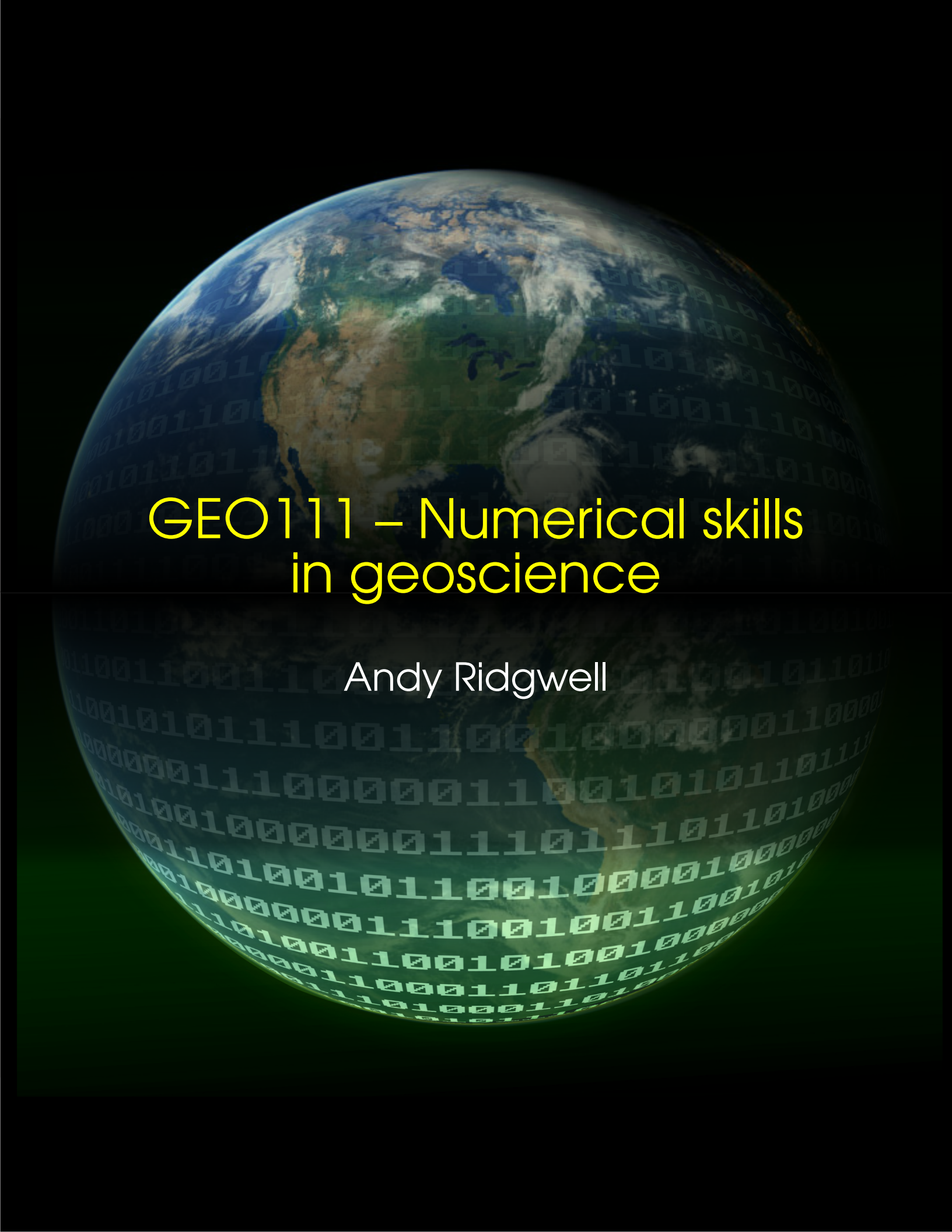


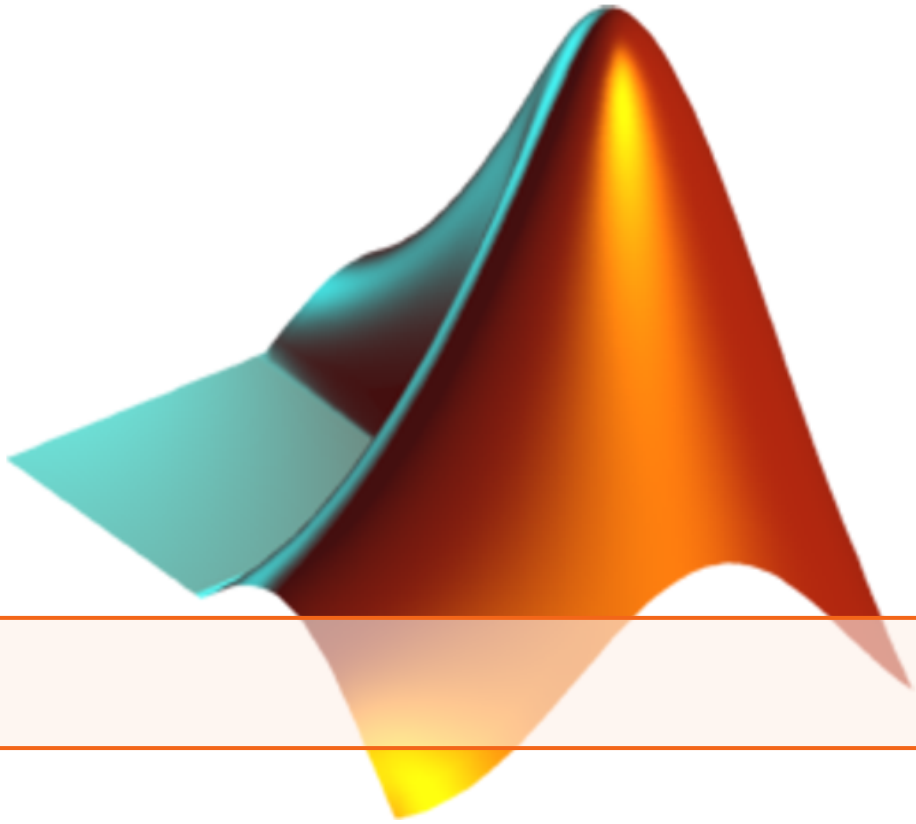
GEO111 – Numerical skills in geoscience

Andy Ridgwell

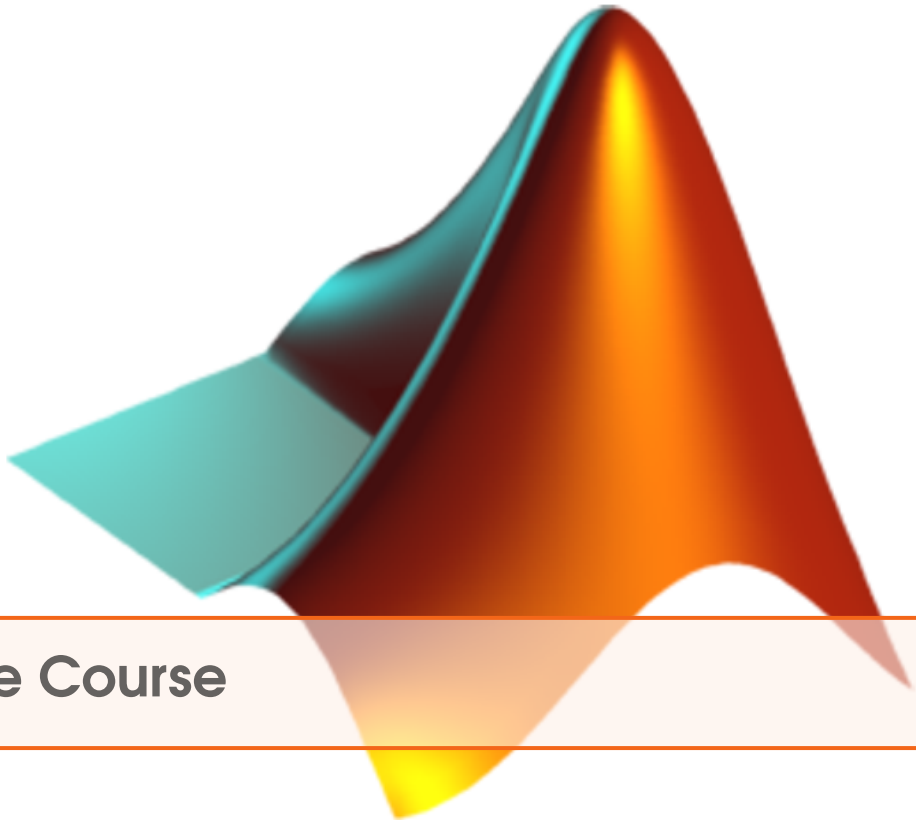
PUBLISHED BY DERPY-MUFFINS INC.

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

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.

0.1 Course Overview

0.1.1 Format

The weekly format of GEO111 is: 1 × 1-hour lecture together with 1 × 3-hour computer practical session (both in Watkins 2111), plus 1 × 2-hour interactive lecture/discussion session of worked problems and examples (also in Watkins 2111). 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 start of each Friday class (2 pm) and be due in the following Friday (at 2 pm). (Longer will be given for project work.). 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 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:

- 6 × weekly micro assessments @ 5% each = 30% total
- Midterm paper – 20%
- Project work – 10%
- Finals paper – 40%

As per the time-table – for the first 6 weeks, a short assessment will be set on Friday at 2 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. The idea is to help reinforce what you should have been 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 Mid-term paper will be a written exam, consisting of a mixture of multiple choice and short-answer format questions. Its purpose is to test basic knowledge of computers and programming, plus general concepts and basic commands you have come across in MATLAB. The testable content

will comprise the material covered in the lectures and lab sessions (in weeks #1-5). The exam will be 2 hours long. No study aids of any sort will be allowed. The mid-term paper will constitute 20% of the total assessment for the course.

The programming project will be a more free-form version of the weekly assessments and you will have 2 weeks in order to complete it. A written plan (no more than 1 page) for the programming project must be agreed with the Instructor during week 8 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 on-line documentation are allowed (but no internet!). This will constitute the remaining 40% of the total assessment of the course. A maximum of 3 hours will be allowed.

0.1.4 Office Hours

Office Hours are Tuesdays ... any time from 8 am to 12 noon, or at other times by prior arrangement. You can also email[1] questions. 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 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.

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]

0.1.5 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 uploaded to iLearn.

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[4].

[1] andy@seao2.org

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

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?`'.

0.1.7 Computers and Software

Watkins 2111 is equipped 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 Course assessment

0.2.1 Weekly Homework

For the first full 6 weeks of the Quarter, a short assessment will be set on Friday at 2 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.

0.2.2 Programming project

During week 8, you will embark on an individual programming project. The hand-in date will be Friday 6th December (the end of Week 10) at 2pm (PST). This project will be worth 10% of the total marks available for the course. Project hand-in will be via iLearn.

The scope and content of the project must be agreed in advance by the Instructor and must be written down (as a 1-page document). Details of what is expected in this project will be made available prior to week 8.

0.2.3 Mid-term

Format of exam

What might be examined on?

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

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

- Detailed usage of MATLAB commands and the exact syntax of commands.

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

Logistics

The Midterm is nominally scheduled for the end of week 5 – from 2 through 4 pm on Friday 1st November, 2019. 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.

0.2.4 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 6th December, 2019. However, you are allowed up until the end of the room booking – 5 pm, if you wish.

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

Study aids and reference materials are allowed, including:

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

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

The Finals will account for 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.

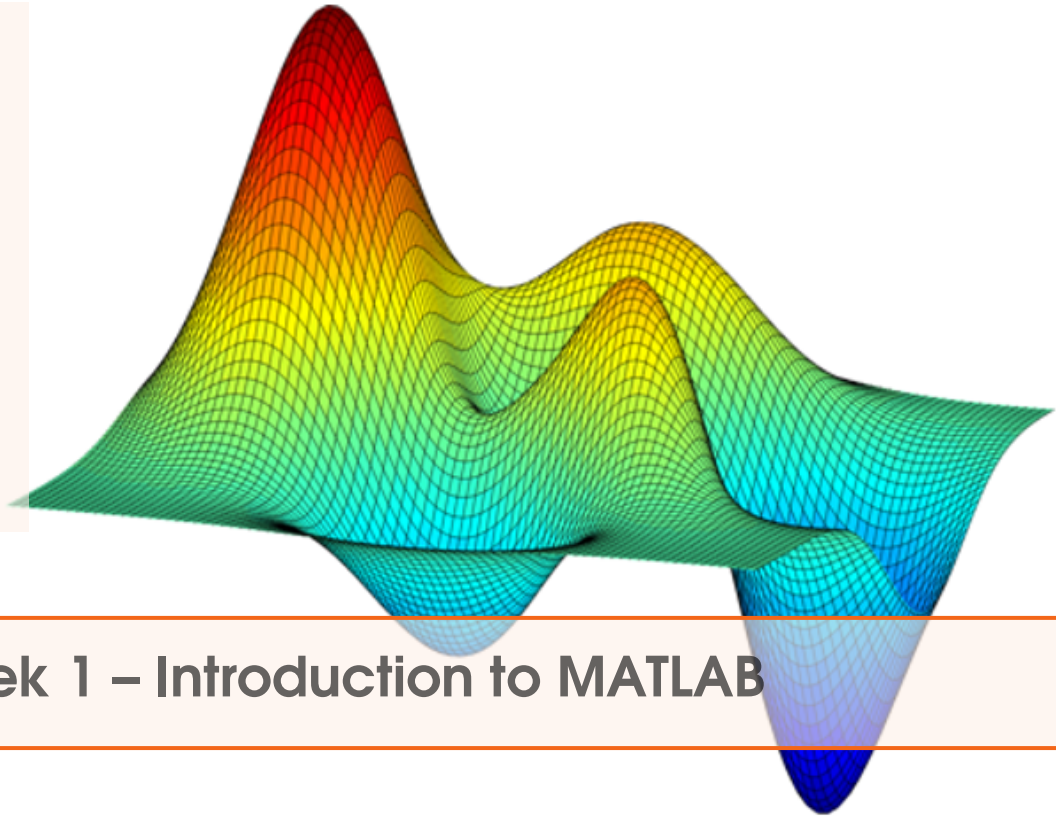
From UCR (with love)

