Exotic Options

We will only discuss a few exotic options. The reason for studying the derivation of an option pricing formula is that it may better equip one to analyze new products. Hopefully, this chapter will be of some assistance in that regard. Of course, one cannot expect to derive a closed-form solution for the value of every product, and often numerical methods will be necessary. For a much more comprehensive presentation of exotic option pricing formulas, the book by Haug [@Haug] and the Excel software that accompanies it are highly recommended. Zhang [@Zhang] is also a comprehensive reference for exotics.

The valuation of an American call option on an asset paying a discrete cash dividend (rather than a continuous dividend) is considered in Section . Under a particular assumption on the volatility, valuing this option is very similar to valuing a compound option. Except for the assumption of a discrete dividend in Section , we will make the Black-Scholes assumptions throughout this chapter: there is a constant risk-free rate r and the underlying asset has a constant volatility σ and a constant dividend yield q.

The order in which exotics are presented in this chapter is based on the simplicity of the analysis—the chapter begins with the easiest to analyze and works towards the more difficult. This order is roughly the inverse of importance, the most important exotics in practice being barriers, baskets, spreads and Asians. In the cases of Asian and basket options, we will explain why there are no simple closed-form formulas (sums of lognormally distributed random variables are not lognormal). We will use these cases and discretely-sampled barriers and lookbacks as examples in the next chapter.

Forward-Start Options

A forward-start option is an option for which the strike price is set equal to the stock price at some later date. In essence, it is issued at the later date, with the strike price set at the money. For example, an executive may know that he is to be given an option grant at some later date with the strike price set equal to the stock price at that date.

Forward-Start Call Payoff

A forward-start call is defined by its maturity date T' and the date $T < T'$ at which the strike price is set. The value of a forward-start call at maturity is

$$\max(0, S(T') - S(T))\;.$$

Let

$$x = \begin{cases} 1 & \text{if } S(T') > S(T)\;, \\ 0 & \text{otherwise}\;. \end{cases}$$

Then, the value of the call at maturity can be written as

$$xS(T') - xS(T)\;.$$

### Numeraires

1. Use $V(t) = \mathrm{e}^{qt}S(t)$ as numeraire to price the payoff $xS(T')$. From the fundamental pricing **?@eq-formula**, the value at date~0 is

$$\mathrm{e}^{-qT'}S(0)E^V[x] = \mathrm{e}^{-qT'}S(0) \times \text{prob}^V(S(T') > S(T))\;.$$

2. To price the payoff $xS(T)$, use the following portfolio as numeraire:[1]purchase e^{-qT} shares of the stock at date 0 and reinvest dividends until date T. This will result in the ownership of one share at date T, worth $S(T)$ dollars. At date T, sell the share and invest the proceeds in the risk-free asset and hold this position until date T'. At date T', the portfolio will be worth $\mathrm{e}^{r(T'-T)}S(T)$. Let $Z(t)$ denote the value of this portfolio for each $0 \le t \le T'$. The fundamental pricing **?@eq-formula** implies that the value of receiving $xS(T)$ at date T' is

$$\begin{aligned} Z(0)E^Z\left[\frac{xS(T)}{Z(T')}\right] &= \mathrm{e}^{-qT}S(0)E^Z\left[\frac{xS(T)}{\mathrm{e}^{r(T'-T)}S(T)}\right] \\ &= \mathrm{e}^{-qT-r(T'-T)}S(0)E^Z[x] \\ &= \mathrm{e}^{-qT-r(T'-T)}S(0) \times \text{prob}^Z(S(T') > S(T))\;. \end{aligned}$$

[1]We are going to use equation~**?@eq-probSnumeraire** at date T' to define the probabilities, because it will not be known until date T' whether the event $S(T') > S(T)$ is true. Thus, we need the price of a numeraire asset at date T'. We would like this price to be a constant times $S(T)$, which is what we will obtain. An equivalent numeraire is to make a smaller investment in the same portfolio: start with $\mathrm{e}^{-r(T'-T)-qT}$ shares. This results in a final value of $S(T)$ at date T'. As will be seen, this is useful for deriving the put-call parity relation for forward-start options.

