**Task 1**

In this task, simulation of Geometric Brownian Motion (GBM) and mean reversal process are applied in Part 1 and Part 2, respectively. Due to the homogeneity of the simulation, the general simulation techniques are used in these two parts.

In this task, it is modeled to 1,000 paths when generating result of Part 1 and Part 2. This is to use the sufficiently large amount of data in computing certain parameters to gain higher accuracy. The usage of random number is required in generating the motion. In this case, pylab.randn is used in generating a set of normalized random numbers. pylab is a library defined in Python. All values generated by using the function above is then divided by , due to the usage of Central Limit Theorem in this task. The generations of random number takes place before the generation of GBM and mean reversal process takes place. Therefore, the generated random number is denoted by dB. In order to get a standard Brownian Motion, dB.cumsum() is used to cumulate the random numbers generated in dB to a series of numbers, B which denotes the current point after it goes up and goes down for given t. The generation of the paths will be described in each part, respectively.

Since it requires only 5 realization of GBM and mean reversal process, the first five paths are being selected and plotted into the graph generated by the two parts.

**Part 1**

Geometric Brownian Motion (GBM) follows the stochastic differential equation listed below,

where the solution of this stochastic differential equation is:

This solution can be expressed in an expected function and variance function, *E(X)* and *Var(X)*, where *X* is a random variable. The solution is very similar to moment generating function of a normal random variable, denote this by *X*, ,

apply some modification by letting *X* be a stochastic random variable, the moment generating function of *X(t)*, where ,

therefore:

can be easily proven from the moment generating function mentioned above. (Sigman, 2006) By applying the formulas mentioned above, with these parameters:

Since the solution of the stochastic differential equation is known, instead of using iterations, the actual solution is presented.

This part is generated from Python.

Computation of mean, variance, P[S(3) > 39] and E[S(3)|S(3) > 39] from the simulation:

Mean, E[S(3)] = 39.3014866876

Variance, Var[S(3)] = 0.0693010343703

P[S(3) > 39] = 0.876

E[S(3)|S(3) > 39] = 39301.4866876

Theoretical variance:

Variance, Var[S(3)] = 0.205985316648

Computation of mean, is theoretically taken from the end of each path of the generated GBM and take the average of it. The mean from the simulation is approximately equal to the mean calculated using the above formula. This is due to the difference of the simulation with the exact formula as it is always depends on the dB generated from the initial stage. At the same time, the variance, also generated from the last column from the variable. However, due to the large number of paths generated (1,000 paths), the variance shall converge to the actual variance. The reason that the variance is less than the actual variance is due to the variance computed is based on the final values obtained only but the actual variance is computed based on t and other variables.

The computation of probability, varies depending on the amount of the last column of S generated each time. In the code, the usage of Boolean data type is to check the condition whether the element of S is satisfying the condition of more than 39 or not. The pylab.sum() function helps in adding up the total amount of the condition when it is true. Then, the definition of probability is used in computing the value. The computation of conditional expectation, is not quite true based on the computation. The equations below show the reason on why the conditional expectation is the sum of all last column of S in the code.

Since there are 1,000 paths of GBM generated, therefore, it is equal to the mean times 1,000 (the amount of paths). The expected value is using summation notation as the amount is countable.

For the convenience of reference, Appendix 1 shows a single run of the Python file “gbm.py”.

**Part 2**

The mean reversal process is a stochastic differential equation that can be reverted to a solution that has this pattern of equation that follows Ornstein-Uehlenbeck process.

Differential equation:

Solution:

However the differential equation that is used to simulate in this task is hard to apply this. Therefore, numerical approach is possible in Python. From this stochastic differential equation, apply some modification to make iteration possible.

Let

This is then expressed in Python as:

for c in range(n):

R[:,c+1] = R[:,c] + (0.064 - R[:,c]) \* dt + 0.27 \* R[:,c] \* (B[:,c+1] - B[:,c])

dt is defined to be the total period divided by number of parts. Since is always equal to dt, the code is using dt in the iterations. Again, B is generated from standard Brownian motion as mentioned in the beginning of the description of this task.

The mean, is calculated using the basic definition of expected value, by summing up the last column of R in the matrix and divided by the number of paths generated. The generation of the probability is similar to the way the probability is generated in the previous part.

Appendix 2 shows a single run of the Python file of “mr.py”.

