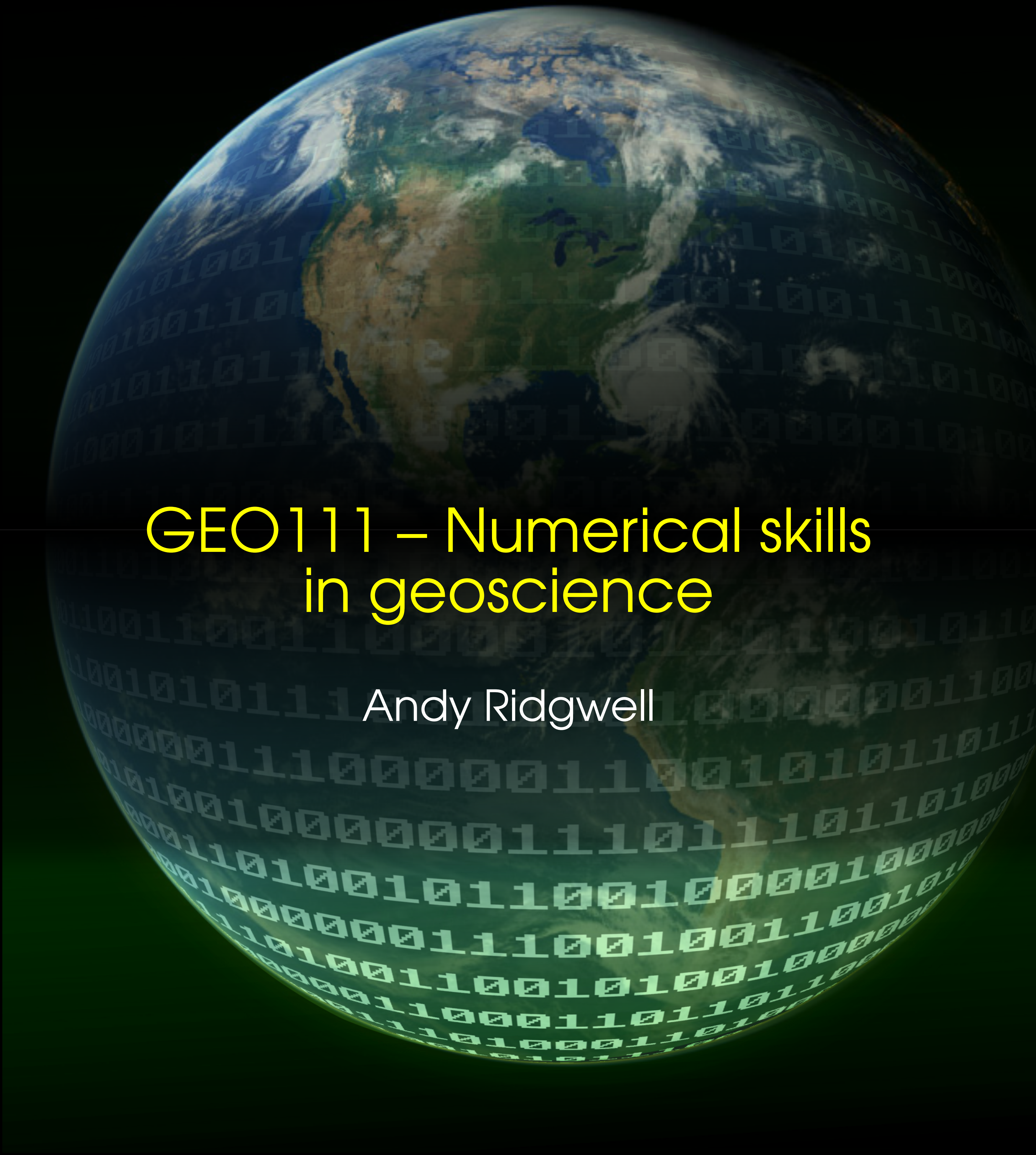
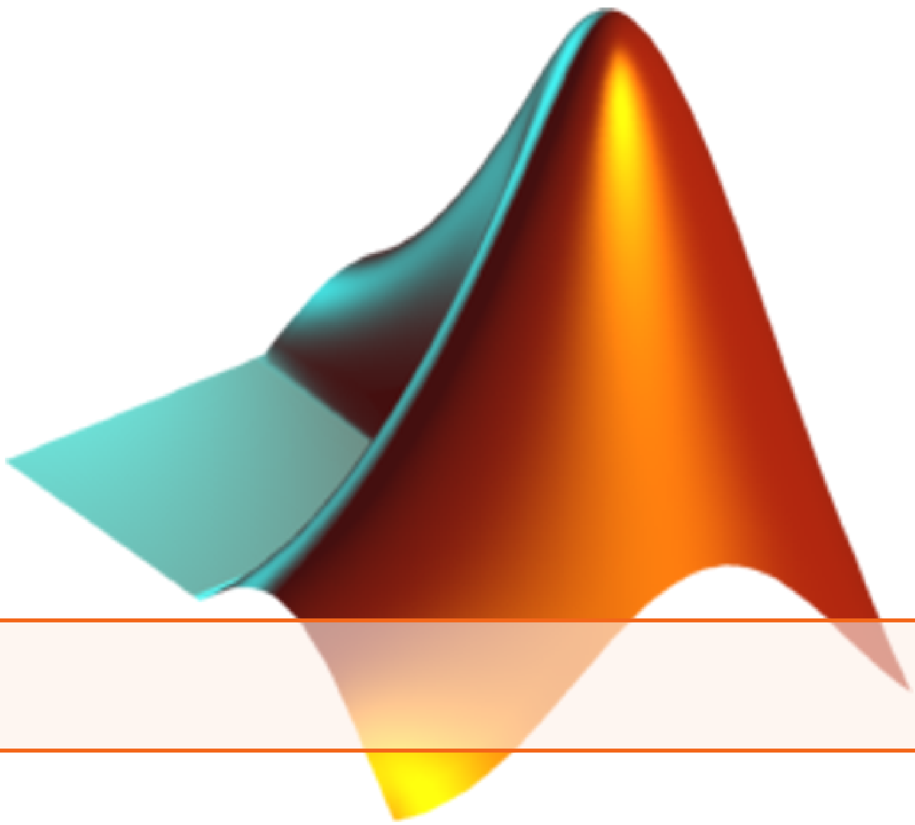


GEO111 – Numerical skills in geoscience

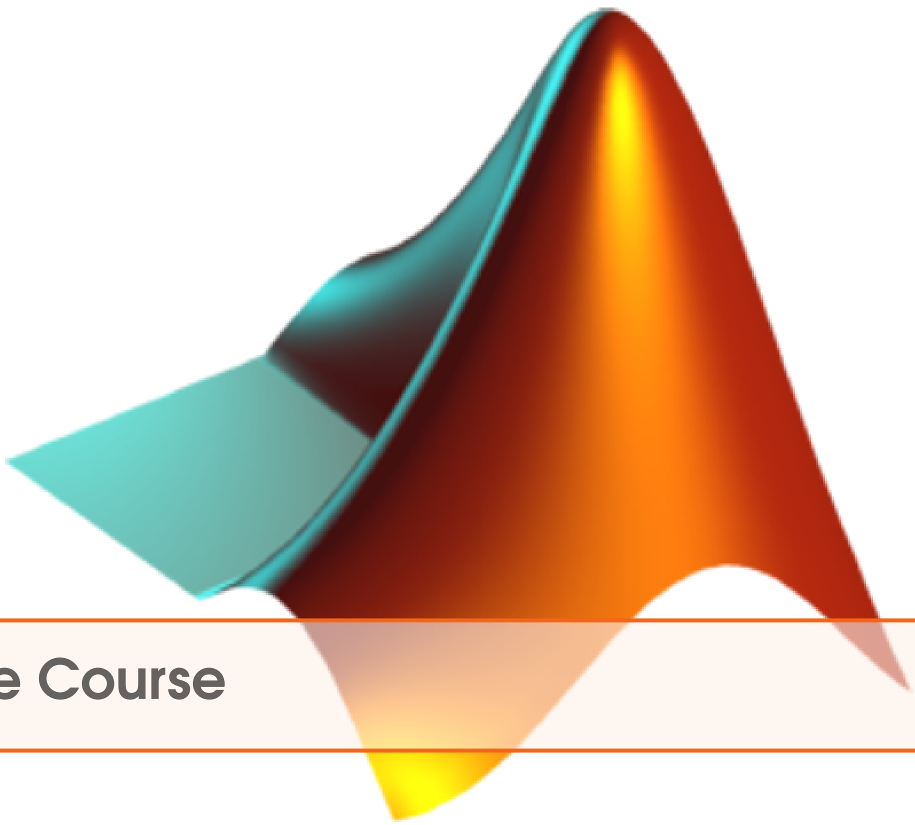
Andy Ridgwell

December 4, 2020

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.
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 ’work’.
- How numerical models are constructed and applied.
- The use of numerical models in addressing scientific questions and testing hypotheses.

and provide transferable skills in

- Problem solving and logical analysis, fault-finding.
- Data processing and visualization.
- Computer programming.

