

Parallel Programming (CS403)
Project-1
Simulation of Brownian Movement of a Stock
Price

Akhilesh Kumar (201351009)
Gaurav Tolani 201352021

September 15, 2016

Contents

| | | |
|----------|--|----------|
| 1 | Problem Statement | 2 |
| 1.1 | What is Brownian Movement? | 2 |
| 1.2 | What is Monte Carlo Simulation? | 2 |
| 1.3 | Geometric Brownian Motion | 2 |
| 2 | Algorithm Used | 3 |
| 3 | Output | 3 |
| 3.1 | Language Used | 3 |
| 3.2 | Outputs(Graphs and execution time) | 3 |
| 4 | Conclusion | 9 |
| 5 | Future/Concluding Work | 9 |
| 6 | Appendix | 9 |
| 6.1 | How to install R | 9 |
| 6.2 | How to run the code | 9 |

1 Problem Statement

1.1 What is Brownian Movement?

Standard definition of Brownian motion is:

Brownian motion is the random motion of particles suspended in a fluid (a liquid or a gas) resulting from their collision with the fast-moving atoms or molecules in the gas or liquid.

Dwelling into probability and statistics, Brownian motion is among the simplest of the continuous-time stochastic (or probabilistic) processes.

In mathematics, the Wiener process is a continuous-time stochastic process named in honor of Norbert Wiener. It is often called standard Brownian Motion.

1.2 What is Monte Carlo Simulation?

Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle.

A Monte Carlo simulation is an attempt to predict the future many times over. At the end of the simulation, thousands or millions of "random trials" produce a distribution of outcomes that can be analyzed.

1.3 Geometric Brownian Motion

A geometric Brownian motion (GBM) (also known as exponential Brownian motion) is a continuous-time stochastic process in which the logarithm of the randomly varying quantity follows a Brownian motion (also called a Wiener process) with drift. It is an important example of stochastic processes satisfying a stochastic differential equation (SDE); in particular, it is used in mathematical finance to model stock prices in the Black-Scholes model.

The geometric Brownian motion (GBM) is the most basic processes in financial modelling.

The formula for GBM is found below, where "S" is the stock price, " μ " (the Greek mu) is the expected return, " σ " (Greek sigma) is the standard deviation of returns, "t" is time, and " ε " (Greek epsilon) is the random variable:

$$\frac{\Delta S}{S} = \mu t + \sigma \varepsilon \sqrt{\Delta t} \quad (1)$$

If we change and rearrange the formula and take log of the above equation on both sides, the final equation which we will get is:

$$S(t) = S(0)e^{(\mu - \sigma^2/2)t + \sigma W(t)} \quad (2)$$

where, $W(t) = \text{Brownian Motion}$, $\sqrt{t}z(t)$, and

$z(t)$ = normal random variable b/w 0 and 1 with mean=0 and variance=1

In this problem, we will try to simulate the Brownian motion of a stock price for a given time using Monte Carlo simulations for different number of iterations and parallelize the algorithm to see the performance change.

2 Algorithm Used

An algorithm for simulating the stock price at time t , given that current price at time $t=0$ is S_0 is as follows:

1. Generate random variable $z \sim N(0,1)$
2. Set $\mu_t = (\mu - \sigma^2/2)t$ and $\sigma_t = \sigma t^{0.5}$
3. Set $S_t = S_0 \times e^{\mu_t + \sigma_t z}$

3 Output

3.1 Language Used

Due to time constraint, for the time being, we have implemented the serial code on the language R. R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

Because of the some inbuilt features which R provide (eg. inbuilt function for generating random variable and graph plotting), the effort for making code was considerably reduced.