Calculating Probabilities

1. As in the case of a share digital, we know that

$$\log S(t) = \log S(0) + \left(r - q + \frac{1}{2}\sigma^2\right) t + \sigma B^*(t)$$

for all $t > 0$, where B^* is a Brownian motion when V is used as the numeraire. Taking $t = T'$ and $t = T$ and subtracting yields

$$\log S(T') - \log S(T) = \left(r - q + \frac{1}{2}\sigma^2\right)(T' - T) + \sigma\left[B^*(T') - B^*(T)\right] \;.$$

Hence, $S(T') > S(T)$ if and only if

$$-\frac{B^*(T') - B^*(T)}{\sqrt{T' - T}} < \frac{\left(r - q + \frac{1}{2}\sigma^2\right)(T' - T)}{\sigma\sqrt{T' - T}} \;.$$

The random variable on the left hand side is a standard normal, so

$$\text{prob}^V(S(T') > S(T)) = \mathrm{N}(d_1) \;,$$

where

$$d_1 = \frac{\left(r - q + \frac{1}{2}\sigma^2\right)(T' - T)}{\sigma\sqrt{T' - T}} = \frac{\left(r - q + \frac{1}{2}\sigma^2\right)\sqrt{T' - T}}{\sigma} \;. \tag{1}$$

2. To calculate the probability $\text{prob}^Z(S(T') > S(T))$, note that between T and T', the portfolio with price Z earns the risk-free rate r. The same argument presented in **?@sec-s_girsanov** shows that between T and T' we have

$$\frac{\mathrm{d}S}{S} = (r - q)\,\mathrm{d}t + \sigma\,\mathrm{d}B^* \;,$$

where now B^* denotes a Brownian motion when Z is used as the numeraire. This implies as usual that

$$\mathrm{d}\log S = \left(r - q - \frac{1}{2}\sigma^2\right)\mathrm{d}t + \sigma\,\mathrm{d}B^* \;,$$

which means that

$$\log S(T') - \log S(T) = \left(r - q - \frac{1}{2}\sigma^2\right)(T' - T) + \sigma(B^*(T') - B^*(T)) \;.$$

Hence, $S(T') > S(T)$ if and only if

$$-\frac{B^*(T') - B^*(T)}{\sqrt{T' - T}} < \frac{\left(r - q - \frac{1}{2}\sigma^2\right)(T' - T)}{\sigma\sqrt{T' - T}} \;.$$

As before, the random variable on the left hand side is a standard normal, so

$$\text{prob}^Z(S(T') > S(T)) = \mathrm{N}(d_2) \;,$$

where

$$d_2 = \frac{\left(r - q - \frac{1}{2}\sigma^2\right)\sqrt{T' - T}}{\sigma} = d_1 - \sigma\sqrt{T' - T} \;. \tag{2}$$

Forward-Start Call Pricing Formula

Combining these results, we have:

Tip

The value of a forward-start call at date 0 is

$$e^{-qT'}S(0)\mathrm{N}(d_1) - e^{-qT-r(T'-T)}S(0)\mathrm{N}(d_2)\ , \tag{3}$$

where d_1 and d_2 are defined in Equation 1 - Equation 2.

Put-Call Parity

Forward-strike calls and puts satisfy a somewhat unusual form of put-call parity. The usual put-call parity is of the form:

$$\text{Call} + \text{Cash} \quad = \quad \text{Put} + \text{Underlying}\ .$$

The amount of cash is the amount that will accumulate to the exercise price at maturity; i.e., it is $e^{-rT'}K$. For forward-start calls and puts, the effective exercise price is $S(T)$, which is not known at date 0. However, the portfolio used as numeraire to value the second part of the payoff will be worth $e^{r(T'-T)}S(T)$ at date T', and by following the same strategy but starting with $e^{-r(T'-T)-qT}$ instead of e^{-qT} shares, we will have $S(T)$ dollars at date T'. The date–0 value of this portfolio should replace Cash in the above. Thus:

Tip

Put-call parity for forward-start calls and puts is

$$\text{Call Price} + e^{-r(T'-T)-qT}S(0) = \text{Put Price} + e^{-qT'}S(0)\ . \tag{4}$$

