GBM

October 14, 2024

0.1 Applications Stochastic Processes to Financial Modelling: Geometric Brownian Motion (GBM)

This Notebook is aimed to show how Geometric Brownian Motion can be used to predict prices of stocks based on historical market data. We aim to:

- Give an overview of the mathematics underlying the Stochastic Process
- Demonstrate how to collect and store the market data and use it to calibrate our model
- Display the results on some graphs

1. Stochastic Differential Equation (SDE) for GBM The SDE for a Geometric Brownian Motion is given by:

$$dX_t = \mu X_t \, dt + \sigma X_t \, dB_t$$

Additionally, by Ito's Lemma, we have:

$$dX_t^2 = \sigma^2 X_t^2 \, dt$$

2. Solution for the SDE The solution to the above differential equation is:

$$X_T = X_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma B_T}$$

This shows that X_T follows a **log-normal distribution**.

3. Expected Value, Variance, and Covariance **Expected Value:**

$$E[X_T] = e^{\log X_0 + \mu T} = X_0 e^{\mu T}$$

Variance:

$$V[X_T] = X_0^2 e^{2\mu T} \left(e^{\sigma^2 T} - 1\right)$$

Covariance:

$$C[X_T, X_S] = E[X_T X_S] - E[X_T]E[X_S] = X_0^2 e^{\mu(T+S)} \left(e^{\sigma^2 \min(T,S)} - 1\right)$$

4. Probability Density Function (PDF) of X_T Given that X_T follows a log-normal distribution, we can derive the PDF using a transformation of the normal distribution. For a normally distributed random variable $ Z N(, ^2) $, its PDF is:

$$f_Z(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

Since $ X_T = e^Z $, we can apply the transformation formula for PDFs:

$$f_X(x) = f_Z(z = \log x) \left| \frac{dz}{dx} \right|$$

This gives the PDF for X_T:

$$f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\log x-\mu)^2}{2\sigma^2}}$$

For our historical data, we shall take market data from the S&P 500 index, a popular stock market index.

We can obtain free historical data in a .csv file from the Nasdaq website : https://www.nasdaq.com/market-activity/index/spx/historical?page=1&rows_per_page=10&timeline=m1

[72]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm


dataframe = pd.read_csv("S&P_DailyData2.csv")
print(dataframe.head())

sp_returns = dataframe.iloc[:,1].values

#sp_returns = [7281.6, 7304, 7302.1, 7300.2, 7331.1, 7367.3, 7490.2, 7458.4,␣
 ↪7507.6, 7511.5, 7545.4, 7510.2, 7495.7, 7474.6, 7510.3, 7418.3, 7318.5]
```

```
         Date  Close/Last     Open     High      Low
0  10/08/2024     5751.13  5719.14  5757.60  5714.56
1  10/07/2024     5695.94  5737.80  5739.34  5686.85
2  10/04/2024     5751.07  5737.48  5753.21  5702.83
3  10/03/2024     5699.94  5698.19  5718.78  5677.37
4  10/02/2024     5709.54  5698.14  5719.63  5674.00
```

Calibrating GBM Model Part 1: Computing Statistical Estimates With the sample dataset of daily S&P 500 figures shown above, we can obtain our figures for the model using methods from statistics. We first need to calculate the mean and standard deviation:

Let X_i = The percentage return of the index on day i

We define the distribution: $X_i \sim e^Z$ where $Z \sim N[\hat{\mu}, \hat{\sigma}^2]$

Above, we take the log of the percentage difference between daily returns, which means that our values will all be sampled from a normal distribution:

$$Y_i = \ln(X_i) \sim N[\hat{\mu}, \hat{\sigma}^2]$$

Note that we are calculating **sample** mean and variance from the dataset:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}(r_i - \hat{\mu})^2$$

By the definition of Brownian motion (Weiner Process), we assume that the daily log returns are indepenent of each other: $X_{i+1} \perp X_i$

If we assume that there are 252 official working days in a calendar year (take out weekends), we can estimate the **Annual Mean and Variance**:

$$\hat{\mu}_a = E\left[\sum_{i=1}^{252} X_i\right] = 252E\left[X_i\right] = 252\hat{\mu}$$

$$\hat{\sigma}_a^2 = V\left[\sum_{i=1}^{252} X_i\right] = 252E\left[X_i\right] = 252\hat{\sigma}^2$$

[73]:
```
sp_log_returns = [np.log((sp_returns[i+1])/(sp_returns[i])) for i in␣
 ↪range(len(sp_returns)-1)]

mean = np.sum(sp_log_returns)*1/(len(sp_log_returns))

diff = sp_log_returns - mean
diff_squared = diff ** 2

variance = 1/(len(sp_log_returns) - 1)*np.sum(diff_squared)

mean_annual = 252 * mean
variance_annual = 252 * variance

#print("Daily log returns of S&P500", sp_log_returns)
print("\n")
print("Mean of Sample Dataset: ", mean)
print("Variance of Sample Dataset: ", variance)
print("\n")
```