3.2 Outputs(Graphs and execution time)

Given the parameters,
 S_0 , initial price of stock= 10,
 μ , expected return=13%,
 σ , standard deviation of returns=15%,
time, $t=100$

1. For 100 simulations,

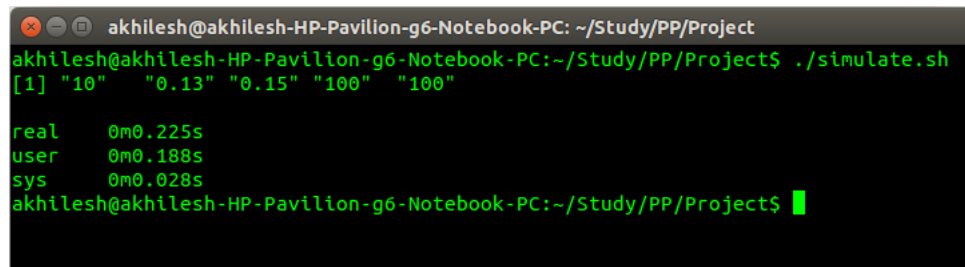
A terminal window screenshot showing the execution of a script. The prompt is 'akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC: ~/Study/PP/Project'. The command './simulate.sh' is executed, returning '[1] "10" "0.13" "0.15" "100" "100"'. Below this, the execution time is displayed: 'real 0m0.225s', 'user 0m0.188s', and 'sys 0m0.028s'. The prompt returns to 'akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project\$'.

Figure 1: Execution time for 100 simulations

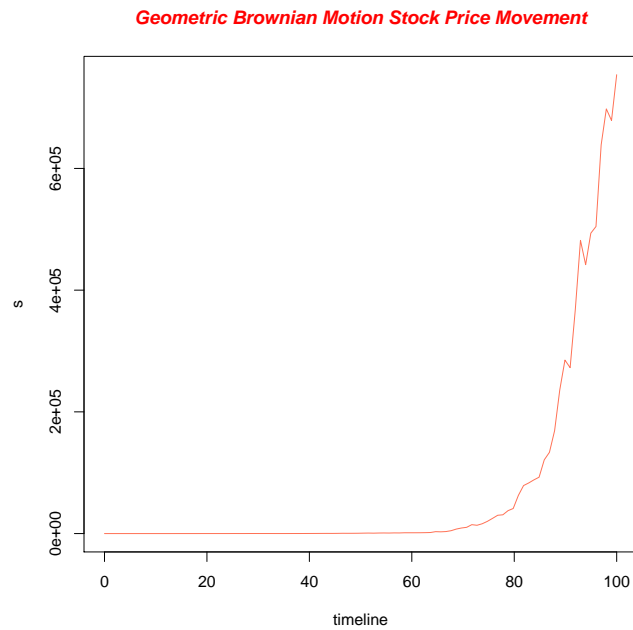


Figure 2: Brownian motion graph for 100 simulations

2. For 1000 simulations,

```
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC: ~/Study/PP/Project
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ ./simulate.sh
[1] "10" "0.13" "0.15" "100" "1000"

real    0m0.255s
user    0m0.208s
sys     0m0.032s
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ █
```

Figure 3: Execution time for 1000 simulations

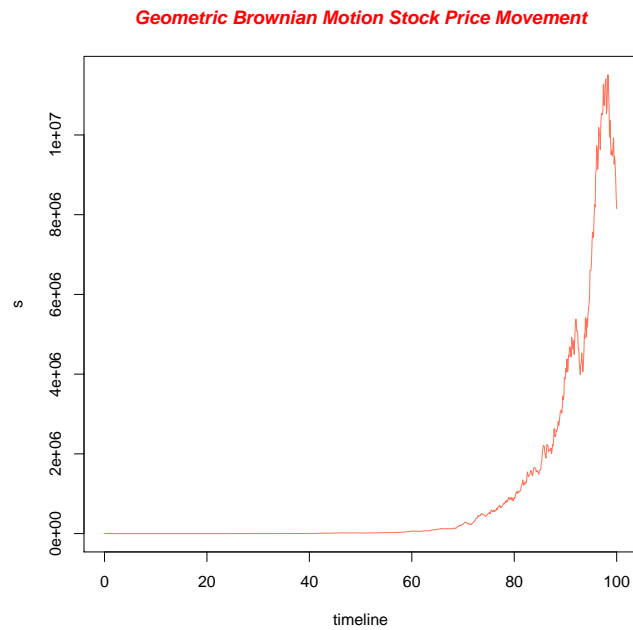


Figure 4: Brownian motion graph for 1000 simulations

3. For 10000 simulations,

```
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC: ~/Study/PP/Project
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ ./simulate.sh
[1] "10"      "0.13"    "0.15"    "100"     "10000"

real    0m0.405s
user    0m0.356s
sys     0m0.036s
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ █
```