The new features in the option pricing formulas in this chapter are the use of the bivariate normal distribution function and sometimes the need to compute a critical (at-the-money) value of the underlying asset price. We will compute the critical values by bisection, in the same way that we computed implied volatilities for the Black-Scholes formula in **?@sec-c__blackscholes**.
The following is a fast approximation of the bivariate cumulative normal distribution function, accurate to six decimal places, due to Drezner [@Drezner]). For given numbers a and b, this function gives the probability that $\xi_1 < a$ and $\xi_2 < b$ where ξ_1 and ξ_2 are standard normal random variables with a given correlation ρ, which we must input.

```python
import numpy as np
from scipy.stats import norm

def binormal_prob(a, b, rho):
    x = np.array([0.24840615, 0.39233107, 0.21141819, 0.03324666, 0.00082485334])
    y = np.array([0.10024215, 0.48281397, 1.0609498, 1.7797294, 2.6697604])
    a1 = a / np.sqrt(2 * (1 - rho ** 2))
    b1 = b / np.sqrt(2 * (1 - rho ** 2))
    if a <= 0 and b <= 0 and rho <= 0:
        total_sum = 0
        for i in range(5):
            for j in range(5):
                z1 = a1 * (2 * y[i] - a1)
                Z2 = b1 * (2 * y[j] - b1)
                z3 = 2 * rho * (y[i] - a1) * (y[j] - b1)
                total_sum += x[i] * x[j] * np.exp(z1 + Z2 + z3)
        return total_sum * np.sqrt(1 - rho ** 2) / np.pi
    elif a <= 0 and b >= 0 and rho >= 0:
        return norm.cdf(a) - binormal_prob(a, -b, -rho)
    elif a >= 0 and b <= 0 and rho >= 0:
        return norm.cdf(b) - binormal_prob(-a, b, -rho)
    elif a >= 0 and b >= 0 and rho <= 0:
        total_sum = norm.cdf(a) + norm.cdf(b)
        return total_sum - 1 + binormal_prob(-a, -b, rho)
    elif a * b * rho > 0:
        rho1 = (rho * a - b) * np.sign(a) / np.sqrt(a ** 2 - 2 * rho * a * b + b ** 2)
        rho2 = (rho * b - a) * np.sign(b) / np.sqrt(a ** 2 - 2 * rho * a * b + b ** 2)
        Delta = (1 - np.sign(a) * np.sign(b)) / 4
        return binormal_prob(a, 0, rho1) + binormal_prob(b, 0, rho2) - Delta
print("BiNormalProb:", binormal_prob(0.1, 0.2, 0.3))
```

```
BiNormalProb: 0.3601084540357551
```

Notice that this function calls itself. This is an example of recursion.
The forward-start call pricing formula is of the same form as the Black-Scholes, Margrabe, Black, and Merton formulas, as discussed in **?@sec-s_matlabimplementations**. We can compute it with our `Generic_Option` pricing function.

```python
def generic_option(P1, P2, sigma, T):
    """
    Inputs:
    P1 = present value of asset to be received
    P2 = present value of asset to be delivered
    sigma = volatility
    T = time to maturity
    """
    x = (np.log(P1 / P2) + 0.5 * sigma ** 2 * T) / (sigma * np.sqrt(T))
    y = x - sigma * np.sqrt(T)
    N1 = norm.cdf(x)
    N2 = norm.cdf(y)
    return P1 * N1 - P2 * N2

def forward_start_call(S, r, sigma, q, Tset, TCall):
    P1 = np.exp(-q * TCall) * S
    P2 = np.exp(-q * Tset - r * (TCall - Tset)) * S
    return generic_option(P1, P2, sigma, TCall - Tset)

# Example usage
S = 100
K = 100
r = 0.05
sigma = 0.2
q = 0.02
T = 1
Div = 5
TDiv = 0.5
TCall = 1
N = 10

print("Forward Start Call:", forward_start_call(S, r, sigma, q, 0.5, TCall))
```

```
Forward Start Call: 6.244873136512808
```

Compound Options

A compound option is an option on an option, for example a call option on a call option or a call on a put. These options are useful for hedging when there is some uncertainty about the need for hedging which may be resolved by the exercise date of the compound option.

As speculative trades, they have the benefit of higher leverage than ordinary options. These options were first discussed by Geske [@Geske].

Call-on-a-Call Payoff

Let the underlying call option have exercise price K' and maturity T'. Consider an option maturing at $T < T'$ to purchase the underlying call at price K.