You will also learn ... extreme patience (with computers) ...

0.1 Course Overview

0.1.1 Format

The weekly format of GEO111 is: 1 $\times$ 1-hour lecture/discussion together with 1 $\times$ 3-hour computer practical session, plus 1 $\times$ 2-hour interactive lecture/discussion session of worked problems and examples. 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.

Each week, 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). There will also be some project work, enabling you to apply your newly learned skills to a subject of particular interest (or relevance to your Majors program). The assessed work will be set at the end of each Friday class (4 pm) and will be due in the following Friday (at 2 pm). If stuck, please make use of Office Hours (see below).

You are expected to have completed all class work set each week, by the start of class the following Monday.

Note that in the course format of 3 hours total per week of lecture/discussion, UCR regulations require that you complete at a minimum, an additional 6 hours of course-related work per week in order to count as 3 credits. This additional work may take the form of the set assessments (and later the programming project), and completing weekly class work. (The 3 hours of designated lab work counts as the 4th course credit.)

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:

- 7 $\times$ weekly micro assessments @ 5% each = 35% total
- Group programming project – 25%
- Finals paper – 40%

As per the time-table – for the first 7 weeks, a short assessment will be set on Friday at 4 pm. The hand-in date will be the following Friday at 2pm (PST). Each micro assessment will be worth 5% of the total marks available for the course. The idea is to help reinforce what you should have been doing and learning that week. 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 assessments ...

The programming project will be a more free-form version of the weekly assessments; you will have 3 weeks in order to complete it and it can be conducted in groups of ca. 2-4. A written plan (no more than 1 page) for the programming project must be agreed with the Instructor during week 9 and prior to embarking on it.

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 any on-line documentation are allowed. A maximum of 3 hours will be allowed.

0.1.4 Instructors

The Instructor for the course is: Prof. Andy Ridgwell
Email: <andy@seao2.org>

The TA for the course is: Pam Vervoort
Email: <pverv001@ucr.edu>

0.1.5 Office Hours

TA Office Hours are Tuesdays and Wednesdays, from 4 pm to 5 pm, or at other times by prior arrangement. This will be via Zoom, and Pam will send a link to everyone via iLearn shortly before 3pm.

Note that 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 plan on treating the Friday class as a kind of group tutorial session plus the ability to complete any unfinished work from the Monday class.

Feel free to contact me at any time[1] with any questions, problems, or concerns, in addition to making use of TA Office Hours,. Contact me by email, and/or we can arrange a short Zoom.

Simply view any situation in which 'normally' you might have arranged to meet the Instructor or TA to discuss something, as a video conference possibility. Exciting!

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[2]! 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.[3]

0.1.6 Communications and Course Material

Group (email) communications, as well as the setting and submission of assessments will be via iLearn.

However, all[4] course materials will appear on my website – www.seao2.info under the Teaching tab and in the 'GEO111' box highlighted in green:

- The GEO111 course guide/syllabus [PDF file format].
- A copy of an old example weekly schedule guide from 2019/20.
- The GEO111 course text [PDF file format].
- An introduction-to-**MATLAB** booklet [PDF file format].
- Any lecture presentation materials [PDF file format].

[1]Tuesdays and Thursdays are best for recieving a quick response.
[2]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.
[3]Instead, some of the programming syntax in MATLAB is genuinely stupid.
[4]Incudling a duplicate copy of the muffin manual PDF.

Note that the GEO111 course guide/syllabus and course text will be updated periodically – at least weekly and quite possibly for the course text, daily at times. Web browsers tend to cache documents, meaning that you may be downloading an 'old' version, even if you have logged out in between ... To force a re-load of the webpage cache in your browser:

- Windows – press the Ctrl-F5 key combination.
- Mac – according to Google (and I am unable to test this) ...
  "In Camino and Firefox, press Shift-Command-R (or hold Shift and click the Reload button). In Safari, hold the Option key and press the Reload button in the toolbar."

0.1.7 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?`'.

0.1.8 Required Software

GEO111 requires the use of the programming language/software suite, **MATLAB**. To access this,

- **Either:**
  On a desktop or laptop, running any of: Windows, MacOS, or linux operating systems, and on which you will install the **MATLAB** software suite. UCR provides you with a free 1-year (renewable) student license. The instructions for obtaining and installing **MATLAB** can be found here:

  https://sitelicense.ucr.edu/matlabstudent.html

  Follow the instructions ... carefully. You will need FIRST to create a (free) account with Mathworks (the **MATLAB** company), then obtain a license key, download and install the software from Mathworks, and then activate the software using the license key you have just obtained. Follow the step-by-step instructions and all should work out A-OK.
- **Or:**
  Via any computer and and operating system (including Chromebook) using a web browser – you will log in and connect to a UCR virtual lab ('VLab') and on that virtual machine, run **MATLAB**.

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

ITS have created an on-line software access system – **Apporto** – UC Riverside's newest virtual computer lab (VLab) service[6]. You simply need to visit http://ucr.apporto.com/ and log in using your UCR NetID credentials to access the app store.

From http://ucr.apporto.com/, click on **RDS Apps Full Desktop**. Once booted into the Windows Desktop, you'll see a folder icon for **MATLAB 2020a** – double-click on this to start **MATLAB**. (The virtual machine also has Word installed and various other softwares.))

For complete instructions on how to use the new VLab, refer to the link given in the foot-note.

You will probably find installing **MALTAB** on your own desktop/laptop best, but the initial download is large and you'll need several GB of free disk space. Via **Apporto** will work best via a good internet connection, and you'll have to work in the Microsoft Windows operating system (what you may not be familiar with if you generally use a Mac). Via **Apporto** may be advantageous if your internet connection is good but your computer very slow.

You can always try both methods and see which one you prefer!

[6]https://ucrsupport.service-now.com/ucr_portal/?id=kb_article&sys_i d=8b5964291b84d49026bd635bbc4bcbd7

0.2 Course assessment

0.2.1 Weekly Homework

For the first 7 weeks of the Quarter, a short assessment will be set each Friday at 4 pm. The hand-in date will be the following Friday at 2pm (PST). Each assessment will be worth 5% of the total marks available for the course. Assessment hand-in will be via **iLearn**.

The aim of the micro-assessments is to help reinforce what you should have been learning.

You will also be required to ensure that all the class work and exercises are completed prior to the start of the class, the following week. (And so may be some mix of ’new’ programming exercises and write-ups of scheduled labs.)

0.2.2 Programming project

During week 7, you will embark on a group programming project. The hand-in date will be Friday 4th December (the end of Week 9) at 2pm (PST). This project will be worth 25% of the total marks available for the course. Project hand-in will be via **iLearn**.

The project will be set at the start of week 7, and you will be given 4 options to choose from.

You will be allowed to, and in fact encouraged, to work in groups to make it more ’fun’ and to help learn from each other.

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.

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 for the end of week 10 – from 2 through 5 pm on Friday 11th December, 2020.

Study aids and reference materials are allowed, including:

- Notes.
- Textbooks (including the GEO111 course text).
- **MATLAB** Help.
- Anything on the course webpage.
- The internet! (but excluding social media ...)

The Finals will account for 40% of the total marks for the course.

0.3 Medical Matters, Disability, and Help! in general

A personal note ...

Life happens. If you are having issues of whatever sort in your life that impact on your ability to complete class work and/or assessments in a timely fashion, please let me know. I cannot offer additional help or guidance, nor be flexible with deadlines etc., if I do not know about any issues (I do not need to know any details!). If you inform me of a valid reason for not being able to meet a deadline only after the deadline, I cannot (and probably will not) make any accommodation.

There are numerous campus resources for supporting you in times or situations of difficulty. I am also genuinely happy to do whatever I can to help. Please do not suffer in silence.

Should you need different accommodations for the form of the class material, exam rooms and examination environment, or anything else like that, please also let me know (in advance!) and I will do everything I can to accommodate you.