**Task 2**

**Part 1**

There are thirty (30) stocks which are the components of the FTSE Bursa Malaysia KLCI Index. Below is the table of stocks of components.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stock Name** | **Stock Code** | **Stock Sector** | **Weightage in FTSE KLCI** | **PE Ratio** | **Net Market Capital (Billion)** |
| AMMB Holdings | 1015 | Finance | 0.016835 | 8.85 | 16.970 |
| Astro Malaysia Holdings | 6399 | Service | 0.015636 | 28.19 | 15.761 |
| Axiata Group | 6888 | Service | 0.054907 | 24.51 | 55.347 |
| British American Tobacco (Malaysia) | 4162 | Consumer Products | 0.018298 | 20.13 | 18.445 |
| CIMB Group Holdings | 1023 | Finance | 0.046323 | 17.82 | 46.694 |
| Digi.com | 6947 | IPC | 0.041498 | 21.02 | 41.830 |
| MISC | 3816 | Service | 0.034984 | 16.00 | 35.264 |
| Genting | 3182 | Service | 0.030710 | 19.13 | 30.956 |
| Genting Malaysia | 4715 | Service | 0.025036 | 21.17 | 25.237 |
| Hong Leong Bank | 5819 | Finance | 0.025476 | 11.91 | 25.680 |
| Hong Leong Financial | 1082 | Finance | 0.016126 | 9.97 | 16.255 |
| IHH Healthcare | 5225 | Service | 0.048370 | 63.63 | 48.758 |
| IOI | 1961 | Plantation | 0.027041 | 65.53 | 27.258 |
| Westports Holdings | 5246 | Service | 0.013802 | 26.58 | 13.913 |
| Kuala Lumpur Kepong | 2445 | Plantation | 0.023849 | 29.28 | 24.040 |
| Malayan Banking | 1155 | Finance | 0.087340 | 12.92 | 88.040 |
| Maxis | 6012 | Service | 0.049016 | 30.14 | 49.409 |
| PETRONAS Chemicals Group | 5183 | Industrial Products | 0.050237 | 21.82 | 50.640 |
| Petronas Dagangan | 5681 | Service | 0.020292 | 37.02 | 20.455 |
| Petronas Gas | 6033 | Industrial Products | 0.042401 | 22.79 | 42.741 |
| PPB Group | 4065 | Consumer Products | 0.018088 | 18.13 | 18.233 |
| Public Bank | 1295 | Finance | 0.073175 | 15.78 | 73.761 |
| RHB Capital | 1066 | Finance | 0.018977 | 9.27 | 19.129 |
| SapuraKencana Petroleum | 5218 | Service | 0.014207 | 12.10 | 14.321 |
| Sime Darby | 4197 | Service | 0.052252 | 20.93 | 52.671 |
| Telekom Malaysia | 4863 | Service | 0.024717 | 33.22 | 24.915 |
| Tenaga Nasional | 5347 | Service | 0.068753 | 9.15 | 69.304 |
| UMW Holdings | 4588 | Consumer Products | 0.011892 | 20.41 | 11.987 |
| YTL Corp | 4677 | Service | 0.017133 | 15.56 | 17.270 |
| KLCC | 5235SS | REITS | 0.012627 | 13.65 | 12.728 |

**Table 1**: Components of FTSE Bursa Malaysia KLCI with the Stock Code, Sectors, Weightage in the Index, P/E Ratio and Net Market Capital in billions of Ringgit. Information adapted from [http://www.malaysiastock.biz/Listed-Companies.aspx on July 26](http://www.malaysiastock.biz/Listed-Companies.aspx%20on%20July%2026), 2015.

**Part 2**

In this part, Kuala Lumpur Kepong (KLK) (2445.KL) is chosen as the stock for the calculation of 5-day moving average as well as the correlation coefficient between itself and FTSE Bursa Malaysia KLCI Index.

In this part, instead of extracting the 5-day moving average out of the data type of DataFrame, another column for calculating 5-day moving average is added to that DataFrame named data. Besides that, implementation of pandas in Python allows the calculation of moving average, in Python is called rolling\_mean, in which it accepts inputs as a column of data from data type DataFrame and an integer that gives the amount of “rolls” for each iteration, which is meant for the amount of input requires for the moving average. For example, if it requires a 10-day moving average, the input receives ten (10) elements in the array and divides it by 10. This can be calculated faster than using loops in the computation.

