International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Effect of Varying Underlying Data of Exotic Options (Asian and Lookback) Using Monte Carlo Scheme

Aumkar Wagle

This is a paper where we will be pricing Fixed and Float Exotic Options using the Monte Carlo scheme. In addition to pricing, we will be varying the data to see how the option price is affected by the variation. Below is a description of the problem at hand and the method in which we will be solving it.

We are, in essence, using the Fundamental Asset Pricing Formula in this framework to price Options. The general formula is as below:

ValueOfAsset=ExpectedValueQ[PV(CashFlows)]$ValueOfAsset=ExpectedValueQ[PV(CashFlows)]$

where Q is some equivalent probability measure.

When we convert this general form of pricing an asset to a particular form of pricing an Option, the formula is as below:

V(S;t)=exp(−r(T−t)ExpectedValueQ(Payoff(S))$V(S;t)=exp(-r(T-t)ExpectedValueQ(Payoff(S))$

where Q is the risk neutral probability measure.

What the above formula is saying is that in order to price the option, we need to follow the below steps:

1. Simulate the risk-neutral random walk starting at today's value of the asset S0 over the required time horizon. This gives one realization of the underlying price path.
2. For this realization calculate the option payoff.
3. Perform many more such realizations over the time horizon.
4. Calculate the average payoff over all realizations.
5. Take the present value of this average, this is the option value.

When we price via expectations, we will be simulating the below random walk (stochastic differential equation):

dSt=rStdt+σStdWt$dSt=rStdt+\sigma StdWt$

where St is the price of the underlying at time t, σ is constant volatility, r is the constant risk-freeinterest rate and W is the brownian motion.

While we can solve the above SDE with one giant leap, it would not be the suitable method because the exotic options that we are pricing in this project are path dependant i.e. the path that the underlying security takes over time (T-t) is of importance in calculating the payoff. The underlying path movements, and hence the payoff, would not be captured if we were to solve the SDE in one leap. But rather, we would need to simulate the entire time step by time step.

Due to the path dependent nature of the options we are pricing, we are increasing the dimensions of the security we are pricing. Dimensionality refers to the number of underlying independent variables. In this case, the path itself is a variable since the highs/lows and average of the path are essential for calculating the payoff. When we have an occurrence of greater number of dimensions, it is tough to have a closed form solution and tedious to setup the partial differential equation, let alone solve it. Thus, the Monte Carlo method is the preferred framework for pricing the given securities.

To achieve the above method of simulating the underlying price path, we will discretize the SDE using the Euler-Maruyama method to get the below equation so it can be used in our simulations:

St+δt=St∗(1+rδt+σ∗sqrt(δt)∗Wt)$St+\delta t=St*(1+r\delta t+\sigma*sqrt(\delta t)*Wt)$

Once we have our many (100000) simulated paths, we will use these to calculate the option payoffs and follow the steps as described earlier using the below sample set of data as an initial example:

Today's stock price S0 = 100 Strike = 100 Time to expiry = 1 year Volatility = 2% Constant risk free interest rate = 5%

We will then vary each of the above parameters (keeping the others constant) to see the impact on the option price and thus, use that as a method to explain the effect that the given parameter has on an option's pricing i.e. the sensitivty of the option price to a change in those parameters.

That being said, the Monte Carlo method is not free from errors. There are two sources of errors:

• If the size of the time step is δt then we may introduce errors of Order O(δt) by virtue of the discrete approximation to continuous events.

• Because we are only simulating a finite number of an infinitenumber of possible paths, the error due to using N realizations of the asset price paths is O(N^−1/2).

We will now start working with Python code to implement the above-described design.

```
#import libraries to be used in analysis
import pandas as pd
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
import cufflinks as cf
pd.set_option('display.max_rows',300)

#create user defined function to simulate underlying path
def simulate_path(s0, mu, sigma, horizon, timesteps, n_sims):
    np.random.seed(2023)
    S0 = s0
    r = mu
    T = horizon
    t = timesteps
    n = n_sims
    dt = T/t
    S = zeros((t,n)) #create grid which has it's rows as the timesteps and columns as a simulation
    S[0] = S0
    for i in range(0, t-1):
        w = random.standard_normal(n)
        S[i+1] = S[i] * (1+ r*dt + sigma*sqrt(dt)*w)
    return S

#create dataframe using pandas to form a table and display the last 5 rows of grid created.
price_path = pd.DataFrame(simulate_path(100,0.05,0.2,1,252,100000))
price_path.tail()
```

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**...**	**99990**	**99991**	**99992**	**99993**	**99994**	**99995**	**99996**	**99997**	**99998**	**99999**
247	82.452651	76.611556	96.127139	104.969419	105.913331	144.899335	107.069491	87.283685	146.641447	98.898211	...	104.511323	97.657080	102.643508	129.313037	85.226197	163.392707	83.861863	88.851576	61.924563	137.385509
248	83.625902	75.827722	96.600573	103.596788	106.015185	147.020837	108.247054	86.245161	146.001810	98.963360	...	104.219166	96.538980	104.257557	129.044406	83.385164	164.666929	83.411607	89.556030	61.695214	137.059986
249	82.015420	76.972786	96.294851	103.555391	106.868361	150.234926	107.513052	85.295482	148.012494	99.377709	...	105.629601	96.199775	102.848104	128.312085	84.227749	164.295674	84.452631	89.853093	61.860018	138.444789