Figure 5: Execution time for 10000 simulations

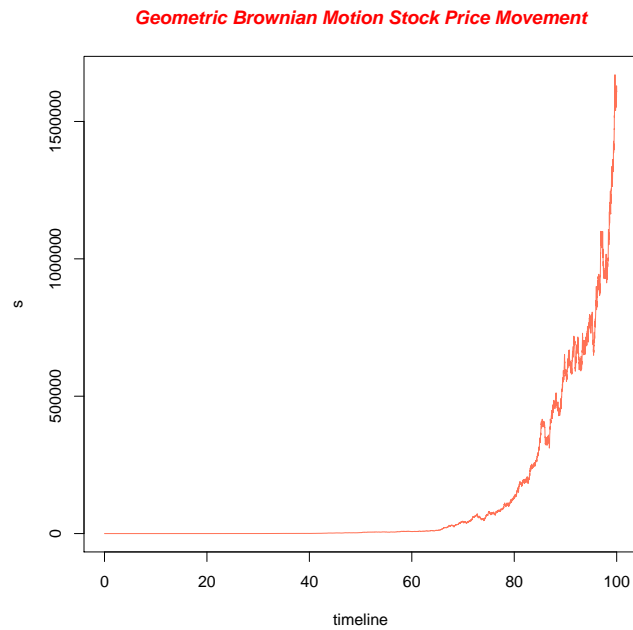


Figure 6: Brownian motion graph for 10000 simulations

4. For 100000 simulations,

```
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC: ~/Study/PP/Project
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ ./simulate.sh
[1] "10"      "0.13"    "0.15"    "100"     "100000"

real    0m1.937s
user    0m1.880s
sys     0m0.040s
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$
```

Figure 7: Execution time for 100000 simulations

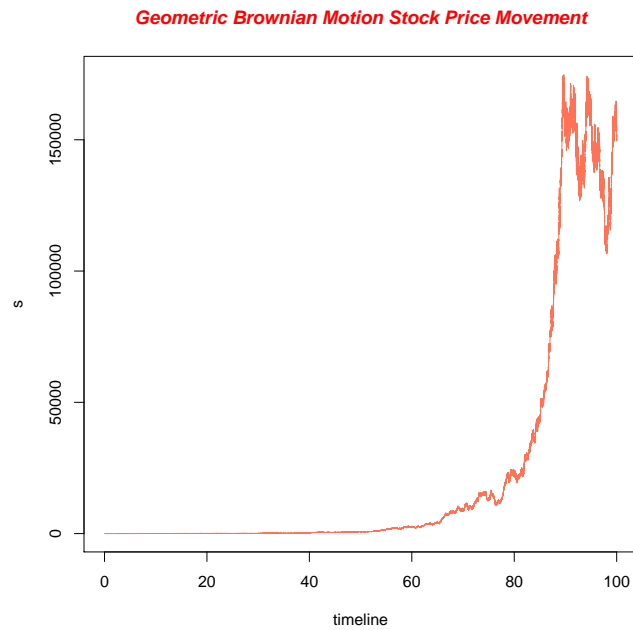


Figure 8: Brownian motion graph for 100000 simulations

5. For 1000000 simulations,

```
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC: ~/Study/PP/Project
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$ ./simulate.sh
[1] "10"      "0.13"     "0.15"     "100"      "1000000"

real    0m15.607s
user    0m15.436s
sys     0m0.140s
akhilesh@akhilesh-HP-Pavilion-g6-Notebook-PC:~/Study/PP/Project$
```

