Applications of Stochastic Processes in Asset Price Modeling

TJHSST Computer Systems Lab Senior Research Project
2008-2009
Preetam D'Souza

February 26, 2009

Abstract

Stock market forecasting and asset price modeling have recently become important areas in the financial world today. The increasing complexity of the stock market and the lucrative field of investment management has fueled breakthrough developments in mathematical stock price modeling. New financial instruments that rely on an underlying asset's price in the future to determine their current price require accurate methods of stock modeling. One method of mathematical modeling uses random or pseudorandom methods known as stochastic processes to determine an asset's price in the future. This project aims to demonstrate the flexibility and accuracy of these stochastic models by implementing them in code and testing them against empirical data.

Keywords: Stochastic processes, Brownian Motion, Financial Derivatives, Asset Price Modeling

Contents

1 Introduction

1.1 Scope of Study

This project examines stochastic processes to predict stock price movements. Given the current price, volatility, and expected return of an arbitrary stock, several stochastic models exist to predict changes in price. The main model that will be implemented and tested is Geometric Brownian Motion, an adaptation of the standard Brownian Motion process. A standard Brownian Motion model assumes that stock prices themselves follow a random walk process. Geometric Brownian Motion, on the other hand, assumes that stock price returns, not specifically the price, follow a stochastic process. The goal of this project is to extensively test both of these models against empirical data for a single stock (IBM) to determine accuracy. Additionally, this project seeks to develop possible variance reduction techniques that improve the validity of both models.

In order to test the models against empirical data, historical prices for a specific stock (such as IBM) in the past over an arbitrary time period are required. Yahoo! Finance has free stock price data in the past for many large companies and data from this website will be used for this experiment.

1.2 Expected results

I expect that the stochastic modeling techniques will approximate a stock's change in price after running many simulated trials and fine-tuning the model. Over several runs, the model should converge to the actual stock price fluctuations. The results of the project can be shown visually through graphs. For example, historical IBM stock prices can be plotted along with the simulated run of the stock to show the accuracy of the model.

If this project is successful, it could be of use to financial companies that use investment models to determine how to hedge their portfolios against risk. Improved methods of variance reduction to improve accuracy of these models also hold value for derivative pricing. Results from this project could also be used to further develop the implemented algorithms to more accurately model stock prices.

In general, data garnered in this experiment will presumably reveal whether stochastic based models are accurate in predicting complex stock price movements. Calibration testing will also potentially reveal possible improvements to the current models.

1.3 Rationale

Implementation of these stochastic models and extensive testing will lead to results that help to determine

the accuracy and validity of these models when they are used to predict stock prices. Several financial firms tend to use these models to price their complex financial instruments and thus require a high degree of certainty that their models are correct to prevent risk and potential losses. Invalid models used to price these instruments can lead to potential mishaps for the entire economy; this can easily be seen in the housing market collapse and subsequent chaos on Wall Street where firms did not know the correct value of their mortgage-backed securities. Eliminating these models and developing improvements to them can lead to greater fundamental knowledge behind human behavior and more accurate asset pricing methods.

2 Theory

Stochastic processes are can be represented with stochastic differential equations (SDEs) that describe changes in different quantities.

Let S be the stock price, μ the drift rate (or mean), σ the volatility (or variance) of the stock, and let dZ represent a Wiener Process.

$$dZ = \phi\sqrt{dt}$$

where ϕ is drawn from a normal distribution $N \sim (0, 1)$. The SDE for Geometric Brownian Motion is given by:

$$\frac{dS}{S} = \mu dt + \sigma dZ$$

Here $\frac{dS}{S}$ represents the return on the stock. Multiplying by S to both sides of the equation one obtains:

$$dS = \mu S dt + \sigma S dZ$$

This shows that the stock price cannot change once $S = 0$, which is a requirement for this model to accurately represent stock prices.

3 Procedures and Methodology

3.1 Structure

After research of the theory behind these models, the actual models were implemented in code. Java was chosen as the prime programming language for all phases of this project. First, an RSS based class was created to retrieve real-time stock price data for an given stock from a free financial website. This will be used in later development to predict present movements of a stock price. A main statistics class was also created to act as a simple resource for calculating the mean, variance, and standard deviation for a list of a stock's historical prices over an arbitrary time period.