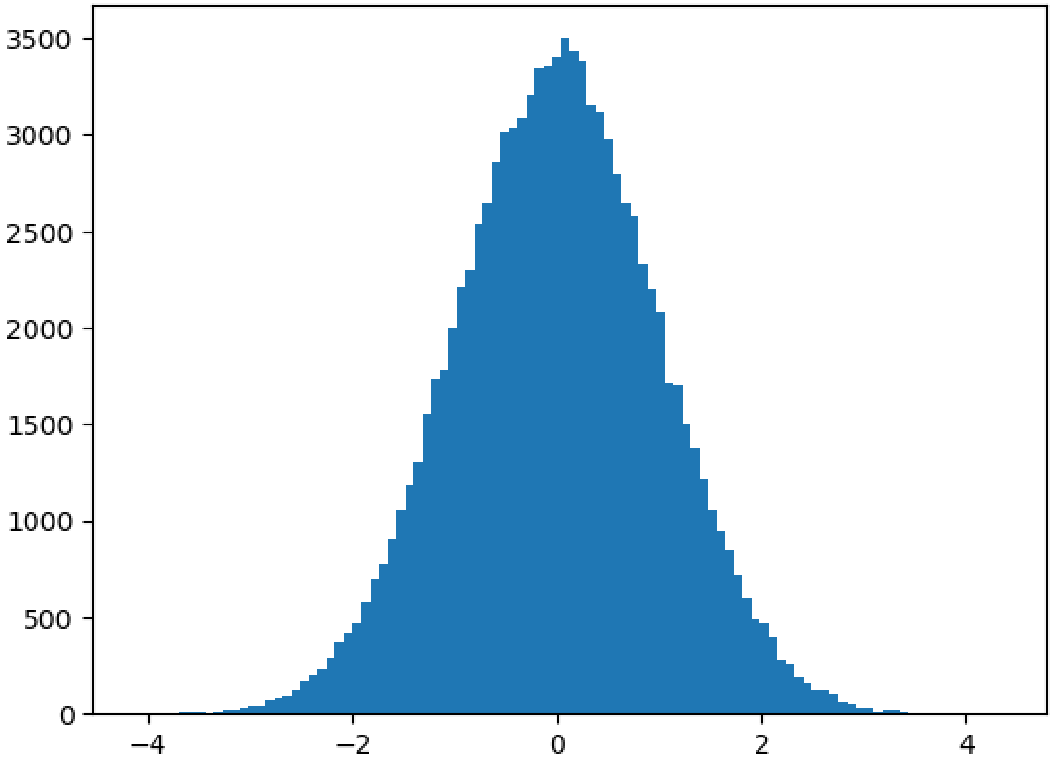
Paper ID: SR24418154649 DOI: https://dx.doi.org/10.21275/SR24418154649

	0	1	2	3	4	5	6	7	8	9	...	99990	99991	99992	99993	99994	99995	99996	99997	99998	99999
250	80.642913	76.783025	95.154389	105.616379	107.497258	149.945284	107.222721	86.734576	148.618368	98.820653	...	107.406821	94.912560	102.935111	125.641846	83.604510	163.470033	83.007683	91.172556	61.333441	136.355664
251	80.782976	76.713537	94.691231	103.588088	108.632172	151.506379	107.393998	85.837756	147.199309	100.592983	...	107.665608	96.874877	102.839756	122.933012	83.634889	163.939416	81.774788	91.142776	62.203926	137.327203

5 rows × 100000 columns

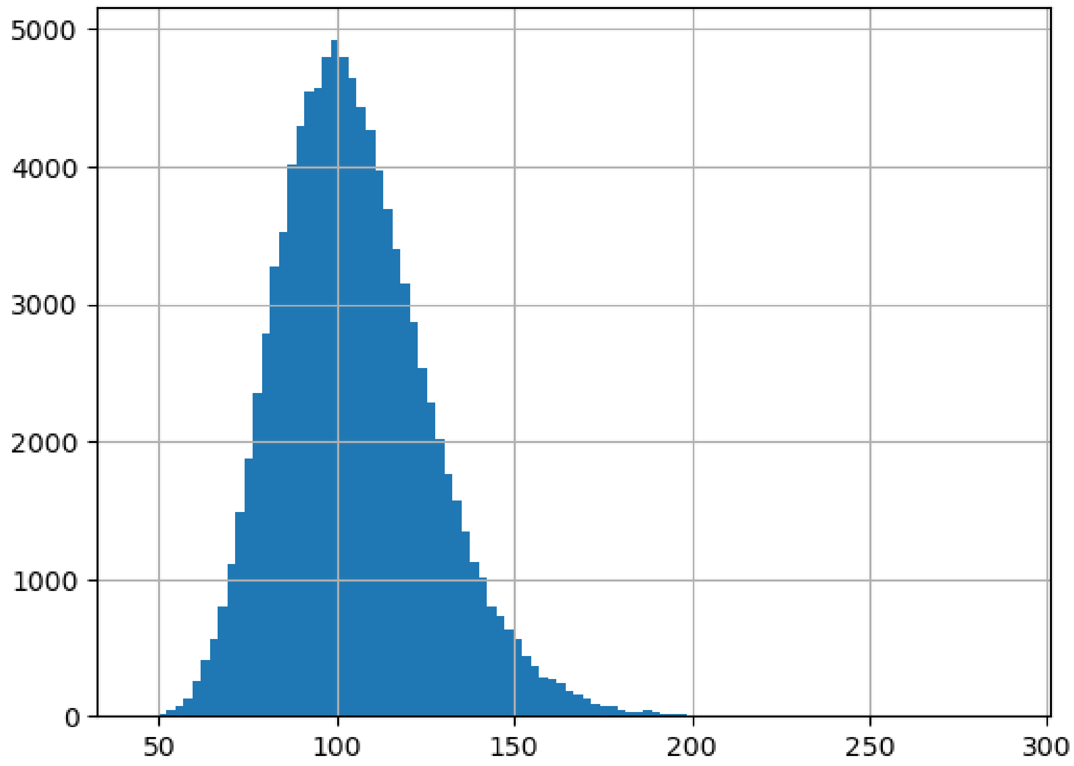
```
#plotting the histogram of random variable to see if normally distributed (bell curve shape)
plt.hist(pd.DataFrame(random.standard_normal(100000)), bins=100);
w = random.standard_normal(100000)
w.std(), w.mean() #checking if mean ~ 1 and standard deviation ~ 0 to know of deviation from normal distribution
```

