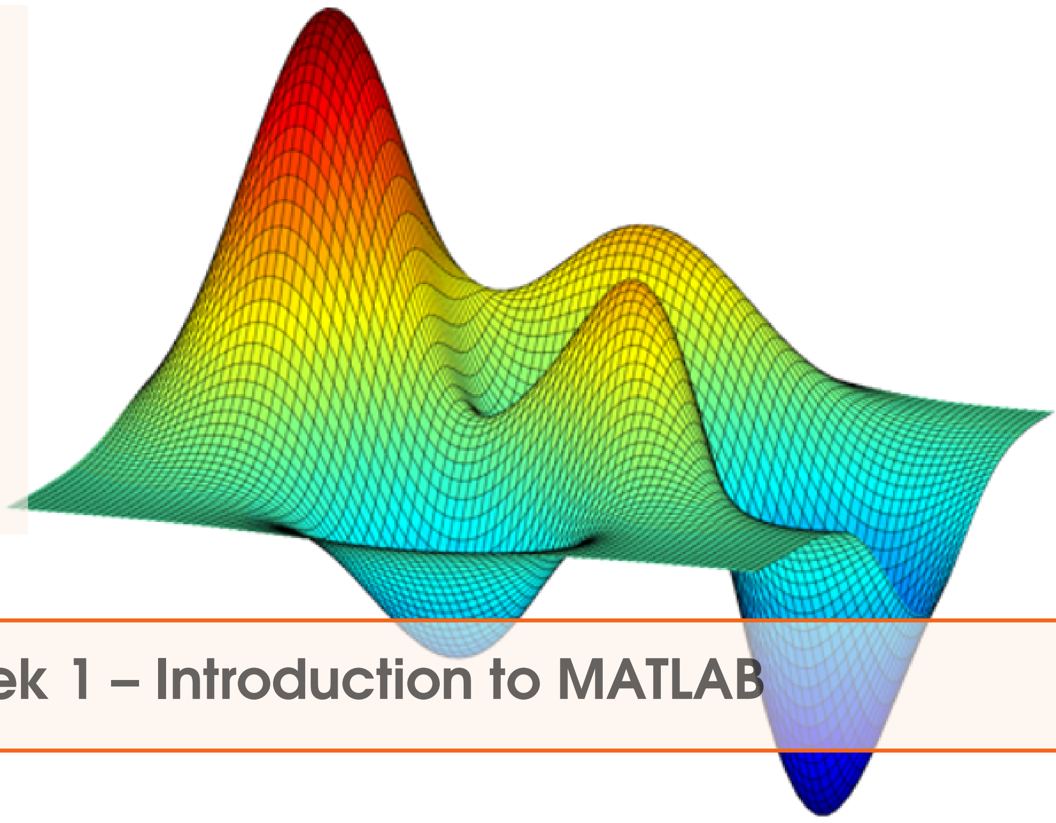
Medical Matters and Disablity: If you have a disability or believe you may have a disability (including any pre-existing health condition that affects your ability to participate in any required class activity), you can arrange for accommodations by contacting Services for Students with Disabilities (SSD) at 951-827-4538 (voice) or specserv@ucr.edu (email). Students needing academic accommodations are required to register with SSD and to provide required disability-related documentation. As such registration applies to each class individually, if you have approved accommodation(s), you are responsible for notifying the instructor and getting paperwork signed shortly after the start of the course, which can be done privately. As the midterm exam is during the fifth week of the course, and SSD itself needs some time (normally at least 10 days) to register you, please make sure you do the registration as early as possible. You and SSD are responsible for coordinating when the exam is taken.

Mental health & wellness: As a student, you may experience a range of issues that can cause barriers to learning, such as strained relationships, increased anxiety, alcohol/drug problems, depression, difficulty concentrating and/or lack of motivation. These mental health concerns or stressful events may lead to diminished academic performance or reduce a student's ability to participate in daily activities. UC offers services to assist you with addressing these and other concerns you may be experiencing. If you or someone you know is suffering from any of the aforementioned conditions, consider utilizing the confidential mental health services available on campus. I encourage you to reach out to the Counseling Center for support (counseling.ucr.edu, 951-827-5531). An on-campus counselor or after-hours clinician is available 24/7.

On a personal note – sh*t happens in life from time-to-time. Please do not suffer in silence – it is difficult, if not impossible, for me to make accommodations if I am not aware of any issues needing accommodating.

Table 1: Schedule of GEO111

Week #	**Monday lecture**	**Monday LAB**	***micro assessment***	**Friday LAB**
01 (10/01)	Introduction to the course & to the MATLAB language and software environment.	Elements of MATLAB and data visualization.	**NONE** (work through 'Getting Started' MATLAB Guide)	(CONTINUED)
02 (10/08)	Fundamentals of computer programming I.	Scripts and functions in MATLAB. Loops and conditionals.	1. Basic MATLAB practice.	Further programming / elements and structures.
03 (10/15)	Fundamentals of computer programming II.	Further MATLAB and data visualization.	2. Data anaylsis and plotting practice.	(CONTINUED)
04 (10/22)	Algorithms and problem-solving.	Further programming tricks and techniques in MATLAB	3. Programming practice	(CONTINUED)
05 (10/29)	How computers work.	Further data analysis and statistics in MATLAB.	**NONE** (revision)	**Mid-term**
06 (11/05)	Introduction to numerical modelling and encoding math.	Basic (zero-D) numerical modelling.	4. Function use practice.	(CONTINUED)
07 (11/12)	UCR HOLIDAY	UCR HOLIDAY	5. Program debugging practice.	How to survive program complexity.
08 (11/19)	Computer program Graphical User Interfaces(GUIs) and GUI-ing in MATLAB.	Basic GUI creation in MATLAB.	**NONE**	UCR HOLIDAY
09 (11/26)	How numerical models 'work'.	Dynamic (time-stepping) modelling – ballistics 101.	6. GUI practice.	More advanced graphics usage and animations.
10 (12/03)	Other computer languages.	Putting it all together – a GUI- and physics-based game(!)	**NONE** (revision)	**Finals**



1. Week 1 – Introduction to MATLAB

Before anything else – read through Chapter #0 – 'How to use this Textbook'.