The main model class is responsible for data parsing and simulation. This class reads in historical price data and utilizes the aforementioned statistical convenience class to calculate inputs to the model. Once these are determined, the simulation process begins. Currently, a Geomet-

ric Brownian Motion model is being used, but this class can easily be adopted to other models as long as they follow the same convention. These stochastic models were implemented using a discrete iterative algorithm to approximate the continuous time forms of the theoretical models. During the simulation, price changes over the given trading period (usually 1 day) are printed out to a file formatted to easily be plotted with Gnuplot. This model supports simultaneous simulations so that several different sample paths for a stock price can be plotted on the same graph.

Stock prices over one year can be intuitively plotted on graphs to display the price fluctuations with a smooth curve drawn through the discrete points. Charts of key prices over a year can also be provided to demonstrate potential variations in the model from the empirical data.

3.2 Model Accuracy

The accuracy of the model with respect to empirical data can be estimated by calculating the Root Mean Squared Deviation (RMSD) of the data sets. This is a measure of the average of the squared errors between the model and the empirical data. Let S_i represent the empirical price and $\hat{S}_i$ represent the simulated price. The RMSD is then defined as:

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{S}_i - S_i)^2}$$

A large value indicates large deviations between the empirical data and the model, while a value close to zero represents a good fit.

4 Current Results

My model calculates simulated price changes for a stock and outputs them to a file that can be easily plotted along with the empirical price graph. Figures 1 and 2 on the next page demonstrate two different simulation runs of the Geometric Brownian model. Each run is plotted with the empirical price curve and three simulation paths. In Figure 1, notice how one simulation path runs far off the drift line while the other two move with the empirical data. In Figure 2, all three simulations hug the drift line and closely approximate the historical price changes. Different sample runs of this model can produce dramatically different results, although they can all be considered valid paths for a stock price to take. The next step for analysis will involve calculating the RMSD between the historical and simulated price sets.