Figure 9: Execution time for 1000000 simulations

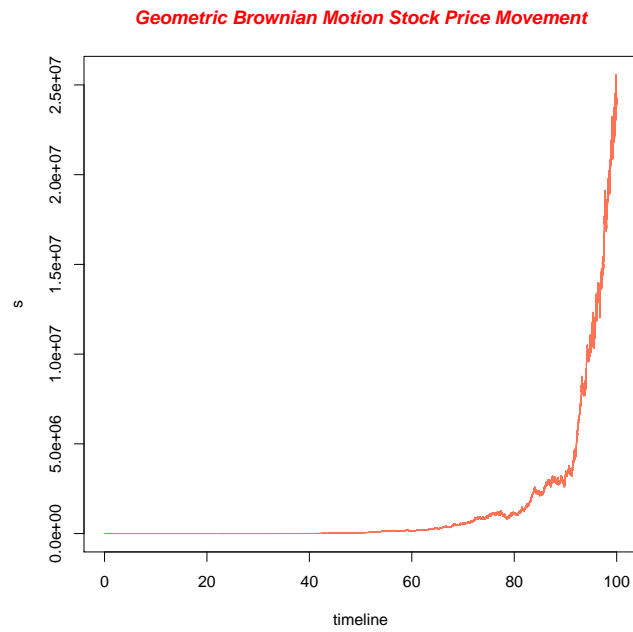


Figure 10: Brownian motion graph for 1000000 simulations

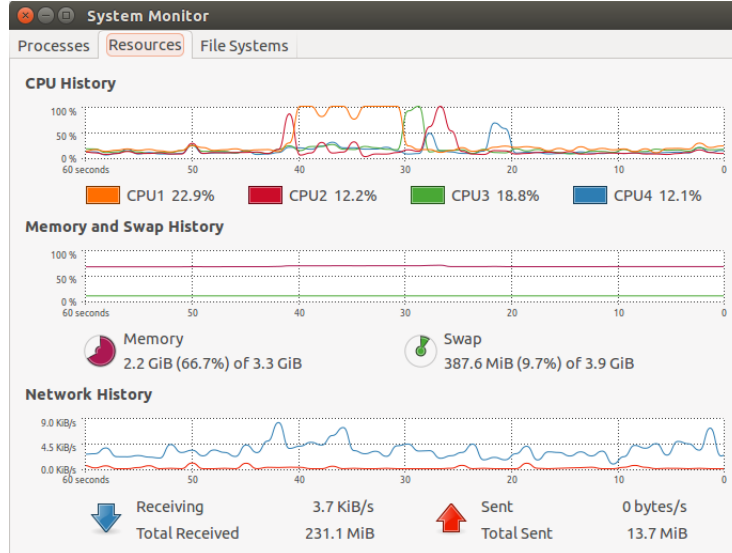


Figure 11: System Monitor for 1000000 simulations

4 Conclusion

We observe that with increasing number of simulations, execution time also increase. Also, we also observe that while executing the code for 10,00,000 simulations, one or two cores reach their 100% processing power. This shows that other cores are idle and their execution capabilities are not utilized by the code. So, the code written is not optimised to utilize the full potential of the machine.

5 Future/Concluding Work

As of now, the code written is in R programming language. A serial code for this algorithm will be written in C and then will be evaluated for performance. After that, using parallel programming APIs like openMP, we will try to parallelize the code and compute the speedup. We can also compare whether for this kind of computational heavy algorithm, parallel code in C is better or serial code in R is better, performance wise.

6 Appendix

6.1 How to install R

Run the following commands in your machine(OS: Ubuntu 14.04):

1. `sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu trusty/"`
`>> /etc/apt/sources.list'`
2. `gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9`
3. `gpg -a --export E084DAB9 -- sudo apt-key add -`
4. `sudo apt-get update`
5. `sudo apt-get -y install r-base`

6.2 How to run the code

Attached are two files:

- GBMStock.R
- simulate.sh

In the simulate.sh file, there are 5 args which are required by the R Script to run. The args stands for:

GBMStock.R "Initial Price of Stock" "mu" "sigma" "Time" "Iterations"

In the terminal, run the **simulate.sh** file by manipulating the arguments and the output will be shown there. In the directory where the R Script is present, a new file "Rplots.pdf" will be created with the corresponding graph in it.