In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Monte Carlo & Euler-Maruyama

This report represents the results found on Asian & Lookback options using Euler-Maruyama scheme for initially simulating the underlying stock price.

Given a risk free interest rate, under the risk neutral framework, we assume that our asset will earn on average this rate. Under the $\mathbb{Q}$-measure and using the fact our option value at (t) would be the discounted value of the expected payoff, we can define the below:

$$V(S,t) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[\text{Payoff}]$$

Before speaking about options, we first define Monte Carlo and Euler-Maruyama scheme. Monte Carlo simulations are used to approximate the value of our random variable Phi. These values can help calculate exotic options payoffs, particularly ones which have path dependencies. Additionally, they're useful for generating various scenarios (or asset paths) according to the underlying stochastic differential equation (SDE) *Resource used: Introduction to numerical methods by Dr. Riaz Ahmad*. Euler-Maruyama is a numerical technique which is used to approximate solutions to our SDE. To understand this method further, given a SDE in the form:

$$dX_t = a(X_t,t)dt + b(X_t,t)dW_t$$

where:

- X_t is the state variable at time t,
- $a(X_t,t)$ is the drift coefficient,
- $b(X_t,t)$ is the diffusion coefficient,
- W_t is a Wiener process (or standard Brownian motion).

The Euler-Maruyama approximation formula over a time step Δt is given by:

$$X_{t+\Delta t} = X_t + a(X_t,t)\Delta t + b(X_t,t)\Delta W_t$$

where $\Delta W_t = W_{t+\Delta t} - W_t$ is the increment of the Wiener process. Given Lecture 3.4 on intro to numerical methods on annotated slide 8, we can use the below form:

$$S_{t+\Delta t} = S_t(1 + r\Delta t + \sigma\sqrt{\Delta t}\Phi)$$

It's important to note, this method has an error of $O(\delta t)$ *Resource used: slide 11 of introduction to numerical methods by Dr. Riaz Ahmad*

Time Sampling

Given our formula above, inherently it's using discrete time steps. This means that we look at asset prices over specified points in time. The more we increase our time steps, we come closer to continuous where we look at the price of our asset at all points in time. These two concepts come into play for our options and can be represented with the below:

- For Discrete Sampling

$$A = \frac{1}{t}\sum_{i=0}^{t} S(i)$$

- For Continuous Sampling

$$A = \frac{1}{t}\int_{0}^{t} S(u)du$$

Asian & Lookback Options

We define an Asian option as a contract where the payoff is determined by the average value of the underlying asset over some period before expiry *Resource used: Exotic Options by Dr. Riaz Ahmad*. For a Lookback, we follow the underlying asset price during the life of our contract, choosing our maximum or minimum price. This means that the holder 'looks back' over time to select the optimal exercise price.

We can view Asian and Lookback option profiles in two variations, fixed strike and floating strike. The fixed strike is a predetermined

exercise price set at the beginning of our contract agreement. For floating strike on an Asian option, we take the average price of the underlying during our contracts duration. Looking at floating strike for a Lookback however, we use the most favorable underlying price as our strike. This means for a call, we take the minimum observed price and for a put, we use the maximum observed price over the life of our option. For the strike profile stated above, we can state that a Lookback will always be in the money.

Under the risk-neutral $\mathbb{Q}$ measure, Asian and Lookback options payoffs are:

- For Fixed-Strike Asian Options

$$C_t^{\text{fixed}} = e^{-r(T-t)}\mathbb{E}(\max(0, S_{\text{avg}} - K))$$

$$P_t^{\text{fixed}} = e^{-r(T-t)}\mathbb{E}(\max(0, K - S_{\text{avg}}))$$

- For Float-Strike Asian Options

$$C_t^{\text{float}} = e^{-r(T-t)}\mathbb{E}(\max(0, S_T - S_{\text{avg}}))$$

$$P_t^{\text{float}} = e^{-r(T-t)}\mathbb{E}(\max(0, S_{\text{avg}} - S_T))$$

For our Lookback options, let S_T be the price of the underlying asset at maturity. we define M_T as the maximum price for our underlying asset for the total duration of the option. Respectively, for the minimum price, we use m_T to get the below payoffs:

- For Fixed Strike Lookback Options

$$C_t^{\text{fixed}} = e^{-r(T-t)}\mathbb{E}(\max(0, M_T - K))$$

$$P_t^{\text{fixed}} = e^{-r(T-t)}\mathbb{E}(\max(0, K - m_T))$$

- For Floating Strike Lookback Options

$$C_t^{\text{float}} = e^{-r(T-t)}\mathbb{E}(\max(0, S_T - m_T))$$

$$P_t^{\text{float}} = e^{-r(T-t)}\mathbb{E}(\max(0, M_T - S_T))$$

Geometric & Arithmetic Averaging

Taking a deeper look into Asian options, we can further define the averaging methodology into two definitions, geometric and arithmetic averaging. Arithmetic is the simple mean of our price over our period T. This methodology is more sensitive to changes in the underlying price given all prices are weighted equally. From an investment perspective, if we have volatile markets, we can use arithmetic averaging in order to capitalize on larger upward swings in our price. As for geometric averaging, we multiply our asset prices and then take the nth root (where n is the number of data points used). Given how geometric averaging operates, this means we're less sensitive to swings in our data set. From a risk management perspective there may be preference in using this method within a higher volatility market in order to have more control over how price swings would impact your option value.

- Geometric Average

$$G = \left(\prod_{i=0}^{t} S(i) \right)^{1/t}$$

$$G = \exp\left(\frac{1}{t} \sum_{i=0}^{t} \log S_i \right)$$

- Arithmetic Average

$$A_{\text{arith}} = \frac{1}{N} \sum_{i=1}^{N} S_i$$

Creating an Option Pricer

Using the concepts and formulas explained above, we develop a class function using python in order to model Asian & Lookback Options using Monte Carlo simulations under Euler-Maruyama methodology. The purpose behind having a class function is so I can call in different components of my model without having to re-write the code each time. Resources used to create a base structure was Kannan python lecture on Monte Carlo Simulations.

In [ ]:
```
# Using Kannan Lecture on Monte Carlo Simulation, we use the monte carlo code as a base to structure the below:
# Monte Carlo Option Pricing
class MonteCarloOptionPricing:
    '''Monte Carlo Option Pricing Engine'''
```

```
def __init__(self, S0:float, strike:float, rate:float, sigma:float, dte:int, nsim:int, timesteps:int=252) -> floa

    self.S0 = S0
    self.K = strike
    self.r = rate
    self.sigma = sigma
    self.T = dte
    self.N = nsim
    self.ts = timesteps
    self.df = np.exp(-self.r * self.T)

@property
def psudeorandomnumber(self):
    return np.random.standard_normal(self.N)

# Discrete simulated path
@property
def simulatepath_discrete(self):
    ''' simulate price path'''
    np.random.seed(4)

    # define dt
    dt = self.T / self.ts

    # simulate paths
    S = np.zeros((self.ts, self.N))
    S[0] = self.S0

    for i in range(0, self.ts-1):
        w = self.psudeorandomnumber
        S[i+1] = S[i] * (1 + self.r * dt + self.sigma * np.sqrt(dt) * w)

    return S

# Continuous simulated path
@property
def simulatepath_continuous(self):
    ''' Simulate price path with extremely fine time steps to approximate continuous sampling '''
    np.random.seed(4)

    fine_ts = self.ts * 100 # As you increase the time step, discrete gets closer to continuous
    dt = self.T / fine_ts
```