Let $C(t,S)$ denote the value at date t of the underlying call when the stock price is S (i.e., C is the Black-Scholes formula). It is of course rational to exercise the compound call at date T if the value of the underlying call exceeds K; i.e., if $C(T,S(T)) > K$. Let S^* denote the critical price such that $C(T,S^*) = K$. To calculate S^*, we need to solve

$$\text{Black\_Scholes\_Call(Sstar,Kprime,r,sigma,q,Tprime-T)} = \text{K}.$$

for S^*. We can do this by bisection or one of the other methods mentioned in **?@secs_impliedvolatility**. It is rational to exercise the compound option when $S(T) > S^*$.

When $S(T) > S^*$, exercise of the compound option generates a cash flow of $-K$ at date T. There is a cash flow (of $S(T') - K'$) at date T' only if the compound call is exercised and the underlying call finishes in the money. This is equivalent to:

$$S(T) > S^* \quad \text{and} \quad S(T') > K' \; . \tag{5}$$

Let

$$x = \begin{cases} 1 & \text{if } S(T) > S^* \; , \\ 0 & \text{otherwise} \; , \end{cases}$$

$$y = \begin{cases} 1 & \text{if } S(T) > S^* \text{ and } S(T') > K' \; , \\ 0 & \text{otherwise} \; . \end{cases}$$

The cash flows of the compound option are $-xK$ at date T and $yS(T') - yK'$ at date T'. We can value the compound option at date 0 by valuing these separate cash flows.

The cash flow $-xK$ is the cash flow from being short K digital options on the underlying asset with strike price S^* and maturity T. Therefore the value at date 0 of this cash flow is $-\mathrm{e}^{-rT}K\mathrm{N}(d_2)$, where

$$d_1 = \frac{\log\left(\frac{S(0)}{S^*}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}, \qquad d_2 = d_1 - \sigma\sqrt{T} \; . \tag{6}$$

Numeraires

The payoffs $yS(T)$ and yK' are similar to share digitals and digitals, respectively, except that the event $y = 1$ is more complex than we have previously encountered. However, we know from the analysis of share digitals and digitals that the values at date 0 of these payoffs are

$$\mathrm{e}^{-qT'}S(0) \times \mathrm{prob}^V(y=1) \quad \text{and} \quad \mathrm{e}^{-rT'}K' \times \mathrm{prob}^R(y=1)\;,$$

where $V(t) = \mathrm{e}^{qt}S(t)$ and $R(t) = \mathrm{e}^{rt}$.

Calculating Probabilities

We will calculate the two probabilities in terms of the bivariate normal distribution function.

1. The event $y = 1$ is equivalent to

$$\log S(0) + \left(r - q + \frac{1}{2}\sigma^2\right)T + \sigma B^*(T) > \log S^*$$

and

$$\log S(0) + \left(r - q + \frac{1}{2}\sigma^2\right)T' + \sigma B^*(T') > \log K'\;,$$

where B^* is a Brownian motion when the underlying asset (V) is used as the numeraire. These conditions can be rearranged as

$$-\frac{B^*(T)}{\sqrt{T}} < d_1 \quad \text{and} \quad -\frac{B^*(T')}{\sqrt{T'}} < d_1'\;, \tag{7}$$

where d_1 is defined in Equation 6, and

$$d_1' = \frac{\log\left(\frac{S(0)}{K'}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T'}{\sigma\sqrt{T'}}\;, \qquad d_2' = d_1' - \sigma\sqrt{T'}\;. \tag{8}$$

The two standard normal variables on the left-hand sides in Equation 7 have a covariance equal to

$$\frac{1}{\sqrt{TT'}}\mathrm{cov}(B(T), B(T')) = \frac{1}{\sqrt{TT'}}\mathrm{cov}(B(T), B(T)) = \sqrt{\frac{T}{T'}}\;,$$

the first equality following from the fact that $B(T)$ is independent of $B(T') - B(T)$ and the second from the fact that the covariance of a random variable with itself is its variance. Hence, $\mathrm{prob}^V(y=1)$ is the probability that $a \le d_1$ and $b \le d_1'$, where a and b are standard normal random variables with covariance (= correlation coefficient) of $\sqrt{T/T'}$. We will write this probability as $\mathrm{M}\left(d_1, d_1', \sqrt{T/T'}\right)$. A program to approximate the bivariate normal distribution function M is provided in **?@sec-s__exotics__matlab**.