(1.002674640493918, -0.003549804899300199)

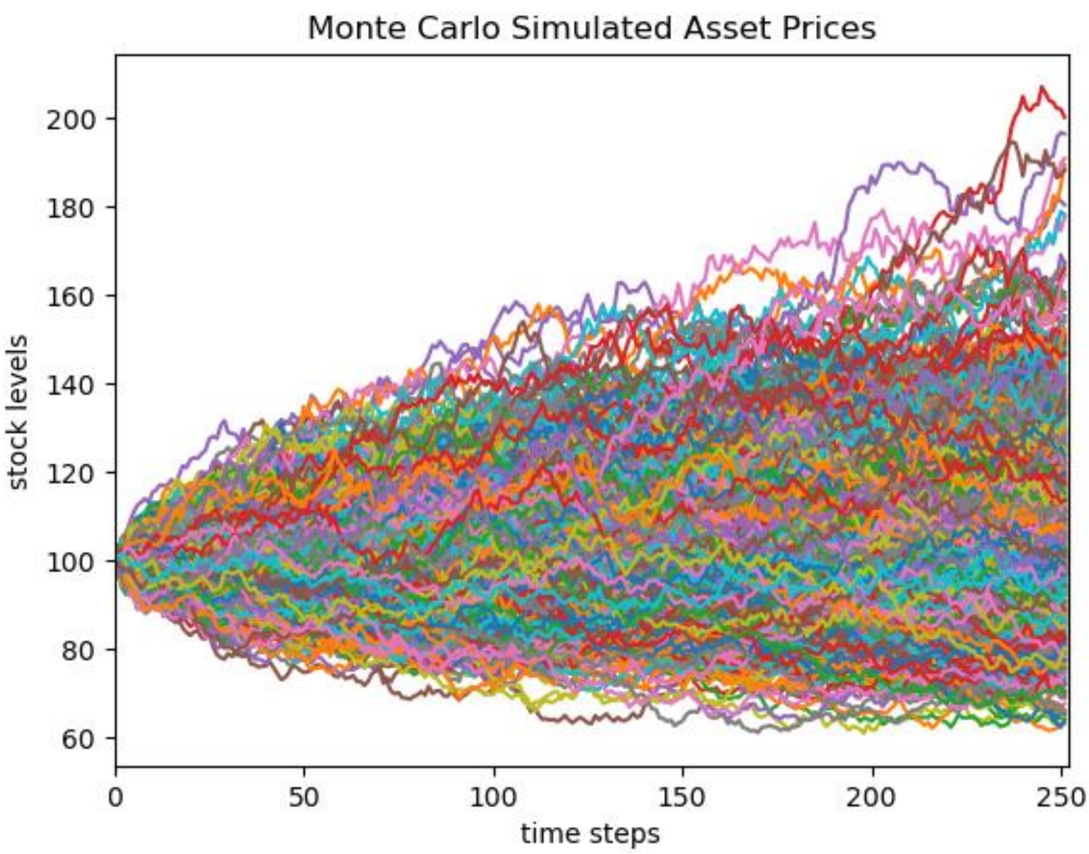



```
price_path.iloc[-1].hist(bins=100);
```


Paper ID: SR24418154649 DOI: https://dx.doi.org/10.21275/SR24418154649




```
#plotting the simulated underlying paths
plt.plot(price_path.iloc[:,:1000])
plt.xlabel('time steps')
plt.xlim(0,252)
plt.ylabel('stock levels')
plt.title('Monte Carlo Simulated Asset Prices');
```



Paper ID: SR24418154649 DOI: https://dx.doi.org/10.21275/SR24418154649

```
#create a user defined function that calculates the payoff of all 8 options and store the result in a list

def exotic_value_calculation(simulate_path,strike,horizon,rate):
    S = simulate_path
    K = strike; r = rate; T=horizon
    mean_path = S.mean(axis=0)
    max_path = S.max(axis=0)
    min_path = S.min(axis=0)
    final_path= S.iloc[-1]

    #Fixed and Float Asian Call and Put Option Payoffs
    Fixed_Asian_Call = exp(-r*T) * mean(maximum(0, mean_path-K))
    Fixed_Asian_Put = exp(-r*T) * mean(maximum(0, K-mean_path))
    Float_Asian_Call = exp(-r*T) * mean(maximum(final_path - mean_path, 0))
    Float_Asian_Put = exp(-r*T) * mean(maximum(mean_path - final_path, 0))

    #Fixed and Float Lookback Call and Put Option Payoffs
    Fixed_Lookback_Call = exp(-r*T) * mean(maximum(0, max_path-K))
    Fixed_Lookback_Put = exp(-r*T) * mean(maximum(0, K-min_path))
    Float_Lookback_Call = exp(-r*T) * mean(maximum(final_path - min_path, 0))
    Float_Lookback_Put = exp(-r*T) * mean(maximum(max_path - final_path, 0))

    Exotic_Results = [Fixed_Asian_Call, Fixed_Asian_Put, Float_Asian_Call, Float_Asian_Put, Fixed_Lookback_Call,
Fixed_Lookback_Put, Float_Lookback_Call, Float_Lookback_Put]
    return Exotic_Results
```

We will now vary the underlying price to see the effect on the prices of the options.

```
#varying underlying stock price S0 to see efect on option price

S0_range=range(50,150,5)

New_Results=[]
for s0 in S0_range:
    price_path = pd.DataFrame(simulate_path(s0, 0.05, 0.20, 1, 252, 100000))
    Exotic_Results=exotic_value_calculation(price_path,100,1,0.05)
    New_Results.append(Exotic_Results)

#creating a table of results
df1=pd.DataFrame(New_Results)
df1.columns=['Fixed_Asian_Call', 'Fixed_Asian_Put', 'Float_Asian_Call', 'Float_Asian_Put', 'Fixed_Lookback_Call',
'Fixed_Lookback_Put', 'Float_Lookback_Call', 'Float_Lookback_Put']
df1.index=S0_range
df1.index.name="Underlying_Level"
round(df1,4)
```