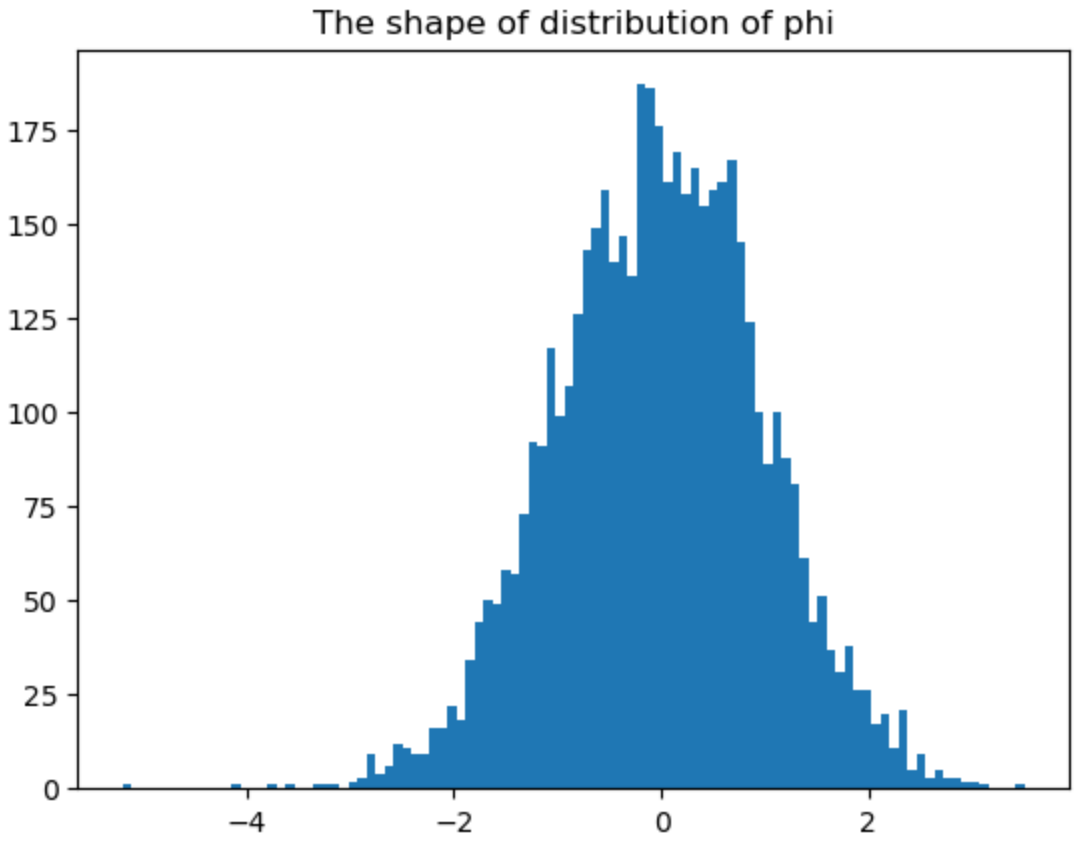
```
        # simulate paths with more steps
        S = np.zeros((fine_ts, self.N))
        S[0] = self.S0

        for i in range(0,fine_ts-1):
            w = self.psudeorandomnumber
            S[i+1] = S[i] * (1 + self.r * dt + self.sigma * np.sqrt(dt) * w)

        return S

    @property
    def asianoptiondiscrete(self):
        ''' calculate asian option payoff'''

        # Simulated path
        S = self.simulatepath_discrete

        # Arithmetic average the price
        A = S.mean(axis=0)

        # Geometric average the price
        G = np.exp(np.mean(np.log(S), axis=0))

        # Calculate the discounted value of the expeced payoff

        # Discrete arithmetic fixed strike
        asian_call_arifixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, A - self.K))
        asian_put_arifixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - A))

        # Discrete geometric fixed strike
        asian_call_geofixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, G - self.K))
        asian_put_geofixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - G))

        # Discrete arithmetic floating strike
        asian_call_arifloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S[-1] - A))
        asian_put_arifloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, A - S[-1]))

        # Discrete geometric floating strike
        asian_call_geofloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S[-1] - G))
        asian_put_geofloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, G - S[-1]))
```

```
        return [asian_call_arifixed, asian_put_arifixed,
                asian_call_geofixed, asian_put_geofixed,
                asian_call_arifloat, asian_put_arifloat,
                asian_call_geofloat, asian_put_geofloat]

    @property
    def asianoptioncontinuous(self):
        ''' calculate asian option payoff'''

        # Simulated path
        S = self.simulatepath_continuous

        # Arithmetic average the price
        A = S.mean(axis=0)

        # Geometric average the price
        G = np.exp(np.mean(np.log(S), axis=0))

        # Calculate the discounted value of the expeced payoff

        # Continuous arithmetic fixed strike
        asian_call_arifixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, A - self.K))
        asian_put_arifixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - A))

        # Continuous geometric fixed strike
        asian_call_geofixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, G - self.K))
        asian_put_geofixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - G))

        # Continuous arithmetic floating strike
        asian_call_arifloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S[-1] - A))
        asian_put_arifloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, A - S[-1]))

        # Continuous geometric floating strike
        asian_call_geofloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S[-1] - G))
        asian_put_geofloat = np.exp(-self.r*self.T) * np.mean(np.maximum(0, G - S[-1]))

        return [asian_call_arifixed, asian_put_arifixed,
                asian_call_geofixed, asian_put_geofixed,
                asian_call_arifloat, asian_put_arifloat,
                asian_call_geofloat, asian_put_geofloat]

    @property
```

```
    def lookbackdiscrete(self):
        ''' calculate asian option payoff'''

        # Simulated path
        S = self.simulatepath_discrete

        # Maximum and minimum prices of the asset during the option's life
        S_max = S.max(axis=0)
        S_min = S.min(axis=0)
        # Final prices at the last time step
        final_S = S[-1, :]

        # Calculate the discounted value of the expeced payoff

        # discrete fixed strike
        lookback_call_fixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S_max - self.K))
        lookback_put_fixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - S_min))

        # discrete floating strike
        lookback_call_float = np.exp(-self.r*self.T) * np.mean(np.maximum(0, final_S - S_min))
        lookback_put_float = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S_max - final_S))


        return [lookback_call_fixed, lookback_put_fixed,
                lookback_call_float, lookback_put_float]

    @property
    def lookbackcontinuous(self):
        ''' calculate asian option payoff'''

        S = self.simulatepath_continuous

        # Maximum and minimum prices of the asset during the option's life
        S_max = S.max(axis=0)
        S_min = S.min(axis=0)
        # Final prices at the last time step
        final_S = S[-1, :]

        # Calculate the discounted value of the expeced payoff

        # continuous fixed strike
        lookback_call_fixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S_max - self.K))
```

```
        lookback_put_fixed = np.exp(-self.r*self.T) * np.mean(np.maximum(0, self.K - S_min))

        # continuous floating strike
        lookback_call_float = np.exp(-self.r*self.T) * np.mean(np.maximum(0, final_S - S_min))
        lookback_put_float = np.exp(-self.r*self.T) * np.mean(np.maximum(0, S_max - final_S))


        return [lookback_call_fixed, lookback_put_fixed,
                lookback_call_float, lookback_put_float]
```

In [ ]:
```
phi = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000, timesteps=252)

plt.hist(phi.psudeorandomnumber, bins = 100)
plt.title('The shape of distribution of phi')
plt.show()
```



In [ ]:
```
columns = ['Mean',
           'Standard Deviation']

rows = ['Values']

values = [np.mean(phi.psudeorandomnumber),
          np.std(phi.psudeorandomnumber)]

shape_dist = pd.DataFrame(np.array(values).reshape(1, 2), index=rows, columns=columns)
shape_dist
```

Out[ ]:

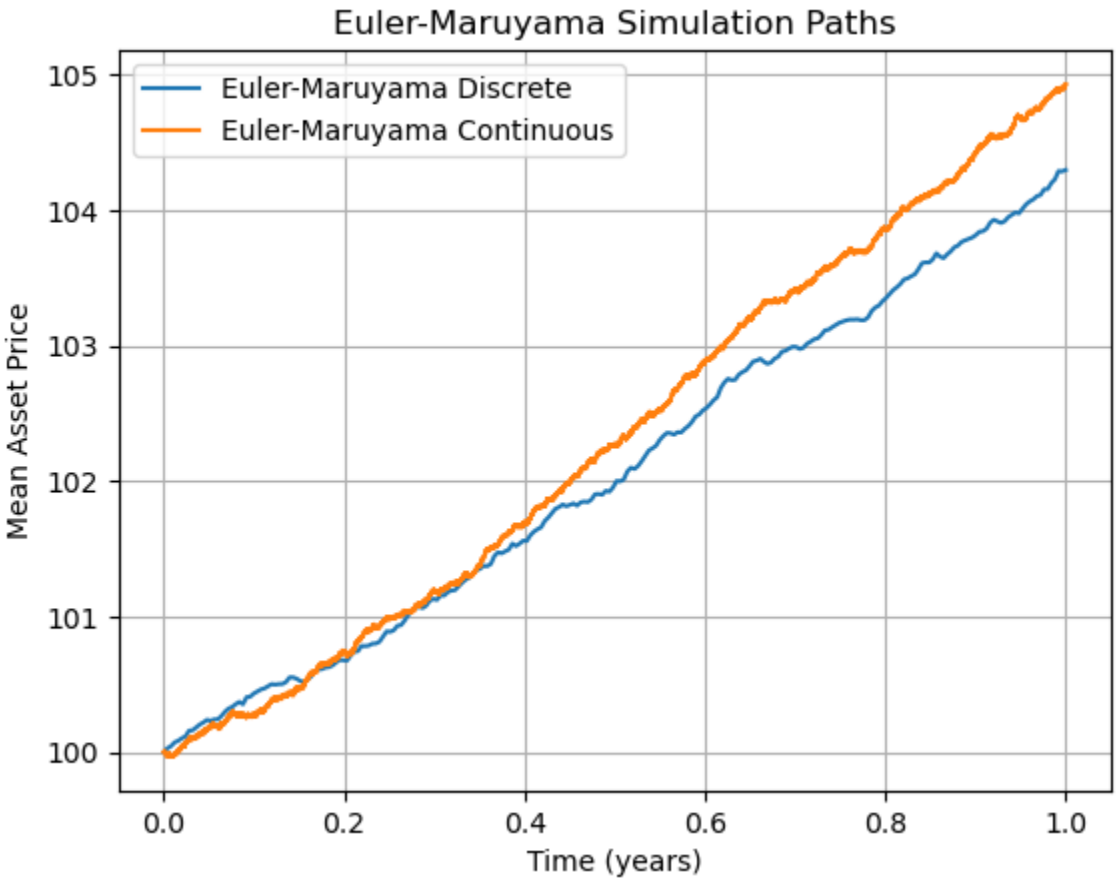
	Mean	Standard Deviation
Values	-0.006574	1.004543

Doing a simple check on our mean and standard deviation, we see that our mean is close to 0 and our standard deviations is close to one. While we could increase our accuracy by using more simulated values, for the purpose of consistency in this report, we use 5000 simulations for most pricings.