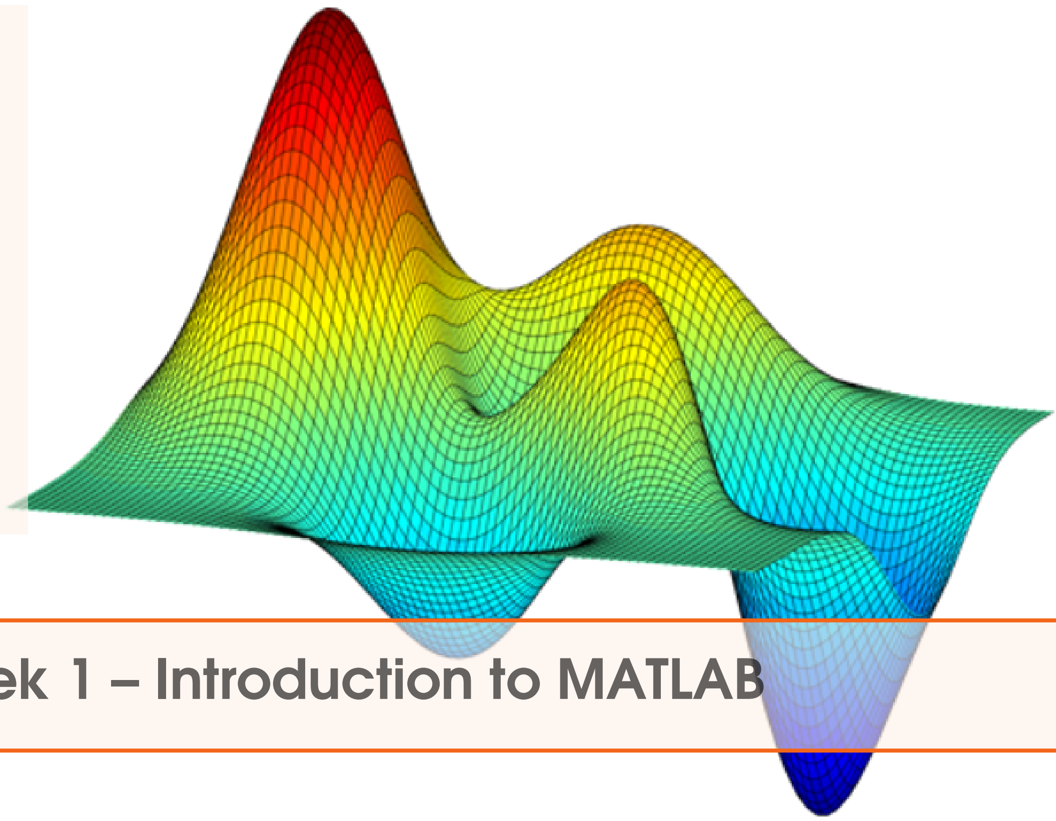
From UCR (with love)

Medical Matters and Disability: 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.

Table 1: Schedule of GEO111

Week #	**Monday lecture**	**Monday LAB**	**Assessment set**	**Friday LAB / discussion**
01 (10/05)	Introduction to the course & to the MATLAB language and software environment.	Elements of MATLAB and data visualization.	**1.** Basic programming practice and exercises.	How computers 'work'. + MONDAY CONTINUED
02 (10/12)	Fundamentals of computer programming I.	Scripts and functions in MATLAB. Loops and conditionals.	**2.** Program structure practice and exercises.	Other computer languages. + MONDAY CONTINUED
03 (10/19)	Fundamentals of computer programming II.	Further MATLAB and data visualization.	**3.** Data anaylsis and plotting practice.	MONDAY CONTINUED
04 (10/26)	Fundamentals of computer programming III.	Algorithms, problem solving, tricks and techniques	**4.** Data processing exercise.	MONDAY CONTINUED
05 (11/02)	Statistics and encoding math.	Further data analysis and statistics in MATLAB.	**5.** Loops exercise.	MONDAY CONTINUED
06 (11/09)	Numerical modelling and encoding physics(!).	Basic (zero-D) numerical modelling.	**6.** GUI fun(!).	MONDAY CONTINUED
07 (11/16)	Computer Graphical User Interfaces(GUIs). **Programming project SET**	Basic GUI creation in MATLAB.	**7. TBA**	Apps and object-orientated programming
08 (11/23)	Basic climate physics.	Building a 0D climate model!	**NONE**	UCR HOLIDAY – Thanksgiving
09 (11/30)	More physics in programs.	Dynamic (time-stepping) modelling – ballistics 101.	**NONE**	MONDAY CONTINUED **Programming project DUE**
10 (12/07)	TBA	Putting it all together – a GUI- and physics-based game(!)	**NONE**	**Final exam**



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. On Friday, we will tackle Sections 1.4, 1.6, 1.7 and 1.8 (which covers data loading and saving, and plotting).

It is likely that the Monday sections will not take you all of Monday class ... so move on to Friday's material when you are done!

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]** (available to download from my website)

- 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 for the week (not assessed ...)

In your own time – 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.

1.4 Assessment #1

The assessment for week #2 is as follows, and will require you to submit to iLearn:

a single JPEG (JPG) format file (and no other file type)

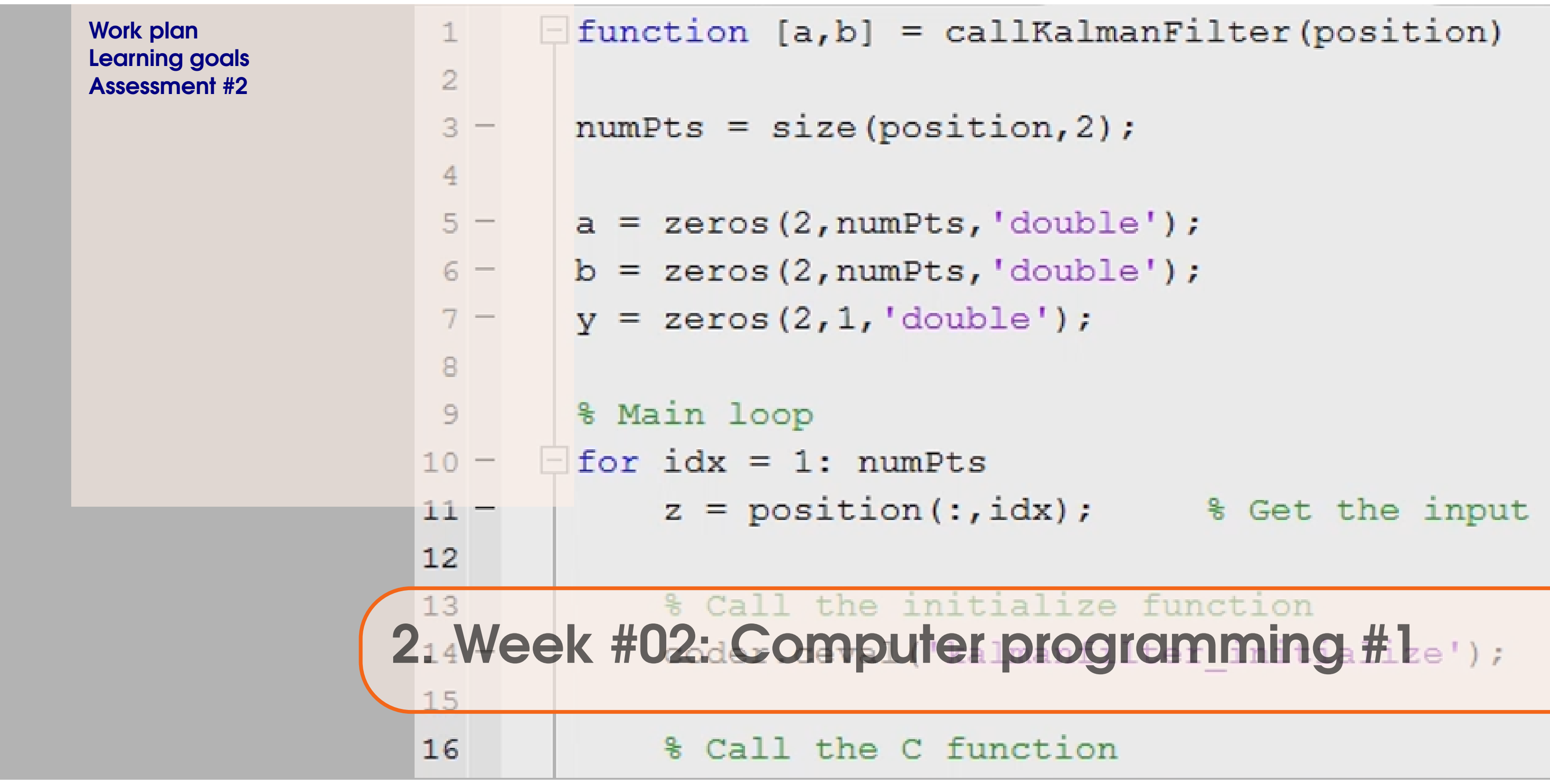
The deadline for submission will be:

2 pm, Friday 16th October.

