**ECE 4960 Spring 2019: Scientific and Numerical Computation**

By Edwin C. Kan

|  |  |  |  |
| --- | --- | --- | --- |
| **Logistics** | **Tuesday**  10:10am – 11:25am  Phillips 407 | **Wednesday Lab**  7:30pm – 9:00pm  Phillips 403 | **Thursday**  10:10am – 11:25am  Phillips 407 |
| Oliveira Chaps. 6-8 Einarrson Chap. 8  Bindel Chap. 1 | **1/22**  Class introduction; Software in the real world | **1/23**  No lab | **1/24**  Source of errors in computing: Overview |
| Bindel Chap. 2;  Einarsson Chaps. 1, 2 | **1/29**  Standards of integers and floating-points | **1/30**  Language tradeoff; structure and objects; | **1/31**  Conditioning of functions and round-off errors |
| Bindel Chap. 3  Oliveira Chaps. 3,9 Einarrson Chap. 13 | **2/5**  Local analysis: Euler methods and Taylor expansion | **2/6**  Source code control, modular programming and regression test suites | **2/7**  Local analysis: Orders of approximation and Richardson methods |
| Einarsson Chap. 3  *Coding 1: Exception handling* | **2/12**  Integration and Gaussian quadrature | **2/13**  Lab 1 practicum: Program review | **2/14**  Sparse and full matrix operations |
| Bindel Chap. 4 | **2/19**  Matrix conditioning and pivoting in solvers | **2/20**  Linear algebra by software: Blas as a tradeoff between flexibility and efficiency | **2/21**  Error analysis: perturbation and noise injection |
| Einarsson Chaps. 4 – 6  Chapra Part 3 | **2/26**  February break:  No class | **2/27**  No lab | **2/28**  Nonlinear equation and optimization: Jacobian matrix |
| Bindel Chap. 5;  Oliveira Chaps. 14, 15  *Coding 2: Matrix* | **3/5**  Newton’s method for nonlinear problems | **3/6**  Lab 2 practicum: Program review | **3/7**  Other search methods for nonlinear problems: Direction and step size |
| Bindel Chap. 6  Oliveira Chap. 13 | **3/12**  Least-square and parametric optimization | **3/13**  Management of memory: violation and leaks | **3/14**  Computational geometry and spline fitting |
| Bindel Chap. 7  *Coding 3: Parametric optimization* | **3/19**  Statistical methods: Monte Carlo sampling | **3/20**  Lab 3 practicum: Program review | **3/21**  Ordinary differential equation and local analysis |
| Chapra Part 5 | **3/26**  Euler and predictor-corrector methods | **3/27**  Introduction to SPICE | **3/28**  Runge Kutta methods |
|  | **4/2**  Spring break:  No class | **4/3**  Spring break:  No lab | **4/4**  Spring break:  No c**l**ass |
| Chapra Part 7 | **4/9**  Error estimation and hp adaptivity | **4/10**  Makeup class if necessary | **4/11**  Multi-step methods and TR-BDF2 |
| Bindel Chap. 8  *Coding 4: Nonlinear circuit simulation* | **4/16**  1D finite-difference PDE solver: elliptic | **4/17**  Lab 4 practicum: Program review | **4/18**  Finite-difference parabolic PDE solver: parabolic |
| Bindel Chap. 9  Chapra Part 8 | **4/23**  1D finite-difference PDE solver: hyperbolic | **4/24**  Case study for large-scale scientific software | **4/25**  2D and 3D finite-difference solvers: equation assembly |
| *Coding 5: Proposed major revision* | **4/30**  Basics of 1D finite-element methods | **5/1**  Lab 5 practicum:  Proposal and planning | **5/2**  2D and 3D finite-element solvers: Triangulation |
|  | **5/7**  Large-scale software: Robust handling | **5/8**  No lab  (Semester ends) | **5/9** |

**Course description**: When scientific programs grow into large scales, robust programming and testing techniques are crucial. Object-oriented modules need minimalistic well-defined procedural interface to go with source-code control and extensive regression testing. Errors from logic, precision, exception, formulation and implementation are unavoidable and require specific techniques to be detected, estimated and corrected, such as the Wilkinson principle, memory access interception and *hp* adaptivity. This course will start from the mathematics, design, maintenance and testing practices to scientific computing for interface and simulation of the physical real world. Topics include precision, local approximation, sparse matrix, optimization, geometry and differential equations. Computational conditioning as well as stability and error estimation will be examined. Students can choose their most comfortable general-purpose development platforms, and C/C++ with ECE applications will be used for illustration. The Lab section will introduce software techniques and conduct program review.

**Pre-requisites:** ECE 2400 or CS 2110. Introductory scientific computing will be helpful but not required.

**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. B. Einarsson, Ed., *Accuracy and Reliability in Scientific Computing*, SIAM 2005.
4. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers, 7th Ed*., McGraw-Hill, 2015.
5. (Optional) S. McConnell, *Code Complete: A Practical Handbook of Software Construction, 2nd Ed*., Microsoft Press, 2004.
6. (Optional) A. Allain, *Jumping into C++*, Cprogramming.com, 2015.

**Program assignments:** There are 5 program assignments as the main efforts throughout the semester. Each assignment will contain required smaller modules for design and testing purposes, as well as a culminating program. The grouping regulation will be described in each assignment. Groups of 2 students are the usual cases. Good program practices of version control and object models are mandate to facilitate code reuse, which not only enhances productivity, but more importantly improves reliability and ease of debugging.

**Blackboard site:** 12487\_2019SP: ECE 4960 Special Topics. All class materials will be on Blackboard.

**Video notes:**

* From Spring 2017: <http://cornell.videonote.com/channels/1013/videos>
* From live Spring 2018: <http://cornell.videonote.com/videos/1000284/play>

**Grading:** Reading material reviews (by multiple choice questions on Blackboard, due on end of Sunday in the applicable week): 10%; Weekly program assignment (by your Git repository, due on end of Friday in the applicable week): 10%; Project assignment: 13% each; Final hacking exam: 15%.

**Student Outcomes:**

1. The student can use software engineering tools such as source code control and automated test suites to manage large-scale software development and testing.
2. The student can analyze the source of errors through modular testing and external observation of asymptotic and known cases.
3. The student can translate a physical real-world problem into a software solution.