Euler-Maruyama Simulation Paths Comparison

In [ ]:
```
EM = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000, timesteps=252)

# Simulate and calculate the mean across simulations for each timestep
mean_discrete_path = np.mean(EM.simulatepath_discrete, axis=1)
mean_continuous_path = np.mean(EM.simulatepath_continuous, axis=1)

# Create an array representing the time in days for each timestep (this must scale for the continuous)
time_steps = np.linspace(0, EM.T, num=EM.ts)
fine_time_steps = np.linspace(0, EM.T, num=EM.ts * 100)

plt.plot(time_steps, mean_discrete_path, label='Euler-Maruyama Discrete')
plt.plot(fine_time_steps, mean_continuous_path, label='Euler-Maruyama Continuous')

plt.xlabel('Time (years)')
plt.ylabel('Mean Asset Price')
plt.title('Euler-Maruyama Simulation Paths')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



The chart above gives a plot between two simulation paths using Euler-Maruyama methodology, continuous and discrete. Notably, both lines start at the same point and have an increasing upward trend. From a volatility perspective, the discrete method looks to have more flattening of prices (which can be seen at time 0.43 and 0.78) while our continuous is more smooth overall. The reason behind the difference of smoothness are related to our continuous path having finer time scaling and therefore less apparent volatility between our individual points.

When comparing our two mean asset price values, an interesting outcome is continuous produces higher valuations. There are a few reasons for why this is occurring. First, the discrete path has larger time steps when approximating the stochastic process. This means that when capturing our points, there may be missing results during intermediate upturns. Using the same idea for missing

values, it's important to note that Euler-Maruyama is used to simulate our stochastic different equation (SDE). The randomness in the paths generated might not be fully represented under the discrete method. Diving deeper into our SDE, certainly we need to factor in the integration of both drift (trend) and volatility (random shocks). Depending on how drift and volatility operate with one another over time can lead to different outcomes given the frequency of updates. As continuous method is a more accurate integration of our drift given time intervals which are more fine, this could result in a higher value.

Asian Option Comparison

In [ ]:
```
mc_option_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000)

asian_prices_discrete = mc_option_pricing.asianoptiondiscrete
asian_prices_continuous = mc_option_pricing.asianoptioncontinuous

rows = ['Asian Call', 'Asian Put']
columns = ['Discrete Arith Fixed Strike', 'Discrete Arith Floating Strike',
           'Continuous Arith Fixed Strike', 'Continuous Arith Floating Strike',
           'Discrete Geo Fixed Strike', 'Discrete Geo Floating Strike',
           'Continuous Geo Fixed Strike', 'Continuous Geo Floating Strike'
          ]

combined_prices = [
    # Discrete Arithmetic Fixed Strike Call & Put
    asian_prices_discrete[0], asian_prices_discrete[1],
    # Discrete Arithmetic Floating Strike Call & Put
    asian_prices_discrete[4], asian_prices_discrete[5],
    # Continuous Arithmetic Fixed Strike Call & Put
    asian_prices_continuous[0], asian_prices_continuous[1],
    # Continuous Arithmetic Floating Strike Call & Put
    asian_prices_continuous[4], asian_prices_continuous[5],

    # Discrete Geometric Fixed Strike Call & Put
    asian_prices_discrete[2], asian_prices_discrete[3],
    # Discrete Geometric Floating Strike Call & Put
    asian_prices_discrete[6], asian_prices_discrete[7],
    # Continuous Geometric Fixed Strike Call & Put
    asian_prices_continuous[2], asian_prices_continuous[3],
    # Continuous Geometric Floating Strike Call & Put
    asian_prices_continuous[6], asian_prices_continuous[7],
```

```
]

data = pd.DataFrame(np.array(combined_prices).reshape(2, 8), index=rows, columns=columns)

data
```

Out[ ]:

	Discrete Arith Fixed Strike	Discrete Arith Floating Strike	Continuous Arith Fixed Strike	Continuous Arith Floating Strike	Discrete Geo Fixed Strike	Discrete Geo Floating Strike	Continuous Geo Fixed Strike	Continuous Geo Floating Strike
Asian Call	5.516280	3.548093	5.592283	3.471385	5.665728	3.459068	5.839570	3.352918
Asian Put	5.310308	3.672464	5.792579	3.341340	5.449892	3.578439	6.051043	3.229185

The table above provides numerical insight into how Asian options behave under various forms. Whats consistent between both calls and puts with fixed strikes is the geometric averaging results in a higher option price compared to arithmetic for both discrete and continuous samplings. The opposite can be seen however for floating strike where arithmetic averaging produces higher values on both calls and puts. From these observations, it's clear the methodology of how you're averaging plays a significant role in option pricing. The phenomenon however between fixed strike and floating strike having different behaviors from averaging methodology comes from the statistical properties which should be investigated.

For geometric averaging, we're less sensitive to higher fluctuations in values compared to arithmetic technique. While it may appear counterintuitive to why option prices would be higher given we're valuing using a method with less sensitivity, it's important to remember how the expectation of the payout would behave. For fixed strike, we would have a higher option price as geometric averaging would not reduce the option price as much when we have lower outliers in our asset pricing. In addition, asset prices typically follow a log-normal distribution *Resource used: Module 2 Asset Returns: Key Imperial Stylized Facts by Stephan Taylor*. As a geometric average is more aligned with this type of distribution, we would expect payoff results to be higher than arithmetic average.

Looking over arithmetic averaging, we are more sensitive to outliers in our data. In other words, this method captures skewness of our data more directly than geometric averaging. The payoff for our floating strike options depends on the difference on the final periods price and the average price. If we have skewness in our results which gives a higher average price, we would see higher

option pricing for both calls and puts.

While we analyzed continuous and discrete time sampling in the above graph, we also see how this plays a role in values between fixed strike and floating strike. Though continuous produces higher option valuations for fixed strike, we see the opposite effect on floating strike. Using the payout definition of floating strike, we know the expectations of option valuations comes from the excess or shortfall relative to the final asset price. As continuous results are more stable as it captures and represents the asset price over time, we would see lower average prices used in our payout model. This result would ultimately lead to a reduction in the options value compared to a more volatile average used from discrete sampling.

Lookback Option Comparison

In [ ]:
```
mc_option_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000)

lookback_prices_discrete = mc_option_pricing.lookbackdiscrete
lookback_prices_continuous = mc_option_pricing.lookbackcontinuous

rows = ['Lookback Call', 'Lookback Put']
columns = ['Discrete Fixed Strike', 'Discrete Floating Strike', 'Continuous Fixed Strike', 'Continuous Floating Strik

combined_prices = [
    lookback_prices_discrete[0], lookback_prices_discrete[2],
    lookback_prices_continuous[0], lookback_prices_continuous[2],
    lookback_prices_discrete[1], lookback_prices_discrete[3],
    lookback_prices_continuous[1], lookback_prices_continuous[3]
]

data = pd.DataFrame(np.array(combined_prices).reshape(2, 4), index=rows, columns=columns)

data
```

Out[ ]:

	Discrete Fixed Strike	Discrete Floating Strike	Continuous Fixed Strike	Continuous Floating Strike
Lookback Call	17.766790	16.153518	18.888443	17.148263
Lookback Put	12.064433	13.677706	12.454951	14.195131

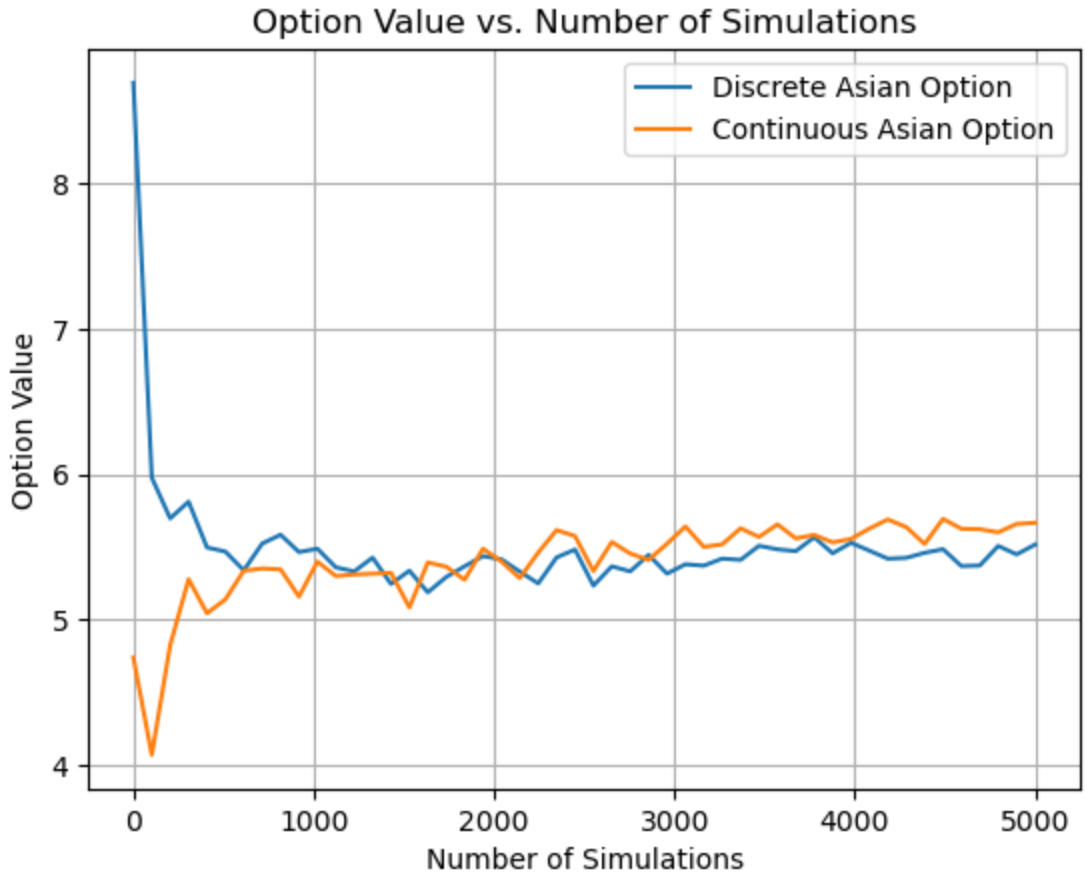
The table above represents Lookback option prices under fixed strike and floating strike using discrete & continuous time sampling.

A stark difference between Lookback and Asians that we can determine is Lookback having higher prices. Intrinsically, Lookback options are always in the money which would result in the exotics having a naturally occurring higher value to reflect this property. Unlike Asians, we also see more consistency between the behavior of time samplings. A unique feature of Lookback options is the payoff modeling. As we capture more price paths, it impacts the exotic valuations given we have more data to find new potential extremes. This detail also reflects the sensitivity of the option due to potential outliers.

An observation for various pricings not only stem from time samplings, but fixed strike and floating strike. To understand this, lets review how the two payoffs operate. Fixed strikes have a guaranteed payoff for the best asset price during the options duration. For a floating strike, the payoff is determined by the excess of the final asset price over the minimum average price for the duration of our option. Given this difference, we can say that for floating strikes, the payoff may be lower to fixed strike if the average is high given strong asset performance.

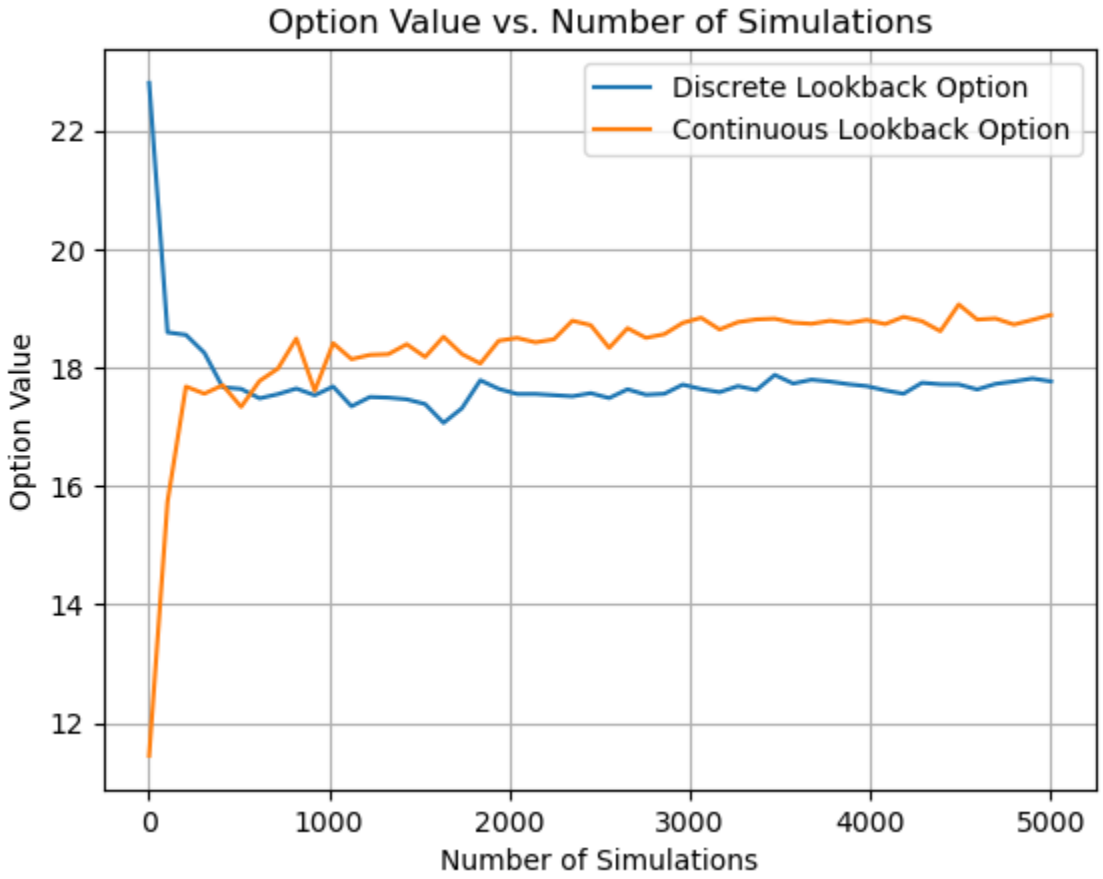
Varying Asian & Lookback Option Pricing for Simulations

In [ ]:
```
Simulation_Number = np.linspace(1, 5000, 50, dtype=int)
asian_val_discrete = []
asian_val_continuous = []