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 (09/30)	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' and the internet. + MONDAY CONTINUED
02 (10/07)	Fundamentals of computer programming I.	Scripts and functions in MATLAB. Loops and conditionals.	**2.** Program structure practice and exercises.	Other computer languages, html. + MONDAY CONTINUED
03 (10/14)	Fundamentals of computer programming II.	Further MATLAB and data visualization.	**3.** Data anaylsis and plotting practice and exercises.	MONDAY CONTINUED
04 (10/21)	Fundamentals of computer programming III.	Algorithms, problem-solving, and further programming tricks and techniques	**NONE** (revision)	MONDAY CONTINUED
05 (10/28)	Statistics and encoding math.	Further data analysis and statistics in MATLAB.	**4.** Statisical analysis practice and exercises.	**Mid-term**
06 (11/04)	Numerical modelling and encoding physics(!).	Basic (zero-D) numerical modelling.	**5.** Graphics fun(!).	MONDAY CONTINUED
07 (11/11)	UCR HOLIDAY – Veterans Day	UCR HOLIDAY	**6.** Program debugging practice.	How to survive program complexity.
08 (11/18)	Computer Graphical User Interfaces(GUIs).	Basic GUI creation in MATLAB.	**Programming project**	More advanced graphics usage and animations.
09 (11/25)		Dynamic (time-stepping) modelling – ballistics 101.	**Programming project**	UCR HOLIDAY – Thanksgiving
10 (12/02)		Putting it all together – a GUI- and physics-based game(!)	**NONE**	**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 (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.

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

1.4 Assessment #1

The assessment for week #1 is based partly on Chapter 1 of the course text, and partly on the information in the **MATLAB®7 – Getting Started Guide** that you read earlier.

The required work will be in the form of a series of .jpg (or jpeg) graphics files (not any other format) that you will upload to Blackboard. The hand-in deadline is 2 pm Friday 11th October. A late hand-in penalty will be applied.

There are 5 graphics to create and hand in, each weighted 20%. You will be penalized for not providing the graphics in .jpg file format, and for missing or inappropriate axis labels and graph title.

1. Based on the work in Section 1.7.3 of the course text, create a single panel plot containing both the global mean and local Riverside temperature anomalies in units of ^oF . The graph should be adequately labelled. Your choice of colors, symbols, fonts/font sizes etc. Aim for the clearest possible presentation. (The instructions for processing the data etc. are exactly as per Section 1.7.3.)
   Name the file: image1.jpg
2. Following Section 1.8.4 in the course text (itself following from 1.7.1), plot, from 0 to 300 Ma (millions of years ago), the mean proxy CO_2 value. Scatter plot (`scatter`) the points (i.e. no lines), and make the symbols squares. For bonus marks, color-code the points with their age. For further bonus points, add error bars (see page 54, and the function `errorbar`).
   Name the file: image2.jpg
3. Load the deglacial CO_2 data (from my website) – file deglacial_co2.txt. Plot plot ice age (y-axis) vs. depth (x-axis) in the ice core as a red dashed line. View the file for column information. Referring to the portion of chapter 3 in the **MATLAB®7 – Getting Started Guide** ... make the plot, 'square'.
   Name the file: image3.jpg
4. Also using the deglacial CO_2 data (from my website) – plot- CO_2 (y-axis) vs. age (x-axis). Plot the data as filled green stars (no lines). Referring to the portion of chapter 3 in the **MATLAB®7 – Getting Started Guide**, add grid lines to the plot.
   Name the file: image4.jpg
5. As a 5th separate figure, taking the plot from (4) and adding on top of this – a dotted black line of the first 102 data points (only) of the deglacial CO_2 data (vs. age).
   Name the file: image5.jpg

Remember: Office Hours on Tuesday if you are stuck (all the answers are in Chapter 1 of the course text and the **MATLAB®7 – Getting Started Guide** and some of the required work is stuff that you should have already completed in class).

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

numPts = size(position,2);

