AMSC/CMSC 664 Advanced Scientific Computing
Spring 2018

Updates from last semester:

- **We won't grade code if it's undocumented or uncommented.**
- **You MUST use git unless you ask for an exception.**
- **We won't grade anything that is emailed to us.**
- **You must have a detailed bullet list of units that your code have, and for each unit a description of at least one unit test that you will perform.**
- **Your code must have unit tests, approximately one per serious code file, for the different UNITS of code. Your unit tests should have fine-scale granularity. A wholistic test that runs your entire system is NOT a unit test.**

Description

AMSC/CMSC 663-664 is a two-semester project course in which each student will identify and carry out a scientific computing project with focus on:

• Understanding of scientific computing algorithms related to the project
• Code Development, including
  o Modularity, portability, memory management
  o Post-processing, restarting, and writing to databases
  o Interactivity
  o Scientific visualization
  o Documentation and version management tools
  o Debugging and profiling tools
  o Validation
• Verification and unit testing using test problems
• Time permitting, additional considerations may be given, for example implementation of parallel algorithms: OpenMP, MPI, GPU programming; Masking communication costs, load balancing, granularity; parallel numerical linear algebra.

Logistics

Lectures: Tu/Th 9:30-10:45pm.
Location: TBD
Instructors:
  Tom Goldstein: AVW 3141 (pi)
  Jacob Bedrossian: MATH 4413 and CSIC 4129

Prerequisites

You *must* have taken both AMSC/CMSC 660 and 661, or have permission.

Course requirements

The second semester will have the following deliverables: second semester proposal, midterm code review (with a faculty member), final presentation, final report. You will also deliver code and documentation for your project.

Your faculty advisor must provide written feedback on BOTH your final document and final code. **Your project will not be graded until this feedback is received. No exceptions.**

Grading

This course does NOT have an "A for effort" grading policy. Projects will be graded on the quality of code, documentation, presentation, and writing. Projects that do not meet high standards of quality will not receive an "A", and in some circumstances will not receive a passing grade.

Important dates and grading events

Jan 31: First class meeting.
Feb 8: Second semester proposal document due.
March 1-on-1 code reviews.
May 7,9,14,16: Final presentation days.

Requirements for deliverables

Code requirements:

- All code must be commented and **each file must have a heading briefly explaining what the file does** and where in the documentation the reader can learn more details
- The code for the unit tests must be provided, documented, and commented.

Second semester proposal must have the following sections:

- Description (a few paragraphs) of the capabilities that will be added and the work that will be done
- A detailed bullet list of different code units (at the level of files or subroutines) you are planning to create to achieve your goal, and a description of their purpose.
- Validation/Unit tests: For **every** code unit, you must have a description of the unit test(s) you will create for that unit. This should include information on how the unit test will work (i.e., don't just say "check that it produces correct output". You must have a description of inputs will be used and how the outputs will be checked).

Second semester code review:

Each student will meet individually with either Goldstein or Bedrossian, and present their code. Students should prepare a walk-through of the code that shows how it's organized and how it works. Students should plan the meetings to be between 30-60 minutes.

Final presentation and final report

- Background on the problem: why is it important, what is the state of the art (with citations)?

- Project Goals: what did you achieve?
- Describe specific algorithms and how they were implemented
- **In a separate section** at the end of your document: What was specifically achieved this semester, and what specific code units (files) were developed that were not present last semester?
- **In another separate section describe validation methods and unit tests**
- Deliverables: what specific components/code/data/visualizations was generated?
- In a separate section: you must have a detailed bullet list of units that your code have, and for each unit a description of at least one unit test that you was performed.

Documentation

- You must have a user guide in the form of a separate pdf, or a markdown formatted GitHub/GitLab readme.
- Describe the structure and organization of the code, how to use the code, and what the code does (e.g. what algorithms are being used with what parameters by what function calls etc)
- Complete usage examples should be provided.

The oral presentation should last no more than 30 minutes including questions and discussions. A sample presentation document is available. You will be evaluated on the quality of your presentation skills, and clarity of your slides.

Academic integrity

Be aware of the UMD code of academic integrity, found here http://www.president.umd.edu/policies/iii100a.html. All sources must be properly and clearly cited. Text or code can never be copied verbatim from the sources, this is plagiarism and a serious offense. You can, of course, import and use third-party libraries in your code (provided you give full and due credit and discuss the third party in your reports and presentations).