IE523 B, ONL: Financial Computing
Fall, 2023

Instructor: Prof. R.S. Sreenivas
201E Transportation Building (Primary Office)
e-mail: rsree@illinois.edu
MW, 12:30-1:50PM, 114 Transportation Building
TA: Aditya Gopalan (gopalan6@illinois.edu)
TA Office hours: TBA

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 0: Review of C++**: You have to review the material from the *Preparatory Course in Computing* on the Canvas Website on your own. You will find a Sample Quiz and recorded Zoom lectures from the Preparatory Course in Computing held this summer – you need to be familiar with the topics covered in these problems as a pre-requisite to this course. **I will not cover any of this material**.
   (b) **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 *Gurobi API*; Solving *Mixed Integer Linear Programs* (MILPs) within C++ code using the *Gurobi 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 Gurobi.

(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; Where do the probabilities come from? Baxter & Rennie's Parable of the Bookmaker, The *Expected Martingale Measure* or *No-Arbitrage-Measure*.

3. **Computational Finance: Case Studies** (Tentative)

(a) **Lesson 6: 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 7: 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 8: 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.

(d) **Lesson 9: 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.

(e) **Lesson 10: 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.

(f) **Lesson 11: 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);

2 Grade Composition

- ≈ 10 Programming Assignments (70%).

  1. **Programming Assignment 1: K Peg Version of the Tower of Hanoi Problem via Recursion**: You will use nested-recursion and generalize the Frame-Stewart Solution to the 4-Peg Tower of Hanoi Problem. You wrote C++ code that recursed over the #Pegs and the #disks.
  2. **Programming Assignment 2: Sudoku Solver via Exhaustive-Search using Recursion**: We will price various instruments through recursion – you have to get used to thinking recursively. Part 1: write a recursive piece of code that finds *a solution* to an arbitrary Sudoku Puzzle. Part 2: modify this code to find *all solutions* to an arbitrary Sudoku Puzzle.
  3. **Programming Assignment 3: Programming Assignment 3: Dutch Miracle Sudoku Solver via Exhaustive-Search using Recursion**: One more attempt at getting you competent in recursive thinking – this is a Sudoku puzzle that has additional constraints

(above the usual ones). The new constraints can be easily incorporated into your recursive solutions to the previous assignment (as you will see). Part 1: write a recursive piece of code that finds *a solution* to an arbitrary Sudoku Puzzle (with the Dutch Miracle constraints). Part 2: modify this code to find *all solutions* to an arbitrary Sudoku Puzzle (with the Dutch Miracle constraints).

4. **Programming Assignment 4: Finding all Solutions to a Sudoku Puzzle using Integer Linear Programs**: This time, you are going to use Integer Linear Programming (ILPs) to solve the (regular) Sudoku puzzle. art 1: write a recursive piece of code that finds *a solution* to an arbitrary Sudoku Puzzle using an ILP-formulation. Part 2: modify this code to find *all solutions* to an arbitrary Sudoku Puzzle using an ILP-formulation.
5. **Programming Assignment 5: Computing the General Solution to $\mathbf{Ax} = \mathbf{y}$**: You are going to use NEWMAT to find the general solution to a set of linear equations.
6. **Programming Assignment 6: How much would you pay for this card-game?**: You are going to figure out the price-of-admission for this card game, which is essentially an American Option (that we will see later). I want your code to work for a large (i.e. ≥ 1000) deck of cards. You will have to use memoization for this to work.
7. **Programming Assignment 7: How to loose as little as possible**: You are going to verify the results of a coin-toss game described by Addona, Wagon and Wilf (you can find this paper on Canvas). This has a discrete-optimization component to it, which you then verify through simulation of the coin-toss game.
8. **Programming Assignment 8: Repeated Squaring Algorithm**: Write C++ code that computes matrix exponents, where the exponent can be very very large. That is, you will compute $\mathbf{A}^n$ for an arbitrary (square) matrix $\mathbf{A}$ for extremely large values of n. This efficient algorithm is only possible due to recursion.
9. **Programming Assignment 9: Pricing European- and American-Options Using the Trinomial Model with Memoization**: Price an American Option using a Trinomial Model with *Memoization*. Look for details on Canvas. I want your code to be able to handle #stages ≥ 5000, and it should run in no more than 2 seconds.
10. **Programming Assignment 10: Verification of the Adjustment-Method for Pricing European Down-and-Out Continuous-, and European Down-and-Out Discrete-Barrier Options via Monte Carlo Simulation**. Look for details on Canvas.

- In-Class Mid-Term (15%).
  - Tentative Date: October 25, 2023; regular class hours.

- In-Class Final (15%).
  - As per Provost's website, it looks like the final exam will be held 1:30pm-4:30pm., Friday Dec. 15, 2023. You should find the exact date on Course Explorer by October 12, 2023.

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 70% independent of the number.

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). **You will submit your *.cpp and *.h files, along with any PDF-documents that explain your code on Canvas. Please do not e-mail the TA or me.** The TA's office hours are TBA. I will send an announcement once we get the details from the TA. Make sure you e-mail her with your questions before you show up online/in-person. This way, she will be able to answer your questions better.

The mid-term and final examinations consist an in-class written component with short-answers. For the online students, we have to figure out the exact mechanisms for how these exams will be conducted. The on-campus students will have to take it in person (with strict social distancing and other relevant University rules) at an appropriate time and venue.

I intend to use the $\pm$-grading system. My lecture notes for the course can be found on the University of Illinois' Canvas 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 her via e-mail. If you cannot resolve the issue over e-mail, you can meet with her 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!

4 Administrivia

1. The tentative due dates for your programming assignments and exams are shown in table 1. If there are any changes to this schedule I will let you know in class at the appropriate time.

2. We will have the Mid-Term Exam during

   Regular Class Hours, October 25, 2023.

3. According to the Provost's Website, the last day of class is December 7, 2022, and the Final-Exams are be held in the week of December 9, 2022 to December 16, 2022. Please be prepared for changes to this schedule if there are additional COVID-19 related changes. As per the Final Exam Schedule at the Provost's Website, the Final-Exam for this course is to be held

   1:30pm-4:30pm., Friday Dec. 15, 2023.

4. If you have a MAC, you can download XCode for free from the App Store. If you have a Windows machine, you can get Visual Studio for free from UIUC Webstore. **If you install Visual Studio 2017 – keep in mind that the look-and-feel of this release is very different from the previous releases. Do not use any bootlegged-copies of Visual Studio! You can get the legitimate-version for free!**

Programming Assignment 1	15 September 2023
Programming Assignment 2	22 September 2023
Programming Assignment 3	29 September 2023
Programming Assignment 4	6 October 2023
Programming Assignment 5	13 October 2023
Programming Assignment 6	20 October 2023
Mid-Term Exam	25 October 2023
Programming Assignment 7	3 November 2023
Programming Assignment 8	10 November 2023
Programming Assignment 9	17 November 2023
Thanksgiving Break	18 November 2023 - 26 November 2023
Programming Assignment 10	1 December 2023
End-Term Exam	1:30PM - 4:30PM, 15 December 2023

Table 1: Tentative Due Dates for Programming Assignments and Exams.