**GEOL4200 Geosciences and Computers**
**Instructor: Po Chen (pchen@uwyo.edu) Phone: (307)766-3086 Office: ESB2036**

**Overview**
Geosciences, like many other branches of sciences and engineering, are undergoing a major transformation. Modern geoscientists spend more and more time in front of computers. With the rapid development of numerical methods and computing technology, almost every geoscientist has on his/her desk an advanced toolkit consisting of software libraries and multi-core processors that will make scientific discovery more optimal and cost effective. But to effectively harness the newly available computational resources, future geoscientists will need, first and foremost, a solid inter-disciplinary education in modern programming techniques and scientific computing. The purpose of this course is to provide students an integrated introduction to the basic components of modern scientific computing and to illustrate the basic concepts through geoscience applications.

**Attendance Policy**
Each student is expected to attend the lectures to fulfill the academic requirements. For participation in a University-sponsored activity or for unusual circumstances (personal hardship), an authorized absence may be issued to the student by the Director of Student Life or the Director's authorized representative. If a student produces the proof of absence, a makeup session can be arranged with the instructor. http://uwadmnweb.uwyo.edu/legal/Uniregs/ur713.htm

**Course requirements:**
This class is composed of two lectures per week. Students are expected to independently work out the class exercises, homework problems, lab projects, and exams. The instructor has developed a set of PowerPoint presentations as well as lecture notes for this class and will periodically post them in the course website via *Wyoweb*. The lecture notes however do not contain formula proofs, equation derivations and solutions to class exercises, so class attendance and participation is key to learning the materials.

**Grading Policy:**
In this course, emphasis is placed on the homework problems and final exam/project due to the time-consuming nature of these assignments. The final grades will be given based on your homework and term project (or exams). The appropriate percentage is shown:

| | |
|---|---|
| Homework | 50% |
| Mid-term Exam | 25% |
| Final Exam | 25% |

Note that each homework exam has a standalone grade of 100 points. When determining the final grade, these will be normalized reflecting the percentage distribution above. The final letter grade is given based on the numerical grade:

| S | U |
|---|---|
| >60 | < 60 |

| A | B | C | D | F |
|---|---|---|---|---|
| 90-100 | 80-89 | 70-79 | 60-69 | <60 |

**Textbook:**
**An Engineer's Guide to MATLAB (3rd Edition)** by Edward B. Magrab et al. (e.g., at Amazon.com)

**Tools:**
Some exercise and homework problems can be solved by hand or using Excel spreadsheet. Matlab will be used extensively through out the course.

**Questions & Answers**
Questions for the instructor: (1) during lecture; (2) office hour.

**Policy on Late homework, make-up exams, grade of incomplete**
Policy for this class:
• Unless otherwise stated, each homework is expected to be handed in to the instructor in the beginning of the class one week after the homework is assigned; If not handed in on time, each day it's delayed, 10 points will be taken out of the total grade (100) of that homework until no points remain.

• Unless otherwise stated, each homework is expected to be completed and handed in the beginning of the next lecture.

If a student can provide valid proofs of absence, the above rules do not apply. Within a reasonable time (1 week), the student is expected to hand in the late homework to the instructor or arrange with the instructor on a make-up exam. It is the student's responsibility to contact the instructor to make arrangement in a timely manner and in advance if at all possible, failing to do so will result in the forfeiture of the relevant points.

*Grade of incomplete*: During the semester, if a student has suffered severe problems (e.g., physical or mental incapacitation) and cannot complete the course as a result, he/she may be issued an "I" (incomplete) grade. Best to be avoided to reduce the frustrations and confusions for both the student and the instructor. The UW regulation on this is long and complex: http://uwadmnweb.uwyo.edu/legal/Uniregs/ur720.htm

**Academic dishonesty**
As defined by UW, academic dishonesty is:
*An act attempted or performed which misrepresents one's involvement in an academic task in any way, or permits another student to misrepresent the latter's involvement in an academic task by assisting the misrepresentation.*

UW has a time-tested procedure to judge such cases, and serious penalties may be assessed.

So, do not cheat and do not help others cheat! In this class, if a student is caught cheating, he or she will not only lose the full point of the assignment/test, but may also be assigned a "F" for the course.