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

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

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

    % Call the C function
```

2. Week #02: Computer programming

2.1 Work plan

Work through Chapter #2 of the GEO111 course text – aim to complete Sections 2.1 through 2.3 during the Monday am lab, and just Section 2.4 for 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 feel free to work on the week #1 assignment.

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 ...

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
- loops

Specific **MATLAB** commands/functions 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 ...`
- misc functions: `disp`, `input`, `strcmp`

2.3 Assessment #2

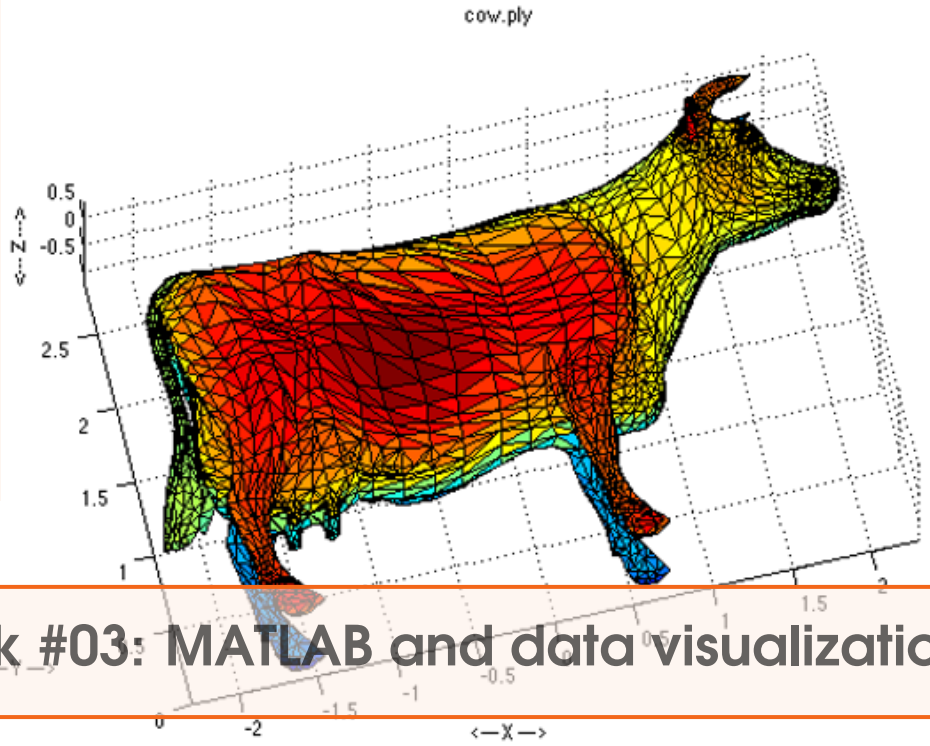
1. Write a *function* that does the following:
   Takes 1 input (integer).
   Returns a single output (integer).
   The output (returned by the function) should be an integer equal to the sum of the squares of all integers up to and including the input number.
   For example, if the function input was the integer `n`, the output would be equal to:

   ```
   sum([1:n].^2)
   ```

   In this notation, `[1:n]` creates a vector of all integers from `1` to `n`; `[1:n].^2` then squares each and every element in the vector, and `sum([1:n].^2)` sums the squares ...
   However, instead, use a loop in your function and a calculate a partial sum within the loop.
   Hand in the **m-file**.

2. Provide an **Excel** format file that contains data that you have generated yourself, or been given in another class, or even, found on the internet. Write a *script* **m-file** that loads and plots the data (and correctly labels it etc.). (The data should not be anything given in GEO111.)
   Hand in the both the **Excel** and **m-file**.

For the `.m` files, you will be marked for basic program structure, appropriate indenting, 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.



3. Week #03: MATLAB and data visualization

Note that the manual (and new instructions in Chapter 0 – 'How to use this textbook') has been updated with a better clarification as to what (code) you are intended to type and when to create new m-files, versus simply reading (and digesting).
Lines of code that goes with the discussion in the text and which is not necessarily intended for you to type in (although you may still want to, simply to try it out), is given in a light Courier font.

Lines of code that are intended for you to type in – either at the command line or place in an **m-file** are given in a **bold Courier font**. Additionally, code to type in, where possible/appropriate, will include the same context-colors as **MATLAB**.

Instructions where you should do or try something out, rather than read and digest, where possible are given in **bold**. (Note that you might want to try out other (light font) code to get a complete picture of the art of programming.)

This new text guidance/clarification will be progressively added to the text as the course progresses.

3.1 Work plan

First ... complete the final loop-related sections of Chapter 2:

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

Then ... work through Section 3.1 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*.

Then on Friday:

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.

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 Assessment #3

The 3rd assessment requires the development of a script **m-file**, that produces a plot. You will need to upload BOTH the **m-file** AND the jpeg format plot that results from your **m-file**, to iLearn. (I will not necessarily run your **m-file**, but will mark it for structure (e.g. indenting) and commenting.)

