Math 573 – Reliable Mathematical Software

Course Description from Bulletin: Many mathematical problems cannot be solved analytically or by hand in a reasonable amount of time, so turn to mathematical software to solve these problems. Popular examples of general-purpose mathematical software include Mathematica, MATLAB, the NAG Library, and R. Researchers often find themselves writing mathematical software to demonstrate their new ideas, or using mathematical software written by others to solve their applications. This course covers the ingredients that go into producing mathematical software that is efficient, robust, and trustworthy. Students will write their own packages or parts of packages to practice the principles of reliable mathematical software (1-0-0).

Enrollment: Elective for graduate and undergraduate students

Textbook(s): Lecture notes and articles

References: R. K. Johnson, *The Elements of MATLAB Style,* Cambridge University Press, Cambridge, 2010.

S. Oliveira and D. E. Stewart, *Writing Scientific Software: A Guide to Good Style*, Cambridge University Press**,** Cambridge, 2006.

Other required material: MATLAB

Prerequisites: MATH 350 or permission of instructor

Objectives:

1. Students should learn what qualities good mathematical software must have to be useful for a general audience in the medium to long term.
2. Students should learn what software engineering practices are needed to produce quality mathematical software.
3. Students should learn to spot flaws in software written by others.
4. Students should produce a small software package, or part of a larger package, that follows good mathematical software development practice.

Lecture schedule: One 75-minute session per week for ten weeks

Course Outline: Hours

1. Characteristics of quality mathematical software 6
   - Low computational complexity or small truncation errors
   - Efficient code
   - Robustness to round-off error
   - Ease to use and modify
   - Robustness to user input
   - Performance guarantees
   - Reproducibility of results

2. Practices that ensure quality mathematical software 6
   - Software repositories
   - Documentation of software
   - Documentation of procedures for writing good software
   - Suites of tests
   - Installation and uninstallation
   - Theoretical justifications of success
3. Project presentations 3

Assessment:	Exercises	35%
	Project and Presentation	65%

Syllabus prepared by: Sou-Cheng Choi (U Chicago) and Fred J. Hickernell
Date: August 2, 2013