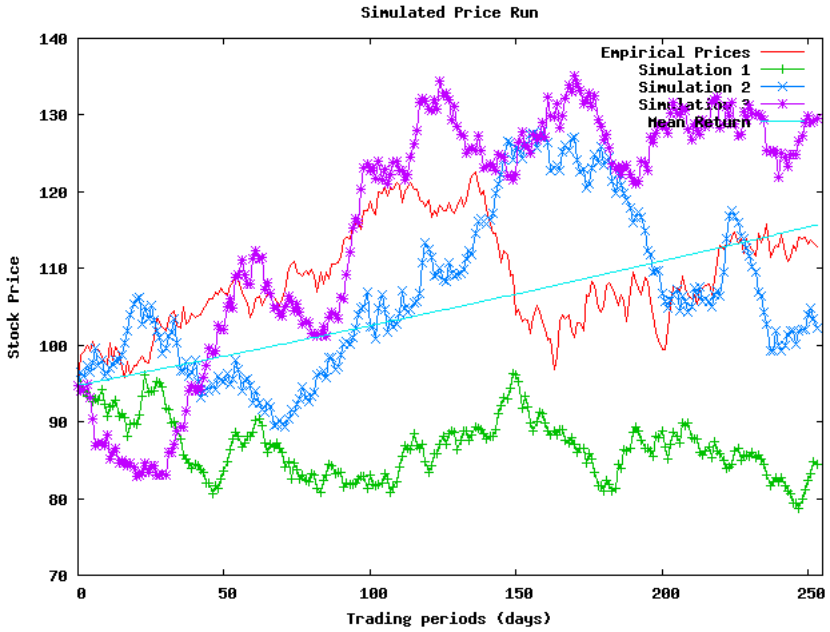

Figure 1: Simulated Price Run 1

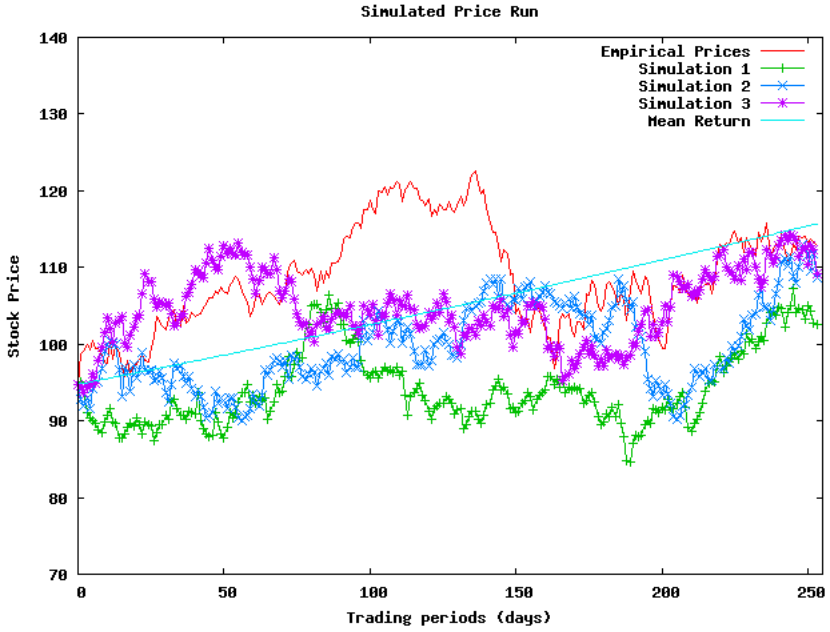

Figure 2: Simulated Price Run 2

A Source code

A.1 Main model class

```
// Preetam D'Souza
// 2.17.09

/* Input: Historical stock prices in .csv format
 * Process: Parse historical price data and obtain useful statistics such as
 *          mean and variance of stock prices. Simlulate the prices using
 *          the disrete time form of the Geometric Brownian process.
 * Output: Simulated stock prices over the set time frame.
 *         Format: <Time Step> <empirical> <sim_1> <sim_2> ... <sim_n> <avg> <mean>
 */

import java.util.*;
import java.io.*;


public class Model1
{
        static final int NUM = 3; // number of simulations

        public static void main(String[] args) throws Exception
        {
                double drift;
                double volatility;
                double dt;      // time step
                double cur_p; // initial price

                //Input Parsing
                String[] months = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
                ArrayList<Double> prices = new ArrayList<Double>();

                System.out.print("\nParsing Historical Prices...");
                PrintWriter pout = new PrintWriter(new BufferedWriter(new FileWriter("IBM.txt")));
                PrintWriter sout = new PrintWriter(new BufferedWriter(new FileWriter("simulation_avg.txt")));
                Scanner sc = new Scanner(new File("IBM2.csv"));
                sc.nextLine();
                pout.printf("#Date\t \tOpen \tHigh \tLow \tClose\n");
                while(sc.hasNext()) {
                        String[] s = sc.nextLine().split("\\,");
                        pout.printf("%s-%s-%s \t%s \t%s \t%s \t%s\n", s[0].substring(8,10),
                                        months[Integer.parseInt(s[0].substring(5,7))-1],
                                        s[0].substring(2,4), s[1], s[2], s[3], s[4]);
                        prices.add(Double.parseDouble(s[1]));
                }
                System.out.println("Done!\n");

                // Statistics calculations of drift rate and volatility
                System.out.print("Calculating input data statistics...");
                double[] p = new double[prices.size()]; // prices (recent to latest order)
                double[] r_norm = new double[prices.size()-1]; // normal returns
                double[] r_log  = new double[prices.size()-1]; // log returns
                for(int i=0;i<p.length;i++) p[i] = prices.get(i);
                for(int i=0;i<r_log.length;i++) {
                        r_norm[i] = (p[i]/p[i+1])-1.0;
                        r_log[i] = Math.log(p[i]/p[i+1]);
                }

                dt = (1.0/p.length); // each time step is one day
                Stat s = new Stat();
                drift = s.mean(r_norm);
                volatility = s.sdev(r_log)*Math.sqrt(p.length); // annualized volatility
                cur_p = p[p.length-1];
                System.out.printf("Done!\nDrift: %f\tAnnualized Volatility: %f\n\n", drift, volatility);

                //Simulation
                double dS;
                double mean_line = cur_p;
                double avg = 0;

                System.out.print("Running Simulation...");
```

```
                sout.printf("0 %.5f ", p[p.length-1]);
                GeoBrownian[] g = new GeoBrownian[NUM];
                for(int i=0;i<g.length;i++) {
                        g[i] = new GeoBrownian(cur_p,drift,volatility);
                        avg += g[i].getPrice();
                        sout.printf("%.5f ", g[i].getPrice());
                }
                sout.printf("%.5f %.5f\n", avg/((double)NUM), mean_line);


                mean_line += (mean_line*drift);
                for(int t=1;t<254;t++){
                        avg = 0;
                        sout.printf("%d %.5f ", t, p[p.length-t-1]);
                        for(int i=0;i<g.length;i++) {
                                g[i].process(dt);
                                avg += g[i].getPrice();
                                sout.printf("%.5f ", g[i].getPrice());
                        }
                        sout.printf("%.5f %.5f\n", avg/((double)NUM), mean_line);
                        mean_line += (mean_line*drift);
                }
                System.out.println("Done!\n");


                sout.printf("#Mean: %f\n#Volatility: %f\n", drift, volatility);
                sc.close();
                pout.close();
                sout.close();
        }
}
```