Because there is no assessment set in week 4, and because this (week 3) assessment is being set later than usual, you have until Friday November 1st at 2 pm PST (the start of the MidTerm) to submit to iLearn. Note that this gives you the opportunity of attending Office Hours on Tuesday 29th October, if you have any problems/questions.

OK, so 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, instead I have made the file available in the week 03 data section of my teaching webpage – 'USGS Ridgecrest earthquake **data**'.

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 in the week 03 data section of my teaching webpage – 'USGS Ridgecrest earthquake **m-file**'.

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 to loading the data and extracting time and earth quake magnitude.

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 line at the very end 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 the 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.)

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 (and no joining lines). The circle fill should be in an orange color (see Section 3.4.2 of the course text) for magnitudes less than 2.5, and red for magnitudes equal to and greater than 2.5.
  How to do this? You can split the complete (all magnitudes) dataset into 2 sets of vectors using `find`. Use the `scatter` function twice (with `hold on`).
  Bonus marks if you can plot the edges of the circles in white (as per in the WikipediA figure).
- Replace the x-axis labels with: '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'.
- Label 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.)
- Calculate the cumulative number of aftershocks >= M2.5 as a function of time, and plot as a black line (on top of the data points). This should look like the WikipediA figure (except that your earthquake data extends down to magnitude 1.0).
  How to do this? Refer back to when you were calculating a partial sum, except now you need to store the value of the partial sum at every time point (rather than only need the final sum at the end). i.e. you need to create a vector (e.g. in a loop), the same length as the earthquake vector, but which contains integer values equal to the number of >= M2.5 events up to and including the event at that time. For example, if your list of events looked like this:

  $(7.1 \quad 6.3 \quad 2.3 \quad 4.3 \quad 3.0 \quad 1.2 \quad 1.1 \quad 4.0 \quad 3.9 \quad \ldots)$

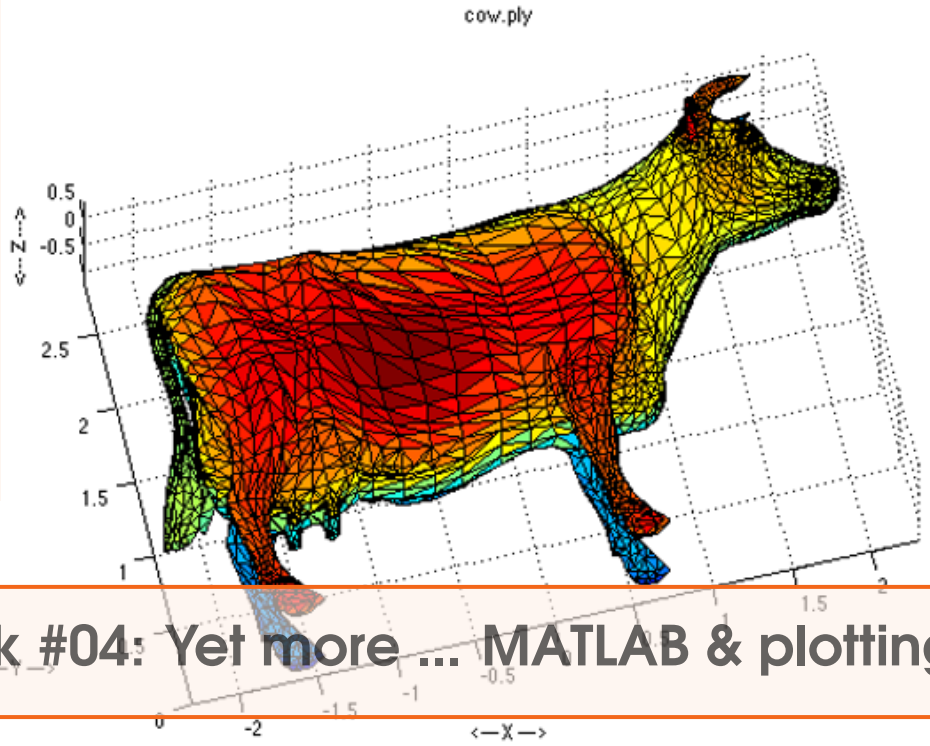
  then your cumulative number of >= M2.5 events will look like:

  $(1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 4 \quad 4 \quad 5 \quad 6 \quad \ldots)$

  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 >=M2.5 events.
- A legend.
- Grid lines.
- A horizonal black dashed line at a value of M2.5.



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

4.1 Work plan

You should by now be familiar with all 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 office hours to come see me.

For Monday:

1. **Section 3.3. – Further data processing**
   Additional data processing techniques, including interpolation.
2. **Section 3.4. – Even nicer graphing and graphics**
   Drawing lines and shapes. Placing text in figures.

For Friday:.

We'll use the time to recap on the course to far, briefly discuss the format of the Mid Term the following week, and go thorough some code examples. Also come prepared with questions and needed clarifications (and we can do some 'live' some **MATLAB**.)