2. The calculation for $\text{prob}^R(y=1)$ is similar. The event $y=1$ is equivalent to

$$\log S(0) + \left(r - q + \frac{1}{2}\sigma^2\right) T + \sigma B^*(T) > \log S^* ,$$

and

$$\log S(0) + \left(r - q + \frac{1}{2}\sigma^2\right) T' + \sigma B^*(T') > \log K' ,$$

where B^* now denotes a Brownian motion under the risk-neutral measure. These are equivalent to

$$-\frac{B^*(T)}{\sqrt{T}} < d_2 \quad \text{and} \quad -\frac{B^*(T')}{\sqrt{T'}} < d_2' . \tag{9}$$

Hence, $\text{prob}^R(y=1) = \mathrm{M}\big(d_2, d_2', \sqrt{T/T'}\big)$.

Call-on-a-Call Pricing Formula

We conclude:

Tip

The value of a call on a call is

$$-\,\mathrm{e}^{-rT} K \mathrm{N}(d_2) + \mathrm{e}^{-qT'} S(0) \mathrm{M}\big(d_1, d_1', \sqrt{T/T'}\big) \\ - \mathrm{e}^{-rT'} K' \mathrm{M}\big(d_2, d_2', \sqrt{T/T'}\big) , \tag{10}$$

{#eq-marketmodel5}
where d_1 and d_2 are defined in Equation 6 and d_1' and d_2' are defined in Equation 8.

Put-Call Parity

European compound options with the same underlyings and strikes satisfy put-call parity in the usual way:

$$\text{Cash} + \text{Call} = \text{Underlying} + \text{Put} .$$

The portfolio on each side of this equation gives the owner the maximum of the strike and the value of the underlying at the option maturity. In the case of options on calls, put-call parity is specifically

$$\mathrm{e}^{-rT} K + \text{Value of call on call} \\ = \text{Value of underlying call} + \text{Value of put on call} ,$$

where K is the strike price of the compound options and T is their maturity date. Likewise, for options on puts, we have

$$\begin{multline*}\mathrm{e}^{-rT}K + \text{Value of call on put} \\ = \text{Value of underlying put} + \text{Value of put on put}\;.\end{multline*}$$

Thus, the value of a put on a call can be derived from the value of a call on a call. The value of a put on a put can be derived from the value of a call on a put, which we will now consider.

Call-on-a-Put Pricing Formula

Consider a call option maturing at T with strike K with the underlying being a put option with strike K' and maturity $T' > T$. The underlying of the put is the asset with price S and constant volatility σ. The call on the put will never be in the money at T and hence is worthless if $K > \mathrm{e}^{-r(T'-T)}K'$, because the maximum possible value of the put option at date T is $\mathrm{e}^{-r(T'-T)}K'$. So assume $K < \mathrm{e}^{-r(T'-T)}K'$.

Let S^* again denote the critical value of the stock price such that the call is at the money at date T when $S(T) = S^*$. This means that S^* solves

'Black_Scholes_Put(Sstar,Kprime,r,sigma,q,Tprime-T) K'.

We leave it as an exercise to confirm the following.

Tip

The value of a call on a put is

$$\begin{multline*}-\mathrm{e}^{-rT}K\mathrm{N}(-d_2) + \mathrm{e}^{-rT'}K'\mathrm{M}\Big(-d_2, -d_2', \sqrt{T/T'}\Big) \\ - \mathrm{e}^{-qT'}S(0)\mathrm{M}\Big(-d_1, -d_1', \sqrt{T/T'}\Big)\;, \quad (11)\end{multline*}$$

{#eq-callonaput}
where d_1 and d_2 are defined in Equation 6 and d_1' and d_2' are defined in Equation 8.