for nsim in Simulation_Number:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=int(nsim))

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(Simulation_Number, asian_val_discrete, label='Discrete Asian Option')
plt.plot(Simulation_Number, asian_val_continuous, label='Continuous Asian Option')

plt.xlabel('Number of Simulations')
plt.ylabel('Option Value')
plt.title('Option Value vs. Number of Simulations')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
Simulation_Number = np.linspace(1, 5000, 50, dtype=int)
lookback_val_discrete = []
lookback_val_continuous = []

for nsim in Simulation_Number:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=int(nsim))

    lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(Simulation_Number, lookback_val_discrete, label='Discrete Lookback Option')
plt.plot(Simulation_Number, lookback_val_continuous, label='Continuous Lookback Option')

plt.xlabel('Number of Simulations')
```

```
plt.ylabel('Option Value')
plt.title('Option Value vs. Number of Simulations')
plt.legend()
plt.grid(True)
plt.show()
```

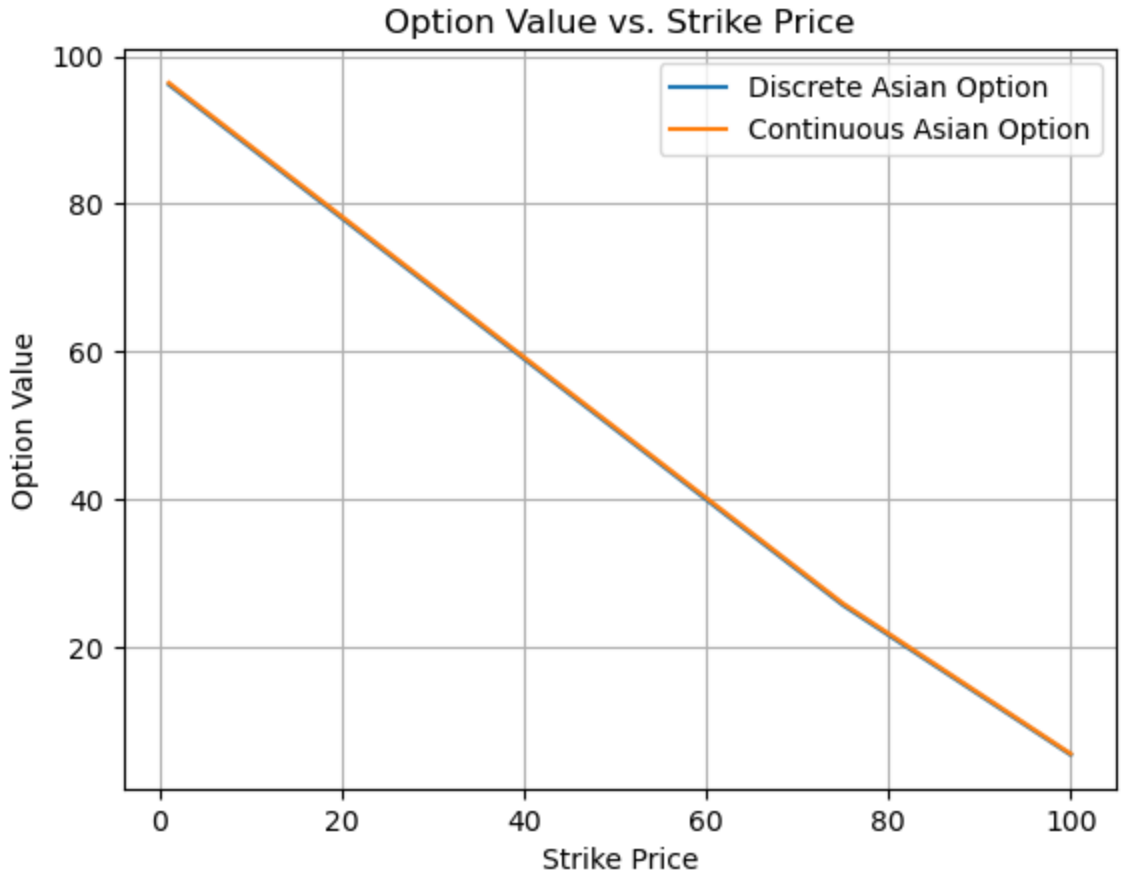


When comparing the impact of the option value due to the number of simulations ran, we see an interesting occurrence. While initially both continuous and discrete give widely different values, over a number of simulations, we converge to a similar price on both time samplings. The relationship describes the accuracy when increasing our simulations. While we may have a higher accuracy, the trade off becomes speed. From observing the graph, after 2000+ simulations, there does not appear to be a large difference in option values. Using this fact, in a trading environment where time is a critical component, we may lower the number of simulations ran from 5000 to 2000. Likewise, it's important to still understand that when we increase the number of simulations, we have a smoothening in our graph. If the purpose of using Monte-Carlo is for risk management and you can wait for a period of

time, increasing our simulated values still produces benefits and can be helpful in deeper option analysis.

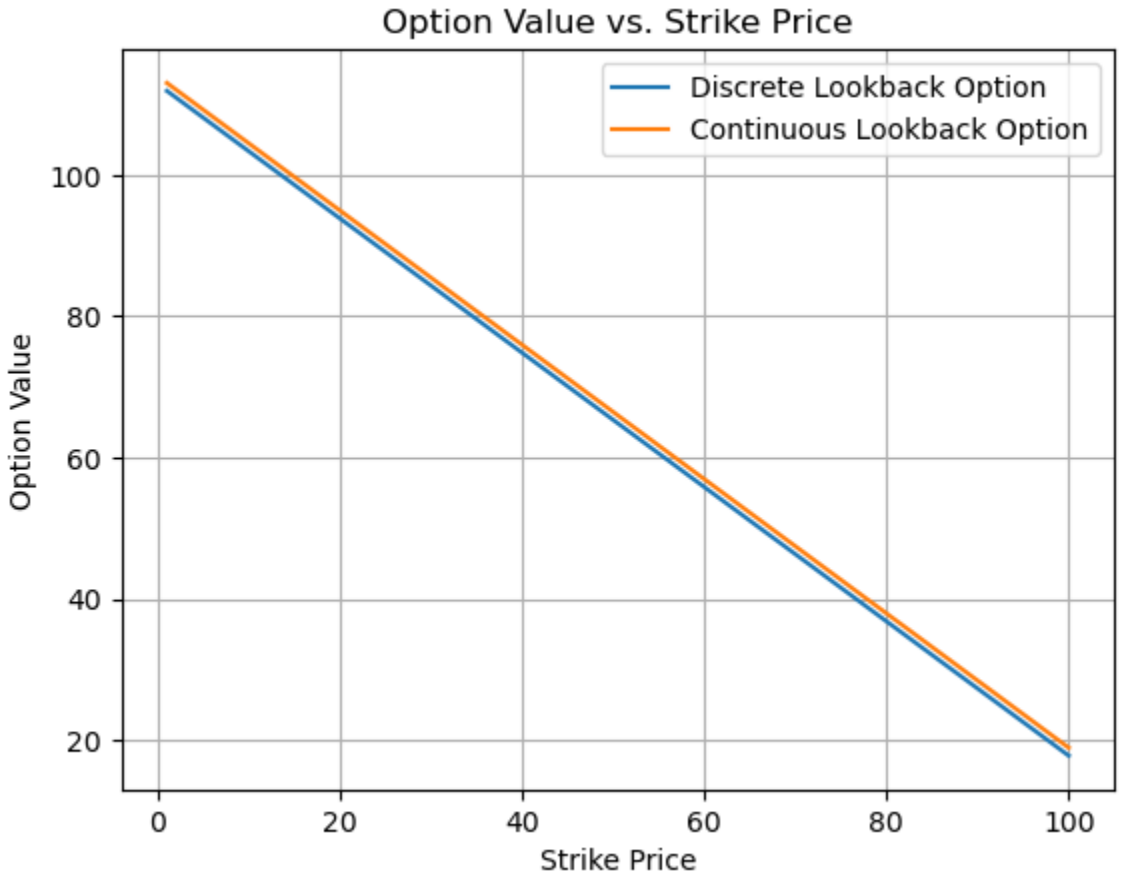
Varying Asian & Lookback Option Pricing for Strikes

In [ ]:
```
Simulation_Number = np.linspace(1, 100, 5, dtype=int)
asian_val_discrete = []
asian_val_continuous = []