4.2 Learning goals

Topics and methodologies you should be familiar with:

- further data filtering (e.g. using `find`)
- basic stats
- data interpolation

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

- `find` ... which takes a little getting used to ...
- various basic stats functions
- `interp1`

5. Week #5 – Coding and algorithms

5.1 Work plan

Monday 28th Oct:

1. Section 4.1 – introducing the concept of 'nested loops'
2. Section 4.2.1 – worked through examples of algorithms (/problem-solving programs)
3. Section 4.2.2 – writing your own sort algorithm (if you have time ...)

Friday 1st Nov:

Mid-term

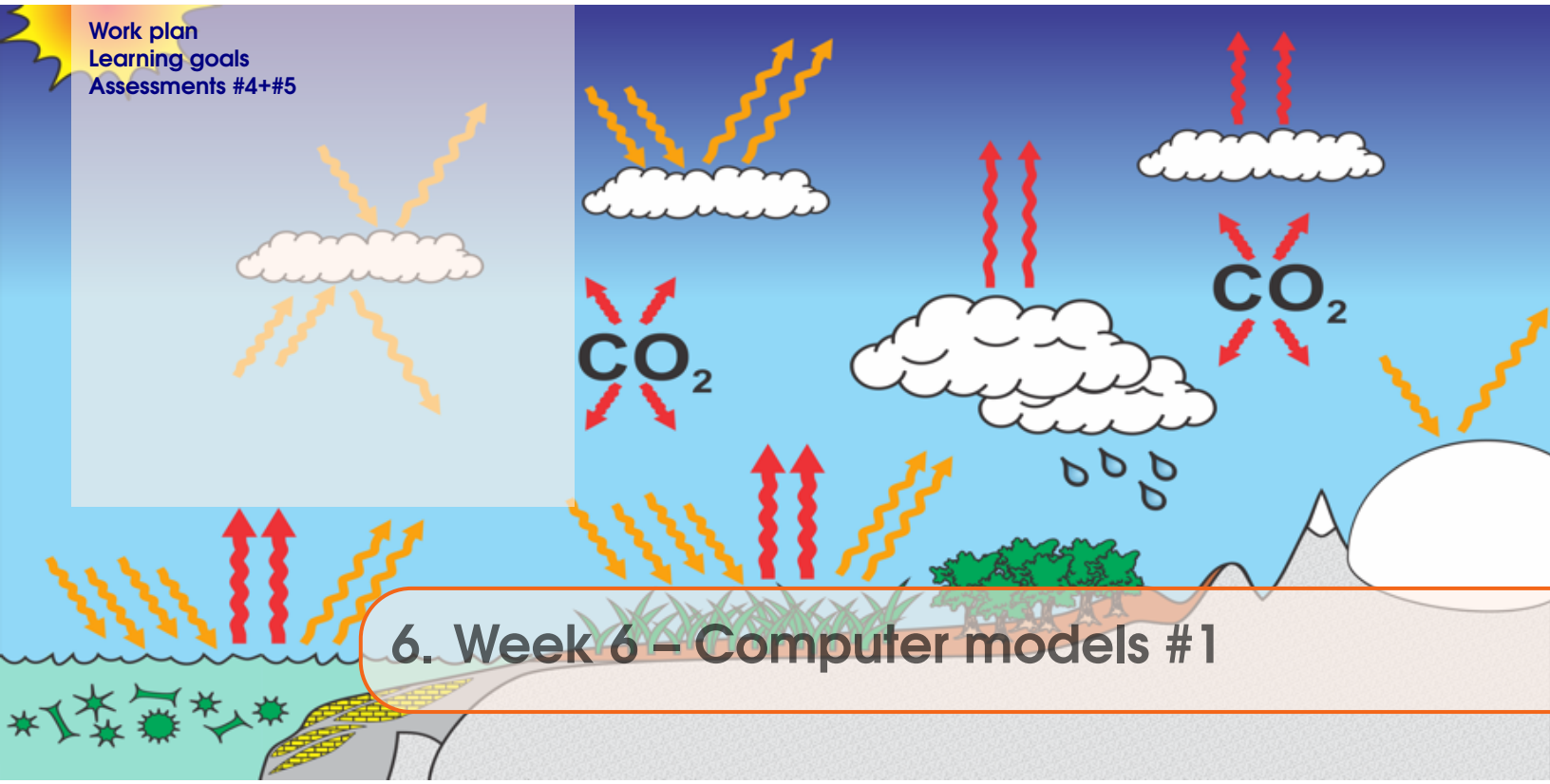
5.2 Learning goals

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`



6. Week 6 – Computer models #1

6.1 Work plan

For week 6 – Monday November 4th + Friday 8th Nov:

Monday ...

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.

Friday ...

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

6.1.6 Explore and plot, how sensitive Earths surface temperature is to 2 simultaneously changing parameters.

6.2 Learning goals

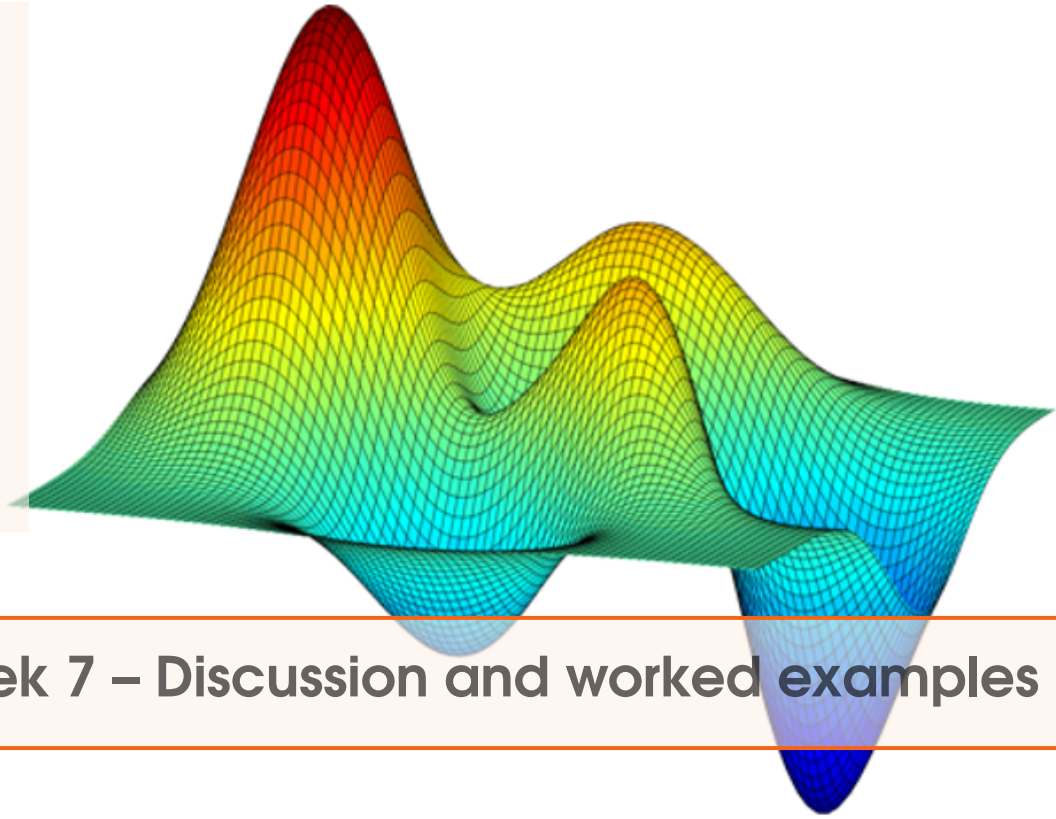
- Appreciate how simple physical models can be encoded in ... code (and **MATLAB**).
- Remind yourself how to re-write equations.
- More practice with loops, and creating arrays of data as a loop progresses.

6.3 Assessments #4+#5

This is a double-assessment (and worth 10% of the course total rather than 5%), spanning 2 weeks AND including the Monday 11th class time (that does not formally occur due to Veterans Day and a UCR holiday). The hand-in date/time will be: **Monday 18th Nov at 8 am**.

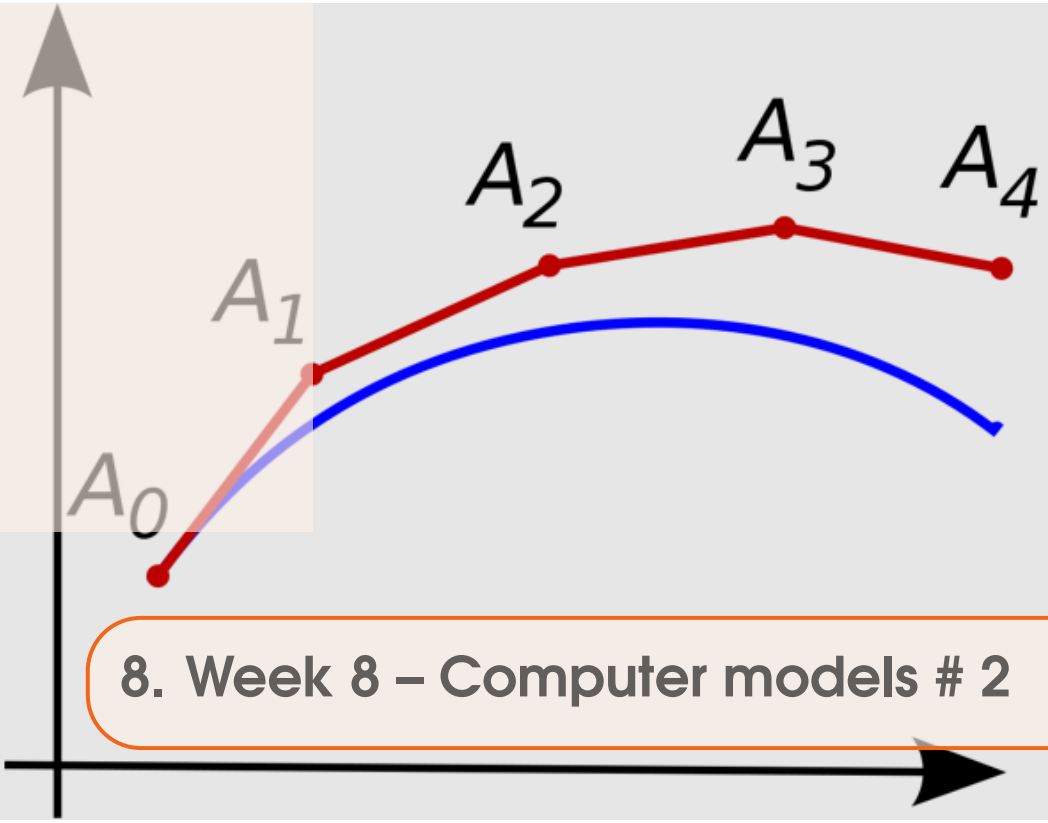
Your task is to work through and complete the game outlined in Section 5.1. Upload to iLearn all **m-files** required to run the program. Sufficient commenting, explaining at every (main) step what your program is doing is required in order to achieve full marks – perhaps as much as 20% of total marks will be reserved for commenting and code formatting (e.g. indenting).

Note that you should use different symbols (than the examples given in the text). Ideally ... try and either draw naughts and crosses, or you might adapt the code to place text instead of drawing shapes ... remembering the `text` command and that you can use larger font sizes.



7. Week 7 – Discussion and worked examples

(Discussion and worked examples)



8. Week 8 – Computer models # 2

8.1 Work plan

For week 8:

1. **Monday November 18th.**

   Work through Section 7.1. 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 22nd.**

   Part 0 of Section 10.1.

8.2 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.

8.3 Assessment #6

Your assessment for week 8 is to complete a game – similar to your tic-tac-toe game, but played a little more like GO, and on a user-defined grid (i.e. you select how large the board is).

You are given the framework (and most) of the code – the **m-file**: gogame.m – can be found on the course webpage, in the green GEO111 (2019/20) box. (You'll also need to download the **m-file**: find_neighs.m.) For the assessment, you need to complete the code and hence create a working game.

The idea of the game is:

- The game is played on an $n \times n$ grid of squares, where n is an even number.
  The code to check for an appropriate value of n – variable name `nmax` – is given to you (checking that n is even and also not too small).
- The game is run from the command line, calling the function `gogame` and passing the size of the game grid, e.g.