Feel free to come for help during Office Hours the week prior to the hand-in date, and the week after if you want any additional feedback on how you did. You can also contact/email us both at any time with questions. A late hand-in penalty will be applied if you did not request an extension (with reason) prior to the submission date.

The purpose of the assessed exercise is to create an appropriately-labelled graph (and save as a jpeg format graphic file), with the following features:

1. The change with time of the number of new daily COVID-19 cases in Riverside County.
   The data file you need – 'Riverside COVID-19 data' – can be found on my website under **Assessment #1**.
   View the file first, e.g. in a text editor (or import into Excel) to find out what the different columns represent. You need to plot 'new_confirmed_cases', vs. 'date' (so you need to find out the column number for both of these). Note that for ease of plotting, I have converted the actual calender date, into a numerical equivalent.
   The line color used in the pot should be RED.
2. On top of the Riverside data and in the same Figure Window, plot a second data time-series of the number of new daily COVID-19 cases in San Bernardino County.
   The data file you need – 'San Bernardino COVID-19 data' – can be found on my website under **Assessment #1**.
   Remember, you will have to use the `hold on` command in order to have a second plot appear on top of the first (and in the same Figure Window), and will have to do this before you plot the 2nd dataset.
   The color of this second line should be BLACK and the line should be DASHED.
3. Add a legend, and ensure that the axes are appropriately labelled and that the plot has a title.
4. For additional marks ... as you will see, the x-axis scale is a hot mess of virtually meaningless numbers for the 'date'. We could transform this such that the plot started at 'day zero' for the COVID-19 outbreak. Will will assume that day `43861` (the first day in the Riverside dataset) corresponds to (day) zero. If you subtract `43861` from the first column of both `riverside` and `san_bernardino` data arrays, you will achieve this transformation.
   See if you can do this – have the plot start at zero on the x-axis, and run to a maximum value of `260` (so you will have to specify the x-axis limits).



```
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 #1

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 and Friday pm labs. Feel free to skip Section 2.3.2 if you are feeling overwhelmed (but still skim through and note the existence of `switch ... case ...`).

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 ...

Remember that in the course text – words in capitals typically indicate that a variable (or string) goes in that location in the code, but that you should come up with your own name (or string). So in place of the occurrence of `MY_VARIABLE` in the text, in your code, you might have `variable1` or `number_of_fruit` or something. (See Section 0.5 in the course text.)

2.2 Learning goals

Topics and methodologies you should be familiar with:

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

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

- the `function` definition
- conditional structures: (a) `if ... end`, (b) `if ... else ... end`, (c) `if ... elseif ... else ... end`
- misc functions: `disp`, `input`, `strcmp`

2.3 Assessment #2

This weeks assessment will be in the form of a series of **MATLAB** .m files that you will upload to iLearn. The hand-in deadline is 2 pm Friday 23rd October.

There are 5 files to create or complete, and hand in, each weighted 20%. Marks will be deducted for poor or insufficient code % commenting.

Please following the naming convention for the 5 filenames when you come to upload to iLearn.

1. Write a *script m-file*, that loads in the COVID-2 data for Riverside county, and plots the cumulative number of cases cases with time (so different from the 1st assessment – view the file in a text editor or **Excel** before loading to work out which column is which). Use scatter to plot as filled, red circles.
   Remember to label axes etc.
   Bonus marks if you can color-code the points by number of daily cases (instead of all red point markers).
   Name the file to upload: myscript.m
2. Write a *function* m-file, with the following properties:
   - Takes a single input.
   - Returns a single output.
   - The value returned by the function should be equal to the factorial of the input – see **MATLAB** help on `factorial`.
     (Yes, I know that you are taking a built-in **MATLAB** function, and simply wrapping it up in your own function ...)

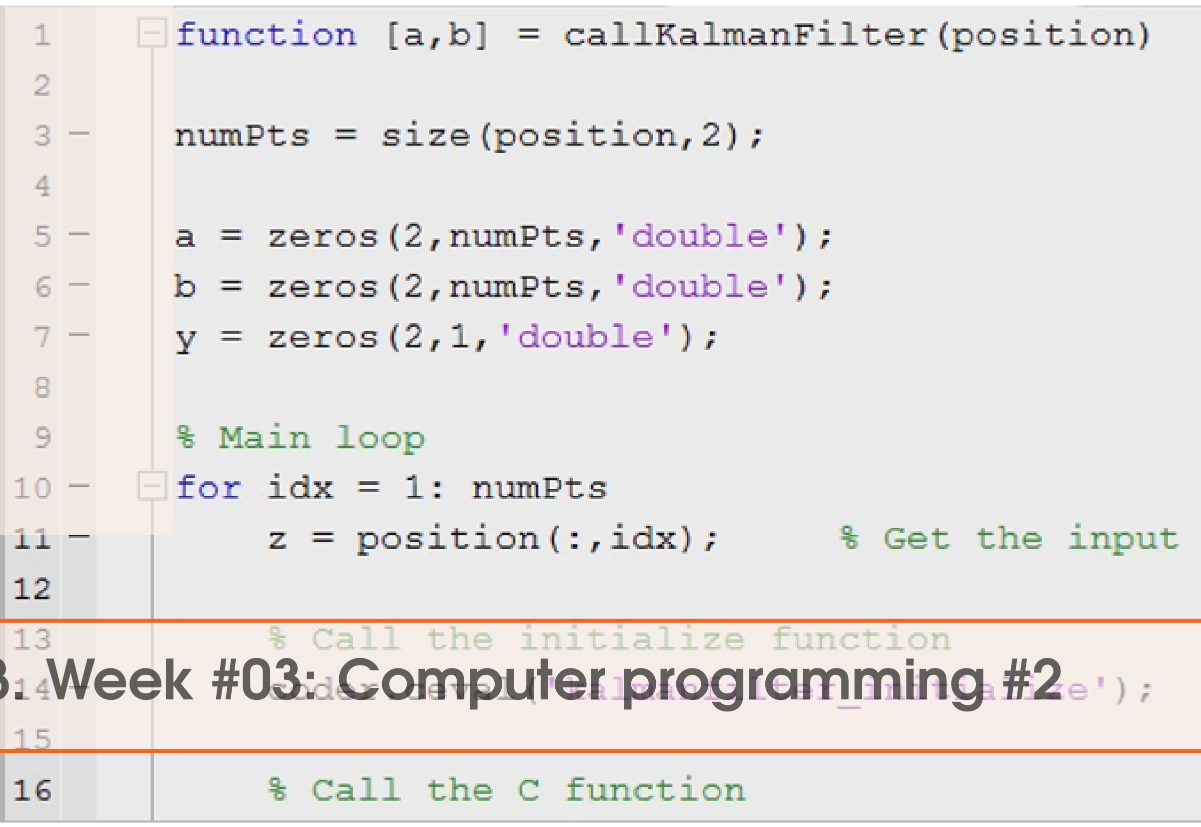
   A successful function will do the following when called at the command line:
   ```
   » myfunction1(10)
   ans =
   3628800
   ```
   (Note, this function only works with positive integers ...)

   Name the file to upload: myfunction1.m
3. Write a *function* m-file, with the following properties:
   - Takes 2 inputs.
   - Returns a single output.
   - The value returned by the function should be equal to the greater of the 2 inputs.
     HINT: If the 2 variables passed into the function are `x` and `y`, and your output is `z`, then you need an `if ...` statement to test whether `x` is greater (or equal to) `y`, and if so, set `z` equal to `x`. `else`, set `z` equal to y.

   A successful function will do the following when called at the command line:
   ```
   » myfunction2(10,20)
   ans =
   20
   ```
   Name the file to upload: myfunction2.m
4. Fix the buggy function debug1.m (which can be found on iLearn). There are several minor bugs in this code.

HINT: there are 3 main bugs in total.
Name the file to upload: debug1.jpg

5. Fix the buggy function debug2.m (which can be found on iLearn). There are several minor bugs in this code.
HINT: there are 3 main bugs in total (that prevent the code working at all). Bonus marks for finding and fixing the 4th minor bug.
Name the file to upload: debug2.jpg



```
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]e');