1.1 Work plan

The work plan for week #1, is to work through Chapter #1 of the GEO111 course text. You should aim to complete Sections 1.1, 1.2, 1.3, and 1.5 during the Monday am lab, and also during the week (in your own time). On Friday, we will tackle Sections 1.4, 1.6, 1.7 and 1.8.

There is a lot of stuff crammed in here and it would be easy to get lost in a mire of commands and instructions. A brief guide to what you will be 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 name and assign information to a *variable* (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) and 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: **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 by the end of the week:
- 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`

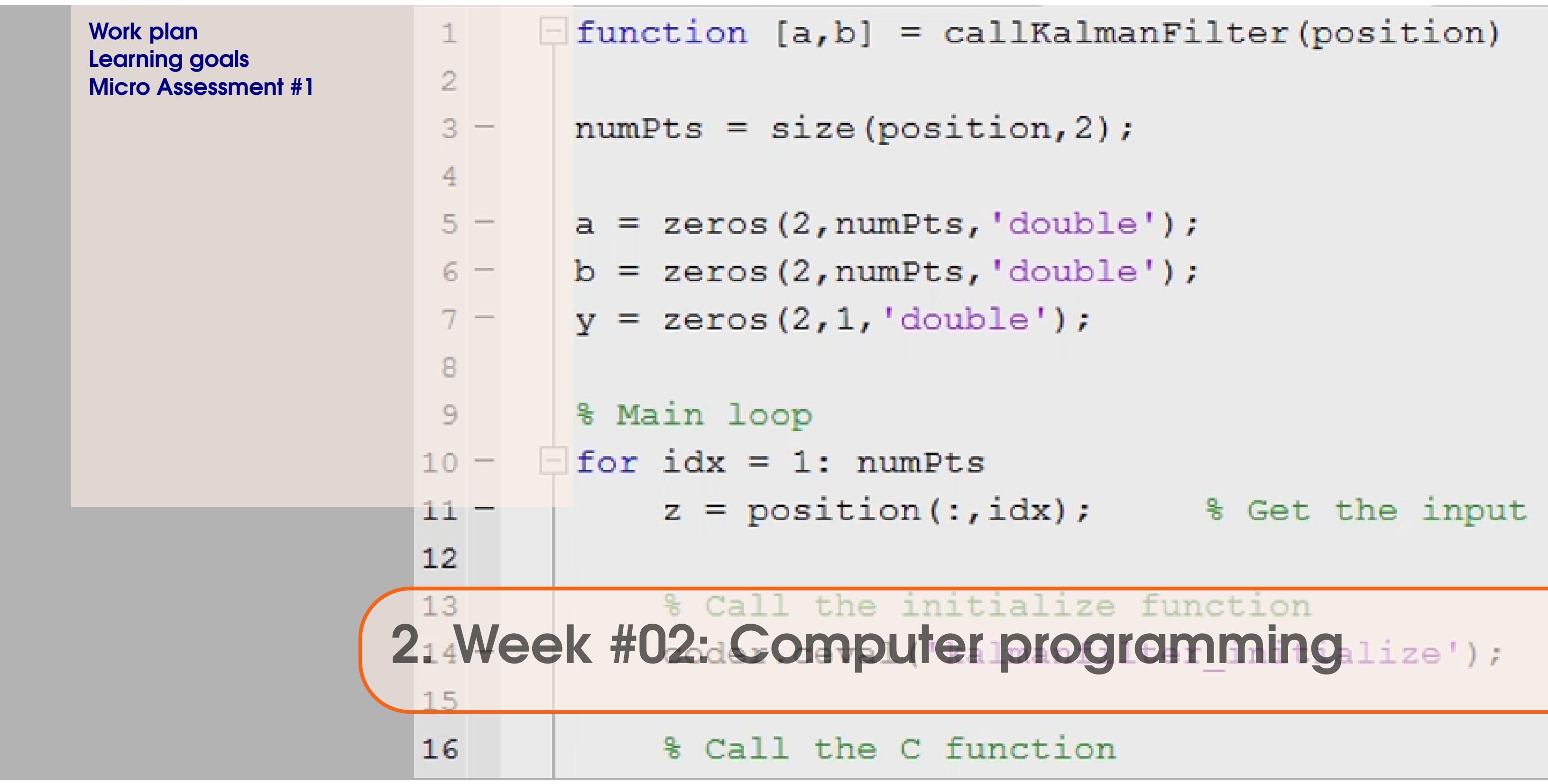
1.3 Homework

Although there is no formal (graded) micro-assessment ... there is important homework – 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* (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.

