ECE 4960 Spring 2016: Computational and Software Engineering

By Edwin C. Kan

TTh 11:40am – 12:55pm Phillips Hall 203

Reading and Assignment	Tuesday	Thursday
Bindel Chap. 1 Einarsson Chap. 2	**1/26** No class	**1/28** Class introduction
Bindel Chap. 2; Einarsson Chap. 1	**2/2** Source of errors in computing: precision and exceptions	**2/4** Source of errors in computing: Exception handling
Bindel Chap. 3	**2/9** Source of errors in computing: Singularity and stability	**2/11** Local analysis and approximation: Expansion and extrapolation
Einarsson Chap. 3 *Coding 1: Exception handling*	**2/16** No Class (February break)	**2/18** Error estimation techniques; computational unit testing
Bindel Chap. 4	**2/23** Linear algebra review: full and sparse systems	**2/25** Matrix conditioning and stabilization
Einarsson Chaps. 4 – 6 Chapra Part 3 *Coding 2: Blas Interface*	**3/1** Error bounds and analysis; perturbation and noise injection	**3/3** Memory usage management: memory access violation and leaks
Bindel Chap. 5; Oliveira Chaps. 14, 15	**3/8** Nonlinear solver overview	**3/10** Iterative methods and relaxation; local and global optimization
Bindel Chap. 6 Oliveira Chap. 13	**3/15** Approximation: interpolation and least-square fitting	**3/17** Ordinary differential equation: first and second order evaluation
Bindel Chap. 7 *Coding 3: Parametric optimization*	**3/22** Ordinary differential equation: stability and accuracy	**3/24** Discrete Fourier transform and marginal testing
	3/29 No class (Spring break)	**3/31** No class (Spring break)
Chapra Part 5	**4/5** Computational geometry and Boolean operations	**4/7** Computational geometry and Boolean operations
Chapra Part 7 and 8 *Coding 4: Nonlinear circuit simulation*	**4/12** 1D finite-difference and finite-element solvers	**4/14** Error estimation and HP adaptivity
Parts in Bindel Chaps. 8, 9	**4/19** 2D and 3D finite-difference and finite-element solvers	**4/21** 2D and 3D finite-difference and finite-element solvers
McConnell Parts I, V; Oliveira Chaps. 3, 9; Einarrson Chap. 13	**4/26** Large-scale software architecture: Modular objects and testing	**4/28** System-level software reliability evaluation and validation
Oliveira Chaps. 6, 7, 8; McConnell Part VI and VII; Einarrson Chap. 8 *Coding 5: Poisson solver*	**5/3** Software engineering framework: Source code control; information sharing and hiding	**5/5** Software engineering framework: Language characteristics; documentation; convention

Course description: This course will introduce the programming, maintenance and testing practices to improve and validate computing. By classifying the computing error sources, we will treat approximation and control errors in algebraic operations, geometry and differentiation. Matrix conditioning and its relation to nonlinear solvers, as well as stability and accuracy estimate in statistical methods, will be examined. The course will then introduce software engineering principles and the team design process. Program craftsmanship and language characteristics for large-scale computing will be overviewed. The course programming assignments will be based on Linux and C/C++ with ECE applications such as SPICE and electrostatics. Optional evening sections will be given to students with mismatched background.

Pre-requisites: ECE 2400 or CS 2110. An introductory course in scientific computing will be helpful but not required.

Logistics: 4 units. Lectures and programming projects. Computer lab: Th 7:30pm – 9:00pm. Registration to the lab is required, and the lab period will be used for background patchup, system setup, questions and answer, as well as makeup lectures.

Reference textbooks: (all reading will be provided on-line)

1. D. Bindel and J. Goodman, *Principles of Scientific Computing*, 2009.
2. S. Oliveira and D. Stewart, *Writing Scientific Software: A Guide to Good Style*, Cambridge 2006.
3. S. McConnell, *Code Complete: A Practical Handbook of Software Construction, 2nd Ed.*, Microsoft Press, 2004.
4. B. Einarsson, Ed., *Accuracy and Reliability in Scientific Computing*, SIAM 2005.
5. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers, 7th Ed.*, McGraw-Hill, 2015.
6. A. Allain, *Jumping into C++*, Cprogramming.com, 2015.
7. Additional papers from Software Engineering conferences.

Grading: Reading material reviews (by multiple choice questions on Blackboard): 15%; Coding 1: 10%; Coding 2: 10%; Coding 3 – 5: 15% each; Hack4960 Final: 20%. According to the program assignment format, members in the same group may not receive the same grade.

Program assignments: There are 5 program assignments as the main stream throughout the semester. The first program will be by groups of 2 students for cross checking. The second program will be by each student where good program practices of version control and object models will be introduced. The last three programs with more extensive interface and testing will be by groups of 3 – 4 students.

Program project policy: For each program assignment, a simple proposal will be submitted and reviewed. For program assignments with multiple students in each group, you are highly recommended to work with different students to expand your horizon in a more realistic setup. Due to the small class this semester, the group constraint will not be enforced.

Hack4960: This unconventional "final exam" will be administered in the posted final exam time and location. It would be similar to the HackXX event with smaller personal efforts, instead of a team project towards a complete software product. Although we have emphasized the importance of software engineering to achieve robust software instead of fastest hacking, this exercise will hopefully demonstrate how systematic programming can help reduce development time. You will need to bring your notebook computer with Internet access. At the end of the exam time, you will submit your code with a short description of operation and validation. The time will be 5/19/2016 (Thurs.) at 7pm.

Platform: The program assignment will be implemented by C++, as execution and memory efficiency is often critical in this level of programs. The suggested platform is Linux, although other IDEs that support multiple authors and version control are allowed as well. You will need to have live access on your laptop computer for program demonstration and Hack4960 final exam. You can install a Virtual Machine (visit www.virtualbox.org) on your laptop computer with Linux Debian OS, or you can remote login to amdpool on ECE servers as well. The software you will definitely need include: compilers: gcc or g++; debugger: gdb (generic or in any wrapper of Eclipse or code::blocks); source control: rcs or cvs or git.