We will use bisection to find the critical price S^*. We can use $e^{q(T'-T)}(K + K')$ as an upper bound for S^* and 0 as a lower bound.^[We set the value of the call to be zero when the stock price is zero. The upper bound works because (by put-call parity and the fact that the put value is nonnegative) $C(T, S) \geq e^{-q(T'-T)}S - e^{-r(T'-T)}K'$. Therefore, when $S = e^{q(T'-T)}(K + K')$, we have $C(T, S) \geq K + K' - e^{-r(T'-T)}K' > K$. }
The following uses 10^{-6} as the error tolerance in the bisection.

```python
def black_scholes_call(S, K, r, sigma, q, T):
    """
    Inputs:
    S = initial stock price
    K = strike price
    r = risk-free rate
    sigma = volatility
    q = dividend yield
    T = time to maturity
    """
    if sigma == 0:
        return max(0, np.exp(-q * T) * S - np.exp(-r * T) * K)
    else:
        d1 = (np.log(S / K) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        d2 = d1 - sigma * np.sqrt(T)
        N1 = norm.cdf(d1)
        N2 = norm.cdf(d2)
        return np.exp(-q * T) * S * N1 - np.exp(-r * T) * K * N2


def call_on_call(S, Kc, Ku, r, sigma, q, Tc, Tu):
    tol = 1e-6
    lower = 0
    upper = np.exp(q * (Tu - Tc)) * (Kc + Ku)
    guess = 0.5 * lower + 0.5 * upper
    flower = -Kc
    fupper = black_scholes_call(upper, Ku, r, sigma, q, Tu - Tc) - Kc
    fguess = black_scholes_call(guess, Ku, r, sigma, q, Tu - Tc) - Kc
    while upper - lower > tol:
        if fupper * fguess < 0:
            lower = guess
            flower = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_call(guess, Ku, r, sigma, q, Tu - Tc) - Kc
        else:
            upper = guess
            fupper = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_call(guess, Ku, r, sigma, q, Tu - Tc) - Kc
    Sstar = guess

    d1 = (np.log(S / Sstar) + (r - q + sigma ** 2 / 2) * Tc) / (sigma * np.sqrt(Tc))
    d2 = d1 - sigma * np.sqrt(Tc)
    d1prime = (np.log(S / Ku) + (r - q + sigma ** 2 / 2) * Tu) / (sigma * np.sqrt(Tu))
    d2prime = d1prime - sigma * np.sqrt(Tu)
    rho = np.sqrt(Tc / Tu)
    N2 = norm.cdf(d2)
    M1 = binormal_prob(d1, d1prime, rho)
    M2 = binormal_prob(d2, d2prime, rho)

    return -np.exp(-r * Tc) * Kc * N2 + np.exp(-q * Tu) * S * M1 - np.exp(-r * Tu) * Ku * M2

# Example usage
S = 100
Kc = 10
Ku = 100
r = 0.05
sigma = 0.2
q = 0.02
Tc = 0.5
Tu = 1


print("Call on Call:", call_on_call(S, Kc, Ku, r, sigma, q, Tc, Tu))
```

```
Call on Call: 3.2568274676824167
```

The implementation of the call-on-a-put formula is of course very similar to that of a call-on-a-call. One difference is that there is no obvious upper bound for S^*, so we start with $2K'$ (= `2*K2`) and double this until the value of the put is below K. We can take 0 again to be the lower bound. Recall that we assume $K < \mathrm{e}^{-r(T'-T)}K'$ and the right-hand side of this is the value of the put at date T when $S(T) = 0$.

```
def black_scholes_put(S, K, r, sigma, q, T):
    """
    Inputs:
    S = initial stock price
    K = strike price
    r = risk-free rate
    sigma = volatility
    q = dividend yield
    T = time to maturity
    """
    if sigma == 0:
        return max(0, np.exp(-r * T) * K - np.exp(-q * T) * S)
    else:
        d1 = (np.log(S / K) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        d2 = d1 - sigma * np.sqrt(T)
        N1 = norm.cdf(-d1)
        N2 = norm.cdf(-d2)
        return np.exp(-r * T) * K * N2 - np.exp(-q * T) * S * N1

def call_on_put(S, Kc, Ku, r, sigma, q, Tc, Tu):
    tol = 1e-6
    lower = 0
    flower = np.exp(-r * (Tu - Tc)) * Ku - Kc
    upper = 2 * Ku
    fupper = black_scholes_put(upper, Ku, r, sigma, q, Tu - Tc) - Kc
    while fupper > 0:
        upper *= 2
        fupper = black_scholes_put(upper, Ku, r, sigma, q, Tu - Tc) - Kc

    guess = 0.5 * lower + 0.5 * upper
    fguess = black_scholes_put(guess, Ku, r, sigma, q, Tu - Tc) - Kc
    while upper - lower > tol:
        if fupper * fguess < 0:
            lower = guess
            flower = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_put(guess, Ku, r, sigma, q, Tu - Tc) - Kc
        else:
            upper = guess
            fupper = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_put(guess, Ku, r, sigma, q, Tu - Tc) - Kc
    Sstar = guess

    d1 = (np.log(S / Sstar) + (r - q + sigma ** 2 / 2) * Tc) / (sigma * np.sqrt(Tc))
    d2 = d1 - sigma * np.sqrt(Tc)
    d1prime = (np.log(S / Ku) + (r - q + sigma ** 2 / 2) * Tu) / (sigma * np.sqrt(Tu))
    d2prime = d1prime - sigma * np.sqrt(Tu)
    rho = np.sqrt(Tc / Tu)
    N2 = norm.cdf(-d2)
    M1 = binormal_prob(-d1, -d1prime, rho)
    M2 = binormal_prob(-d2, -d2prime, rho)
    return -np.exp(-r * Tc) * Kc * N2 + np.exp(-r * Tu) * Ku * M2 - np.exp(-q * Tu) * S * M1
# Example usage
S = 100
Kc = 10
Ku = 100
r = 0.05
sigma = 0.2
q = 0.02
Tc = 0.5
Tu = 1

print("Call on Put:", call_on_put(S, Kc, Ku, r, sigma, q, Tc, Tu))
```

```
Call on Put: 1.2998205534006573
```