	Fixed_Asian_Call	**Fixed_Asian_Put**	**Float_Asian_Call**	**Float_Asian_Put**	**Fixed_Lookback_Call**	**Fixed_Lookback_Put**	**Float_Lookback_Call**	**Float_Lookback_Put**
Underlying_Level								
50	0.0000	46.3526	2.9277	1.6899	0.0037	53.4294	8.3146	6.7226
55	0.0000	41.4756	3.2205	1.8589	0.0228	49.2601	9.1460	7.3949
60	0.0000	36.5986	3.5132	2.0279	0.0947	45.0908	9.9775	8.0672
65	0.0009	31.7225	3.8060	2.1969	0.3028	40.9214	10.8089	8.7394
70	0.0083	26.8528	4.0988	2.3659	0.7801	36.7521	11.6404	9.4117
75	0.0479	22.0154	4.3915	2.5349	1.7028	32.5827	12.4719	10.0839

	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
Underlying_Level								
80	0.2028	17.2932	4.6843	2.7039	3.2723	28.4134	13.3033	10.7562
85	0.6318	12.8452	4.9771	2.8729	5.6571	24.2440	14.1348	11.4285
90	1.5726	8.9090	5.2698	3.0419	8.9592	20.0747	14.9662	12.1007
95	3.2535	5.7128	5.5626	3.2109	13.2017	15.9053	15.7977	12.7730
100	5.7777	3.3600	5.8554	3.3799	18.3384	11.7360	16.6291	13.4453
105	9.1029	1.8082	6.1482	3.5489	24.0115	8.2693	17.4606	14.1175
110	13.0606	0.8889	6.4409	3.7179	29.6846	5.7196	18.2921	14.7898
115	17.4497	0.4009	6.7337	3.8869	35.3576	3.8890	19.1235	15.4620
120	22.0925	0.1666	7.0265	4.0559	41.0307	2.5986	19.9550	16.1343
125	26.8660	0.0632	7.3192	4.2249	46.7038	1.7116	20.7864	16.8066
130	31.7022	0.0223	7.6120	4.3939	52.3769	1.1109	21.6179	17.4788
135	36.5647	0.0078	7.9048	4.5628	58.0499	0.7109	22.4493	18.1511
140	41.4363	0.0024	8.1975	4.7318	63.7230	0.4502	23.2808	18.8234
145	46.3116	0.0007	8.4903	4.9008	69.3961	0.2834	24.1122	19.4956

Observations:

Calculating the price of the option by varying the underlying price and keeping the other parameters constant is essentially telling us how sensitive the option price is to a change in the underlying price. Ofcourse, we aren't actually calculating Greeks here (one of the drawback of the MC Simulation since we would need many option prices. But it is giving us an understanding of how the option price varies when we vary the parameters affecting it's payoff. For example, we can see that as the level of the underlying stock moves farther away from the strike price i.e. goes further out-of-the-money, for a fix asian call option, the price of the option decreases. This makes sense because the chances of the stock crossing the strike price threshold to expire in-the-money have now reduced (keeping other parameters such as volatility and time to expiry constant). Interestingly, we see that floating call and put Asian options are an increasing function of the underlying because they are both linear in S0.

We will now vary the Strike price to see the effect on the prices of the options.

```
#varying underlying strike price to see efect on option price

price_path = pd.DataFrame(simulate_path(100, 0.05, 0.20, 1, 252, 100000)) # reset
K_range=range(50,155,5)

New_Results=[]
for K in K_range:
    Exotic_Results=exotic_value_calculation(price_path,K,1,0.05)
    New_Results.append(Exotic_Results)

#creating a table and plot of results
df2=pd.DataFrame(New_Results)
df2.columns=['Fixed_Asian_Call','Fixed_Asian_Put','Float_Asian_Call','Float_Asian_Put','Fixed_Lookback_Call','Fixed_Loo
kback_Put','Float_Lookback_Call','Float_Lookback_Put']
df2.index=K_range
df2.index.name="Strike Price"
```

round(df2,4)

	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
Strike Price								
50	49.9792	0.0000	5.8554	3.3799	65.8999	0.0005	16.6291	13.4453
55	45.2230	0.0000	5.8554	3.3799	61.1438	0.0045	16.6291	13.4453
60	40.4669	0.0000	5.8554	3.3799	56.3876	0.0214	16.6291	13.4453
65	35.7107	0.0000	5.8554	3.3799	51.6315	0.0799	16.6291	13.4453
70	30.9553	0.0008	5.8554	3.3799	46.8753	0.2420	16.6291	13.4453
75	26.2068	0.0083	5.8554	3.3799	42.1192	0.6197	16.6291	13.4453
80	21.4928	0.0505	5.8554	3.3799	37.3630	1.3693	16.6291	13.4453
85	16.9022	0.2161	5.8554	3.3799	32.6069	2.6760	16.6291	13.4453
90	12.6054	0.6755	5.8554	3.3799	27.8507	4.7325	16.6291	13.4453
95	8.8324	1.6585	5.8554	3.3799	23.0946	7.7082	16.6291	13.4453
100	5.7777	3.3600	5.8554	3.3799	18.3384	11.7360	16.6291	13.4453
105	3.5220	5.8604	5.8554	3.3799	14.0980	16.4921	16.6291	13.4453
110	1.9999	9.0945	5.8554	3.3799	10.6276	21.2483	16.6291	13.4453
115	1.0638	12.9145	5.8554	3.3799	7.8613	26.0044	16.6291	13.4453
120	0.5338	17.1407	5.8554	3.3799	5.7162	30.7605	16.6291	13.4453
125	0.2535	21.6165	5.8554	3.3799	4.0904	35.5167	16.6291	13.4453
130	0.1132	26.2324	5.8554	3.3799	2.8860	40.2728	16.6291	13.4453
135	0.0478	30.9231	5.8554	3.3799	2.0107	45.0290	16.6291	13.4453
140	0.0198	35.6513	5.8554	3.3799	1.3860	49.7851	16.6291	13.4453
145	0.0079	40.3955	5.8554	3.3799	0.9449	54.5413	16.6291	13.4453
150	0.0029	45.1467	5.8554	3.3799	0.6362	59.2974	16.6291	13.4453

