Algorithmics for Financial Modelling
Volatility Trading

MSc Finance 2022

Mevlut Ozdemir

GRENOBLE ECOLE DE MANAGEMENT
BUSINESS LAB FOR SOCIETY

une école CCI GRENOBLE

ASSOCIATION OF MBAS ACCREDITED
AACSB ACCREDITED
EFMD EQUIS ACCREDITED

ACT THINK IMPACT

AIM OF THE COURSE

- A recent article deals with the quantitative skills needed in both investment banks and hedge funds:
  - Dan Stefanica, director of the Baruch Masters in Financial Engineering:
    - **IB:** hire students with fundamental math skills like stochastic calculus
    - **Hedge funds:** interested in students with statistical and data analysis skills

"They are different career opportunities and they require different kinds of skills" Stefanica says.

- In this course, we will touch base on both:
  - Option pricing
  - Time series analysis for relative value volatility trading

Source: https://www.efinancialcareers.co.uk/news/2022/04/maths-jobs-in-finance

MONTE CARLO SIMULATION (1/9)

- Monte Carlo simulation is an indispensable numerical tool in computational finance
- We will work with the Monte Carlo simulations of Geometric Brownian Motions to model the evolution of stock prices or index levels
- Black-Scholes-Merton (1973) theory of option pricing relies on this process in a risk-neutral environment

  – The underlying of the option follows a stochastic differential equation ("SDE"), with:
    - S_t, the value of the underlying at time t
    - r, the constant, riskless short rate
    - σ, the constant instantaneous volatility
    - Z, the Brownian Motion

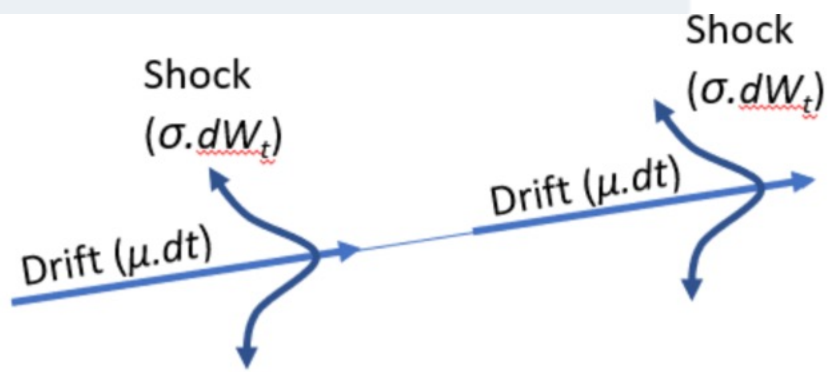
$$\boldsymbol{dS_t = rS_t \cdot dt + \sigma S_t \cdot dZ_t}$$

- The drift is used to model deterministic trends (risk free rate in risk-neutral environment), while the latter term is often used to model a set of unpredictable events occurring during this motion

- Changes in stock prices have the same distribution and are independent to each other

MONTE CARLO SIMULATION (2/9)

Geometric Brownian Motion Illustration:



- Where W_t is as Wiener process or Brownian Motion, and μ (the percentage drift), and σ (the percentage volatility) are assumed to be constants

- For an initial value S_0, the previously mentioned SDE has the following solution (under Itô's interpretation):

$$S_t = S_0\, e^{\left(\mu - \frac{\sigma^2}{2}\right)t \, + \sigma W_t}$$

MONTE CARLO SIMULATION (3/9)

- The geometric Brownian process is **<u>continuous</u>** but it can be discretised over equidistant time intervals and simulated according to the following Equation:

$$\boldsymbol{S_t = S_{t-\Delta t} \cdot e^{\left(\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma \cdot \sqrt{\Delta t} \cdot z\right)}}$$

- With z a random number following a <u>standard normal distribution</u>
- For M times intervals, the length of the time interval is given as $\Delta t \equiv \frac{T}{M}$, with T, the time horizon for the simulation (e.g. the maturity of the option)

- Therefore, for an European Call Option price, we have:

$$\boldsymbol{C_0 = e^{-rT} \cdot \frac{1}{I} \cdot \sum_{I} \max(S_T(i) - K, 0)}$$

- Where $\boldsymbol{S_T(i)}$ is the *i*-th simulated value of the underlying at maturity T for a total of simulated paths ***I*** with *i* = 1,2,…,I

MONTE CARLO SIMULATION (5/9)

