FE545 Design, Patterns and Derivatives Pricing

Introduction and A Simple Monte Carlo Model

Steve Yang

Stevens Institute of Technology

steve.yang@stevens.edu

01/23/2024

Overview

- **Topics:**
  This course covers the design patterns and implementation of financial models using object oriented programming in C++. It discusses advanced applications on quantitative finance with special emphasis on derivatives pricing and their calculations using commonly known formulas such as the Black-Scholes and lattice models. The course uses available simulation techniques such as Monte Carlo simulation and its implementations in financial engineering problems.

- **Textbooks:**
  - **Mark S. Joshi, C++ Design Patterns and Derivatives Pricing, 2nd edition. Cambridge University Press, 2008 [REQUIRED]**

  - Mark S. Joshi, The Concepts and Practice of Mathematical Finance, 2nd edition, Cambridge University Press, 2008 [OPTIONAL]

  - IE. Gamma, R. Helm, R Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 2nd Edition, Addison-Wesley, 2000. [OPTIONAL]

  - R. Lafore, Object-Oriented Programming in C++, 4th edition. Sams, 2002. [OPTIONAL]

Policies

Homework Honor Policy:

- You are allowed to discuss the problems between yourselves, but once you begin writing up your solution, you must do so independently, and cannot show one another any parts of your written solutions. The homework is to be pledged (for undergraduate students).
- Your solutions to the homework and exam problems have to be typed (written legibly) and uploaded to the Canvas course website in one single PDF file (no other file format will be accepted). Any changes to the course schedule or due date of assignments will be announced through the course website.
- Each homework assignment will contain 2-4 problems, and will be posted on the class website. Late homework submission will receive 10% panelty under any circumstances.

Exams & Grades

- **Tools: A Working C++ IDE. Recommendations:**
  a) Windows: Microsoft Visual Studio: https://www.visualstudio.com/free-developer-offers/
  b) macOS: Xcode
  c) Linux: Eclipse with gcc
- **Grades:** Homework Assignments (5 assignments) - 45%; Breakout Excercises - 5%; Quizzes (4) - 20%; Final Exam- 30%.
- **Exams:** (Final) EXAM: May 7th - (Tuesday). These exams will consist of short questions, and mathematical programming problems.

  Exam must be taken at these times No Exceptions!

Derivatives in Financial Market

- The term derivative refers to a type of financial contract whose value is dependent on an underlying asset, group of assets, or benchmark. A derivative is set between two or more parties that can trade on an exchange or over-the-counter (OTC).

- Derivatives can be used to either mitigate risk (hedging) or assume risk with the expectation of commensurate reward (speculation). Derivatives can move risk (and the accompanying rewards) from the risk-averse to the risk seekers.

- Derivatives are usually leveraged instruments, which increases their potential risks and rewards. Common derivatives include futures contracts, forwards, options, and swaps.

Derivative Pricing

- It is important to understand how prices of derivatives are determined. Whether one is on the buy side or the sell side, a solid understanding of pricing financial products is critical to effective investment decision making.

- The unique characteristics of derivatives, however, pose some complexities not associated with assets, such as equities and fixed-income instruments.

- Derivatives are priced by creating a risk-free combination of the underlying and a derivative, leading to a unique derivative price that eliminates any possibility of arbitrage.

- Derivative pricing through arbitrage precludes any need for determining risk premiums or the risk aversion of the party trading the option and is referred to as risk-neutral pricing.

The Theory - Monte Carlo Simulation in Finance

- Our interest in simulating paths of geometric Brownian motion lies primarily in pricing options, particularly those whose payoffs depend on the path of an underlying asset S and not simply its value $S(T)$ at a fixed exercise date T.

- The price of an option may be represented as an expected discounted payoff. This price is estimated through simulation by generating paths of the underlying asset, evaluating the discounted payoff on each path, and average over paths.

- The one subtlety in this framework is the probability measure with respect to which the expectation is taken and the nearly equivalent question of how the payoff should be discounted - how the paths of the underlying asset ought to be generated.