Observations:

Changing the strike price does not affect the Floating strike options (Asian or lookback) since their payoff is not dependant on a fixed strike price. For the 2 fixed strike call options, we see that as the strike price increases, the options become more out-the-money since the underlying price is now further away from the strike price and therefore it's chances of expiring in-the-money are lesser, keeping other parameters constant. It is the other way around for the 2 fixed strike put options, the option becomes more valuable as the strike price is lower as the option now has a greater probability of expiring in-the-money or if it is already in-the-money, then the payoff is greater for a deeper in-the-money put.

We will now vary the underlying volatility to see the effect on the prices of the options.

```
#varying underlying stock volatility to see efect on option price

Vol_range=arange(0.05,0.35,0.01)

New_Results=[]
for vol in Vol_range:
    price_path = pd.DataFrame(simulate_path(100, 0.05, vol, 1, 252, 100000))
    Exotic_Results=exotic_value_calculation(price_path,100,1,0.05)
    New_Results.append(Exotic_Results)

#creating a table and plot of results
df3=pd.DataFrame(New_Results)
df3.columns=['Fixed_Asian_Call','Fixed_Asian_Put','Float_Asian_Call','Float_Asian_Put','Fixed_Lookback_Call','Fixed_Lookback_Put','Float_Lookback_Call','Float_Lookback_Put']
df3.index=Vol_range
df3.index.name="Volatility Level"
round(df3,4)
```

	Fixed_Asian_Call	**Fixed_Asian_Put**	**Float_Asian_Call**	**Float_Asian_Put**	**Fixed_Lookback_Call**	**Fixed_Lookback_Put**	**Float_Lookback_Call**	**Float_Lookback_Put**
Volatility Level								
0.05	2.7114	0.3026	2.7547	0.3002	6.8086	1.8458	6.7091	1.9453
0.06	2.8759	0.4669	2.9203	0.4646	7.4805	2.4692	7.3340	2.6158
0.07	3.0555	0.6461	3.1017	0.6447	8.1818	3.1131	7.9794	3.3155
0.08	3.2451	0.8354	3.2933	0.8350	8.9041	3.7694	8.6374	4.0361
0.09	3.4418	1.0317	3.4921	1.0324	9.6420	4.4330	9.3027	4.7723
0.10	3.6436	1.2331	3.6963	1.2353	10.3927	5.1007	9.9722	5.5212
0.11	3.8491	1.4381	3.9044	1.4421	11.1541	5.7705	10.6439	6.2807
0.12	4.0576	1.6461	4.1153	1.6516	11.9245	6.4407	11.3159	7.0492
0.13	4.2682	1.8561	4.3284	1.8633	12.7030	7.1102	11.9874	7.8257
0.14	4.4806	2.0678	4.5432	2.0767	13.4889	7.7782	12.6575	8.6096
0.15	4.6944	2.2809	4.7595	2.2916	14.2818	8.4444	13.3258	9.4004
0.16	4.9094	2.4952	4.9770	2.5076	15.0813	9.1083	13.9919	10.1976
0.17	5.1253	2.7103	5.1955	2.7246	15.8869	9.7696	14.6555	11.0010
0.18	5.3421	2.9263	5.4148	2.9423	16.6984	10.4281	15.3164	11.8101
0.19	5.5596	3.1429	5.6348	3.1608	17.5156	11.0836	15.9743	12.6249
0.20	5.7777	3.3600	5.8554	3.3799	18.3384	11.7360	16.6291	13.4453
0.21	5.9962	3.5775	6.0764	3.5994	19.1667	12.3851	17.2808	14.2710
0.22	6.2151	3.7954	6.2979	3.8192	20.0004	13.0308	17.9292	15.1020
0.23	6.4345	4.0137	6.5197	4.0395	20.8394	13.6733	18.5744	15.9383

	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
Volatility Level								
0.24	6.6541	4.2322	6.7418	4.2599	21.6838	14.3122	19.2161	16.7799
0.25	6.8740	4.4509	6.9642	4.4806	22.5334	14.9477	19.8545	17.6267
0.26	7.0942	4.6698	7.1868	4.7015	23.3882	15.5796	20.4893	18.4785
0.27	7.3145	4.8888	7.4096	4.9226	24.2482	16.2080	21.1207	19.3355
0.28	7.5350	5.1080	7.6325	5.1438	25.1134	16.8327	21.7485	20.1976
0.29	7.7556	5.3272	7.8556	5.3651	25.9838	17.4538	22.3728	21.0648
0.30	7.9763	5.5464	8.0788	5.5865	26.8593	18.0715	22.9936	21.9371
0.31	8.1971	5.7657	8.3021	5.8079	27.7398	18.6854	23.6109	22.8143
0.32	8.4180	5.9851	8.5254	6.0295	28.6255	19.2957	24.2247	23.6966
0.33	8.6389	6.2044	8.7489	6.2510	29.5162	19.9024	24.8348	24.5838
0.34	8.8599	6.4237	8.9724	6.4727	30.4120	20.5055	25.4414	25.4761

Observations:

Increasing the volatility leads to an increase in the option price for all options, irregardless of put/call, fixed/floating or Asian/Lookback. This is because an increase in volatility will provide for a greater chance of the option expiring in-the-money for a given horizon due to the larger size of the movement in the underlying asset. These larger swings in the underlying asset can result in the option becoming in-the-money and thus, making it more valuable.