Plagiarism is considered a form of cheating. Both students will lose the full points on the particular homework or lab assignments. However, when writing papers, a student may cite other's work, but proper attribution must be given.

**Concerning homework/exams styles**
Four points must be emphasized: (1) For problems involving equations, if appropriate, provide a complete analysis rather than a single number. (2) Be professional in your presentations. If applicable, write down the unit for your results and round off the final number to 1 or 2 decimal points. (3) You can discuss the problems with fellow students, but complete your assignments by yourself. Copying other's work is considered cheating and no points will be given. (4) Hand in the homework on time. Finally, please keep all course materials (notes, exercises, homework/exams/labs) to yourself and do not share them with future students. They must, as you have, work to earn the credit.

**Disclaimer**
The syllabus is subject to changes as deemed necessary by the instructor. If a significant change were to be made, all students will be informed of it and given appropriate reasons for such a change.

**Course Outline**

Part I: Writing codes for people to understand

You wrote some codes to solve a problem, after a month you read it again and could not figure out what you were trying to do. This is a scenario that even the most experienced programmers are struggling with. In this part, we will discuss how to write codes that a person can understand and maintain.

1. Introduction
2. Naming conventions
    a. Variables
    b. Constants
    c. Structures
    d. Functions
    e. General
3. Files and Organization
    a. M Files
    b. Input and Output
4. Statements
    a. Variables
    b. Loops
    c. Conditionals
    d. General
5. Layout, Comments and Documentation
    a. Layout

b. White Space
c. Comments
d. Documentation

Part II: Writing fast Matlab codes

Mablab is a prototyping environment, meaning it focuses on the ease of development with language flexibility, interactive debugging, and other conveniences lacking in performance-oriented languages like C and Fortran. While Matlab may not be as fast as C, there are ways to bring it closer.

1. Introduction
2. The Profiler
    a. Tic/toc
    b. Profile report
    c. Influence of background processes
3. Array preallocation
4. JIT Acceleration
    a. In-place computation
    b. Multithreaded computation
5. Vectorization
    a. Vectorized computation
    b. Vectorized logic
6. Function Inlining
7. Referencing operations
    a. Subscripts vs. indices
    b. Vectorized subscripts
    c. Vector indices
    d. Reference wildcards
    e. Logical indexing
    f. Deleting submatrices with []
8. Solving linear systems
    a. Iterative methods
    b. Functional representation
    c. Special matrices
    d. Inlined PCG
9. Numerical integration
    a. One-dimensional integration
    b. Multi-dimensional integration
10. Signal processing
    a. Moving-average filter
    b. Locating zero-crossings and extrema
    c. FFT-based convolution
    d. Noncausal filtering and other boundary extensions
    e. Upsampling and downsampling
11. Miscellaneous

a. Clip a value without using if statements
b. Convert any array into a column vector
c. Find the min/max of a matrix
d. Flood filing
e. Vectorized use of set on GUI objects

Part III: Writing re-usable codes

Object-oriented programming is a formal programming approach that combines data and associated actions (methods) into logical structures (objects). This approach improves the ability to manage software complexity—particularly important when developing and maintaining large applications and data structures.

1. Introduction
2. Matlab classes overview
3. Defining and organizing classes
    a. User-defined classes
    b. Class definition
    c. Class attributes
    d. Organizing classes
    e. Class precedence
    f. Namespace
    g. Importing classes
4. Value vs. handle class
5. Properties
    a. How to use properties
    b. Defining properties
    c. Property attributes
    d. Mutable/Immutable properties
    e. Property access
    f. Properties containing objects
    g. Dynamic properties
6. Methods
    a. How to use methods
    b. Method attributes
    c. Ordinary methods
    d. Class constructor methods
    e. Static methods
    f. Overloading functions
    g. Object precedence
    h. Class methods for graphics
7. Object arrays
8. Events
    a. Events and listeners
    b. Event attributes
    c. Listening for changes to properties

9. Building on other classes
    a. Hierarchies of classes
    b. Subclasses
    c. Modifying superclass methods and properties
    d. Subclassing multiple classes
    e. Access control
    f. Abstract classes and interfaces
10. Saving and loading objects
11. Enumerations