    % Call the C function
```

3. Week #03: Computer programming #2

3.1 Work plan

For Monday – work through the remainder of Chapter #2 of the GEO111 course text – Section 2.4 and Section 2.5. (Section 2.6 – Even more (and loopier) loops, is optional.)

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

1. **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 ...
2. **Section 2.5. Loops and conditionals ... together(!)**
   Combining conditional and loop structures in the same program code.

For Friday – in order of priority:

1. Make sure that you have finished off anything and everything left over from Monday, because we will move swiftly on to more fun (graphics) things for Week #4.
2. Make sure that you have handed in the 2nd assignment ...
3. See if you can get Section 2.6 to work. An example code is given at the back fo the book to help guide you structure your program.

3.2 Learning goals

Topics and methodologies you should be familiar with:

- loops
- string concatenation

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

- loop structures: `for ...`, `while ...`
- `num2str`, `break`, `exist`

3.3 Assessment #3

This weeks assessment will be in the form of a series of **MATLAB** .m files that you will upload to iLearn. There are just 2 files to create and hand in, each weighted 50%. Marks will be deducted for poor or insufficient code % commenting and structure (indentation). Please follow the naming convention for the filenames when you come to upload to iLearn. The hand-in deadline is 2 pm Friday 30th October.

1. Write a *function* m-file, with the following properties:
   - Takes a single input.
   - Returns a single output.
   - The value returned by the function should be equal to the sum of the squares of all (positive) integers up to and including the value of the input (which is assumed to be an integer).

   HINTS: There are several ways to go about this – for instance, you could construct a loop inside the *function*, which takes its maximum loop count from the value of the input. You'll then need to update a running sum of the square of the loop counter in a similar way to a *function* you worked on in class.
   For bonus marks: solve the problem without the use of a loop.

   Name the file to upload: sumsquares.m

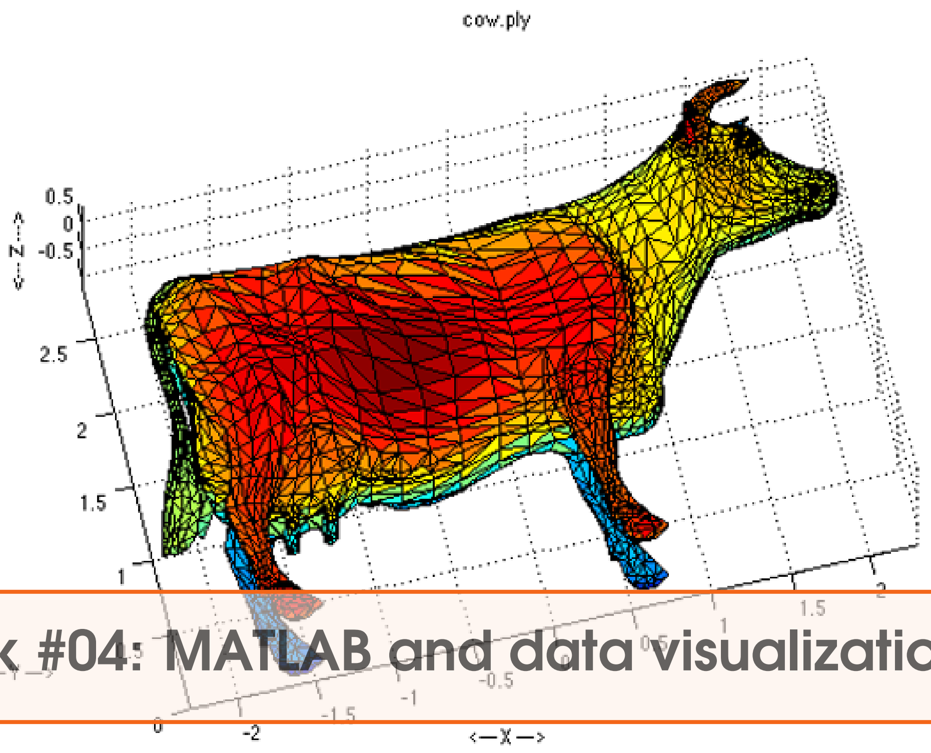
   A successful function will do the following when called at the command line:

   ```
   » sumsquares(10)
   ans = 385
   ```

2. Write a *function* m-file, with the following properties:
   - Takes a single input.
   - Returns a single output.
   - The number returned by the function is equal to the reciprocal of that number, UNLESS, the input number is equal to zero, when a NaN is returned instead.
     HINT: In the *function*, you need to test whether the input is equal to zero, and if it is, set the output equal to `NaN` (otherwise set the output equal to the reciprocal of the input).

     Name the file to upload: recip.m
     e.g.

     ```
     » recip(10)
     ans = 1.000000000000000e-01
     » recip(0)
     ans = NaN
     ```



4. Week #04: MATLAB and data visualization

4.1 Work plan

Work through the following sections of Chapter #3 of the GEO111 course text.

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.

Sections marked [**OPTIONAL**] you are free to treat as ... 'optional'!

For Friday:

1. Sections: **3.3.1** (find!), **3.3.2** (Other data filtering), **3.3.3** (Some useful data manipulations techniques), **3.3.4** (Data interpolation).
2. Ensure you have completed the work from Monday (Sections: **3.1**, **3.2**, **3.4**).
But only after the new stuff in **Section 3.3** ...

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 coutours
- basic data filtering and manipulation techniques
- how to interpolate data

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

- additional gridded plotting: `imagesc`
- 2D plotting: `contour` and `contourf`
- plotting controls: `clabel`, `colorbar`, `colormap`
- `meshgrid`
- `find`, `reshape`, `interp1`

4.3 Assessment #4

There are 2 parts to the 4th assessment (with marks weighted 40:60).

(1) Excel file and `find` exercise

Your first task is to plot the cumulative COVID-19 cases for San Diego county. You will need to upload BOTH the **m-file** (**covid.m**) AND the jpeg format plot (**covid**.jpg) that results from your **m-file**, to iLearn.

The aim is to practice importing data from Excel and **MATLAB** `find`. You can find the Excel file on iLearn – file:latimes-county-totals.xlsx.

You need to plot the cumulative COVID-19 cases, which in the original **Excel** file is the column labelled 'confirmed_cases' (so take a note of the column number, because the header line with the column labels will be stripped out).

Your problem ... is to extract only the data for San Diego county. Luckily, it has a unique identifying ID in column 3 – 73.