4. Generate the paths using the GeoPaths function and compute the theoretically expected end-of-period value of our initial stock price.

```
%time pathsV = GeoPaths_V(M, I, S0, T, r, sigma)
```

```
CPU times: user 686 ms, sys: 54.6 ms, total: 740 ms
Wall time: 739 ms
```

```
#We can now check the theoretically expected end-of-period value of our initial stock price
round(pathsV[-1].mean(),2)
```

```
105.18
```

```
round(S0 * np.exp(r * T),2)
```

```
105.13
```

Expected Value of Geometric Brownian Motion:

$$E(X_t) = x_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)t} E(e^{\sigma B t})$$

- Since $E(e^{uZ}) = e^{u^2/2}$ for every real number u and every standard normal random variable Z, the identity $E(e^{\sigma Bt}) = e^{\sigma^2 t/2}$ follows from the fact that σBt is distributed like $\sigma\sqrt{t}Z$.
- In a nutshell: one gets the same expression of $E(X_t)$ using

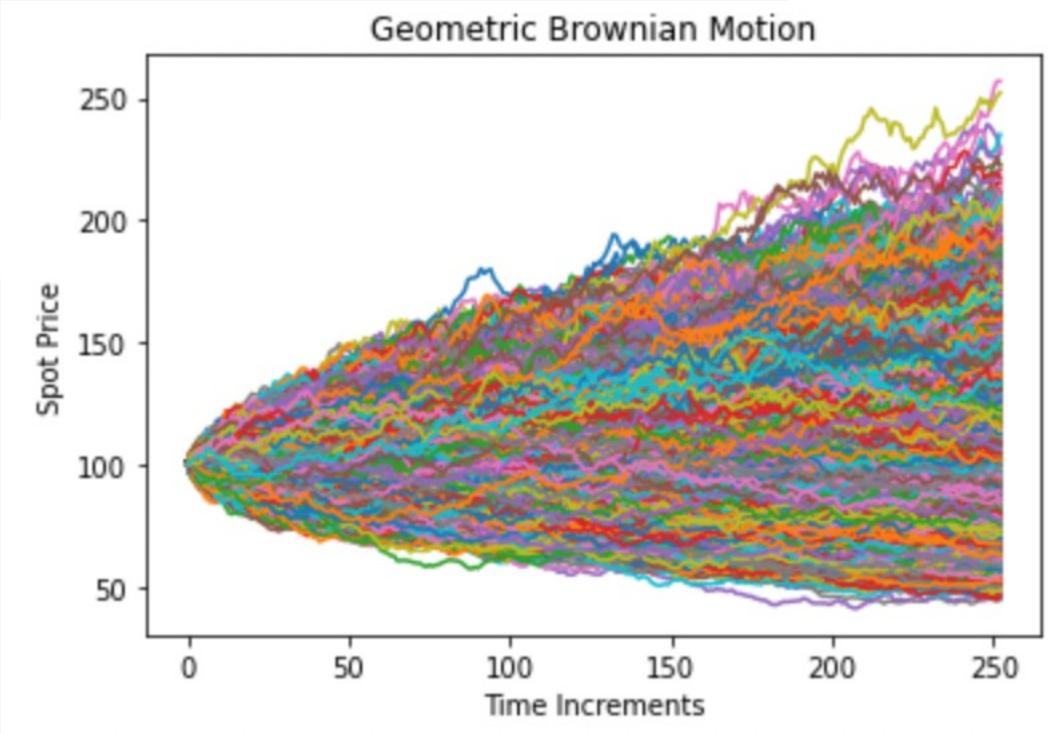
$$E(X_t) = E(X_0)e^{\mu t}$$

MONTE CARLO SIMULATION (6/9)

5. Graphical illustration of the simulations.

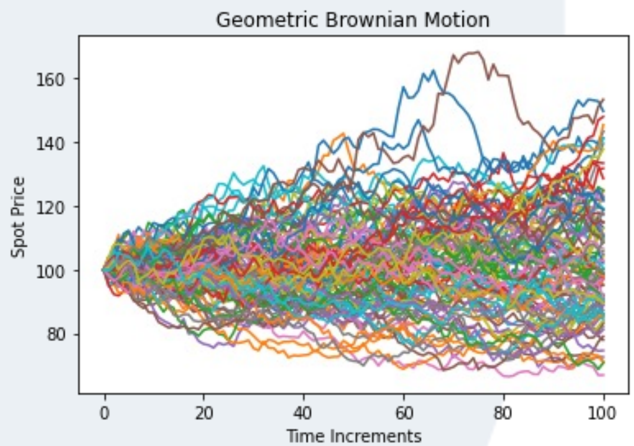
```
plt.plot(pathsV)
plt.xlabel("Time Increments")
plt.ylabel("Spot Price")
plt.title("Geometric Brownian Motion")
```

```
Text(0.5, 1.0, 'Geometric Brownian Motion')
```