```
print("Annual Mean of Sample Dataset: ", mean_annual)
print("Annual Variance of Sample Dataset: ", variance_annual)
```

```
Mean of Sample Dataset:  -0.0007843950483858798
Variance of Sample Dataset:  7.287853976090072e-05


Annual Mean of Sample Dataset:  -0.19766755219324172
Annual Variance of Sample Dataset:  0.01836539201974698
```

Calibrating GBM Model Part 2: Building the Model from Estimates We can now proceed to calibrate our model. Since we will be predicting future values based on historical data, we first define the time increments.

$t = 1$ will correspond to 1 year. Consequently, we let $\Delta t = 1/252$ equal 1 day.

We lastly apply our estimates to our random variable X_T. This can be done by returning to our solution for the SDE

$$X_T = X_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma B_T} \implies \ln \frac{X_T}{X_0} = \left(\mu - \frac{\sigma^2}{2}\right) T + \sigma B_T$$

We convert our equation into the form above since we are taking the log of the percentage difference of the daily returns. We note that the manipulated formula above is Arithmetic Brownian Motion:

$$dX_t = \mu dt + \sigma dB_t \implies X_T = X_0 \mu T + \sigma B_T \sim N\left[X_0 + \mu T, \sigma^2 T\right]$$

$$\therefore \ \ln \frac{X_T}{X_0} \sim N\left[\left(\mu - \frac{\sigma^2}{2}\right) T, \sigma^2 T\right]$$

We now plug in our statistical estimates back into the above equation to complete our model:

$$\sigma^2 = \hat{\sigma}^2$$

$$\hat{\mu} = \left(\mu - \frac{\hat{\sigma}^2}{2}\right) \implies \mu = \left(\hat{\mu} + \frac{\hat{\sigma}^2}{2}\right)$$

Note: These can be inputted back into our original equation: $dX_t = \mu X_t \, dt + \sigma X_t \, dB_t$

EXAMPLE: If $\hat{\mu} = 0.08$ and $\hat{\sigma}^2 = 0.013$ we would get

$$\sigma^2 = \hat{\sigma}^2 = 0.013$$

$$\mu = \hat{\mu} + \frac{\hat{\sigma}^2}{2} = 0.08 + \frac{0.013}{2} = 0.0865$$

$$X_T = X_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma B_T} \implies X_T = X_0 e^{(0.08)T + \sqrt{0.013} B_T}$$

We now converts this formula into an incremental one:

$$X_{t+\Delta t} = X_t e^{(0.08)\Delta t + \sqrt{0.013}\sqrt{\Delta t} N(0,1)}$$

We can also get the distribution of the simluated path by using the lognormal distribution (Assume $X_0 = 7318.5$):

$$X_T \sim LN\left[\ln X_0 + (\mu - \sigma^2/2)T, \sigma^2 T\right] \implies X \sim LN\left[\ln 7318.5 + 0.08\frac{T}{252}, 0.013\frac{T}{252}\right]$$

Since we have already defined $\Delta t = 1/252$ we are now ready to make some graphs!

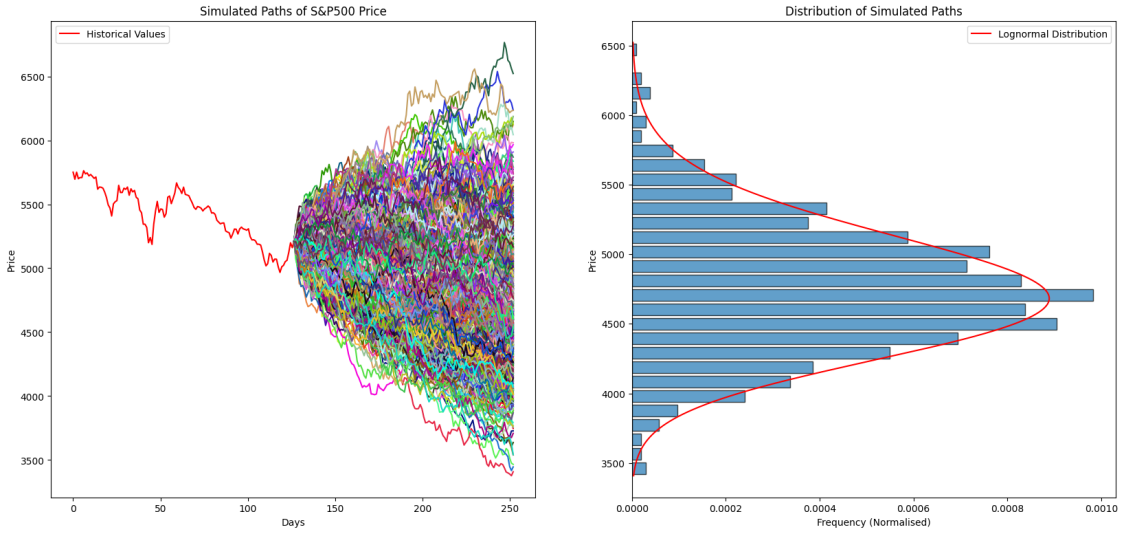
0.1.1 Graph 1: Monte Carlo Simulation

Here is a graph showing the simulation of values for the S&P 500 using the GBM model

[74]:
```
def run_sim(N):
    X = np.zeros(N+1)
    X[0] = X_0