  ```
  >> gogame(8);
  ```

- As per in tic-tac-toe – the players alternate in turn, and can place a counter in any empty (only) grid square.
  The code for this is given to you. Also, the code for how the chosen square is marked in the player's color is given.
- If ... the chosen square is orthogonally surrounded ... i.e. in N, S, E, W directions, by 2 or more squares of the opposing player's color, then those squares are flipped to the current player's color (see game images).
  The code for searching surrounding cells is given to you – the function `find_neighs` (**m-file** find_neighs.m). As is the code for flipping the color of these cells (given to you)
- The winner is the player, when the grid is complete (i.e. all game grid cells are one or the other player color), who owns the most cells. (i.e. there is no pattern-matching, just who has the most colored cells at the end)

The code you need to add in gogame.m is highlighted. Read through all the instructions first, before starting. Your 3 assessment tasks are to:

1. Firstly ... you need to write a function to draw the game grid lines (the background is already colored-in for you). The code (in several places) calls the line:

   ```
   >> redraw_grid(nmax);
   ```

   You need to create and write a *function*, called `redraw_grid` (file redraw_grid.m), that draws the grid lines (see game grid images). The function takes a single input – `nmax`.
   The outside lines – the game grid border – should be simple to draw. The outer edge of the game board should be in a thick solid white line.
   The interior lines of the game grid should also be white, but thinner.
   REMEMBER – the grid is not fixed in size but goes from 0 to *nmax* in both x and y directions.
   The interior lines ... you need to draw vertical lines at x values of $1, 2, \ldots (nmax - 1)$, and horizontal lines at $1, 2, \ldots (nmax - 1)$ (note that lines at values of 0 and at *nmax* you have already drawn because these are the game grid boarders). You should see that you need a loop for drawing the x-axis lines, and also a second loop for drawing the y-axis lines. (One loop following the other – NOT a nested loop.)

Once you have written the grid line plotting function, you can test the complete code by adding the following temporary lines of code:

```
player1_col='b';
player2_col='r';
```

where it says `% set default player colors`. A this point you can get the game to play, but it would still be missing player color choice and a way of identifying the winner.

2. Secondly, in the game model code as given to you, there are 2 currently undefined parameters (variables):
   - `player1_col` – the color of the squares or player 1
   - `player2_col` – the color of the squares or player 2

   and the code will not run until they have been assigned a value.
   These variables are single character strings, and must be one of the short **MATLAB** plotting color codes, e.g. 'r', 'b', 'k', ... (I'll leave you to compile the full list of possibilities from e.g. **MATLAB help**.)
   What you need to do is to create and add code, similar to earlier in the course, such that **MATLAB** asks at the command line, first for the color of player 1, and then for the color of player 2.
   You'll need 2 sections of code – one for each player. Each section will require some sort of loop (`while`?), because you need to keep asking the question until a valid color is picked. (Refer to earlier in the course for a very similar piece of code you wrote.) Unless you can think of something better, you will also need to create a `switch` and a list of the possible color code strings, to test whether the inputted answer is one of the possible color codes. This, if `player_answer` was the variable you assigned the result of `input('Please enter a color','s')` to, the structure will look something like:

```
switch player_answer
   case {'r','b', ...}
      player1_col = player_answer;
      go = false;
   ...
```

   (where go controls whether the `while` loop keeps looping (not shown)).
   Remember – for player 1, all the color codes are possible correct answers. For player 2 – you cannot chose the color of player 1, so you additionally need to test whether that color is already taken – I suggest that, in the code, under `case {'r','b' ... }`, you then test for `player1_col` being selected by player 2 when determining whether to assign the player 2 color or keep asking the question.

3. Lastly ... there is not yet any code for determining whether anyone has won.
   In the highlighted location in the code – after either player 1 or player 2 has taken a turn, you need to test for whether the grid is full – very similar to the code you had in the tic-tac-toe game – you are looking for (`find`) no remaining empty locations in the grid (i.e. values of `0` in the `tokens` array). If ... there are no longer any `0` value elements in the array and `find` returns an empty variable (HINT: `isempty` tests whether a variable is empty ...).
   Then, if the grid is full, you need to test whether player 1 has more counters than player 2 (more elements in the `tokens` array with value `1` than value `2`). Or if player 2 has more than player 1, player 2 wins. HINT – in the code:

```
length(find(tokens==1))
```

`find(tokens==1)` returns the linear indices of all elements in the `tokens` array that are equal to `1`. `length` then gives you the number of elements (i.e. the number of occurrences of a `1` in `tokens`).

The program should announce the winner. Or if a draw, announce that instead.

Finally finally, whether or not player 1 or player 2 wins, or whether it is a draw, you can end the game (and the game loop) in the code with:

```
break;
```

You need to fully complete 2 out of 3 pieces of code for maximum marks. Completing (well) all 3 pieces of code will give you bonus marks. Marks will be given for adequate commenting/indenting.

Deadline is the start of class on Monday 25th.

Office hours this week (week 8) are: Wed morning, Friday lunchtime (12 noon to 2 pm) and after class on Friday.

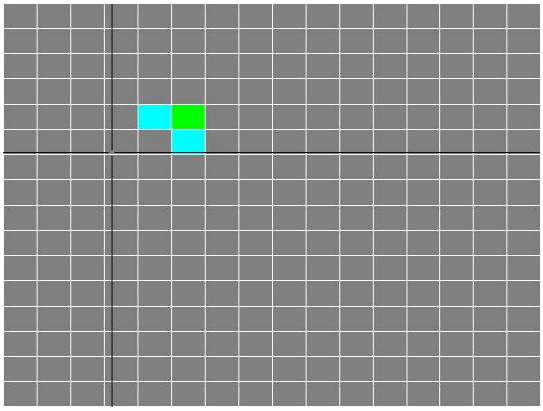


Figure 8.1: 16×16 game grid. Light blue moved first ... but played badly, and light green will place a counter in the bottom left had corner of that cluster of counters, turning both the 2 orthogonally adjacent light blue counters green ... like ...

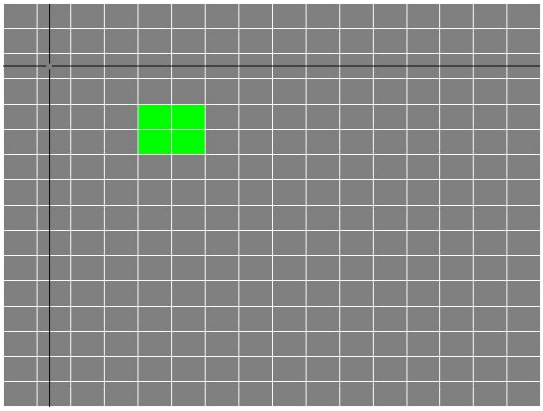
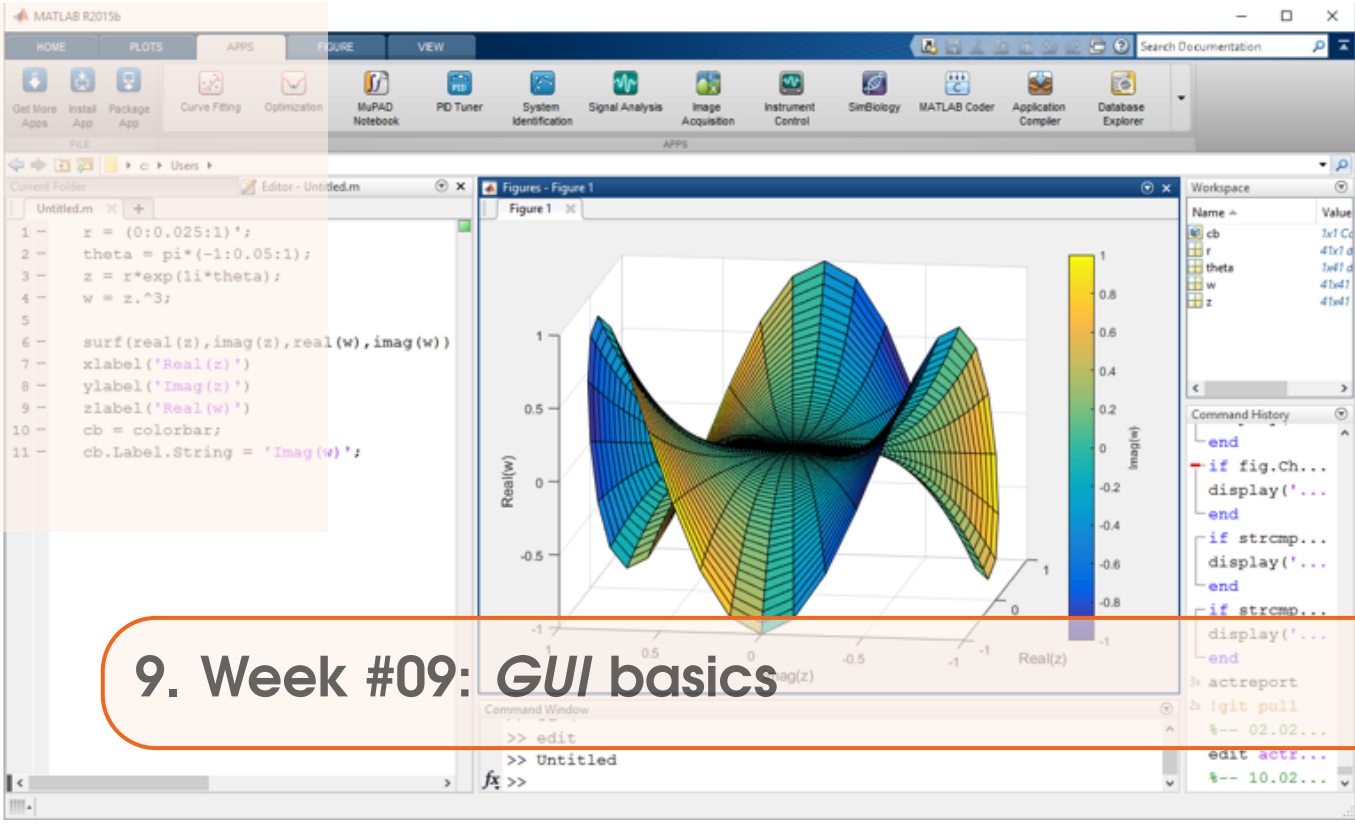


Figure 8.2: ... this. Go green!

Work plan
Learning goals



9. Week #09: *GUI* basics

9.1 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.

9.2 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`

10. Week 10 – Putting it all together

10.1 Work plan

For week 10:

1. **Monday December 2nd.**

   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.)

   An example code is 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 6th.**
   - Course re-cap.
   - Questions+discussion. Worked examples.
   - Finals exam format overview.

10.2 Programming project assessment

Instructions:

- Pick ONE of the programming project options listed below.
- Upload your completed profile files to iLearn (remember to upload **all files** – whether .m or image files – that are required to run the program).
- The hand-in deadline is the start of the Finals exam @ **2 pm, Friday 13th December**.
- Because you only have $1\frac{1}{2}$ weeks to accomplish this and exams also to prepare for ... marking will be on the 'generous' side (as long as you have clearly made an effort).
- Examples of the programs required, to make completely clear what is required, will be show in Friday class. I'll also go briefly through some of the code complications/details.
- Questions can be asked and help given, firstly as part of the Friday (6th December) class. I will also be available in my office for further questions/help between 4.30 and 5.30 on that day.
- I will be available (all days) the week of 9th-13th December for questions/help via email (i.e. not in person).
- Your code needs to be well commented, appropriately indented, and ideally, well laid out and readable, in order to be awarded full marks.

10.2.1 Option #1 – a GUI based tick-tac-toe game

The aim of this first programming project option is to create a simple tick-tac-toe game within a **MATLAB** GUI. The program requirements are as follows (all must be completed for maximum marks):

1. The game grid, rather than an axes object with drawn grid lines etc etc ... should instead be a 3×3 array of button objects, i.e. each button will correspond to one of the tick-tac-toe grid locations.
   Note that the use of button simplifies the mechanics of the game, because you no longer have to identify which grid location a button is clicked in – the GUI will return the corresponding button click function for you.
2. The buttons should initially be white and should be re-colored with a player color (your choice) when a valid click is made.

For bonus marks, additionally include one or more of the following features:

- A pair of text boxes and store the game count. i.e. when a player wins a game, the counter for that player is incremented by one. The game must also be re-set (including the tokens array, and all the colors of the push buttons). [**hard**]
- Add a nought or a cross (text) to the button to further indicate when a player owns it (and which player). [**easy**]

The rest of the game is pretty much be as per the class exercise – you need an array to store the grid of tokens (0 (== empty), 1 or 2 == player values).

You can initialize the tokens array, create the array of winning patterns, etc etc in the `game_OpeningFcn` function (assuming you called the program `game`).

Remember ... the tokens and winning pattern, player turn, etc etc variables are not shared between the `game_OpeningFcn` function and all the `pushbutton_Callback` functions unless you define them as `global` – this needs to occur both in the `game_OpeningFcn` function AND in ALL

the `pushbutton_Callback` functions.

One complication, unlike in the class version with the drawn grid, is identifying which button corresponds to which tokens array location, because the buttons might be labelled `pushbutton1`, `pushbutton2`, `pushbutton3` etc etc, while the array locations are of the form (1,1), (1,2), (1,3) etc. An idea then, when you create the button objects, might be to name them e.g. `pushbutton11`, `pushbutto12`, `pushbutton13` ... corresponding to array locations are of the form (1,1), (1,2), (1,3) etc and so easy to remember. Then, in e.g. `pushbutton13_Callback` you would know that you had to test whether array location (1,3) was occupied or not.

Testing for a win could be a little messy in the code, because when any of the `pushbutton_Callback` functions are called, after a successful player move, you need to test for a win. To do this simply and well, you could write a function to contain the win test code, and each `pushbutton_Callback` function would call this same function. Because this bit is far form being entirely straightforward, I am not fully expecting the game to include a fully working test of a win in return for full marks. (But maybe give it a try?)

10.2.2 Option #2 – a GUI based planetary temperature simulator

The aim of this second programming project option is to create a GUI to control and display your calculation of the equilibrium temperature of the Earth's surface. The program requirements are as follows (all must be completed for maximum marks):

1. Create an axes object to display a plot of time (in units of Ga since the formation of the Sun) vs. global mean surface temperature – exactly as per was plotted in the class exercises on the EBM. The plot should include a labelled vertical line corresponding to the current age of the Sun (i.e. today), and labelled horizontal line corresponding to the freezing point of water. (Your choice of whether the temperature units are in K or C.)
2. A pair of sliders – one for age (in Ga), and one for albedo.
   As per the exercises in class, each slider should by a text box acting as a label (e.g. 'Time (Ga)', 'Albedo'). Each should also be accompanied by a text box displaying the current value.
3. When either slider is clicked on and changed in value, the planetary temperature should be re-calculated using the values of time and albedo that need to be read in from the sliders (depending on which is adjusted). The calculate temperature value should be displayed on the plot (in the axes object) as a filled circle. The filled circle needs to be plotted at the appropriate (time, temperature) location in (x,y) space in the plot.
4. The filled circle should be red when the planetary temperature is above freezing, and blue when below (or equal) to freezing).
5. When a new temperature is calculated – before it is plotted (as an appropriately colored circle) – the old circle should be deleted (just as per how the scatter point in the animated ball trajectory exercise was deleted).

Complications include making sure that the values of the time and albedo parameters are shared across both `slider_Callback` functions, i.e. you need to declare these two variables as `global`, at the start of both `slider_Callback` functions. When you (`scatter`) plot a (age, temperature) point, you will need to save the handle to a variable (as per the ball trajectory model) ... but declare this as `global` in both slider functions so that you retain memory its value. You will also need to plot an initial value as the program starts up (in `ebm_OpeningFcn` assuming yo called the program `ebm`)

and save the handle of the scatter point in the same variable ... declared also in `ebm_OpeningFcn` as `global`.

For bonus marks: Add a text box to display the temperature (i.e. in addition to the temperature being plotted on the (age, temperature axes plot). The text should change in color as per the point in the plot – red if warmer than freezing, otherwise blue.

10.2.3 Option #3 – DaisyWorld!

Follow through the text, and create the DaisyWorld program outlined in Section 6.2 of the **MATLAB** text.

The model builds up, step-by-step. You'll receive full marks for creating the model described in **Section 6.2.2 – 'dumb daisy' daisy-world**.

You will be awarded bonus marks if you successfully complete the more complex version of the model described in **6.2.3 – 'clever daisy' daisy-world**.