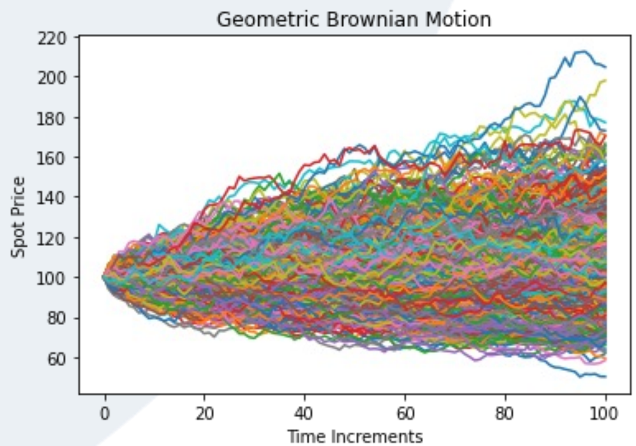


MONTE CARLO SIMULATION (7/9)

- For I = 100 simulations we have:



- For I = 1 000 simulations we have:



MONTE CARLO SIMULATION (8/9)

6. Visualise our paths with panda DataFrame.

```
dfPaths = pd.DataFrame(pathsV)
dfPaths
```

	0	1	2	3	4	5	6	7	8	9	...	79990	79991	
0	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	...	100.000000	100.000000	10
1	99.214707	101.521326	99.705702	102.810693	100.951573	98.714322	102.161774	101.686868	100.761419	102.718267	...	97.660315	100.206785	10
2	98.328987	103.083317	101.503466	102.439083	100.912186	98.339314	100.853888	104.085620	100.518851	101.829092	...	97.840298	96.762069	9
3	95.480670	102.898995	101.519612	103.280418	101.196324	98.863807	102.156547	103.095630	100.587787	101.451430	...	97.968235	97.412638	9
4	94.975468	101.049892	101.551434	102.593222	102.841234	98.559597	103.223554	102.718577	98.714678	102.453362	...	99.898162	98.370608	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
248	95.852395	92.811615	84.238333	117.616459	139.970422	145.479575	88.385365	120.289744	102.461865	76.976763	...	126.177313	103.509932	12
249	96.193060	92.301956	83.748622	115.620731	141.683298	144.273905	88.201674	123.071717	101.800922	77.544489	...	124.372074	102.686352	12
250	96.986261	90.993718	83.001330	117.113735	142.551954	145.450586	89.043980	124.419853	98.745970	77.782330	...	124.044201	101.532233	12
251	99.582065	89.210804	83.712939	116.521016	141.962961	146.935081	88.452780	125.853103	98.751255	76.458634	...	125.038746	101.722157	12
252	98.765755	88.864500	84.392452	117.278296	142.556239	147.053148	88.311983	128.533583	99.203621	76.132339	...	125.915259	102.107424	12

253 rows × 80000 columns

MONTE CARLO SIMULATION (9/9)

- Now that we have our simulations, we can apply any type of payoff to find the fair value of the financial product

- Let's start with the vanilla call option:

  - We initiate the strike K
  - ***np.maximum(ST – K, 0).mean()*** is the average final payoff
  - ***np.exp(-r * T)*** is the discount factor of this payoff

```
St = pathsV[-1]          #european observation, we take the last increment -> maturity
K = S0 * 1               #strike
```

```
#The fair final coupon of the payoff with the discount factor
hT = np.maximum(St - K, 0)
fairCoupon = np.mean(hT) * np.exp(-r * T)
print(round(fairCoupon/K,3)*100,'%')
```

```
10.5 %
```

BOOKS AND REFERENCES

- Mohamed Bouzoubaa, Adel Osseiran, *Exotic Options and Hybrids: A Guide to Structuring, Pricing and Trading,* 2010
- Frans de Weert, *Exotic Options Trading, 2008*
- Yves Hilpisch, *Python for Finance 2e: Mastering Data-Driven Finance, 2019*
- Yuxing Yan, *Python for Finance, 2014*
- K. Demeterfi, E. Derman, M. Kamal, J. Zou, *More Than You Ever Wanted To Know About Volatility Swaps*, Goldman Sachs Quantitative Research, 1999
- Davide Silvestrini, Bram Kaplan, Adam Rudd, *Volatility Swaps*, J.P. Morgan Europe Equity Derivatives & Delta One Strategy, 2010