We will now vary the time to maturity of the options to see the effect on the prices of the options.

```
#varying time to maturity to see efect on option price

Horizon_range=arange(0.5,1.5,0.01)

New_Results=[]
for horz in Horizon_range:
    price_path = pd.DataFrame(simulate_path(100, 0.05, 0.2, horz, 252, 100000)) # timesteps=252 meaning simulating by days
    Exotic_Results=exotic_value_calculation(price_path,100,1,0.05)
    New_Results.append(Exotic_Results)

#Creating a table and plot of results
df4=pd.DataFrame(New_Results)
df4.columns=['Fixed_Asian_Call','Fixed_Asian_Put','Float_Asian_Call','Float_Asian_Put','Fixed_Lookback_Call','Fixed_Lookback_Put','Float_Lookback_Call','Float_Lookback_Put']
df4.index=Horizon_range
df4.index.name="Horizon"
round(df4,4)
```

Horizon	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
0.50	3.7689	2.5702	3.7855	2.5636	12.0674	8.8056	11.2261	9.6468
0.51	3.8137	2.5909	3.8312	2.5846	12.2082	8.8803	11.3498	9.7387
0.52	3.8583	2.6112	3.8767	2.6054	12.3480	8.9541	11.4725	9.8297
0.53	3.9026	2.6313	3.9219	2.6259	12.4870	9.0270	11.5943	9.9197
0.54	3.9466	2.6512	3.9669	2.6461	12.6251	9.0990	11.7153	10.0089
0.55	3.9904	2.6708	4.0117	2.6661	12.7624	9.1702	11.8353	10.0972
0.56	4.0340	2.6901	4.0562	2.6859	12.8989	9.2404	11.9546	10.1847
0.57	4.0773	2.7091	4.1004	2.7054	13.0345	9.3099	12.0731	10.2713
0.58	4.1203	2.7280	4.1445	2.7246	13.1694	9.3785	12.1908	10.3572
0.59	4.1631	2.7465	4.1883	2.7436	13.3036	9.4464	12.3077	10.4423
0.60	4.2058	2.7649	4.2320	2.7625	13.4370	9.5135	12.4239	10.5266
0.61	4.2482	2.7830	4.2754	2.7810	13.5697	9.5799	12.5394	10.6102
0.62	4.2903	2.8010	4.3187	2.7994	13.7017	9.6456	12.6542	10.6931
0.63	4.3323	2.8187	4.3617	2.8176	13.8331	9.7105	12.7683	10.7753
0.64	4.3741	2.8362	4.4046	2.8356	13.9638	9.7748	12.8817	10.8569
0.65	4.4157	2.8535	4.4472	2.8533	14.0939	9.8384	12.9945	10.9377
0.66	4.4571	2.8706	4.4898	2.8709	14.2233	9.9013	13.1067	11.0180
0.67	4.4983	2.8875	4.5321	2.8883	14.3522	9.9636	13.2182	11.0975
0.68	4.5394	2.9042	4.5743	2.9056	14.4804	10.0253	13.3292	11.1765
0.69	4.5802	2.9207	4.6163	2.9226	14.6081	10.0864	13.4396	11.2549
0.70	4.6209	2.9371	4.6581	2.9395	14.7352	10.1469	13.5494	11.3327
0.71	4.6614	2.9533	4.6998	2.9562	14.8617	10.2067	13.6586	11.4099
0.72	4.7018	2.9693	4.7413	2.9727	14.9877	10.2661	13.7673	11.4865
0.73	4.7420	2.9851	4.7827	2.9890	15.1132	10.3248	13.8754	11.5626
0.74	4.7820	3.0008	4.8240	3.0053	15.2382	10.3830	13.9831	11.6382
0.75	4.8219	3.0163	4.8651	3.0213	15.3627	10.4407	14.0902	11.7133
0.76	4.8617	3.0317	4.9061	3.0372	15.4867	10.4979	14.1968	11.7878
0.77	4.9013	3.0469	4.9469	3.0529	15.6102	10.5545	14.3029	11.8618

Horizon	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
0.78	4.9407	3.0619	4.9876	3.0686	15.7332	10.6107	14.4086	11.9353
0.79	4.9800	3.0768	5.0282	3.0840	15.8558	10.6663	14.5138	12.0084
0.80	5.0192	3.0916	5.0687	3.0993	15.9779	10.7215	14.6185	12.0809
0.81	5.0583	3.1062	5.1090	3.1145	16.0996	10.7762	14.7228	12.1530
0.82	5.0972	3.1207	5.1492	3.1296	16.2209	10.8304	14.8266	12.2247
0.83	5.1359	3.1350	5.1893	3.1445	16.3417	10.8842	14.9300	12.2959
0.84	5.1746	3.1492	5.2293	3.1592	16.4621	10.9375	15.0330	12.3666
0.85	5.2131	3.1633	5.2692	3.1739	16.5821	10.9904	15.1355	12.4370
0.86	5.2515	3.1772	5.3090	3.1884	16.7017	11.0429	15.2377	12.5069
0.87	5.2898	3.1910	5.3486	3.2028	16.8209	11.0949	15.3394	12.5764
0.88	5.3280	3.2047	5.3882	3.2171	16.9398	11.1465	15.4408	12.6455
0.89	5.3660	3.2183	5.4276	3.2313	17.0582	11.1977	15.5418	12.7142
0.90	5.4040	3.2317	5.4670	3.2453	17.1763	11.2485	15.6424	12.7825
0.91	5.4418	3.2450	5.5062	3.2593	17.2941	11.2989	15.7426	12.8504
0.92	5.4795	3.2583	5.5454	3.2731	17.4114	11.3490	15.8425	12.9179
0.93	5.5172	3.2714	5.5844	3.2868	17.5285	11.3986	15.9420	12.9850
0.94	5.5547	3.2843	5.6234	3.3004	17.6451	11.4479	16.0412	13.0518
0.95	5.5921	3.2972	5.6623	3.3139	17.7615	11.4968	16.1400	13.1183
0.96	5.6294	3.3100	5.7011	3.3273	17.8775	11.5453	16.2385	13.1844
0.97	5.6666	3.3226	5.7398	3.3406	17.9932	11.5935	16.3366	13.2501
0.98	5.7037	3.3352	5.7784	3.3538	18.1086	11.6413	16.4344	13.3155
0.99	5.7407	3.3476	5.8169	3.3669	18.2237	11.6888	16.5319	13.3805
1.00	5.7777	3.3600	5.8554	3.3799	18.3384	11.7360	16.6291	13.4453
1.01	5.8145	3.3722	5.8938	3.3928	18.4529	11.7828	16.7260	13.5097
1.02	5.8512	3.3844	5.9320	3.4056	18.5671	11.8293	16.8226	13.5737
1.03	5.8879	3.3964	5.9703	3.4183	18.6809	11.8754	16.9189	13.6375
1.04	5.9245	3.4084	6.0084	3.4309	18.7945	11.9213	17.0148	13.7009
1.05	5.9609	3.4202	6.0465	3.4434	18.9078	11.9668	17.1105	13.7641

