

IE523: Financial Computing

Fall, 2016

Instructor: Prof. R.S. Sreenivas

201E Transportation Building (Primary Office)

155 Coordinated Science Laboratory (Secondary Office)

e-mail: rsree@illinois.edu

MW, 2:00-3:20PM, 106B8 Engineering Hall

Teaching Assistant: Arun Raman (raman12@illinois.edu)

Office hours: Tue-Thu 11:00AM-12:30PM

Course Description: This course will introduce you to programming and computational concepts in C++ using examples that are relevant to Financial Engineering.

Primary Text: Lessons with Code Samples written by me.

1 Tentative Syllabus

“You must fill your heads with wisdom before you can break boards with it.”

— Karate instructor on “The Simpsons”.

Lectures

1. Overview of C++.

- (a) **Lesson 1:** Recursion in C++; Tower of Hanoi Problem; The Master Theorem; Karatsuba’s $O(n^{1.585})$ algorithm for multiplying two n bit numbers; Strassen’s $O(n^{2.81})$ algorithm for multiplying two $n \times n$ matrices.

2. Methods.

- (a) **Lesson 2: Review of Linear Algebra:** Linear Independence, Bases and Subspaces; Orthogonality, Orthogonal Projection, Pseudo-Inverses; General Solution to $\mathbf{Ax} = \mathbf{b}$; The *NEWMAT* C++ library for linear algebra problems; Installing the *Lpsolve API*; Solving *Mixed Integer Linear Programs* (MILPs) within C++ code using the *Lpsolve API*.
- (b) **Lesson 3:** Root-finding by Method of Bisection; Newton’s Method; Secant Method; Descartes’ Rule of Signs; Application to IRR-computation in C++.
- (c) **Lesson 4:** Taylor’s Expansion and its uses; Bond Duration, Convexity, etc.; Bond Immunization in C++; Maximizing Convexity subject to Duration-matching constraints: An illustration “by-hand” and *lp-solve*.

- (d) **Lesson 5: Statistics and Simulation:** (C++ code for) Pseudo Random number generation and Uniform Distributions; (C++ code for) Inverse Transform Technique for other distributions; Efficient Discrete Random Variate generation; (C++ code for) Geometric Distributions; (C++ code for) Binomial Random Variates; The *Box-Muller Transform* and (C++ code for) Unit-Normal Variate generation; (C++ code for) Multivariate Gaussian; Generating constrained Random Variates; Generating Random Variates from Empirical Data; Basics of Discrete-time Markov Chains and some related computational problems (in C++); *Infinitesimal Perturbation Analysis* via Examples.
- (e) **Lesson 6: Frequency Domain Methods:** *Fourier Transform* and the *Fast Fourier Transform* (FFT) algorithm; MATLAB Signal Processing Toolkit Review; (C++ code for) Spectral analysis of financial data and related observations.

3. Computational Finance: Case Studies (Tentative)

- (a) **Lesson 7: Computational Aspects of Option Pricing:** Models for the dynamics of Asset Price; Ito calculus and *Delta Hedging*; Short introduction to the *Black-Scholes PDE* for the value of a derivative instrument; (C++ code for) *Black-Scholes formula* for the price of an European Option; Put-Call parity; Binomial Trees and Binomial Lattices; The *Martingale Property*; Option Pricing using the Binomial Lattice via Recursion (in C++); Pricing a path-dependent (American-Asian) Option using Recursion (in C++); Computational limitations of the Binomial Model; Discretization of asset-price in to b -many values; (C++ code for an) $O(b^2T)$ -algorithm for pricing American Option of duration T discrete-steps using Dynamic Programming; (C++ code for an) $O(b^3 \log T)$ -algorithm for pricing an European Option using Dynamic Programming and the method of *repeated-squaring*; Replication Portfolios; (C++ code for) Edirisinghe et al's approach to pricing European Options with transaction costs using Linear Programming. (C++ code for) Carr and Madan's approach to price an European Option with the *Fast Fourier Transform* (FFT).
- (b) **Lesson 8: Pricing Exotic Options – Barrier Options:** (C++ code for) Pricing using Truncated Binomial Lattices; “Overestimation of price” phenomenon; (C++ code for) Adjusted Binomial Lattices using the Baron-Adesi, Fusari and Theal correction-term; Compendium of Closed-Form Expressions for European Barrier Options; Pricing an European Discrete Barrier Option – role of *Random Walks*, *Weiner Processes*, *Brownian-bridges* in asset pricing; (C++ code for) Computing the price of an European Discrete Barrier Option using Brownian-bridge correction-terms.