You need `find`, to find all the rows for which the value in column 3 is 73. Either assign all these rows to a new array, or delete all the rows that do not meet the criteria (of column 3 == 73).

Then make a nice plot of the cumulative COVID-19 cases (anything you like, as long as it is adequately labelled).

(Refer to the examples in the text for help, or even the code in the subsequent exercise.)

(2) Plotting/graphics exercise

Your second task the development of a script **m-file**, that produces a plot. You will need to upload BOTH the **m-file** (**ridgecrest.m**) AND the jpeg format plot (**ridgecrest.jpg**) that results from your **m-file**, to iLearn.

The aim of the assessed exercise is to produce an annotated figure of the M7.1 Ridgecrest, CA Earthquake Sequence. For background, read this USGS piece:

https://tinyurl.com/y4gw9tnk

as well as the WikipediA entry:

https://en.wikipedia.org/wiki/2019_Ridgecrest_earthquakes

Your figure is going to end up (hopefully!) looking a little like a combination of the '*Recorded earthquakes during the first 3 days of the July 2019 Ridgecrest earthquake sequence.*' time-line figure in the USGS piece, and the '*The July 2019 Ridgecrest earthquakes consist of three main shocks of magnitudes 6.4, 5.4, and 7.1, each followed by a flurry of aftershocks of substantially lower magnitude.*' time-line figure on the WikipediA page.

I have managed to obtain data similar to that plotted in these published figures from the USGS (https://earthquake.usgs.gov/earthquakes/search/). You do not need to try and obtain this data yourself – it is available in iLearn as file: query.csv.

First, view the file (it is text / ASCII) and think about how you might load/import it. Try using the **MATLAB** Import Data wizard to load the data and view it.

Because the data format has ... some difficulties associated with it, I have written a script **m-file** to import and carry out an initial processing of the data. This **m-file** can be found on iLearn.

You are welcome to use and add to, this **m-file** as part of your answer. Or copy-paste the code to your own **m-file.** (Or completely write your own code got loading the data and extracting time and earth quake magnitude if you wish!)

View the `usgs.m` **m-file** and make sure that you understand what it is doing. I have added lengthy comments to the code to try and fully explain what each bit is doing. Try it out (i.e. run the **m-file**, having checked first that the data file is in the same location as the **m-file**). There is a `scatter` function on line 77 that can be un-commented to get a crude initial visualization of the data. Note that the data looks similar to the dataset in the USGS article, except I am pretty sure that they screwed up their time axis :o) (My download also did not include quite as much data after the M7.1 event as on the USGS webpage or in the WikipediA article figure, and so expect it to be slightly shorter in time.)

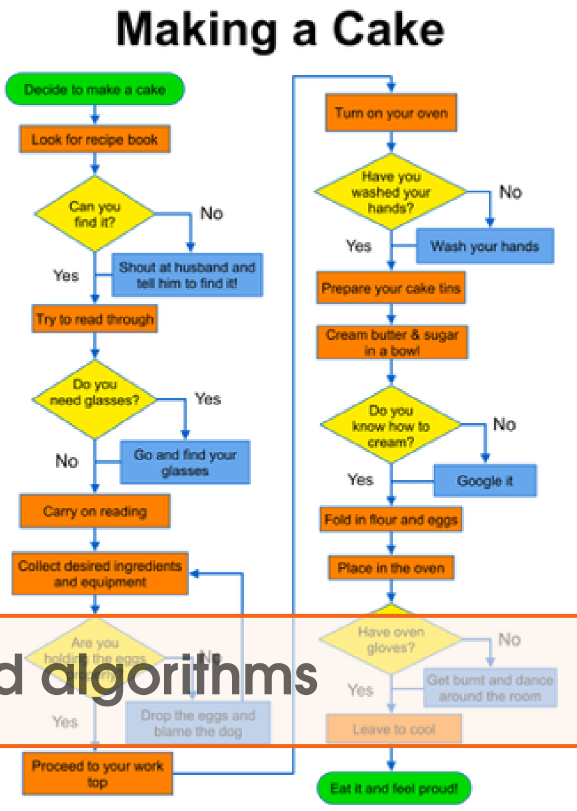
Your fist task is to complete the data extraction – line 94. Simply view how the first part of the data extraction was done, and write your own for the 2nd part to create 2 vectors – one of all events greater than or equal to magnitude 3.0, and a second vector with the corresponding time.

Then you need to plot it all up. For full marks – your plot should have all the following features (in addition to all the 'usual' necessary plot annotations):

- Data points as filled circles (using `scatter`). The circle fill should be in an orange color (see section on color in course text) for magnitudes less than 3.0, and red for magnitudes equal to and greater than 3.0.
  If you completed the code earlier in the program, you will have 2 sets of vectors for the 2 different sets of magnitude values (with each set getting colored differently).
  Bonus marks if you can plot the edges of the circles in white (as per in the WikipediA figure).
- x-axis labels displaying: 'July 4th', July 5th', July 6th', 'July 7th' – similar to in the USGS figure but omitting the time of day on the line below. Your labels should ideally align with the start of each day, i.e. a day value of 4.0 on the x-axis would correspond to (and be replaced by) 'July 4th'.
- Labels for the M6.4 and M7.1 events using the **MATLAB** `text` function, as per in the WikipediA figure (with the magnitude value).
  (You may need to do a little trial-and-error to refine where the text appears.)
- A line plot of the cumulative number of aftershocks as a function of time. (Plot as a black line on top of the data points.)
  You are going to need to plot this on a 2nd right-hand axis, because the scales are different (between magnitude vs. cumulative count of >= M2.5). You do this as follows:

  ```
  yyaxis right
  plot(x,y);
  ```

  (after you have scatter-plotted the points in the **m-file**), where `x` is your time vector, and `y` the cumulative number of events.
- A legend.
- Grid lines (use `help`, or internet search, or refer to the **MATLAB** Setting Started Guide you should have read the first part of).
- A horizonal black dashed line at a value of M3.0.



5. Week #5 – Coding and algorithms

5.1 Work plan

Monday 2nd November:

1. Section 4.2.1 – worked through examples of algorithms (/problem-solving programs)
   (You can skip over the sections labelled OPTIONAL if you wish.)
2. Section 4.2.2 – more worked through examples of algorithms (/problem-solving programs)
   (You can skip over the sections labelled OPTIONAL if you wish.)
3. Section 4.1 – introducing the concept of 'nested loops'

Friday 6th Nov:

If you have not completed it already – Section 4.1 – 'nested loops'.

5.2 Learning goals

Topics and methodologies you should be familiar with:

- loops, particularly nested loops
- solving problems in code

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

- `rand`

5.3 Assessment #5

Your task for the 5th assessment is to develop a *function* **m-file** that produces a plot. You will need to upload just the **m-file** (**gamegrid.m**) to iLearn.

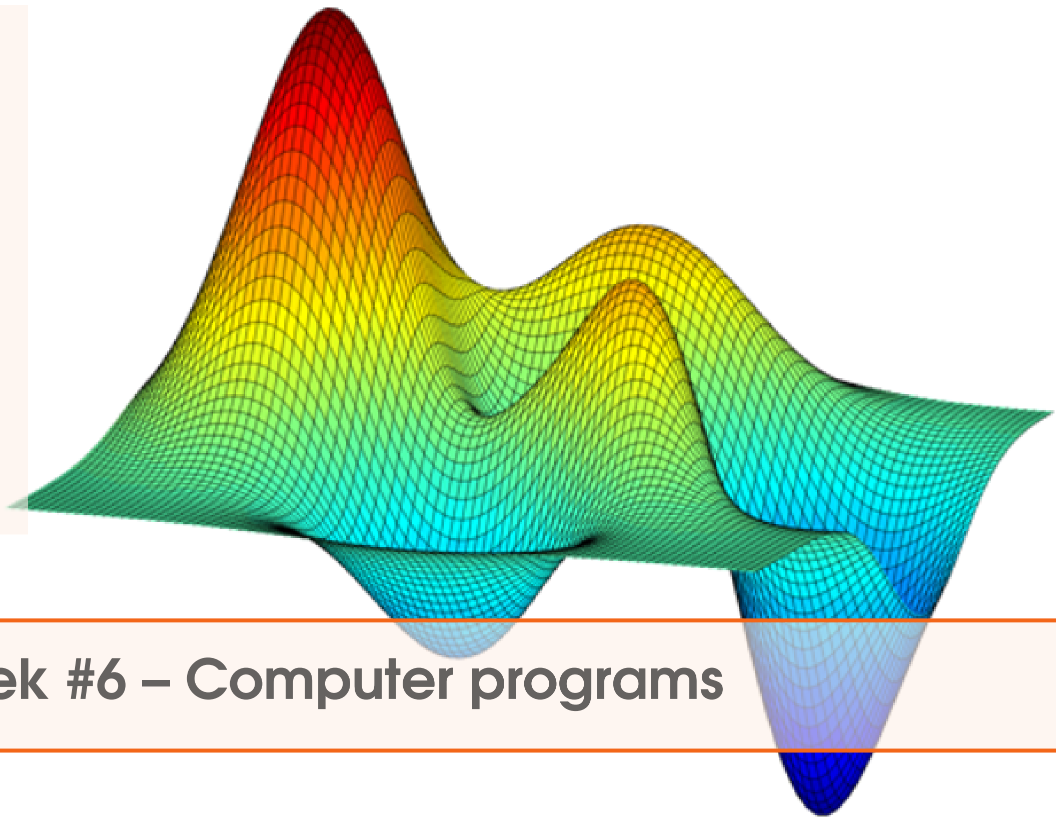
The *function* should:

- Take 2 inputs (but return no outputs).
- Plot a grid of squares (e.g. see the examples in Section 4.1).

  The grid of squares should:
  - Be of any color you like except red.
  - But, have a red border to the rectangles (refer back to what we went through in the class and the `'EdgeColor'` property of `patch` ...).
- Finally, the dimensions of the grid should be set by the 2 inputs to the *function*.
  i.e. if you were to call:
  ```
  » gamegrid(5,7)
  ```
  then the grid of squares would be 5 across (columns), by 7 high (rows).

This is very similar to both Section 4.1 examples, except being in the form of a *function* rather than a *script* **m-file**, and having the number of rows and columns set by the 2 inputs to the *function*. The way you might (but don't have to) go about is to:

1. Take your *script* **m-file** for the 3x3 grid plotting, and turn it into a *function* with 2 inputs.
2. Then ... worry about changing the inner and outer loop limits. Remember that the 1st input to the function sets the number of columns (x), and the 2nd input the number of rows (y).
3. Sort out the colors.
4. Done!



6. Week #6 – Computer programs

6.1 Work plan

Monday 9th November:

Section 5.1 (programs and games) – ’Tic-tac-toe’

Friday 13th Nov:

Finish Section 5.1 (and finish anything else you have not been able to finish to date).
Next week, will be something completely different ...

6.2 Learning goals

Topics and methodologies you should be familiar with:

- structuring more complex programs

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

- `ginput`
- `isempty`

6.3 Assessment #6

There are 2 parts to the 6th assessment exercise (with marks weighted 40:60).
Both require the use of **MATLAB GUIDE**, which will generate both an **m-file** AND a **.fig** file – both (**.m** and **.fig**) files must be uploaded to iLearn for each of the 2 parts of the assessment (so two **.m** and two **.fig** files (a grand total of four files for the entire exercise)).
The iLearn deadline is: 2 pm, Friday 20th November.

(1) Click-button app

Create an app with the following properties:

(i) 2 Push Buttons, labelled 'CLICK ME' and 'CLICK ME NEXT'.

(ii) That when you have clicked on BOTH buttons, the app ends (i.e. window closes).

(*) For bonus marks, ensure that the window only closes if the buttons are clicked in the order: 'CLICK ME' and 'CLICK ME NEXT' (and not the other way around).

HINT: Create a pair of variables to represent the 'clicked' state of each button. Initialize the value of each variable to *false*. After a click, test whether both are *true*.
Refer to: *6.1.3 Updating object properties (do you like bananas?)*

Call the app: **twoclick**. Submit both **twoclick.m** and **twoclick.fig** files to iLearn.

(2) Click-counter app

Create an app with the following properties:

(i) A single Push Button, labelled 'CLICK ME'.

(ii) A Static Text Box, initially labelled '0' (zero).

(iii) A second Static Text Box, immediately above the first, labelled 'COUNT'.
(This is just a label for the count in the 1st box, and plays no further role in the program.)

(iv) A program (function) action, that when you click on the push button with the mouse, the number displayed in first Static Text Box increases by 1.

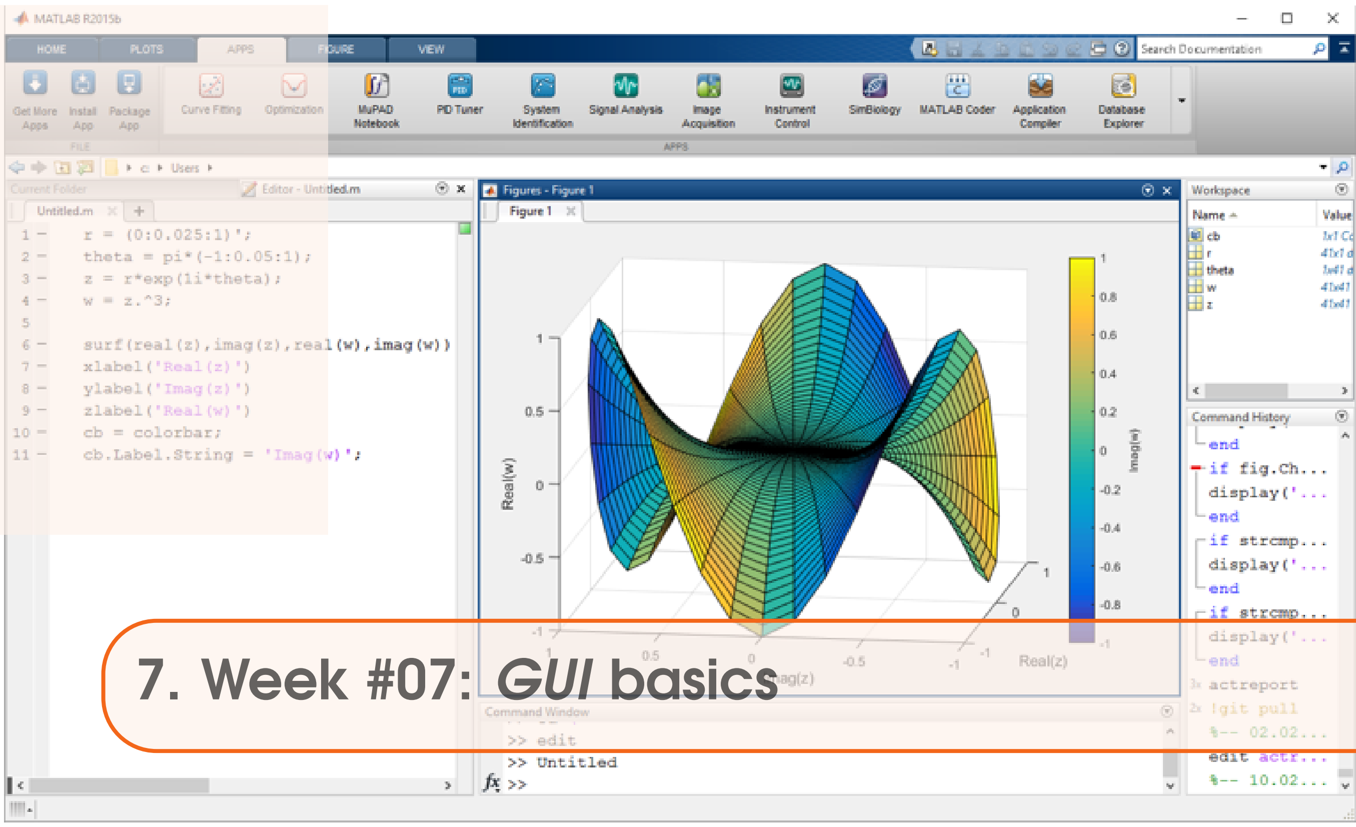
(v) A program (function) action, that when the value displayed in the Static Text Box reaches 10 (i.e. after 10 clicks of the button), the program exits

HINT: You need to keep count of the number of times the button has been clicked. You will need a variable that is defined as global, and this line of code will need to be in 2 places:

1. The start of the OpeningFcn function.
2. The start of the pushbutton Callback function.

(You will also need to initialize the variable value to zero in the OpeningFcn, and then increment its value somewhere in the pushbutton Callback function.)
Don't forget `num2str`!
(Alternatively, you could use the text in the text box as the way of keeping track of the current count – you can find the numerical value of the text in the textbox by str2num, add one to this value, and then convert back to a text format using num2str and then update the textbox display.)

Call the app: **tenclick**. Submit both **tenclick.m** and **tenclick.fig** files to iLearn.