Horizon	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
1.06	5.9973	3.4320	6.0845	3.4559	19.0209	12.0120	17.2059	13.8269
1.07	6.0336	3.4437	6.1224	3.4682	19.1336	12.0569	17.3011	13.8895
1.08	6.0699	3.4552	6.1602	3.4805	19.2461	12.1015	17.3959	13.9517
1.09	6.1060	3.4667	6.1980	3.4926	19.3583	12.1458	17.4905	14.0137
1.10	6.1421	3.4781	6.2357	3.5047	19.4703	12.1899	17.5848	14.0754
1.11	6.1781	3.4895	6.2734	3.5167	19.5820	12.2336	17.6788	14.1368
1.12	6.2140	3.5007	6.3110	3.5287	19.6935	12.2770	17.7726	14.1979
1.13	6.2498	3.5119	6.3485	3.5405	19.8047	12.3202	17.8661	14.2588
1.14	6.2856	3.5229	6.3859	3.5523	19.9157	12.3631	17.9594	14.3194
1.15	6.3213	3.5339	6.4233	3.5640	20.0264	12.4057	18.0525	14.3797
1.16	6.3569	3.5448	6.4607	3.5756	20.1369	12.4481	18.1452	14.4398
1.17	6.3925	3.5556	6.4979	3.5871	20.2472	12.4902	18.2378	14.4996
1.18	6.4279	3.5664	6.5351	3.5986	20.3572	12.5320	18.3301	14.5591
1.19	6.4633	3.5771	6.5723	3.6100	20.4671	12.5736	18.4222	14.6184
1.20	6.4987	3.5877	6.6094	3.6213	20.5766	12.6149	18.5140	14.6775
1.21	6.5340	3.5982	6.6464	3.6325	20.6860	12.6560	18.6057	14.7363
1.22	6.5692	3.6086	6.6834	3.6437	20.7952	12.6968	18.6971	14.7949
1.23	6.6043	3.6190	6.7204	3.6548	20.9042	12.7374	18.7882	14.8533
1.24	6.6394	3.6293	6.7573	3.6658	21.0129	12.7777	18.8792	14.9114
1.25	6.6744	3.6395	6.7941	3.6768	21.1214	12.8178	18.9700	14.9693
1.26	6.7094	3.6497	6.8309	3.6877	21.2298	12.8576	19.0605	15.0269
1.27	6.7443	3.6598	6.8676	3.6985	21.3379	12.8973	19.1508	15.0844
1.28	6.7791	3.6698	6.9043	3.7093	21.4459	12.9367	19.2410	15.1416
1.29	6.8139	3.6798	6.9409	3.7200	21.5536	12.9758	19.3309	15.1986
1.30	6.8486	3.6896	6.9775	3.7306	21.6612	13.0148	19.4206	15.2554
1.31	6.8832	3.6995	7.0141	3.7412	21.7686	13.0535	19.5102	15.3119
1.32	6.9178	3.7092	7.0505	3.7517	21.8758	13.0920	19.5995	15.3683
1.33	6.9524	3.7189	7.0870	3.7621	21.9828	13.1303	19.6886	15.4244

	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
Horizon								
1.34	6.9869	3.7285	7.1234	3.7725	22.0896	13.1684	19.7776	15.4804
1.35	7.0213	3.7381	7.1598	3.7829	22.1963	13.2063	19.8664	15.5361
1.36	7.0557	3.7476	7.1961	3.7931	22.3027	13.2439	19.9550	15.5917
1.37	7.0900	3.7570	7.2324	3.8033	22.4090	13.2814	20.0434	15.6470
1.38	7.1243	3.7664	7.2686	3.8135	22.5152	13.3186	20.1316	15.7022
1.39	7.1585	3.7757	7.3048	3.8236	22.6211	13.3557	20.2197	15.7572
1.40	7.1927	3.7850	7.3410	3.8336	22.7270	13.3925	20.3076	15.8119
1.41	7.2268	3.7942	7.3771	3.8436	22.8326	13.4292	20.3953	15.8665
1.42	7.2608	3.8033	7.4132	3.8535	22.9381	13.4657	20.4828	15.9209
1.43	7.2949	3.8124	7.4492	3.8634	23.0434	13.5019	20.5702	15.9751
1.44	7.3288	3.8215	7.4852	3.8732	23.1486	13.5380	20.6574	16.0292
1.45	7.3628	3.8304	7.5212	3.8830	23.2536	13.5739	20.7444	16.0830
1.46	7.3966	3.8393	7.5571	3.8927	23.3584	13.6096	20.8313	16.1367
1.47	7.4305	3.8482	7.5930	3.9023	23.4632	13.6451	20.9181	16.1902
1.48	7.4643	3.8570	7.6289	3.9119	23.5677	13.6805	21.0046	16.2436
1.49	7.4980	3.8658	7.6647	3.9215	23.6721	13.7156	21.0911	16.2967

Observations:

Increasing the time to expiry of an option leads to an increase in it's price for all options, irregardless of put/call or fixed/floating or Asian/Floating. This is because it provides for a greater amount of time until which the option can move from being out-the-money to in-the-money. This makes the option more valuable and therefore, we see an increase in option price for as time to maturity increases.