A.2 Geometric Brownian class

```
// Preetam D'Souza

/*
 * This class simulated discrete changes in stock price
 * using a geometric brownian process.
 */

import java.util.*;

public class GeoBrownian
{
        private double cur_price;
        private double mu;
        private double sigma;

        public GeoBrownian(double price, double drift, double volatility)
        {
                cur_price = price;
                mu = drift;
                sigma = volatility;
        }

        public double getPrice()
        { return cur_price; }

        public void setPrice(double price)
        { cur_price = price; }

        public double getDrift()
        { return mu; }

        public void setDrift(double drift)
        { mu = drift; }

        public double getVolatility()
        { return sigma; }

        public void setVolatility(double volatility)
        { sigma = volatility; }

        // price - current stock price
```

```
        // mu - drift parameter
        // sigma - stock volatility
        // dt - time step (usually 1/trading periods)

        public double process(double dt)
        {
                double phi;
                double dS, dZ;

                Random r = new Random();
                phi = r.nextGaussian();
                dZ = phi*Math.sqrt(dt);
                dS = cur_price*(mu*dt + sigma*dZ);
                cur_price += dS;
                return dS;
        }
}
```

A.3 Statistics class

```
// Preetam D'Souza
// *Based on Philip Barker's DataDisperion package*

/* Convenience class to implement statistics
 * algorithms for calculating the mean, variance
 * and standard deviation for a data set.
 */

// Stat.java

import java.util.*;

public class Stat
{
        public Stat() {}

        public static double mean(double[] x)
        {
                double total=0.0;
                for(int i=0;i<x.length;i++)
                        total += x[i];
                return total/x.length;
        }

        public static double variance(double v1[])
        {
                double sumd=0.0;
                double total=0.0;
                for(int i=0;i<v1.length;i++) {
                        total += v1[i];
                        sumd += Math.pow(v1[i],2);
                }
                return (sumd-(total*(total/v1.length)))/((v1.length)-1);
        }

        public static double sdev(double v1[])
        {
                return Math.sqrt(variance(v1));
        }
}
```

References

[1] Barker, Philip. Java Methods for Financial Engineering, Springer Publishing, May 2007

[2] Charnes, John. "Using Simulation for Option Pricing" School of Business, The University of Kansas.

[3] Chance, Don. "Essays in Derivatives", Wiley Publishing, August 1998

[4] Forsyth, Peter. "An Introduction to Computational Finance without Agonizing Pain", School of Computer Science, University of Waterloo.

[5] Straja, Sorin. "Stochastic Modeling of Stock Prices", Montgomery Investment Technology, Inc.