Note that any of this material could appear in the Midterm ...



```
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
    [illegible]alize');

    % 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 #1

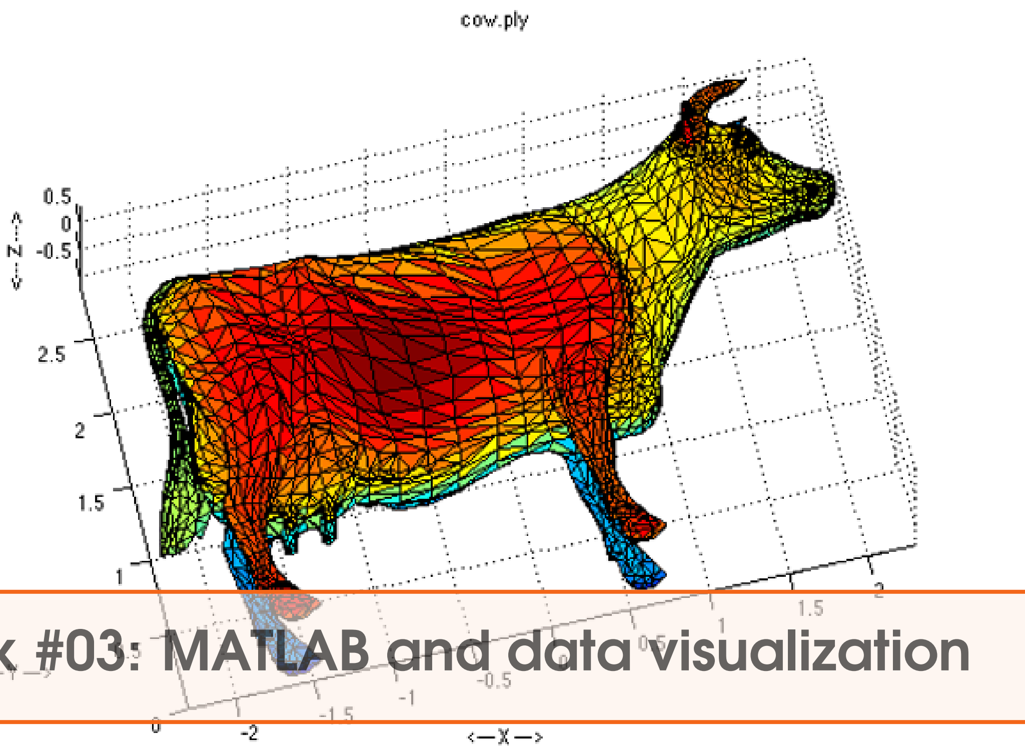
Your (first) assessment is in 2 parts:

1. Firstly, to complete Section 1.7.2 and 1.7.3 – global vs. local (Riverside) temperature analysis and plotting (these sub-sections have now been numbered .... but otherwise are unchanged). Export and submit a **.png** format graphic to complete the assignment. The graphic should contain 2 plots on the same set of axes. One plot should be the observed global surface temperature anomaly in units of ^oF . The other should be the observed trend in local annual mean temperature in Riverside, on the same scale (and units). You should remove (delete from the data array before plotting) whatever years of Riverside temperature you judge unreliable or biased in some way. The graph should be 'adequately' labelled. Your choice of colors, symbols, fonts/font sizes etc. Aim for the clearest possible presentation.
2. Secondly, to write a function that does the following:
   Takes 2 inputs (numbers).
   Returns a single output (number).
   The output (what is returned by the function) should be a number, that is equal to the larger of the 2 input numbers. Unless ... the inputs are equal, when a `NaN` should be returned instead.
   You should submit the `.m` file to complete the assignment.
   Hint: follow the instructions (exactly) for create a *function m-file*. You will need to test for the 2 inputs being equal ... if so – return a `NaN`. If not, find out which is the larger one, and return that.
   Bonus marks ... if you can test for a string wrongly given as input, and display an appropriate (error) message indicating this (and exit the function without returning anything). (You may need to do some Gooling and/or looking through **MATLAB** help.)

Submit by email (andy@seao2.org) the respective files files. The name of each file should start start with your surname (and then contain what makes for a helpful filename).

For the `.m` file, 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.

The deadline (for the emails to appear in my in-box) is 8.10 am on Monday 15th October.



3. Week #03: MATLAB and data visualization

3.1 Work plan

First ... ensure that you have completed *all of Chapter 2*. For Section 2.6 – '*Even more (and loopier) loops*' – Pam will take you through the steps and code and discuss the concepts, which illustrate more about loops and some more involved and clever use of array indexing. Expect that this will take at least 2 hours of the Monday class.

Then ... work through Section 3.1 of Chapter #3 of the GEO111 course text. (We will then work through Sections 3.2 and 3.4 on 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. **Section 3.2. – 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. **Section 3.4. – 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

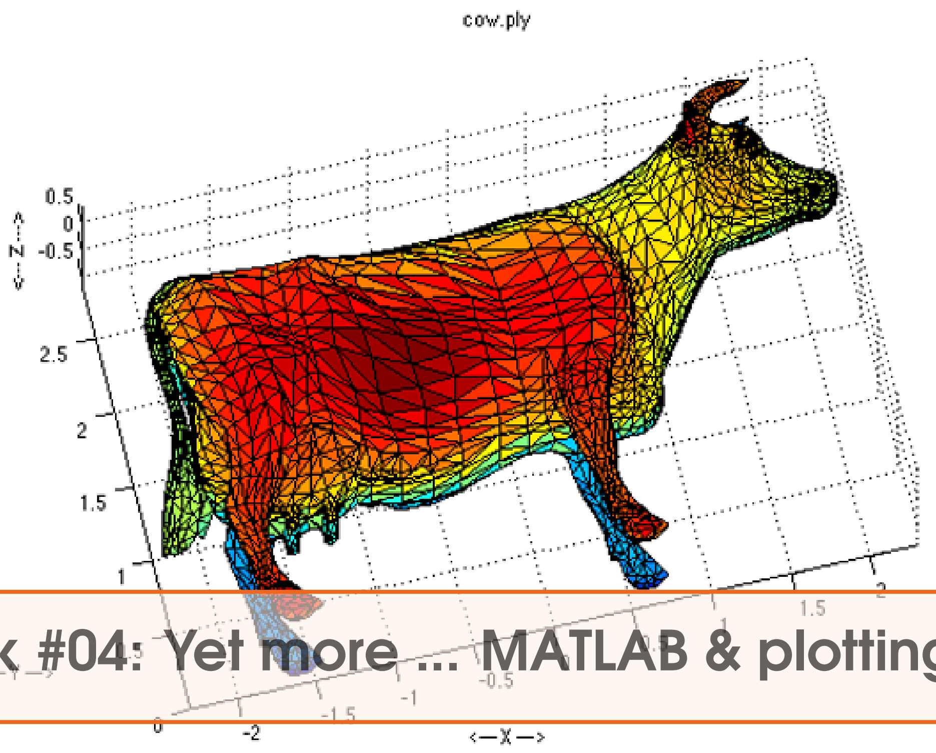
- setting the number and value, and also labelling, of coutours

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

[none for this week]



4. Week #04: Yet more ... MATLAB & plotting

4.1 Work plan