for strike in Simulation_Number:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=strike, rate=0.05, sigma=0.2, dte=1, nsim=5000)

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(Simulation_Number, asian_val_discrete, label='Discrete Asian Option')
plt.plot(Simulation_Number, asian_val_continuous, label='Continuous Asian Option')

plt.xlabel('Strike Price')
plt.ylabel('Option Value')
plt.title('Option Value vs. Strike Price')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
Simulation_Number = np.linspace(1, 100, 5, dtype=int)
Lookback_val_discrete = []
Lookback_val_continuous = []

for strike in Simulation_Number:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=strike, rate=0.05, sigma=0.2, dte=1, nsim=5000)

    Lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    Lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(Simulation_Number, Lookback_val_discrete, label='Discrete Lookback Option')
plt.plot(Simulation_Number, Lookback_val_continuous, label='Continuous Lookback Option')

plt.xlabel('Strike Price')
```

```
plt.ylabel('Option Value')
plt.title('Option Value vs. Strike Price')
plt.legend()
plt.grid(True)
plt.show()
```

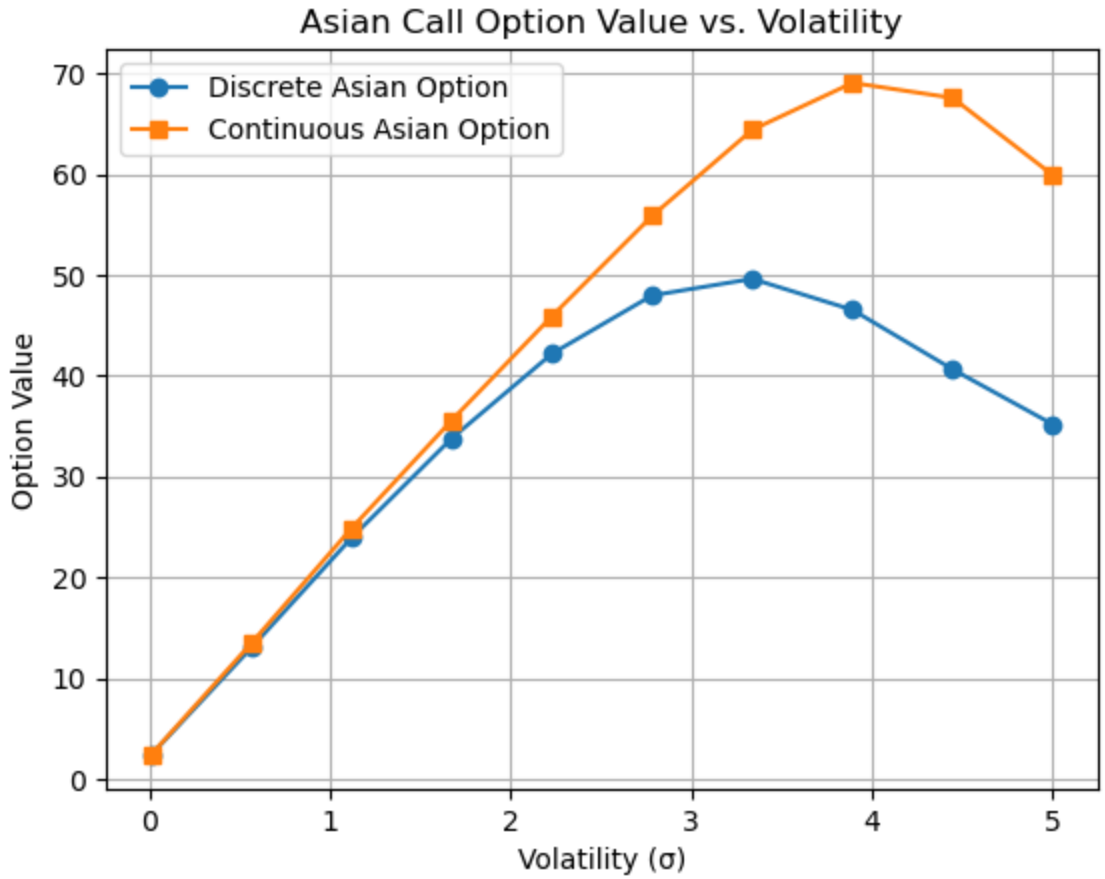


The above two graphs demonstrates the relationship between strike and option pricing for both Asian and Lookbacks. In both cases, even though the expectation of the payouts may be different, we see linearity in the relationship. As we increase the strike price, the value of the option goes down for both continuous and discrete time samplings. This can be explained by the intrinsic value of option pricing where for calls, your payout is determined by the amount by which your asset goes above the strike.

Varying Asian & Lookback Option Pricing for Volatility

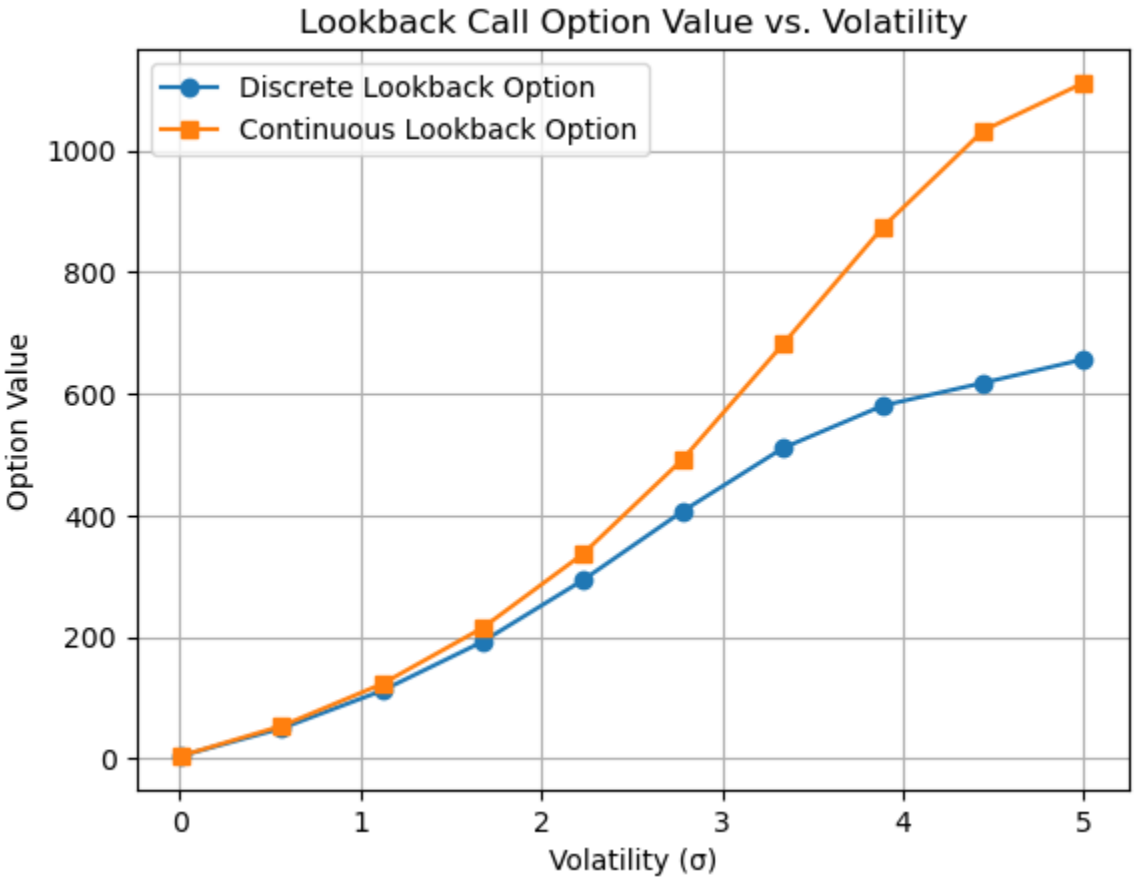
In [ ]:
```
Sigma_Range = np.linspace(0.01, 5, 10, dtype=float)
asian_val_discrete = []
asian_val_continuous = []

for sigma in Sigma_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=sigma, dte=1, nsim=5000)