Monte Carlo Simulation in Finance

- We start by assuming the existence of a constant continuously compounded interest rate r for riskless borrowing and lending. A dollar invested at this rate at time 0 grows to a value of

$$\beta(t) = e^{rt}$$

- Similarly, a contract paying one dollar at a future time t (a zero-coupon bond) has a value at time 0 of e^{-rt}. In pricing under the risk-neutral measure, we discount a payoff to be received at time t back to time 0 by dividing by $\beta(t)$; i.e. β is the numeraire asset.
- Suppose the asset S pays no dividends; then, under the risk-neutral measure, the discounted price process $S(t)/\beta(t)$ is a martingale:

$$\frac{S(u)}{\beta(t)} = \mathbf{E}\left[\frac{S(t)}{\beta(t)} | \{S(\tau), 0 \leq \tau \leq u\right]. \tag{1}$$

Monte Carlo Simulation in Finance

- Equation (1) shows that if S is a geometric Brownian motion under the risk-neutral measure, then it must have $\mu = r$; i.e.,

$$\frac{S(u)}{\beta(t)} = rdt + \sigma dW(t) \tag{2}$$

  In a world of risk-neutral investors, all assets would have the same average rate of return - investors would not demand a higher rate of return for holding risky assets.

- In the case of an asset that pays dividends, we know that the martingale property continues to hold but with S replaced by the sum of S, any dividends $D(t)$ paid by S, and any interest earned from investing the dividends at the risk-free rate r and δ rate of dividend yield.

$$\frac{D(t)}{dt} = \delta S(t) + rdD(t),$$

Strength and Weakness of Monte Carlo Simulation

- The advantage of the Monte Carlo simulation method is to deal with path dependent options. The superiority of the Monte Carlo simulation method is that it can simulate the underlying asset price path by path, calculate the payoff associated with the information for each simulated path, e.g. S_{max} or S_{avg}, and utilize the average discounted payoff to approximate the expected discounted payoff, which is the value of path-dependent options.

- However, the advantage of the Monte Carlo simulation method causes the difficulty to apply this method to pricing American options. This is because it is difficult to derive the holding value (or the continuation value) at any time point t based on one single subsequent path.

- Someone may try to apply the multiple-tier Monte Carlo simulation to estimating the holding value and thus price American options, but this method is infeasible for a large number of time points, n.

- Note that in the tree-based model, the holding value for each node is determined through $e^{-r\Delta t}(P_u \cdot C_u + P_d \cdot C_d)$, where C_u and C_d are the option values corresponding to the upper and lower branches and they already take the possible early exercise in the future into account.

- Tilley (1993) was the first one trying to price American options by simulation. The main idea is to devise a method based on the Monte Carlo simulation to decide early exercise boundary.

Geometric Brownian Motion Simulation

- A stochastic process $S(t)$ is a *geometric* Brownian motion if $\log S_t$ is a Brownian motion with initial value $\log S(0)$; in other words, a geometric Brownian motion is simply an exponential Brownian motion.

- Accordingly all methods for simulating Brownian motion become methods for simulating geometric Brownian motion through exponentiation.

- Geometric Brownian motion is the most fundamental model of the value of a financial asset. More fundamentally, the percentage changes are independent for $t_1 < t_2, ..., t_n$.

$$\frac{S(t_2) - S(t_1)}{S(t_1)}, \frac{S(t_3) - S(t_2)}{S(t_2)}, \dots, \frac{S(t_n) - S(t_{n-1})}{S(t_{n-1})} \tag{3}$$

Geometric Brownian Motion - Basic Properties

- Suppose W is a standard Brownian motion and X satisfies

$$dX(t) = \mu dt + \sigma dW(t),$$

  so that $X \sim BW(\mu, \sigma^2)$.

