Monte-Carlo Simulation for American Options

Analytical Finance I

Authors:
Ehsan Ehsan Ullah
Fayaz Ali
Tawfiqullah Qutbuddin
Teacher: Jan Röman

Date: October 17, 2016

Abstract

The aim of this report is to price American call options using monte-carlo simulation in Python. The authors will consider an American call option and price it using monte-carlo estimation. The authors will plot the Monte-Carlo estimation and compare the result by binomial model.

Introduction

The Monte Carlo method was invented by John von Neumann, Stanislaw Ulam and Nicholas in 1940s while they were working on the project named (Manhattan Project) of nuclear weapon. Monte Carlo Simulation is an extensively used technique for problem solving, which approximates the probability of certain outcomes by running multiple trials using random variables and it was named after the Monte Carlo Casino.

Monte Carlo Simulation

Monte Carlo simulation try to pursue the 'time dependence' of a model for which change, or growth, does not proceed in some conscientiously predefined fashion (e.g. according to Newton's equation of motion) but comparatively in a stochastic process which depends on a sequence of random numbers which is generated during the simulation. [1]

Monte Carlo Simulation in Finance

Financial Analysts use Monte Carlo Simulation quite often to model different scenarios. Below are few scenarios in which commonly Monte Carlo Simulation gets used. [2]

Real Option Analysis

In real option analysis, stochastic models use Monte Carlo Simulation to identify a project net present value.

Portfolio Analysis

Monte Carlo Simulation are used for portfolio evaluation. For each simulation the demeanor of the aspect impacting the component instruments is simulated over time, the value if the instruments is determined, and portfolio value then observed.

Option Analysis

Monte Carlo simulation can be used to generate various alternative price paths for underlying share for options on equity. Similarly, in bond and bond options, the annualized interest rate is dubious variable, which can be simulated using Monte Carlo analysis.

American Options

There are two basic type of options, a put and a call option. A put option gives the holder the right to sell the underlying asset by a certain date for a certain price. In the contract, there is a price decided for exercise that is called strike price or exercise price.

American options can be exercised anytime until maturity (expiration date).

Every option contract has two sides; on one side there is an investor who buys the option (takes the long position), on the other side there is an investor who sells the option (take the short position). This leads to

different payoffs for the buyer and seller of the option. The profit of the seller of the option means the buyer obtained a loss and vice-versa. ‘

Options Payoff:

$\{S_T\}$ is the final price of the underlying asset, K is the strike price)

The payoff from a long position in an American call option is

$$max(S_T - K, 0)$$

This means that the option will be exercised when $S_T > K$.

The payoff from a short position in an American call option is

$$-max(S_T - K, 0)$$

Which is the opposite of the long position.

In the same way the payoff from a long position in an American put option is

$$max(K - S_T, 0)$$

The payoff from a short position in an American put option is

$$-max(K - S_T, 0)$$

There are many models to calculate the price of an option, but in this report the authors will use Monte Carlo simulation to estimate the price of an American option and compare with the binomial model.

Results Comparison

We consider a two year American call option on a non-dividend-paying stock, where the stock price is 100, exercise price is 105, risk free interest rate is 2% per annum, volatility is 0.3.

In order to find the call option price we run the python code and the outcomes for the options are:

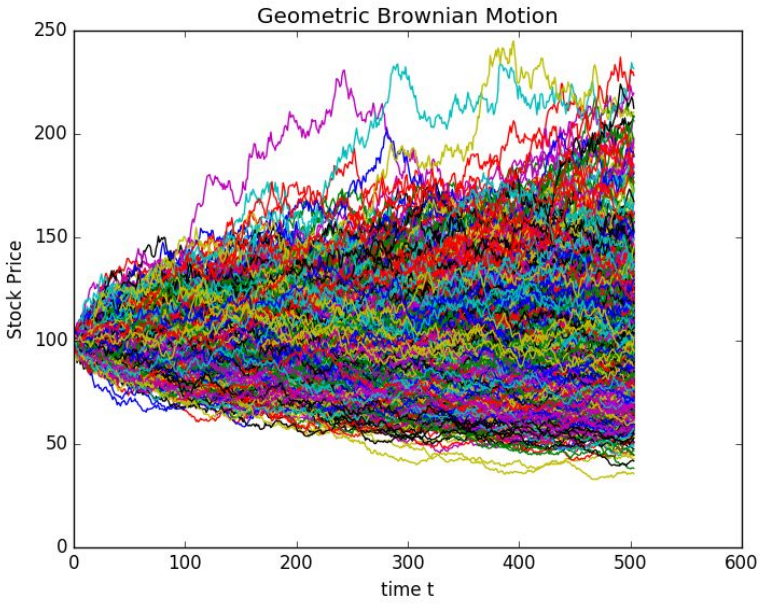

figure 1.1

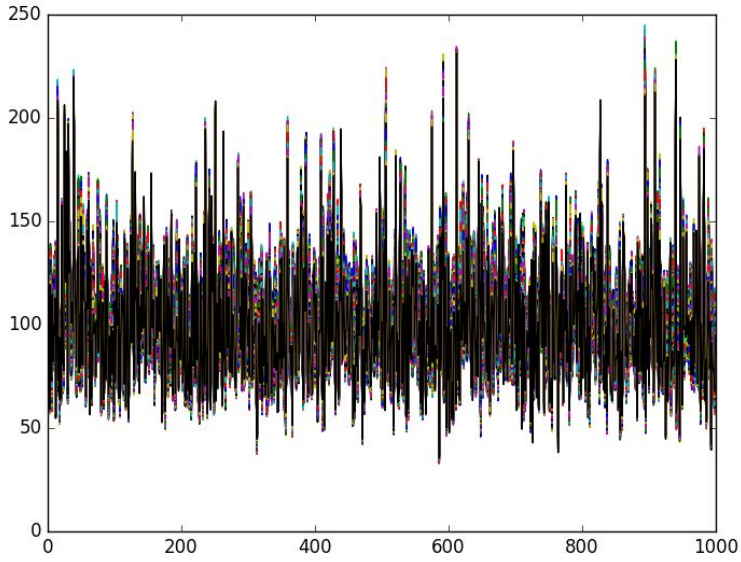

Figure 1.2

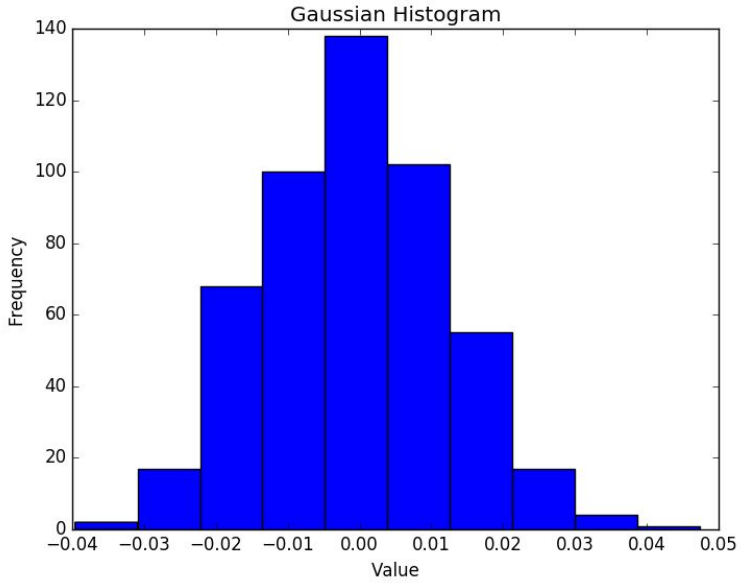

Figure 1.3

We get the call option price 12.8275059715.

To find the price of call option using binomial model we use the following online option pricing calculator:
http://www.hoadley.net/options/binomialtree.aspx?tree=B

Strike price:	105.00	**Volatility:**	30.0 %
Underlying asset price:	100.00	**Interest rate:**	2.00 %
Days to expiration:	504		
Dividend: Enter an amount ($.cc) for discrete dividend, or an annual yield (eg 3.5 = 3.5% pa)	0.00	**Days to ex-dividend:** Enter days for discrete dividend; leave blank or zero for yield	0
Option type:	(•) Call ( ) Put	**Exercise style:**	(•) American ( ) European
No. Tree Steps (1- 150): (max. 10 displayed)	100	Calculate	