You should by now have completed *all of Chapter 2*. (If not – please complete in your own time.) This provides the basic introduction to the key elements of programs and programming – functions, loops, and conditionals. We will need this knowledge and familiarity throughout the rest of the course. If you need help .. please ask! There are scheduled TA hours and you can also come see me. You should also have taken a quick tour through Section 3.1 to become aware of other data import methods.

For Monday: ensure you have finished Sections 3.2 and 3.4 from last week, which were:

1. **Section 3.2. – 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.
2. **Section 3.4. – Even nicer graphing and graphics**
   Drawing lines and shapes. Placing text in figures.

(This will be your *last* chance to 'catch up' if you are at all behind.)

For Friday – Section 3.3.

1. **Section 3.3. – Further data processing**
   Additional data processing techniques including interpolation.

Plus ... the last (new) part of **Section 3.4** on designing your own color scales (which will form part of next weeks assessment).

Make sure that you are up to speed by Monday 29th and the start of Week 5.
(This is your unofficial 2nd part of the weeks assessment.)

4.2 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 contours

specific MATLAB commands you should be familiar with:

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

Of the new concepts and functions – `meshgrid` is the oddest and most difficult to grasp – feel free to ask for help / live demonstrations ... Also, `find` takes a little getting used to – we will cover this in some details on Friday, but feel free to ask more about it on Monday.

4.3 Micro Assessment #2

There is a single part to the micro assessment:

The good and the bad (and the ugly) of graphical visualization.

A change to the micro assessment – for the assessment this week (and this week only) – your assessment (files) need to be submitted (by email) PRIOR to 12 noon on Friday (26th) so we can discuss the results in class.

Find either one ’good’, or one ’bad’ (in your personal judgement), examples of scientific plotting/graphing/visualizaton.

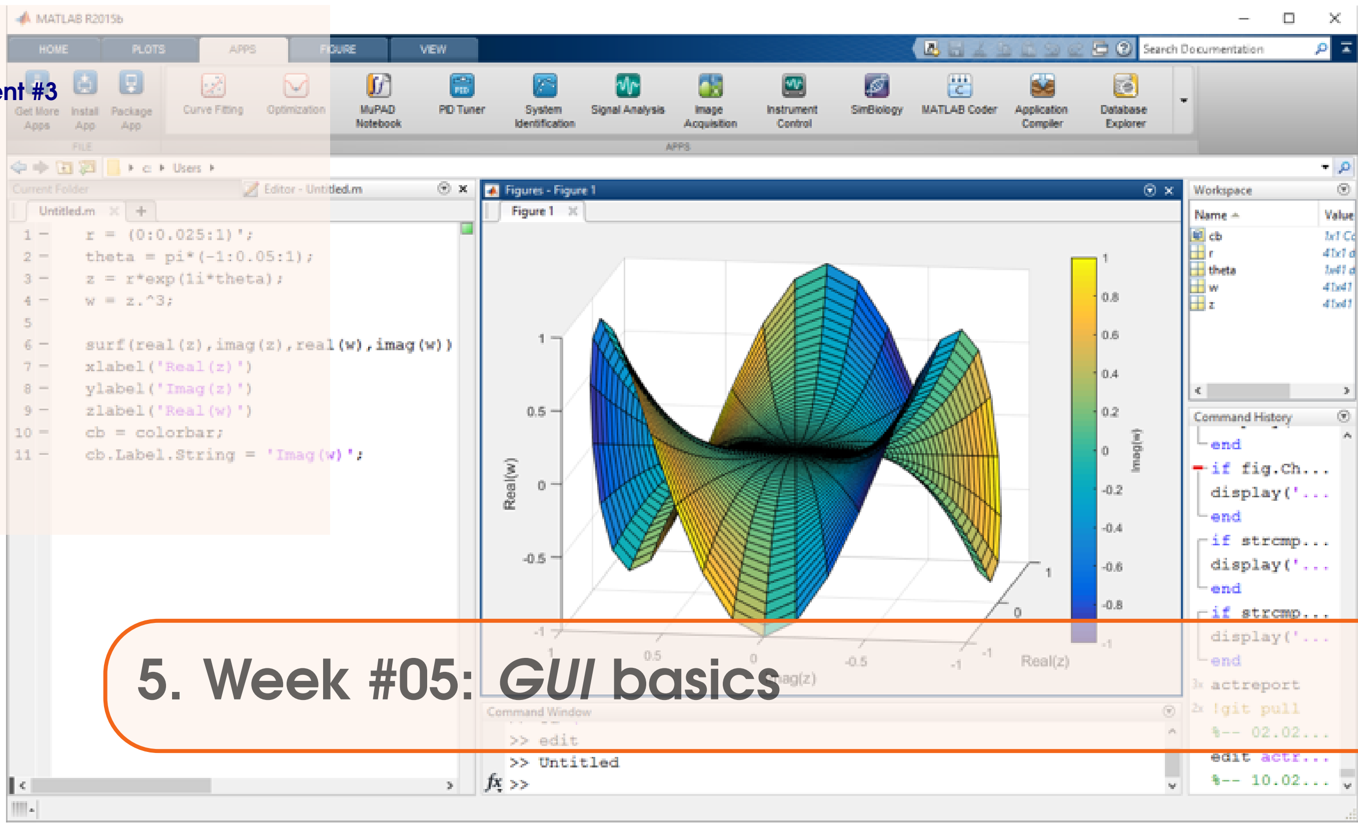
Your source can be from any scientific paper or book (restricted to the physical sciences). It could also be from a lecture or other presentation (including your own).

You need to save, download, or otherwise extract (e.g. the Take a Snapshot (from the Edit menu) in Adobe) to copy a section of a page of a PDF file to the clipboard (you then need to paste the clipboard contents into a drawing or graphics package to save).

Ask for help if you are having trouble extracting the image and saving it. (If you really cannot get the image out of e.g. a PDF file, then hand in the complete PDF file.)

Hand in (by email), the file, in whatever graphic format you have managed to extract/save it.

Be prepared on Friday to spend (no more than) 2-4 minutes discussing what is good or bad (depending on your choice) about the figure in terms of colors, amount of information, symbols and labels ... whatever makes it successful, or unsuccessful in getting the authors point and information across to the reader. Also be prepared to briefly explain what (the data) is being plotted and what the authors are trying to show, illustrate, or highlight (so it requires you to read the paper).