- (c) **Lesson 9: Recursion for Statistical Computing:** Median (and k -th *order statistic*) selection using sorting; The Blum-Floyd-Pratt-Tarjan $O(n)$ *Median-of-Median* algorithm for efficient picking of the k -th order statistic in a list; Experimental Determination of the relevant constants in implementation of algorithms; Moving-Median filtering vs. Moving-Average Filtering for out-lier elimination in noisy real-time data.
- (d) **Lesson 10: Pricing an American-Asian Option using the Hull-White Interpolation Method:** The intractability of path-dependent option pricing; (C++ code for) Hull and White’s interpolation method; Experimental observations of accuracy vs. grid-size trade-off.
- (e) **Lesson 11: Simulation:** Simulating Random-walks in C++; Pricing an European Option by Simulation; Computing the *greeks* from a single-run simulation using Infinitesimal Perturbation Analysis; Can we run a simulation “backwards” and price an American Option?; Pricing American Options and the “Monte-Carlo within Monte-Carlo” problem; Longstaff and Schwartz’s solution to the “Monte-Carlo within Monte-Carlo” problem; (C++ code for) Least Squares Monte Carlo.
- (f) **Lesson 12: Fractional Brownian Motion Models:** Geometric Brownian Motion and the “fat-tail” problem; *Fractional Brownian Motion* overview; the *Hurst exponent*; (C++ code for) Estimating the Hurst exponent from sample-paths; Using the Hurst exponent in the validation of Financial Models.
- (g) **Lesson 13: Epilogue:** The “*do not drink the kool-aid*” spiel; Videos of – Nicholas Taleb, Robert Merton and Benoit Mandelbrot (on Bachelier, Brownian Motion, Markets and Risk-Management); Problems/Issues with the Fractional Brownian Motion model; *Statistical* and *Computational undecidability* and its implications to *agnosticism* in model estimation and validation.

2 Grade Composition

- ≈ 10 Programming Assignments (50%).
- (≈ 10) Quizzes (10%)
- (Take-Home + In-Class) Mid-Term (20%).
- (Take-Home + In-Class) Final (20%).

3 General Instructions

I plan to have ≈ 10 programming assignments for this course. Since the pace of the course will be dictated by the class needs/skills, the exact number of

these assignments might vary. The contribution to your final grade will be 50% independent of the number.

I am planning to have ≈ 10 quizzes/assessments, that you can take on Compass. Their contribution to your final grade will be 10% independent of the number. These will be short, mostly multiple-choice, questions. I am trying this Compass-function out for the first-time, and if we run into issues, we might switch to doing them on paper. Stay tuned for more details!

Since this is a 500-level course on programming, your proficiency in the course material will be tested primarily in your programming assignments. You must turn-in an electronic version of your code on or before the date they are due. Please do not ask for extensions in the 11-*th* hour. The TA and I have a very demanding schedule this semester and delays intrude into the other tasks that we need to get done. You will get full-credit if your submitted code works when compiled and run on a data-set of my choice. We will revert back to you if there is an error/problem with your submission. You then have three days to turn-in a corrected version, at the loss of 20 points. This process is repeated at most two times (i.e. inclusive of your first attempt, you have three chances at getting the programming assignment right).

The TA would require that you follow these submission guidelines for the programming assignments:

1. The *.cpp and *.h files should be named/labelled according to this format: `first_name_last_name_assignment_number.cpp` and/or `first_name_last_name_assignment_number.h`
2. Send the file(s) as attachments to his e-mail address (i.e. do **not** send the entire “solution” folder from your Mac/Windows machine – it takes too much space!).

The graded quiz can be picked from the TA during his office hours –

Tue-Thu 11:00AM - 12:30PM.

Make sure you e-mail him with your questions before you show up at his desk. This way, he will be able to answer your questions better.

The mid-term and final examinations consist of two parts (1) a take-home component, that is a programming project, and (2) an in-class written component with short-answers. You have a week to design, compile and test your C++ code. You turn-in your code on the due date (no extensions, please!). Just as with the programming assignments, we will get back to you if your code does not do what it is supposed to. You have three days to turn-in a corrected version for a loss of 20 points. This process is repeated at most two times (i.e. inclusive of your first attempt, you have three chances at getting the programming component of the mid-term and final exams right).

The mid-term exam will be held during class-hours at a date that is to be announced. I expect it to be during the first week of November. The details regarding the final exam schedule can be found at [this link](#). Since we have a full class (≈ 60 students), I will arrange for an alternate venue for our exams.

If I am not able to find a suitable classroom, the exams will most likely be held in the same classroom where we meet regularly.

I intend to use the \pm -grading system. My lecture notes for the course can be found on the University of Illinois' [Compass Website](#). I suggest you print the appropriate lesson before class and follow-along. This will free you from the tedium of copying material off the board during class, you can use that time to follow the material presented in class instead. It is your responsibility to check the above URL regularly for updates/due-date-announcements as the course progresses.

The TA can help you with any programming related questions you might have. If you have coding-related questions, you can address it to him via e-mail. If you cannot resolve the issue over e-mail, you can meet with him at a time that is to-be-announced.

I will be happy to speak to you if you are having trouble with the algorithmic-aspects of the course material. I prefer to work on an “interrupt-driven” mode – as opposed to having a fixed-meeting time each week. If you need to speak to me for any reason, just send me an e-mail, and I will let you know the times when I can meet with you. I will let you know where, among my two offices, we will meet.

Good luck and I am looking forward to seeing you do well in this course!