Figure 1.4

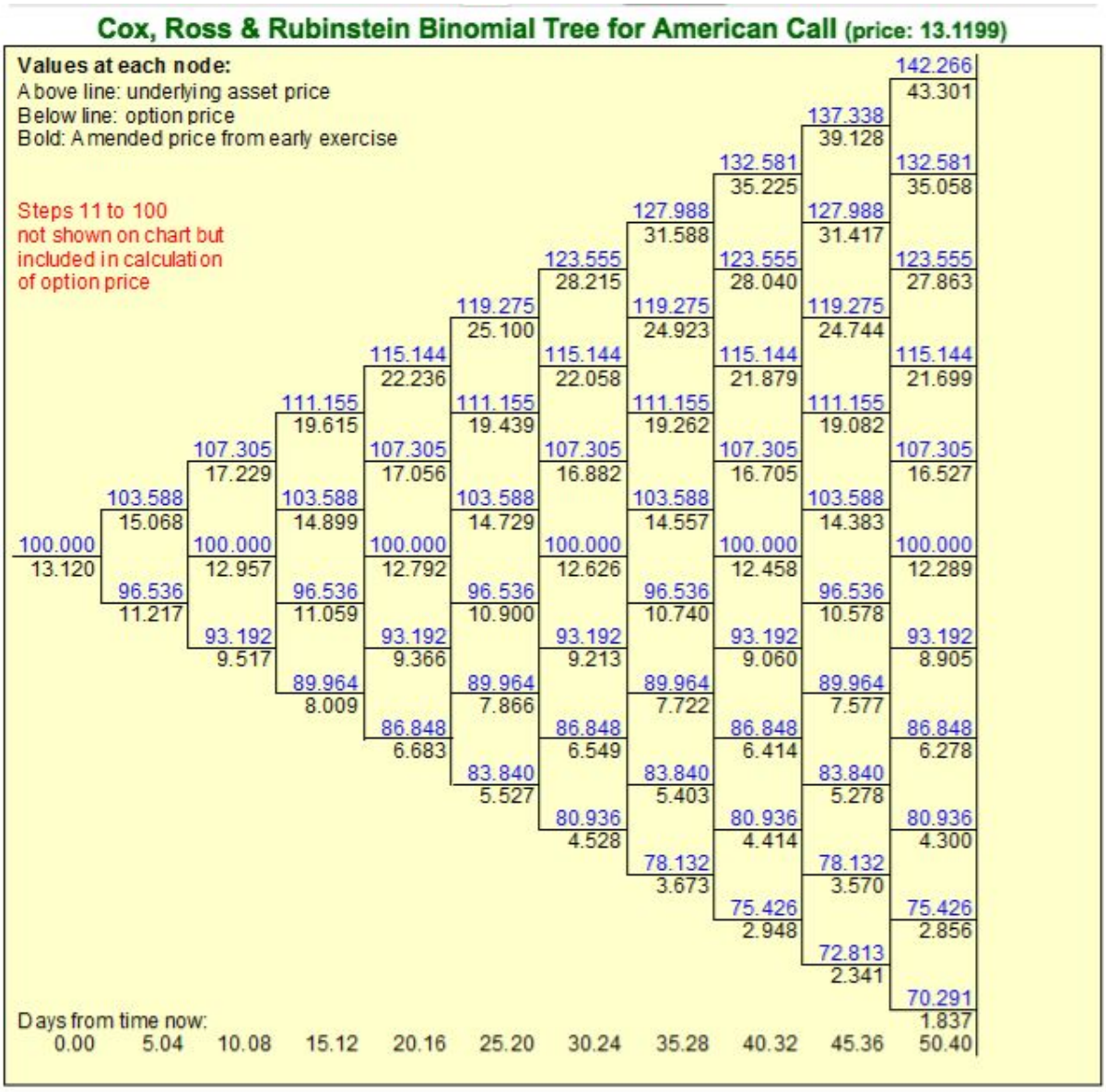

Figure 1.5

As we can see from figure 1.5 the price of American call option is 13.1199 which is close to the result we were expecting.

1. Landau, David P., and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.
2. Raychauduri, S. (2008, December). Introduction to monte carlo simulation. In 2008 winter simulation conference (pp. 91-100) IEEE.

Appendix

Python Code for Monte-Carlo simulation

```
import math
import random
import pylab
from random import gauss
from math import exp, sqrt
import matplotlib
import matplotlib.pyplot as plt
from numpy import *
import numpy as np

# Monte carlo simulation using brownian motion
# n is the number of simulations, T is time to maturity,
# sigma is the volatility , S0 is the initial stock price, K is the strike price.


def monte_carlo_simulation(n,T,sigma,S0,K):
  dt=T*252
  # generate random numbers
  u = random.randn(dt) * sigma / sqrt(dt)

  #lets plot the histogram
  plt.hist(u)
  plt.title("Gaussian Histogram")
  plt.xlabel("Value")
  plt.ylabel("Frequency")
  plt.show()

  # creating a geometric random walk
  z= cumprod(1+random.randn(n,dt)*sigma/sqrt(dt),1)*S0
  plt.plot(z)
  plt.show()
  plt.title("Geometric Brownian Motion")
  plt.xlabel("time t")
  plt.ylabel("Stock Price")


  for i in z: plt.plot(i)
  plt.show()
  plt.hist(z[:,-1],40)
  plt.show()

  # to compute the payoff of the option
  payoffs = (z[:, -1] - 100) * ((z[:, -1] - 100) > 0)
  #print(payoffs)
```

```
    price = mean(payoffs)
    print(price)


# calling the function monte_carlo_simulation(n,T,sigma,S0,K)
monte_carlo_simulation(1000,2,0.3,100,105)
```