```
r = (0:0.025:1)';
theta = pi*(-1:0.05:1);
z = r*exp(1i*theta);
w = z.^3;

surf(real(z),imag(z),real(w),imag(w))
xlabel('Real(z)')
ylabel('Imag(z)')
zlabel('Real(w)')
cb = colorbar;
cb.Label.String = 'Imag(w)';
```

```
>> edit
>> Untitled
>>
```

5. Week #05: *GUI* basics

5.1 Work plan

Monday: Work through Section 9.1 of the GEO111 course text. A brief guide as to what you will be doing/seeing:

1. **Section 9.1. MATLAB GUI basics**
   How to design a simple *GUI* in **MATLAB**. Basic interfacing (linking) of the *GUI* with a program.

Friday: Mid-term

5.2 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`

5.3 Micro Assessment #3

Your assessment for this week is to write a script (**m-file**) program that achieves the following:

1. Loads in 12 monthly data fields of sea surface temperature (SST).
2. Plots a global map (-180 to 180 E longitude, -90 to 90 N latitude) of each monthly SST distribution using `contourf`.
3. Creates a single .avi animation file with 12 frames corresponding to the 12 months of the SST data.

Firstly – the 12 files you need are:

1. SSTmonth1.nc
2. SSTmonth2.nc
3. ...

10. ...
11. SSTmonth11.nc
12. SSTmonth12.nc

and are available form the course website.

Note that they have very similar filenames, differing only in the month number. (You have seen earlier how to automatically create a series of filenames that differ only by a single number in the name – see Section 2.4.3.)

Also note that the file format is *netCDF* (`.nc`) – refer to Section 3.1.3 (the ’easy’ method). The code you will need to load in the data is as follows:

```
dataall = ncread('SSTmonth1.nc','t_mn');
```

which will read in the variable `t_mn` from the first monthly file and assign it to the variable `dataall`. The only complication is that the data variable is 3D(!), with dimensions of: lon, lat, AND depth in the ocean. You can extract the surface lat-lon layer by:

```
datasur = dataall(:,:,1);
```

(i.e. selecting all longitudes (rows) and latitudes (columns) but only the first depth layer). Otherwise, this is very similar to the exercise you did previously.

For the animation, the code is as per the end of Section 2.4.3:

```
% Prepare the new file.
vidObj = VideoWriter('SSTseasonalcycle.avi');
open(vidObj);
% your loop ...
for ...
% form filename ...
% load data ...
% plot data ...
% make plot nice!  ...
currFrame = getframe;
```

```
writeVideo(vidObj,currFrame);
end
% Close the file.
close(vidObj);
```

For `contourf`, you'll need to create a lon-lat grid using `meshgrid`. I'll give you for free, the grid vectors needed:
longitude: `[-177.5:5:177.5]`
latitude: `[-87.5:5:87.5]`
(we can discuss later, exactly where these came from)

HINTS.

1. Create a temporary script(`.m`) file (or work at the command line) to work out your data loading and plotting strategy. Pick an example month (any one) and develop how you are going to plot this and make it nice. Work out the orientation of the plot (using the transpose operator, and/or `flipud`, and/or `fliplr` if necessary). Also at this point, work out the use of `meshgrid` to create yourself the arrays of latitude and longitude you need. Don't forget axis labels and a `colorbar` (and title?). (See 'Marking' (below) for `colormap` instructions.)
2. Then create the script (`.m`) file that you will submit. In this, first create the loop and the formation of the filename and check this works before doing anything else. Then check that you can successfully load each of the 12 data sets in the loop. You might try a simple visual check by using `imagesc` to plot each (2D, surface temperature) data field within the loop.
3. Then add in the code you developed for making a nice contour plot. Note that you will need to specify a vector of color contours so that you can give all the individual plots the same scale – see Section 3.2.1. Note that the code for creating the vector of contour values, and the lon and lat arrays created using `meshgrid` can go outside (before) the loop (they need not be re-calculated each month).
4. Finally, add in the code (given) for creating the animation.
5. (Check it correctly creates and saves an animation.)

Marking.

1. 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.
2. Maximum marks will only be awarded for choosing a (good) alternative to the default `colormap`.
3. Extra marks will be awarded for creating your own `colormap` – see Section 3.4.3.
4. Partial marks (only) will be awarded for a simple `imagesc` plot based animation, or a solution that does not use a loop to load in and plot all the frames of the animation.
5. A 10%-per-day late hand-in penalty will be applied.

Submit by email (andy@seao2.org) as an .m files. Please include somewhere a comment line with your name in it to ensure there can be no file mix-up, or better: include your name at the start of the filename.

The deadline (for the emails to appear in my in-box) is the start of class (8.10 am) on Monday 5th November.

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

6.1 Work plan

For Monday 5th 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. Specifically, we will recap on:
    - operators
    - `find`
    - string concatenation
    - indexing (of regions of arrays)
    - loops
2. There will be a chance to go through any outstanding micro-assessment **MATLAB** questions.
3. Section 4.1 – introducing the concept of 'nested loops'
4. Section 5.1 – putting nested loops to work!

(This work is to be finished by the end of the class on Friday 9th.)

6.2 Learning goals

More familiarity with loops, particularly nested loops
New **MATLAB** commands/functions:

- `axes`

6.3 Micro Assessment #4

This assessment has a single component:

1. Create a *script* **m-file** that draws a chessboard (https://en.wikipedia.org/wiki/Chessboard). You will need the following components/information:
   - A pair of nested loops (each counting from 1 up to the maximum number of rows/columns in a chess board (8×8, apparently)).
   - A **Figure** window to draw in, with a set of invisible axes and axis scaling from 0 to 9 in each dimension – exactly as per outlined in Section 5.1.
   - Use of the **MATLAB** function `patch` to draw the squares.
   - Some cunning way of deciding whether each square should be black or white and specify the (x,y) coordinates of the 4 vertices...

   How to decide which square should be black and which white is the only genuinely difficult part. Looking at a chess board picture and working along the bottom row first, from left to right, and then upwards – the first (bottom) row starts with a black square and then alternates in color. The 3rd row up is identical, as is the 5th ... etc.
   You should then see that odd numbered rows (counting upwards), have black squares in odd-numbered columns. Conversely, even-numbered rows, have black squares in even-numbered columns. Don't do anything else until you are satisfied about this – picturing and having a mental image of how you will go about addressing the problem (and hence write the code) is key.
   So, as you loop-loop (from 1 to 8) in both dimensions, you need to test whether the rows and column numbers are odd or not. If we assign whether or not the column number (x) is odd to the variable `colodd`, and the whether or not the row number (y) is odd to variable `rowodd` – setting these equal to `1` ('true') if the row/column is odd, and `0` ('false') if even, we have the following situation:

```
if (colodd && rowodd),
        % black
elseif (~colodd && ~rowodd)
        % black
else
        % white
end
```

   Or, if you prefer:

```
if ((colodd==1) && (rowodd==1)),
        % black
elseif ((colodd==0) && (rowodd==0))
        % black
elseif ((colodd==1) && (rowodd==0))
        % white
elseif ((colodd==0) && (rowodd==1))
        % white
end
```

The first form is just a slightly more compact version of the more explicit second version (that you might prefer). Go through this code mentally and:

(a) Satisfy yourself that all possible combinations of values of `colodd` and `rowodd` are accounted for.

(b) That the black and white squares are correctly distributed. Go through in your head (or on paper), for the first couple of values of column and row (e.g. 1 through 3 or 4 for both), that this is the case (i.e. that black and white squares will correctly alternate).

The final piece of the puzzle is to determine whether a column (or row) value is odd or even. Basically, if a number is even, it is divisible by zero with no remainder, and if it is odd, the remainder is `1`. **MATLAB** has a function for this – `mod` – see **help** and work out for yourself how to use it (it is not that hard – it takes 2 parameters as input – the number you want to test, and the base (divisor), which will be `2`; `mod` returns the remainder, which will either be `1` or `0` – exactly the values/format you need to assign to your variables `colodd` and `rowodd`).

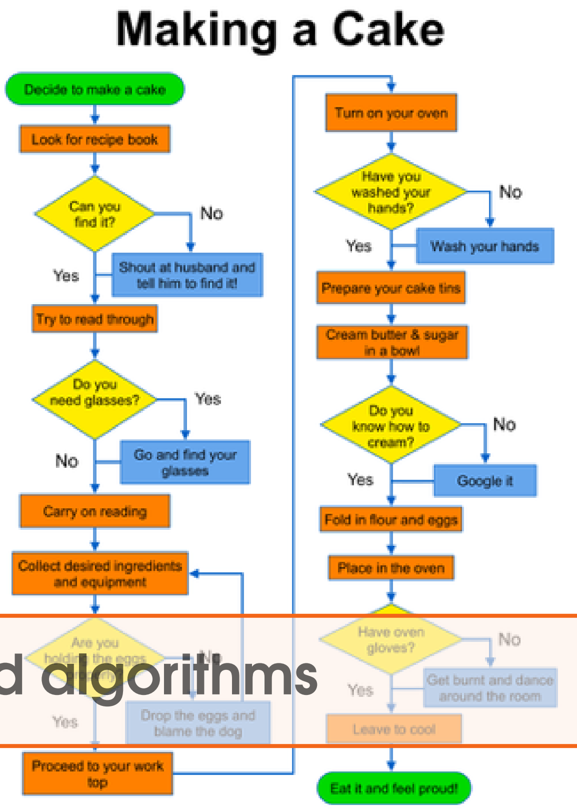
Overall, your code structure might look like:

- Comments describing the program ...
- Create the **Figure** window and drawing space (including scaling the axes from 0-9).
- Define parameters for the maximum number of columns and rows needed.
- Nested loop, looping through both column and row number. Within which you will:
  (a) Determine whether the row and column numbers are odd or even.
  (b) Determine whether the square is black or white and draw (`patch`) the square.
  Hint: For the coordinate parameters to be passed to `patch`, if your current location in the loop is `col=1`, `row=1` – the first (bottom left hand corner) square, the coordinates are:
  (0,0), (1,0), (1,1), and (0,1)
  and for which patch takes input:
  `patch([0 1 1 0], [0 0 1 1],'black');`
  The mental leap is then to notice that if your column (x) and row (y) values are held in variables `col` and `row`, respectively, you could write:
  `patch([col-1 col col col-1], [row-1 row-1 row row],'black');`
  See: last part of Section 4.1.
- (end the nested loop)

Submit by email (andy@seao2.org) the file. The name of the file should start start with your surname (and then contain whatever makes for a helpful filename ... ideally also somethgin indicating which assessment exercise it is for).

For the `.m` file, you will be marked for basic program structure and commenting of the code, in addition to the code correctly working! Maximum marks (100%) will be awarded for neat and compact code and/or clever solutions.

The deadline (for the emails to appear in my in-box) is 8.10 am on Monday 12th November.



7. Week #7 – Coding and algorithms

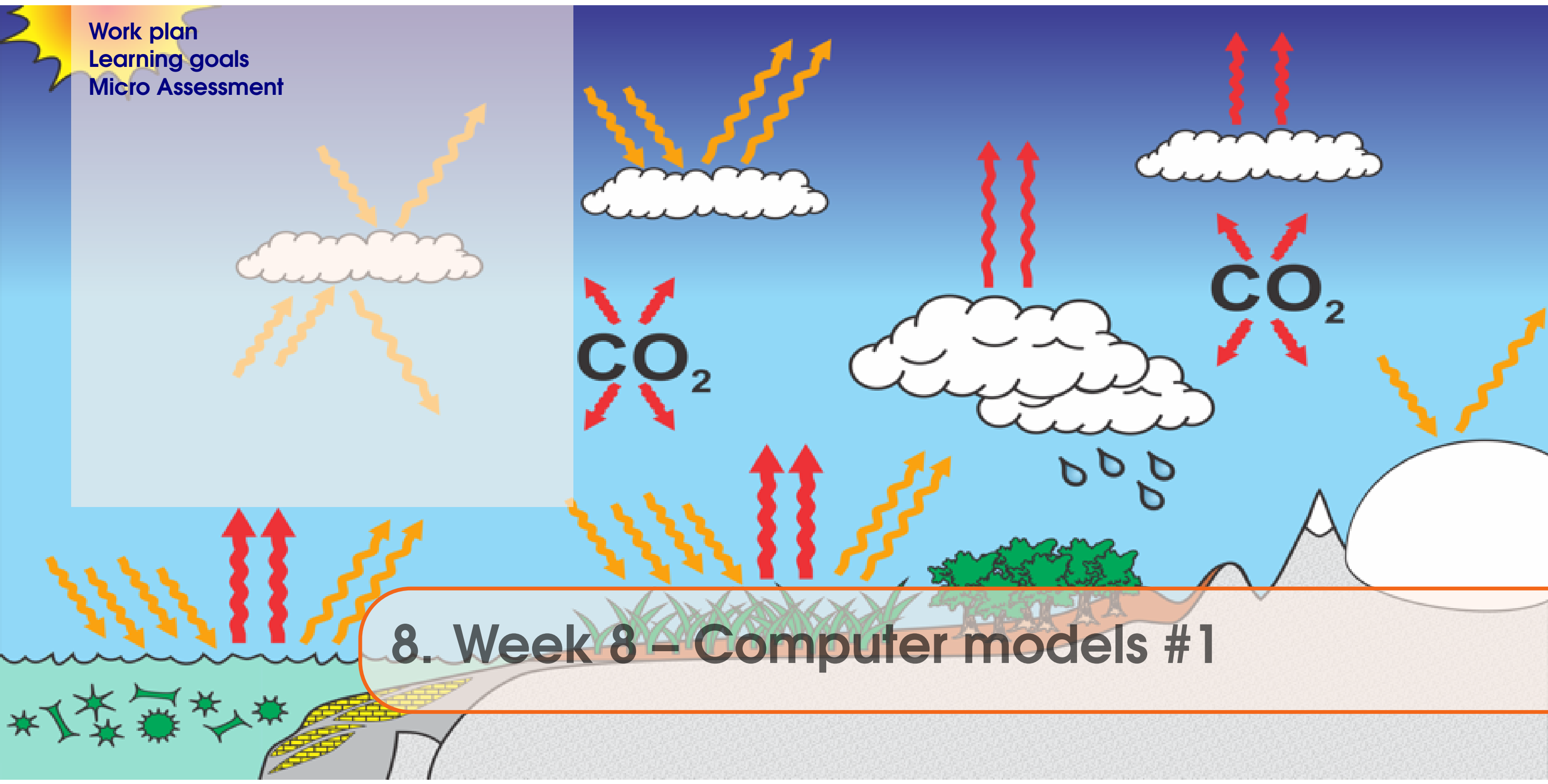
7.1 Work plan

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). But otherwise ...

1. Section 4.2.1 – worked through example code algorithm
2. Section 4.2.2 – second example (also worked through)
3. Section 4.2.2 – write your own sort algorithm that sorts in ascending order AND removes duplicates

7.2 Learning goals

(Solving problems in code.)



8. Week 8 – Computer models #1

8.1 Work plan

For week 8 – Monday November 19th:

main ...

Work through the main part of Section 6.1 – there are 4 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 ...

6.1.1 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 ...)

6.1.2 Turn this into a function and test it.

6.1.3 Create a function to calculate the value of the solar constant, given a value for 'time' (since the Suns formation).

6.1.4 Put all the functions together – calculating how Earths surface temperature may have evolved through geological time.

if you are bored ...

6.1.5

Explore and plot, how sensitive Earths surface temperature is to various parameter choices, using the climate model function.

8.2 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.

8.3 Micro Assessment

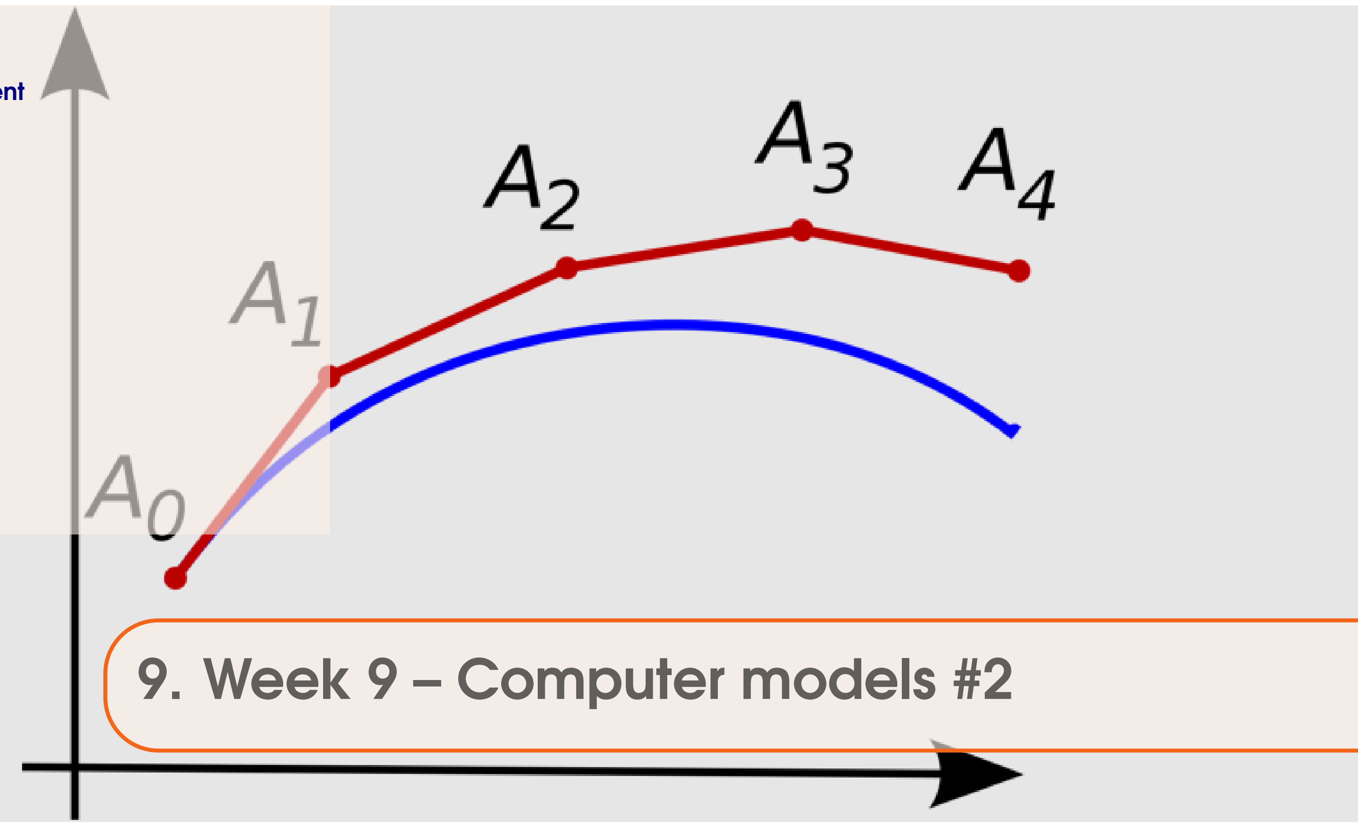
This weeks micro-assessment has 2 parts, based on the simple 0D climate and 'solar evolution' models that you wrote.

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 of $1368Wm^{-2}$, 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 single correct answer for any of these, depending on the albedo value you find and assume.)
2. Secondly – based on Section 6.1.6 (and having also worked through 6.1.5) – write a script **m-file** that calculates and plots how surface temperatures varies (in 2D) as a function of solar constant and albedo, over the range:
   - solar constant: $1000 - 1400Wm^{-2}$
     (with a suggested step size of $50Wm^{-2}$)
   - albedo: $0.0 - 0.5$
     (with a suggested step size of 0.05)

For (1), simply provide the answers to (i) through (v) as text in an email – give the albedo value you assumed alongside the mean global surface temperature of the planet in units of degrees Centigrade. For (2), submit the **m-file** that generates the required (and fully labelled/annotated) plot.

The deadline (for the emails to appear in my in-box) is 8.10 am on Monday 26th November.



9. Week 9 – Computer models #2

9.1 Work plan

For week 9:

1. **Monday November 26th.**

   Work through Section 7.1 and then *Part* 0 of Section 10.1. These involve:
   (a) 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.
   (b) Extending the graphics capability of the ballistics model.

   Also read the intro to Chapter 7.

2. **Friday November 30th.**

   Continue on from Monday.
   If you finish ahead of time, then try creating the time-dependent version of the 0D EBM and then adding greenhouse gas forcing and (driving with historical greenhouse gas concentration trends) – Section 7.2.

9.2 Learning goals

- More practice with, and exposure to, numerical models.
- More practice with scripts and functions, loops and counters.
- Learning some more advanced graphics usage.

9.3 Micro Assessment

For this 6th and final Micro Assessment – work through all the questions in the example Finals exam paper, following the instructions exactly, and email in your completed **m-file**.

You can find the uncompleted **m-file** (which you will complete as directed in the instructions) and associated data files on the course webpage (http://www.seao2.info/teaching.html).

The deadline (for the emails to appear in my in-box) is 8.10 am on Monday 3rd December.

NOTE: the GUI questions at the end are optional for this assessment (if you attempt them – they will be awarded additional marks over and above the 5% total available for this assessment.)

10. Week 10 – Putting it all together

10.1 Work plan

For week 10:

1. **Monday December 3rd.**

   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.)

   There is an example set of code and GUI files provided for guidance – see the blue box on the left hand side of the course webpage (under week 10, at the bottom).

2. **Friday December 7th.**
   Will be the Finals exam ... (see Intro).