- If we set $S(t) = S(0)\exp(X(t)) \equiv f(X(t))$, then an application of Itô's formula shows that

$$\begin{aligned} dS(t) &= f'(X(t))dX(t) + \frac{1}{2}f''(X(t))dt \\ &= S(0)exp(X(t))[\mu dt + \sigma dW(t)] + \frac{1}{2}\sigma^2 S(0)exp(X(t))dt \\ &= S(t)(\mu + \frac{1}{2}\sigma^2)dt + S(t)\sigma dW(t). \end{aligned} \quad (4)$$

Geometric Brownian Motion - Basic Properties

- In contrast, a geometric Brownian motion process is often specified through an SDE of the form

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma dW(t). \tag{5}$$

  an expression suggesting a Brownian model of the "instantaneous returns" $\frac{dS(t)}{S(t)}$.
- Comparison of (4) and (5) indicates that the models are inconsistent and reveals an ambiguity in the role of "μ". In (5), $S(t)$ has drift $\mu S(t)$ and implies

$$dlogS(t) = (\mu - \frac{1}{2}\sigma^2)dt + \sigma dW(t). \tag{6}$$

  as can be verified through Itô's formula.

- If we assume a stock with continuous compound rate of r, the algorithm to price a call option by Monte Carlo is clear. We draw a random variable x, from an $N(0,1)$ distribution and compute

$$f(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}x}), \tag{7}$$

  where $f(S) = (S-K)_+$.
- We do this many times and take the average. We then multiply this average by e^{-rT} and we are done.

A Simple Monte Carlo Call Option Pricer in C++

- A Simple Implementation of a Monte Carlo Call Option Pricer
  - SimpleMCMain1.cpp
  - Random1.h
  - Random1.cpp
  - Input:

$$\begin{aligned} T &= 1 \\ S(0) &= 50 \\ K &= 50 \\ \sigma &= 0.30 \\ r &= 0.05 \end{aligned}$$

A Simple Monte Carlo Call Option Pricer in C++

SimpleMCMain1.cpp

```
// SimpleMCMain1.cpp
// requires Random1.cpp

#include <Random1.h>
#include <iostream>
#include <cmath>
using namespace std;

double SimpleMonteCarlo1( double Expiry,
         double Strike,
         double Spot,
         double Vol,
         double r,
         unsigned long NumberOfPaths)
\{
      double variance = Vol*Vol*Expiry;
      double rootVariance = sqrt(variance);
      double itoCorrection = 0.5*variance;

      ...
\}
```

A Simple Monte Carlo Call Option Pricer in C++

- A Simple Implementation of a Monte Carlo Call Option Pricer
- Critiquing the Simple Monte Carlo Routine
  - **Reusability** is the use of existing assets in some form within the software product development process; these assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation.
  - The important attributes of reusability are **clarity** and **elegance** of design.
    - Clarity: if the code is easy to understand, others will tend not to create their own but reuse it.
    - Elegance: if the code is clear but difficult to adapt, others will simply abandon it.
  - **Agility** is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

- Identifying Objects and Classes
  - When you approach a programming problem in an object-oriented language, you no longer ask how the problem will be divided into functions, but how it will be divided into objects. The match between programming objects and real-world objects is the happy result of combining data and functions: The resulting objects offer a revolution in program design.
  - A **class** serves as a plan, or blueprint. It specifies what data and what functions will be included in objects of that class. Defining the class doesn't create any objects, just as the mere existence of data type int doesn't create any variables.
- What Will Classes Buy Us?
  - Classes encapsulate natural financial concepts.
  - Our code becomes clearer - easy to read.
  - Separate interface from implementation.

Key Points

- Options can be priced by risk-neutral expectation.
- Monte Carlo uses the Law of Large Numbers to approximate this risk-neutral expectation.
- Reuse is as much a social issue as a technical one.
- Procedural programs can be hard to extend and reuse.
- Classes allow us to encapsulate concepts which makes reuse and extensibility a lot easier.
- Making classes closely model real-world concepts makes them easier to design and to explain.