    for i in range(N):

        # GBM Equation
        X[i+1] = X[i]*np.exp(mean_annual*dt + np.sqrt(variance_annual*dt)*np.
 ↪random.standard_normal())

    return X


def plot_sim(N, M, plot3D):

    # We will plot the historical data alongside the simluated values to␣
 ↪visualise how the historical data influences the simluation
    full_curve = np.zeros(len(sp_returns) + N+1)

    for i in range(len(sp_returns)):
        full_curve[i] = sp_returns[i]
```

```
    t = np.linspace(0,N,N+1)
    full_t = np.linspace(0, N + len(sp_returns)-1, N + len(sp_returns)-1)

    final_vals = []
    # We plot our results below. Our historical data is in red, while the␣
↪simulated values are colored randomly

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20,9))

    ax1.plot(full_t[full_t<len(sp_returns)], full_curve[:len(sp_returns)],␣
↪color='r', label="Historical Values")
    for i in range(M):
        X = run_sim(N)

        for i in range(N):
            full_curve[i + len(sp_returns)] = X[i]

        final_vals.append(full_curve[-2])
        ax1.plot(full_t[full_t>=len(sp_returns)-1], full_curve[len(sp_returns):
↪len(full_curve)-1], color=np.random.rand(3))

    ax1.set_xlabel("Days")
    ax1.set_ylabel("Price")
    ax1.legend()
    ax1.set_title("Simulated Paths of S&P500 Price")


    # Making sure density = True ensures that the area under the histogram␣
↪equals 1, thus allowing you to plot a pdf line over it!
    ax2.hist(final_vals, bins=30, orientation="horizontal", edgecolor="black",␣
↪alpha=0.7, rwidth=0.8, density=True)

    shape, loc, scale = lognorm.fit(final_vals, floc=0)  # Fit the lognormal␣
↪distribution
    x_vals = np.linspace(min(final_vals), max(final_vals), 10000)  # Generate x␣
↪values for the distribution

    # this function calculates the lognormal pdf given the input parameters␣
↪shape, loc and scale
    pdf_vals = lognorm.pdf(x_vals, shape, loc, scale)

    ax2.plot(pdf_vals, x_vals, color='r', label="Lognormal Distribution")
    ax2.set_xlabel("Frequency (Normalised)")
    ax2.set_ylabel("Price")
    ax2.legend()
```

```
    ax2.set_title("Distribution of Simulated Paths")

    if plot3D == True:
        plt.close()

    return [pdf_vals, x_vals]