We will now vary the risk-free interest rate to see the effect on the prices of the options.

```
#varying rate to see efect on option price

Mu_range=arange(0.01,0.1,0.01)

New_Results=[]
for mu in Mu_range:
    price_path = pd.DataFrame(simulate_path(100, mu, 0.2, 1, 252, 100000))
    Exotic_Results=exotic_value_calculation(price_path,100,1,0.05)
    New_Results.append(Exotic_Results)

#Creating a table and plot of results
df5=pd.DataFrame(New_Results)
df5.columns=['Fixed_Asian_Call','Fixed_Asian_Put','Float_Asian_Call','Float_Asian_Put','Fixed_Lookback_Call','Fixed_Lookback_Put','Float_Lookback_Call','Float_Lookback_Put']
```

```
df5.index=Mu_range
df5.index.name="Rate"
round(df5,4)
```

	Fixed_Asian_Call	Fixed_Asian_Put	Float_Asian_Call	Float_Asian_Put	Fixed_Lookback_Call	Fixed_Lookback_Put	Float_Lookback_Call	Float_Lookback_Put
Rate								
0.01	4.6546	4.1701	4.6460	4.1432	15.9230	13.2558	14.2430	14.9358
0.02	4.9205	3.9575	4.9304	3.9441	16.4994	12.8624	14.8116	14.5502
0.03	5.1962	3.7515	5.2266	3.7504	17.0937	12.4778	15.3987	14.1729
0.04	5.4818	3.5523	5.5349	3.5624	17.7067	12.1024	16.0045	13.8046
0.05	5.7777	3.3600	5.8554	3.3799	18.3384	11.7360	16.6291	13.4453
0.06	6.0837	3.1746	6.1881	3.2031	18.9889	11.3785	17.2726	13.0947
0.07	6.4000	2.9963	6.5335	3.0322	19.6579	11.0299	17.9350	12.7528
0.08	6.7265	2.8248	6.8915	2.8672	20.3459	10.6903	18.6163	12.4199
0.09	7.0630	2.6601	7.2623	2.7080	21.0535	10.3594	19.3167	12.0962

Observations:

For the fixed call options, we see that the option price increases as the interest rate increases. This is because the interest one can earn at higher rates makes the opportunity cost of buying the underlying less attractive. The opposite is the case for the fixed Asian Put. When one owns the underlying asset and is looking to sell it in the future, a greater interest rate would mean that much of a missed opportunity to earn that risk free rate by deposting the proceeds of the underlying that would be sold.

Remarks

While one difficulty that would usually be encountered with Monte Carlo simulations is that of added time required for computation, in this exercise, i didnt really come accross this. Well, it took about 1.5-2.0 mins for each varying data exercise that was conducted. Usually, the time consumption issue occurs due to the resimulation that takes place when varying the data but it wasnt too time consuming in my exercise.

We have been varying the parameters/variables affecting an option's price such as the underlying price,volatility, time to expiry, etc and then looked at how the option price is changing. If one was to read the aforesaid statement, then one would ideally think about Greeks since we are checking how the option price changed when the parameters/variables affecting it are changed. However, we are not actually looking at Greeks (calculating Greeks) since this is a simulation methodology and not a closed form answer. For example, the Greeks themselves have closed form solutions. However, this simulation methodology we used can indeed aid in our understanding of how the option price is affected for a change in the variables used in it's pricing.

In addition to the above, this is the first time i am pricing exotic options such as floating strike Asian Options and looking at it's first order sensitivities was interesting when compared to plain vanilla options (especially change in the underlying).

Conclusion

While it is possible to price these types of options using a closed form solution, using the Monte Carlo method has it's own set of advantages. The advantages that can be drawn from this method as seen in the aforeperformed exercise are described below:

1)The mathematics that you need to perform a Monte Carlo simulation is very basic. We didnt really solved an complex math functions in this exercise. Apart from discretizing the SDE using the Euler-Maruyama method, there werent any mathematical changes or solving that was involved.

2)It is computationally quite efficient in high dimensions. Despite the addition of the additional dimension (path dependence), we didnt face much of a challenge in our exercise. This would not be the case had we used the Partial Differential Equation or finite difference method approaches. The effort in getting some answer is very low i.e. it is hard to make mistakes with this method.

3)The models used are flexible and can be changed without much work. We had one basic modeel for the pricing framework which was used in a loop to vary the information that affected the payoff. We didnt need to keep rebuilding our model for any of the different variations or pricing functions. Thus we saw that many contracts can be priced at the same time without much effort.

4)The MC method is the most widely used method in the market today, about 60% of market participants use this simulation methodology for a range of purposes. This can be for securities pricing, risk analysis, portfolio optimization, etc. Given that it is such a widely used method, more people know of it's usage and thus are able to interpret/understand the method much more easily as compared to other methods, thus making it more 'colleague friendly'. People are acclimitized to seeing the results from this method, they accept this technique, and are more liekly to believe the answers that this method aims to solve.

However, this method does have it's drawbacks. For example, it is tough to find Greeks using this method as discussed briefly earlier, and it doesnt cope well with early exercise options.

That being said, we can conclude that using the MonteCarlo scheme for pricing Exotic Options is viable due to its many advantages and it's allowance for easy manipulation of variable data to see it's affect on option prices.

References

[1] Kannan Singaravelu, Monte Carlo Simulation Python Lab
[2] Sebastien Lleo, CQF_June_2023_M3L2
[3] Riaz Ahmed, CQF_June_2023_M3L4
[4] Riaz Ahmed, CQF_June_2023_M3L5
[5] Paul Wilmott, Paul Wilmott introduces Quantitative Finance 2nd Edition
[6] Stackoverflow: https://stackoverflow.com/