```
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
>>
```

7. Week #07: *GUI* basics

7.1 Work plan

Monday: Work through Section 6.1 of the GEO111 course text. A brief guide as to what you will be doing/seeing:

1. **Section 6.1. MATLAB GUI basics**
   How to design a simple *GUI* in **MATLAB**. Basic interfacing (linking) of the *GUI* with a program.

Friday: TBA

7.2 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`

7.3 Assessment #7

There is a single part to the 7th assessment exercise. Hand in, to ilearn, all the **.m** files needed to run the program. (It is up to you what you name the files, but the exercise recommends specific filenames and you may as well follow that same naming convention.)

The iLearn deadline is: 2 pm, Friday 4th December.
(So you have 2 weeks to complete this.)

NOTE: This is the final micro-assessment of the course. yay!

Parameter sensitivity experiments using the EBM

Follow the instructions in **Section 2.1.6** of **MODELS.pdf** and adapt as per the additional instructions below. (Read through Section 2.1.5 first.)

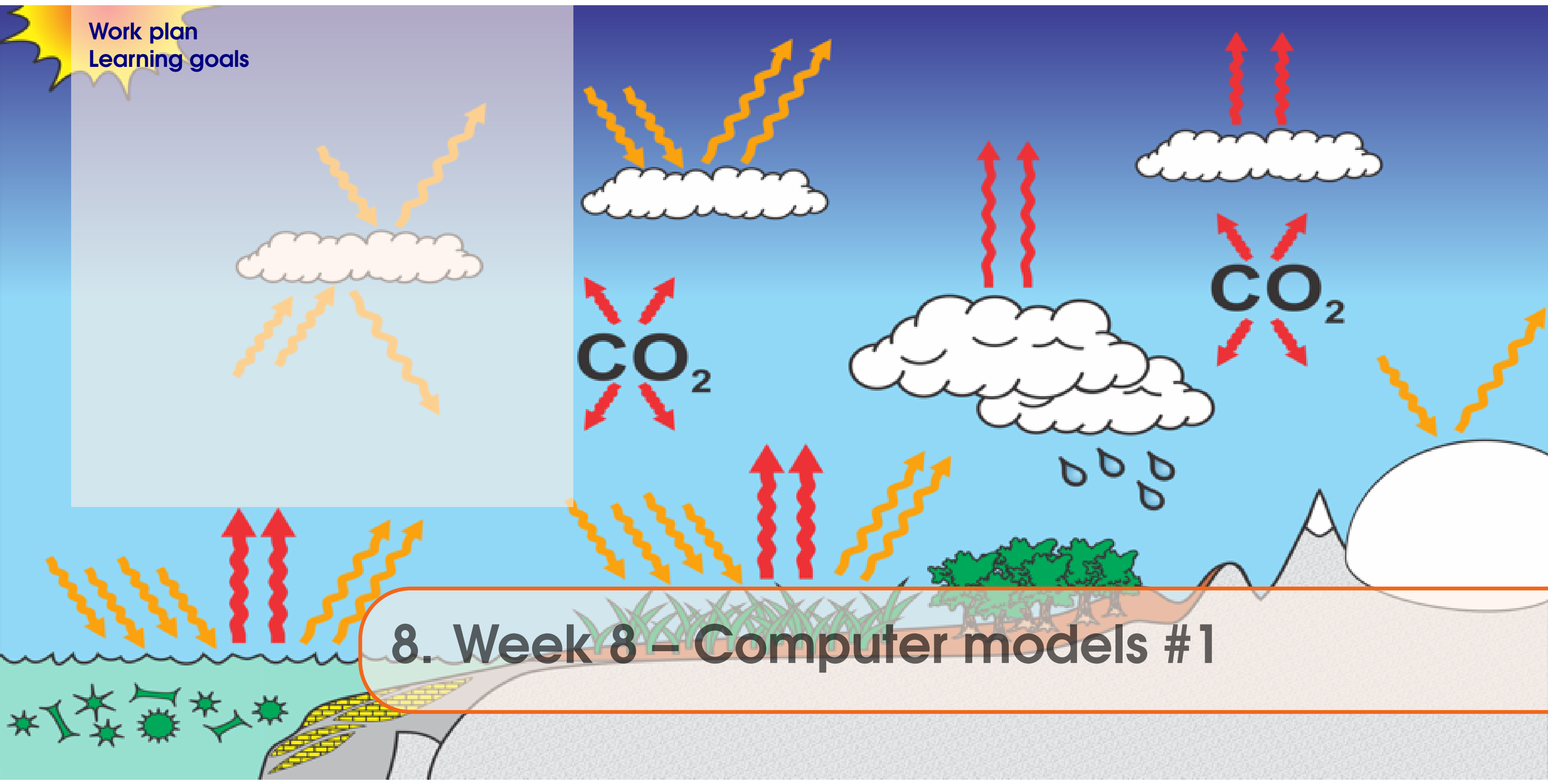
Your program should produce a 2D array of surface values (in degrees C), with:

- A range of solar constant values running from 1000 to 1500 Wm^{-2} in steps of $50\,Wm^{-2}$.
- A range of albedo values running from 0.1 to 0.5 in steps of 0.05.

Your program should also produce a 2D filled color-contoured plot similar to Figure 2.11 (in **MODELS.pdf**) (but with different axes limits), and include:

- All axis labels and a title.
- Contour labels.
- A color-bar.

(Bonus marks if the color bar is also labelled.)



8. Week 8 – Computer models #1

8.1 Work plan

For week 8:

Monday 23rd November

We will be working from the e-book: **MODELS.pdf** (available on iLearn).

Work through Section 2.1 – there are 4 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 ... (Also read the 'climate primer' PDF).

2.1.1 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 ...)

2.1.2 Turn this into a function and test it.

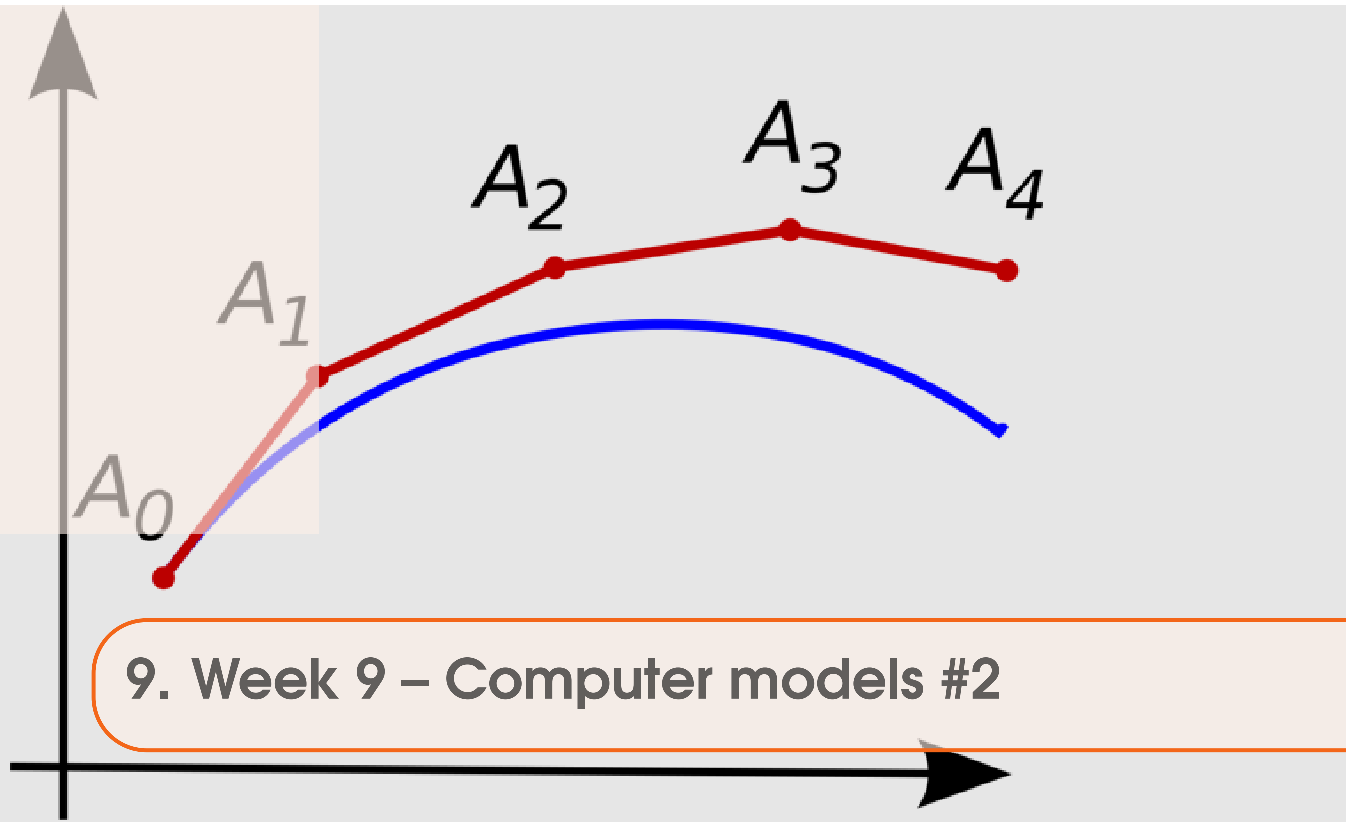
2.1.3 Create a function to calculate the value of the solar constant, given a value for 'time' (since the Suns formation).

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

Friday (Thanksgiving)...

8.2 Learning goals

- Appreciate how simple physical models can be encoded in ... code (and **MATLAB**).
- Remind yourself how to re-write equations.
- More practice with loops, and creating arrays of data as a loop progresses.



9. Week 9 – Computer models #2

9.1 Work plan

For week 9:

(a) **Monday November 30th.**

We will be working from the e-book: **MODELS.pdf** (available on iLearn).

Work through Section 3.1 – there are 4 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 ...

Part I – Creating a model of the trajectory of a thrown ball (or any object), considering only horizontal travel.

Part II – Creating a model of the trajectory of a thrown ball (or any object), considering only vertical travel and experiencing the force of gravity.

Part III – Considering both horizonal and vertical travel and testing for when the ball hits the ground.

Part IV – Extending the graphics capability of the ballistics model.

(b) **Friday December 4th.**

Finals overview.
Course re-cap / Q&A / revision.
And ... complete any unfinished work from Monday.

And don't forget that both the group project and final (7th) micro-assessment are due in by 2 pm ...

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.

10. Week 10 – Putting it all together

10.1 Work plan

(a) **Monday December 7th.**

For week 10, we will be working from the e-book: **MODELS.pdf** (available on iLearn).

i. First, ensure that you have completed the ballistics program from last week.

ii. Second – work through Part 0 of Section 5.1
(Adapting the ball/trajectory model to incorporate a picture (sprite) rather than a plot/scatter point.)

Then, you have 2 options:

I. Creating a GUI game based on the ball/trajectory model, in five steps:
   A. Create the GUI.
   B. Set up the graphics.
   C. Add in the ball/trajectory model.
   D. Activate the Sliders.
   E. Determine when the ball 'hit' the Pokemon.
   F. (Further refinements to the App.)

   This is a good revision-friendly step-by-step guide through building up the working program.
   (A complete example code is provided for guidance on iLearn.)

OR

II. Take the complete working code and tackle one of more of the 'Part VI – final game refinements' tasks (page 98).

(A complete example code is provided for working with on iLearn.)

(b) **Friday December 11th. – Finals.**