When computing correlation coefficient of the data, an array of strings is set up to denote the requirement of the data downloader to download “^KLSE” and “2445.KL” on the values of closing only for the specified dates. After receiving the data, using the closing.corr() to compute the correlation matrix. closing is an array with data type DataFrame and corr asks Python to compute the correlation matrix of the data that closing had received. The closing.corr(), which is denoted by correlation in “download\_data.py”, gives a symmetrical matrix. If an input of correlation is typed after the computation, observation will be as follows:

2445.KL ^KLSE

2445.KL 1.000000 0.318993

^KLSE 0.318993 1.000000

It is logical to obtain this result. As 2445.KL is identical to 2445.KL, as well as ^KLSE is identical to ^KLSE, both of their correlation with respect to themselves are always perfectly positively correlated, therefore the diagonal part is always 1. When goes to the non-diagonal part, the correlation between 2445.KL and ^KLSE is 0.318993, which is the same when they swap their positions. This means that 2445.KL and FTSE Bursa Malaysia KLCI Index are weak-positively correlated with respect to each other. For example, if there is an increase in the stock price of KLK, there will be generally a slight increase in FTSE Bursa Malaysia KLCI Index. In taking out the value from the matrix, the “slice” technique is used. The printer slices the second column out of the correlation matrix and use printer as the basic of printing the correlation coefficient to the display screen.

The way of creating the graph is different from the way used in the previous task. The call of the storage DataFrame is used instead of asking pylab in doing so. data[‘5-day Moving Average’].plot() plots the values on the 5-day Moving Average column to the graph with the index (horizontal axis) as date labeled in the data with default features. The usage of title and color is to write a title on the graph and to change the colour from its default settings, respectively.

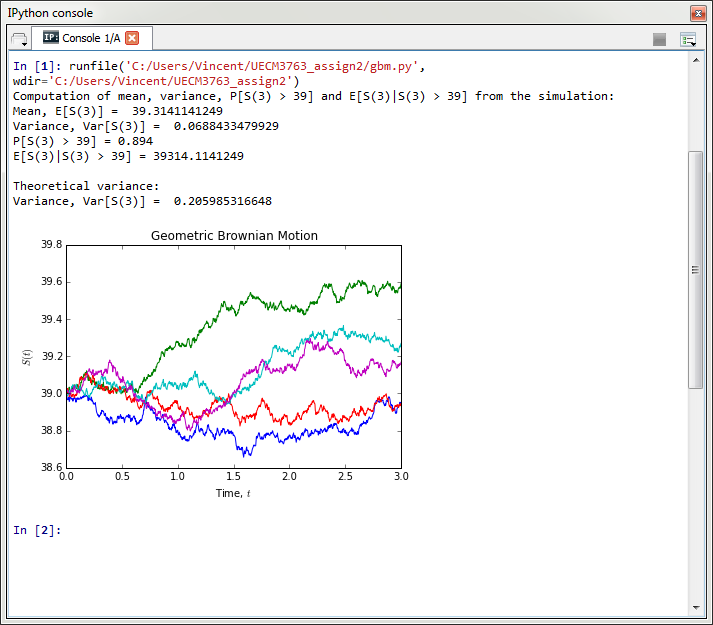
Please refer to Appendix 3 for the 5-day moving average plot by the Python file “download\_data.py”.

**References**

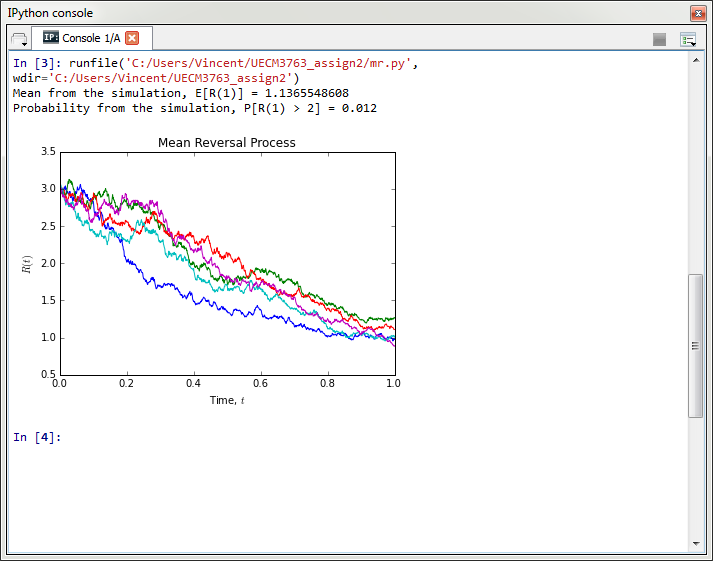
Malaysiastock.biz. (n.d.). *KLSE listing companies.* Retrieved July 26, 2015, from <http://www.malaysiastock.biz/Listed-Companies.aspx>