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(Sigma_Range, asian_val_discrete, 'o-', label='Discrete Asian Option')
plt.plot(Sigma_Range, asian_val_continuous, 's-', label='Continuous Asian Option')

plt.xlabel('Volatility (σ)')
plt.ylabel('Option Value')
plt.title('Asian Call Option Value vs. Volatility')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_19876\1465888267.py:70: RuntimeWarning: invalid value encountered in log
  G = np.exp(np.mean(np.log(S), axis=0))
```



In [ ]:
```
Sigma_Range = np.linspace(0.01, 5, 10, dtype=float)
lookback_val_discrete = []
lookback_val_continuous = []

for sigma in Sigma_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=sigma, dte=1, nsim=5000)

    lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(Sigma_Range, lookback_val_discrete, 'o-', label='Discrete Lookback Option')
plt.plot(Sigma_Range, lookback_val_continuous, 's-', label='Continuous Lookback Option')

plt.xlabel('Volatility (σ)')
```

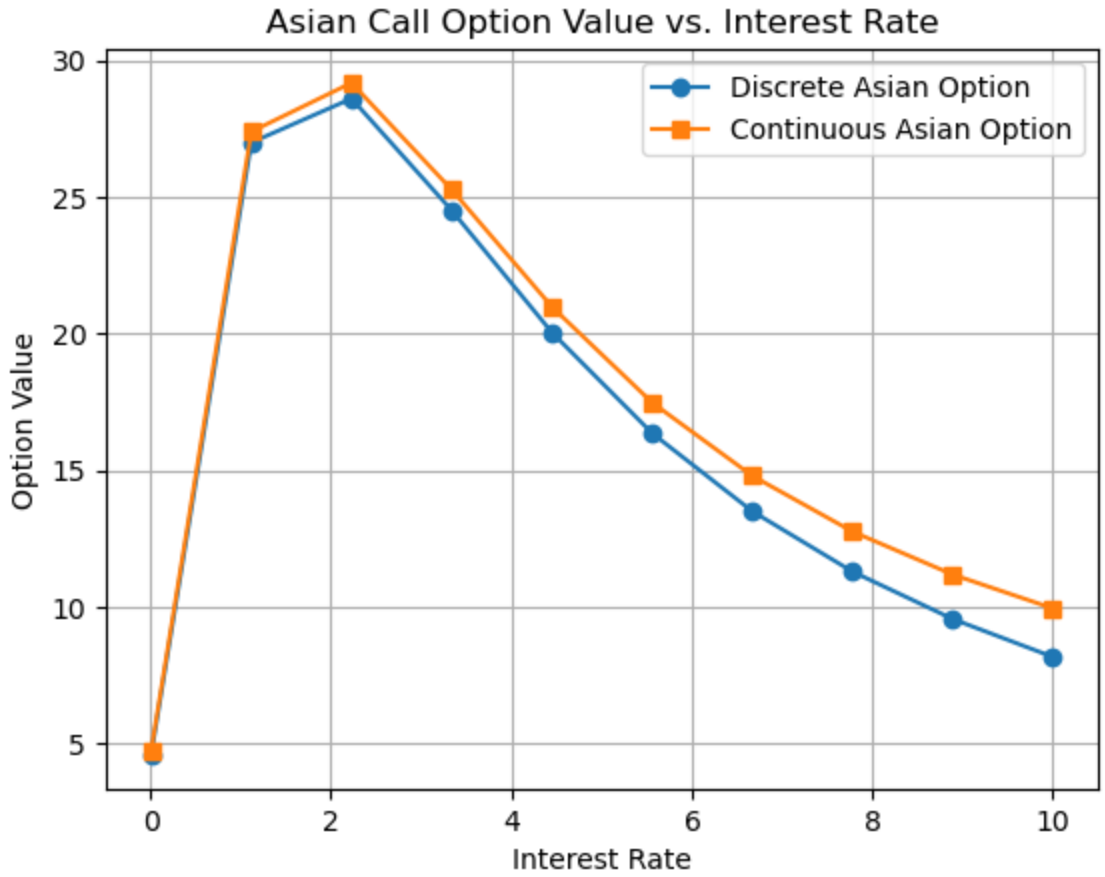
```
plt.ylabel('Option Value')
plt.title('Lookback Call Option Value vs. Volatility')
plt.legend()
plt.grid(True)
plt.show()
```



Unlike strike and number of simulations, volatility is a non linear relationship to option valuations. For volatility, we do see a sharp increase in option prices, but as volatility increases, the relationship no longer behaves as expected. For Asian options due to their payoff structuring, the incremental increase of volatility has less influence on the average, resulting in a mean reversion for option price. Lookback options on the other hand do not see this reversion effect, but instead have a plateau or convergence. Ultimately, the phenomenon seen is determined on the expectations of the payout.

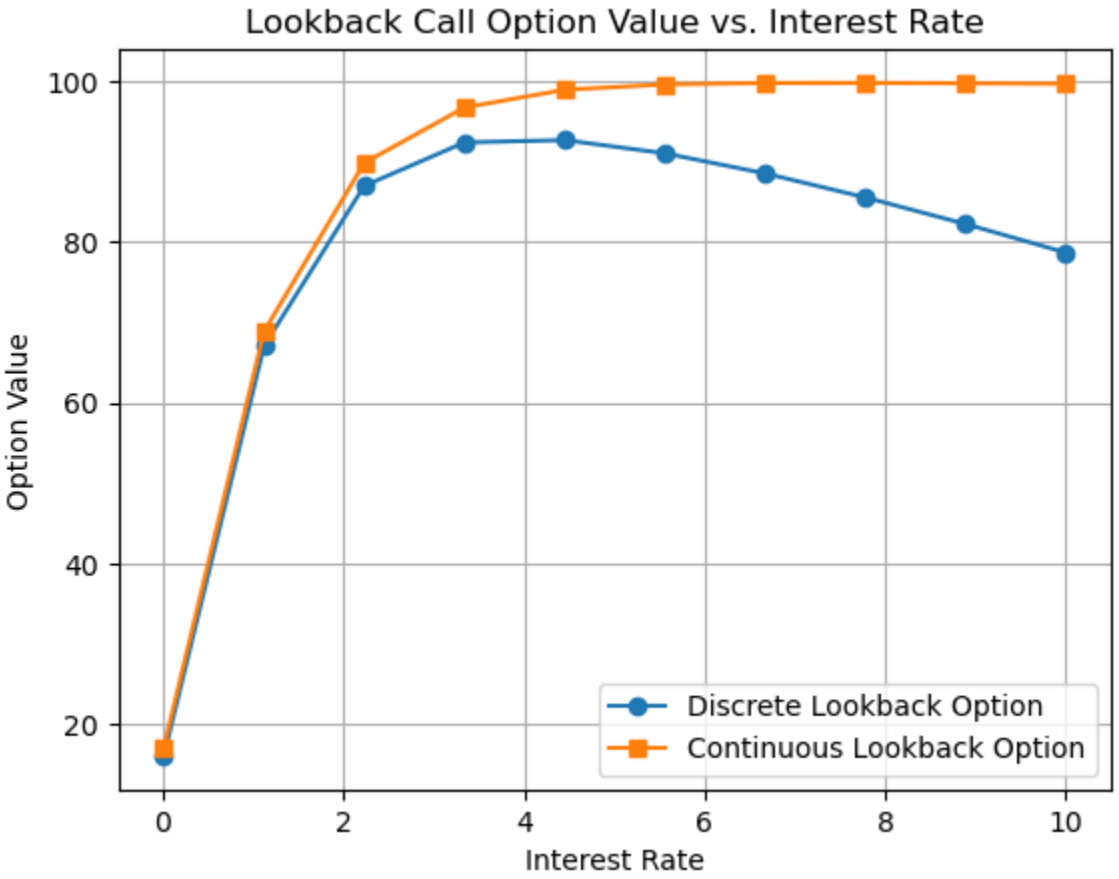
Varying Asian & Lookback Option Pricing for Rate

In [ ]:
```
Rate_Range = np.linspace(0.01, 10, 10, dtype=float)
asian_val_discrete = []
asian_val_continuous = []

for rate in Rate_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=rate, sigma=0.2, dte=1, nsim=5000)