# N = Number of days
# M = Number of simulations
N = 252
M = 1000

# t is our time
t = 1
dt = 1/N

# X_0 is our initial value
X_0 = sp_returns[-1]

graph1 = plot_sim(int(N/2), M, False)
```

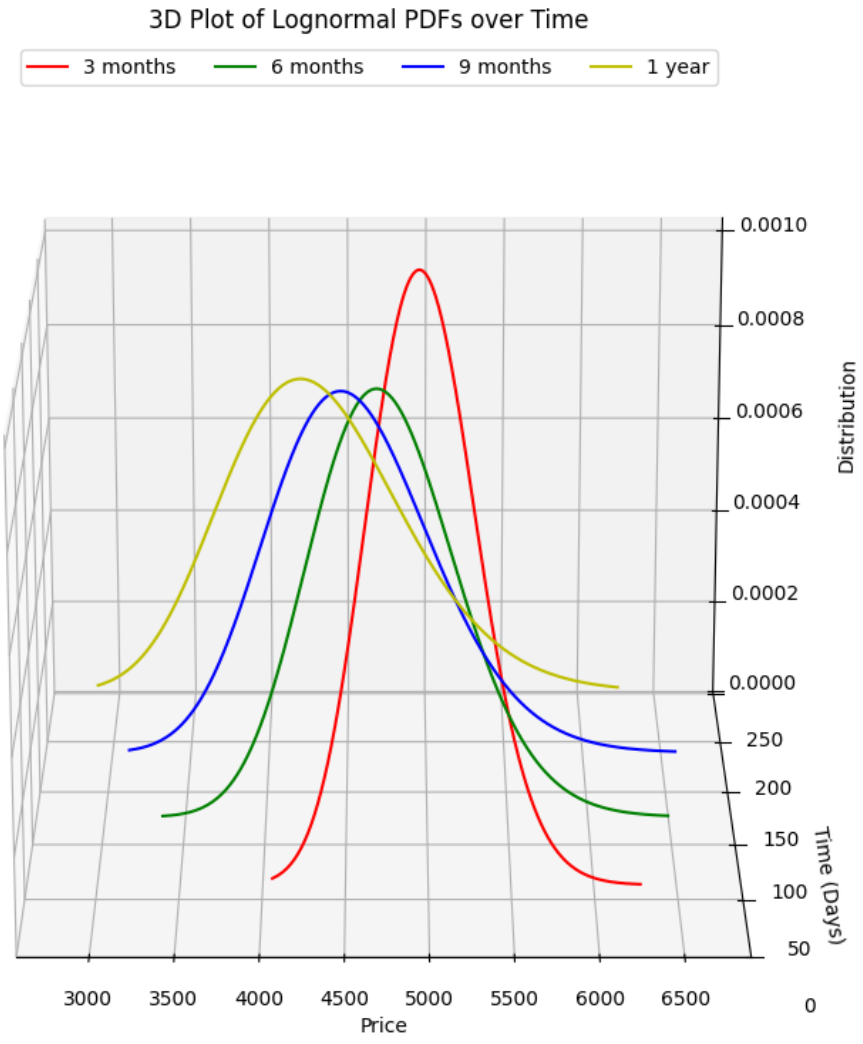


0.1.2 Graph 2: Change of Distribution Over Time

This 3D graph demonstrates how the distribution of the stock price spreads out as t increases. Note that this is expected based on the fact that the mean and variance depends on the value of T:

$$X_T \sim LN\left[\ln X_0 + (\mu - \sigma^2/2)T, \sigma^2 T\right]$$

[80]:
```
# Initialize 3D plot
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection="3d")

times = [N // 4, N // 2, 3 * N // 4, N]  # Time points for PDFs
written_times = ["3 months", "6 months", "9 months", "1 year"]
colors = ['r', 'g', 'b', 'y']

# Store PDF values for each time
for idx, time in enumerate(times):
    pdf_vals, x_vals = plot_sim(time, M, True)  # Get PDF and x_vals at each␣
 ↪time step
    ax.plot(x_vals, time, zs=pdf_vals, zdir='z', color=colors[idx], label =␣
 ↪written_times[idx])

ax.set_xlabel('Price')
ax.set_ylabel('Time (Days)', labelpad=20)
ax.set_zlabel('Distribution', labelpad=40)
ax.set_title('3D Plot of Lognormal PDFs over Time')

ax.set_xlim(min(x_vals)*0.9, max(x_vals)*1.1)  # Adjust the Price axis to give␣
 ↪some padding
ax.set_ylim(0, N)  # Time axis: adjust from 0 to N days
ax.set_zlim(0, max(pdf_vals)*1.5)  # Adjust the z-axis to have more space



ax.zaxis.set_tick_params(pad=15)
ax.yaxis.set_tick_params(pad=15)


ax.view_init(elev=20., azim=-90, roll=0)
ax.legend(ncols=4, loc="upper center")

plt.show()
```



0.2 References

1. **Oosterlee, C.W. and Grzelak, L.A.** (2019). *Mathematical Modeling And Computation In Finance: With Exercises And Python And Matlab Computer Codes.* World Scientific Publishing Company. ISBN: 9781786347961.
   Available at: https://books.google.ie/books?id=TsPKDwAAQBAJ

2. **Quantpie:** "Geometric Brownian Motion (GBM): solution, mean, variance, covariance, calibration, and simulation" YouTube. Available at: https://www.youtube.com/watch?v=98xF6b0PZpo
   Published on Oct 25, 2018.