Sigman, K. (2006). Geometric Brownian motion. *Back to our study to geometric BM.* Columbia University: New York.Retrieved July 25, 2015, from <http://www.columbia.edu/~ks20/FE-Notes/4700-07-Notes-GBM.pdf>

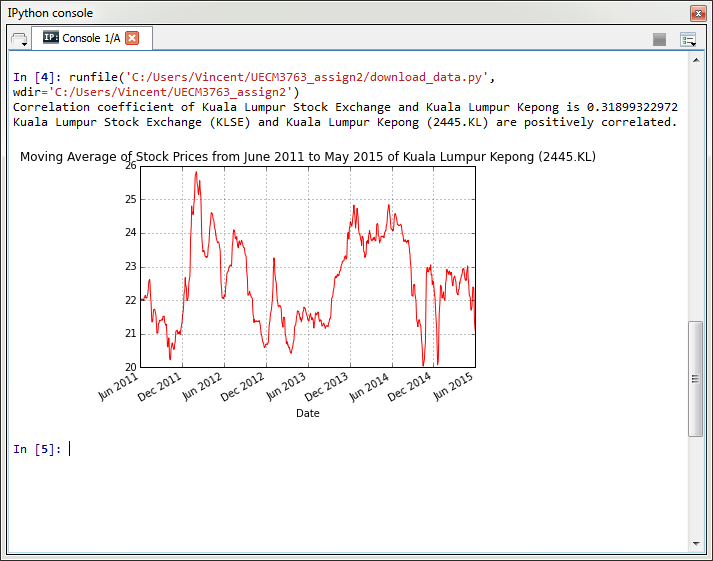
**Appendix**



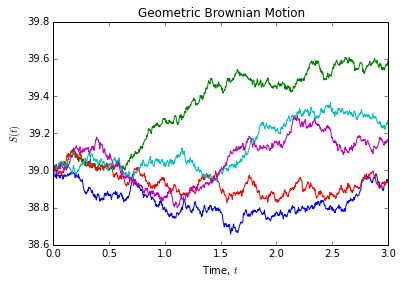
**Appendix 1:** Result generated from the Python file “gbm.py”



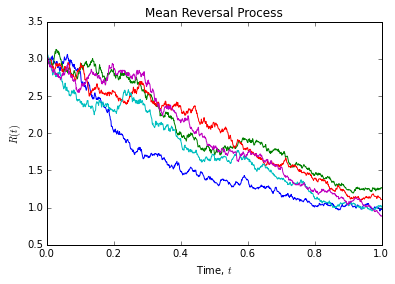
**Appendix 2:** Results generated from the Python file “mr.py”



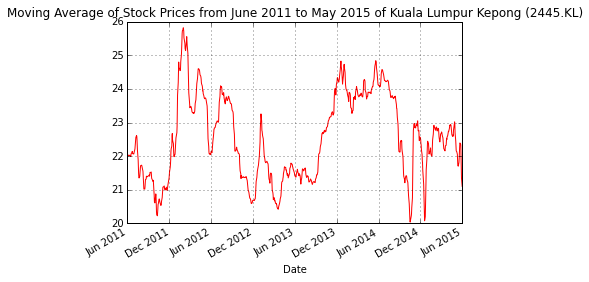
**Appendix 3:** Result generated from the Python file “download\_data.py”



**Appendix 4:** Graph of 5 realizations of Geometric Brownian Motion (GBM)



**Appendix 5:** Graph of 5 realizations of Mean Reversal Process



**Appendix 6:** Graph of 5-day moving average of Kuala Lumpur Kepong (2445.KL) from 1 June 2011 to 1 June 2015