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(Rate_Range, asian_val_discrete, 'o-', label='Discrete Asian Option')
plt.plot(Rate_Range, asian_val_continuous, 's-', label='Continuous Asian Option')

plt.xlabel('Interest Rate')
plt.ylabel('Option Value')
plt.title('Asian Call Option Value vs. Interest Rate')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
Rate_Range = np.linspace(0.01, 10, 10, dtype=float)
lookback_val_discrete = []
lookback_val_continuous = []

for rate in Rate_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=rate, sigma=0.2, dte=1, nsim=5000)

    lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(Rate_Range, lookback_val_discrete, 'o-', label='Discrete Lookback Option')
plt.plot(Rate_Range, lookback_val_continuous, 's-', label='Continuous Lookback Option')

plt.xlabel('Interest Rate')
```

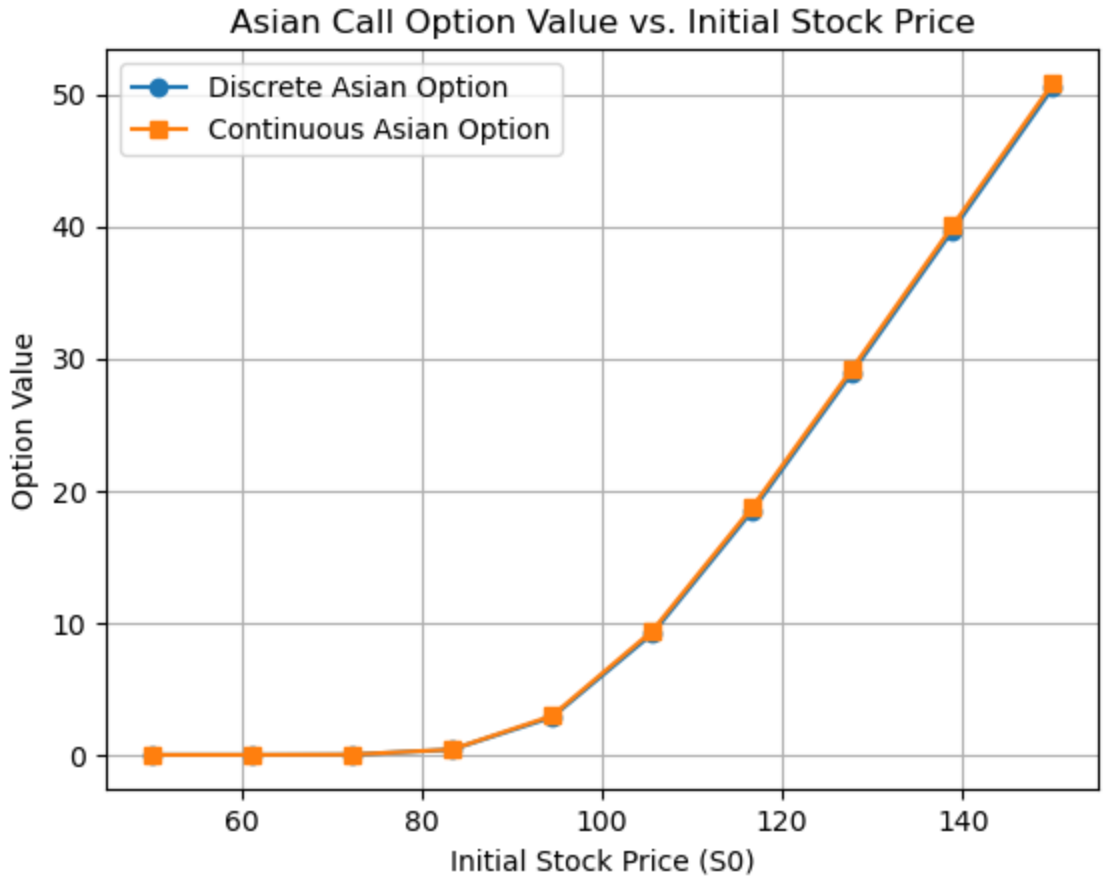
```
plt.ylabel('Option Value')
plt.title('Lookback Call Option Value vs. Interest Rate')
plt.legend()
plt.grid(True)
plt.show()
```



Asian & Lookbacks initially behave the same when interest rates increase, but for incremental increase beyond an interest rate value of 2, we start to see differences in the relationship. For Asians, we look at the payout structuring for both expectations, and discounting. For the expectation, since the option is average price dependent, we have a mitigation on spot price which strongly influences our valuation. On the other hand, for discounting, as the rate increases, the present value of the payout decreases. This combination leads to the reversion effect for Asians. When viewing Lookbacks, we also see the impact between discounting and payoffs. While the relationship is not as pronounced as we see with Asians, the discrete form alludes to similar behavior in which the discounting of the payoff starts decreasing the option value.

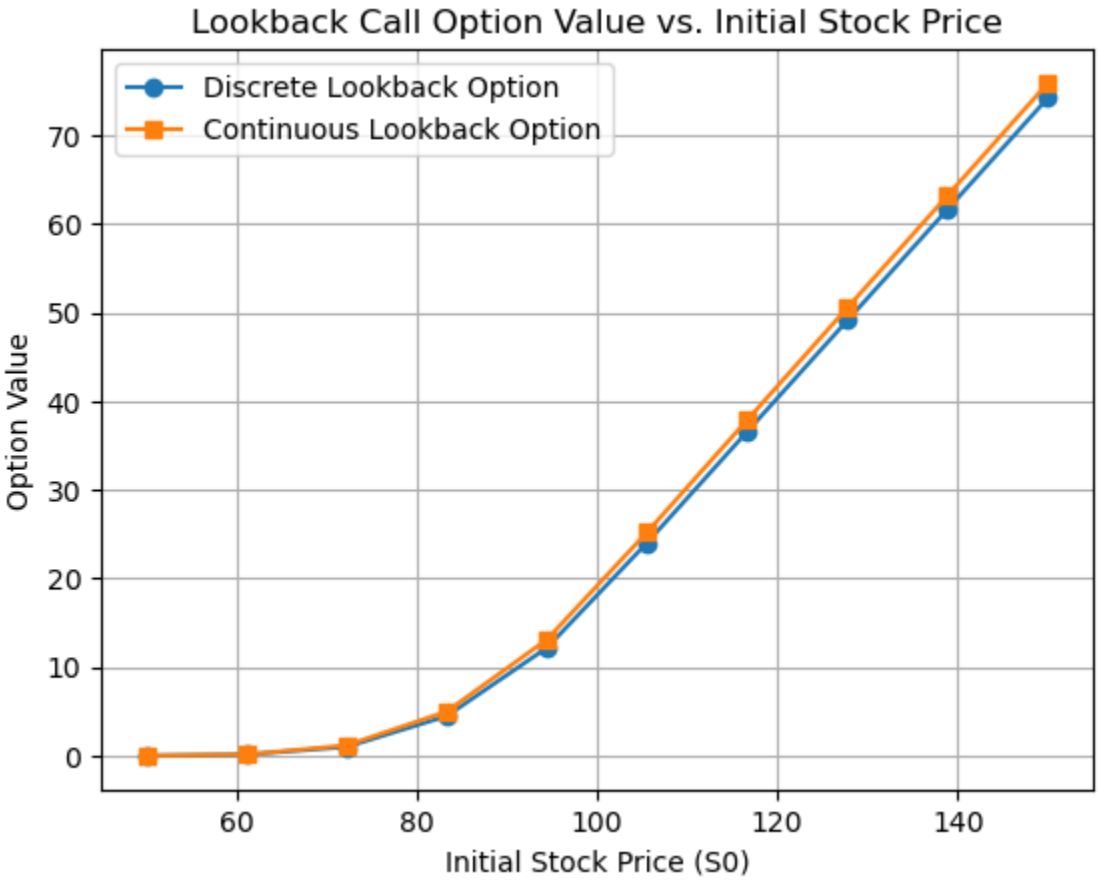
Varying Asian & Lookback Option Pricing for Spot

In [ ]:
```
S0_Range = np.linspace(50, 150, 10, dtype=float)
asian_val_discrete = []
asian_val_continuous = []

for S0 in S0_Range:
    mc_pricing = MonteCarloOptionPricing(S0=S0, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000)

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(S0_Range, asian_val_discrete, 'o-', label='Discrete Asian Option')
plt.plot(S0_Range, asian_val_continuous, 's-', label='Continuous Asian Option')

plt.xlabel('Initial Stock Price (S0)')
plt.ylabel('Option Value')
plt.title('Asian Call Option Value vs. Initial Stock Price')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
S0_Range = np.linspace(50, 150, 10, dtype=float)
lookback_val_discrete = []
lookback_val_continuous = []

for S0 in S0_Range:
    mc_pricing = MonteCarloOptionPricing(S0=S0, strike=100, rate=0.05, sigma=0.2, dte=1, nsim=5000)

    lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(S0_Range, lookback_val_discrete, 'o-', label='Discrete Lookback Option')
plt.plot(S0_Range, lookback_val_continuous, 's-', label='Continuous Lookback Option')

plt.xlabel('Initial Stock Price (S0)')
```

```
plt.ylabel('Option Value')
plt.title('Lookback Call Option Value vs. Initial Stock Price')
plt.legend()
plt.grid(True)
plt.show()
```

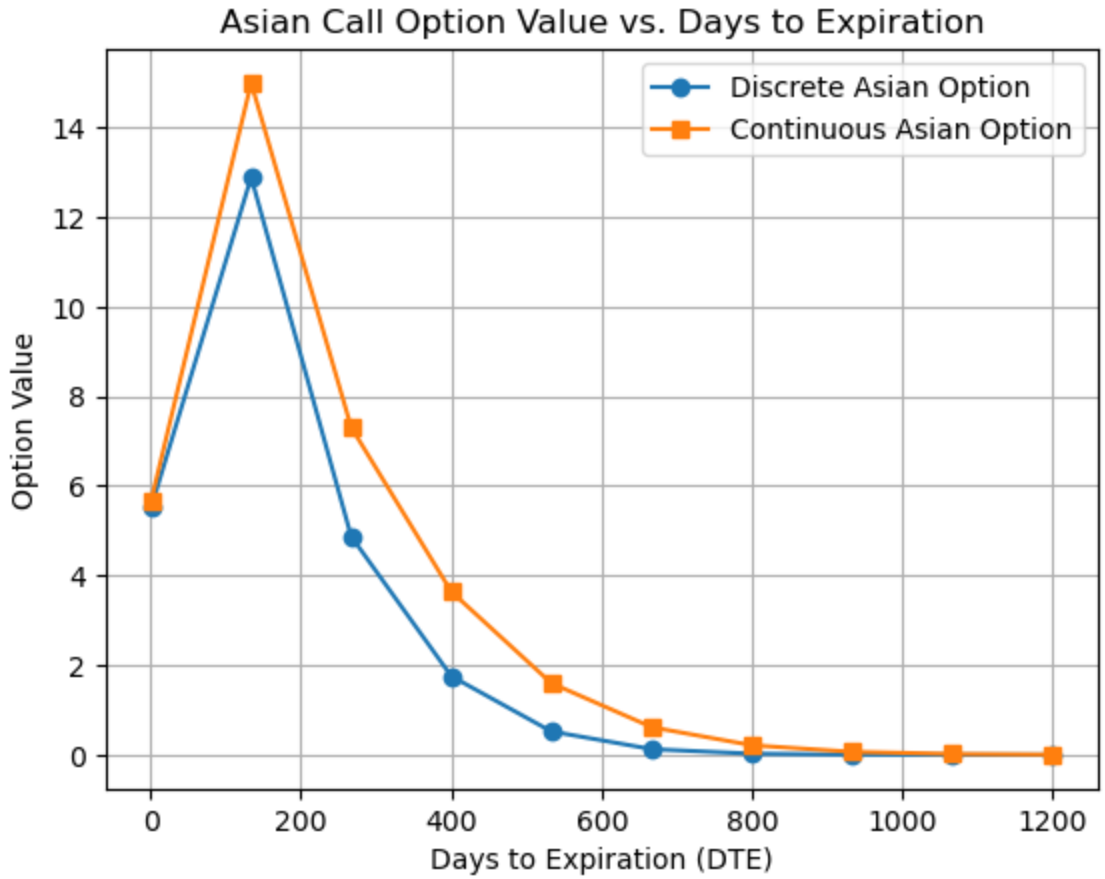


In both both exotic options, there's a clear relationship on an increasing spot price strengthening our option value. Initially, the options are worth very little given our spot is less than the strike. However, as spot increases, so does the intrinsic value of the option which is expected given the payout model we defined at the start of the report.

Varying Asian & Lookback Option Pricing for Time to Expiry

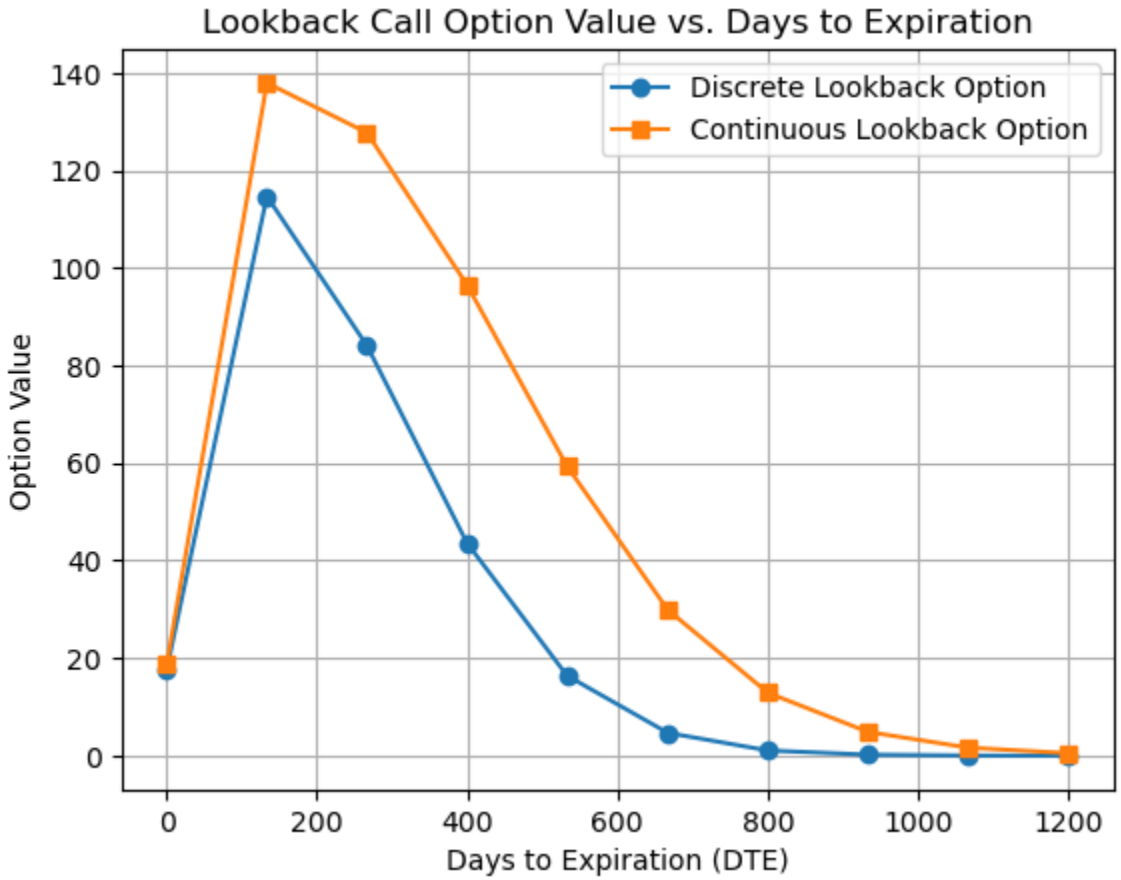
In [ ]:
```
DTE_Range = np.linspace(1, 1200, 10, dtype=int)
asian_val_discrete = []
asian_val_continuous = []

for dte in DTE_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=dte, nsim=5000)

    asian_val_discrete.append(mc_pricing.asianoptiondiscrete[0])
    asian_val_continuous.append(mc_pricing.asianoptioncontinuous[0])

plt.plot(DTE_Range, asian_val_discrete, 'o-', label='Discrete Asian Option')
plt.plot(DTE_Range, asian_val_continuous, 's-', label='Continuous Asian Option')

plt.xlabel('Days to Expiration (DTE)')
plt.ylabel('Option Value')
plt.title('Asian Call Option Value vs. Days to Expiration')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\natha\AppData\Local\Temp\ipykernel_19876\1465888267.py:70: RuntimeWarning: invalid value encountered in log
  G = np.exp(np.mean(np.log(S), axis=0))
```



In [ ]:
```
DTE_Range = np.linspace(1, 1200, 10, dtype=int)
lookback_val_discrete = []
lookback_val_continuous = []

for dte in DTE_Range:
    mc_pricing = MonteCarloOptionPricing(S0=100, strike=100, rate=0.05, sigma=0.2, dte=dte, nsim=5000)

    lookback_val_discrete.append(mc_pricing.lookbackdiscrete[0])
    lookback_val_continuous.append(mc_pricing.lookbackcontinuous[0])

plt.plot(DTE_Range, lookback_val_discrete, 'o-', label='Discrete Lookback Option')
plt.plot(DTE_Range, lookback_val_continuous, 's-', label='Continuous Lookback Option')

plt.xlabel('Days to Expiration (DTE)')
```

```
plt.ylabel('Option Value')
plt.title('Lookback Call Option Value vs. Days to Expiration')
plt.legend()
plt.grid(True)
plt.show()
```



For Asian options, initially we see a strong increase in the value of the security given an increase of Days to Expiration (DTE). While expected results would be that levels would remain stable or valuations to still increase as you have more time to end up in the money, we notice a negative trend. Given the payout of the option type where you're underlying price is average dependent, the averaging impact can reduce the effects of DTE and hence the price.

Seeing Lookbacks, we notice the same behavior as we did for Asians. Initially, there's a strong increase in option value given an increase for DTE, but overtime, we see a negative trend. Given the feature of a lookback where you find the optimal price over a

longer period of time, you would expect to see a constant increase. Given we have falling values, this may suggest that during longer horizons, the lookback feature is not as valuable. As DTE increases, the probability of the underlying hitting new extreme values may not increase given enough time has already passed. Therefore, the marginal increase of value added by each day that passes becomes less or negative.

Conclusion

Monte-Carlo can be an effective and accurate tool for exotic pricings. Before pricing an option however, it's always important to understand the behavior of the contract and how the value comes to be. For Asians as an example, we saw how a simple change between using discrete or continuous time samplings can impact values. Additionally, while averaging spot prices is a definition to how Asians are structured, the method in which you're taking the average also matters. Outside of geometric and arithmetic averaging producing different prices, we can use those methods to work around an investment thesis, particularly for volatility and potential risk management. Lookbacks differ from Asians not only in structure, but also in how much higher their prices are. Given their nature around being in the money, they are more expensive than Asian to reflect this intrinsic property. While there may not be as many variants for Lookbacks as there are for Asians, we still see how the type of time sampling effects the option values.

When pricing our options, one of the most important elements is our number of simulations ran. While more simulations lead to a smoothening to the convergence of price, we first need to understand what team within the financial world needs the valuations. In some instances, such as being on a trading floor, you may sacrifice accuracy for time. On a risk desk however, time may not be as important allowing for an increase in number of simulations used. Once we have an understanding for price, the other 5 elements: Spot, interest rate, volatility, strike, and time to expiry all have direct impacts on the options value. Interestingly, when viewing larger numbers, both spot and strike have inverse linear relationships. In other words, while a larger spot will result in a higher option value, a larger strike would subsequently result in a lower option value. Interest rates while similar in both Asians and Lookbacks, we could see how sensitive the two products were in terms of our discount factor applied. If interest rates are large enough, it may produce a downward result in the option value. Volatility is the most fascinating out of the parameters as it is dependent on the payoff structure of your option. For Lookbacks, results were more aligned with expectations given at a certain point of volatility, the option price converges. Asian options on the other hand, produced unexpected outcomes. The modeling for an Asian is dependent on averaging and given a higher volatility, we saw a mean reversion occur for the options price. Finally, for both Asians and Lookbacks, we saw how days to maturity impacted both securities in the same direction, but for different reasons. Ultimately, the falling of option value when viewing increasing DTE stemmed from the features & structuring of the two exotics.

Sources

- Module 2 | Asset Returns: Key Imperial Stylized Facts by Stephan Taylor
- Module 3 | Intro to Numerical Methods by Dr. Riaz Ahmad
- Module 3 | Exotic Options by Dr. Riaz Ahmad
- Module 3 | Understanding Volatility By Dr. Paul Wilmott
- Python Lab | Black Scholes Option Pricing by Kannan Singaravelu
- Python Lab | Monte Carlo Option Pricing by Kannan Singaravelu