American Calls with Discrete Dividends

It can be optimal to exercise an American call option early if the underlying asset pays a dividend. The optimal exercise date will be immediately prior to the asset going ex-dividend. Consider a call option maturing at T' on an asset that will pay a known cash dividend D at a known date $T < T'$. We assume there is no continuous dividend payment, so $q = 0$. For simplicity, we assume that the date of the dividend payment is also the date that the asset goes ex-dividend; i.e., ownership of the asset at any date $t < T$ entitles the owner to receive the dividend at date T. Under this assumption, it is reasonable also to assume that the stock price drops by D when the dividend is paid.

There is some ambiguity about how to define the asset price at the instant the dividend is paid—whether to include or exclude the dividend. We will let $S(T)$ denote the price including the dividend and denote the price excluding the dividend by $Z(T)$, so $Z(T) = S(T) - D$. In fact, it is convenient to let $Z(t)$ denote the price stripped of the dividend value at all dates $t \leq T$, so we will define

$$Z(t) = \begin{cases} S(t) - \mathrm{e}^{-r(T-t)}D & \text{if } t \leq T\,, \\ S(t) & \text{if } t > T\,. \end{cases}$$

Note that Z is the price of the following non-dividend-paying portfolio: buy one unit of the asset at date 0, borrow $\mathrm{e}^{-rT}D$ at date 0 to help finance the purchase, and use the dividend D at date T to retire the debt.

If we assume as usual that the asset price S has a constant volatility, then, using **?@eq-exponential1** for a geometric Brownian motion and letting B^* denote a Brownian motion under the risk-neutral measure, we have

$$\begin{aligned}
S(T') &= [S(T) - D] \exp\left\{(r - \sigma^2/2)(T' - T) + \sigma B^*(T') - \sigma B^*(T)\right\} \\
&= \left[S(0) \exp\left\{(r - \sigma^2/2)T + \sigma B^*(T)\right\} - D\right] \\
&\qquad \times \exp\left\{(r - \sigma^2/2)(T' - T) + \sigma B^*(T') - \sigma B^*(T)\right\} \\
&= S(0) \exp\left\{(r - \sigma^2/2)T' + \sigma B^*(T')\right\} \\
&\qquad - D \exp\left\{(r - \sigma^2/2)(T' - T) + \sigma B^*(T') - \sigma B^*(T)\right\} \; .
\end{aligned}$$

Thus, S will be a sum of lognormal random variables. A sum of lognormals is not itself lognormal, so S will not be lognormal, and we are unable to calculate the option value in a simple way.

We will assume instead that Z has a constant volatility σ. Thus, Z is the price of a non-dividend-paying portfolio, it satisfies the Black-Scholes assumptions, and we have $S(T') = Z(T')$. To value a European option, we would simply use $Z(0) = S(0) - \mathrm{e}^{-rT}D$ as the initial asset price and σ as the volatility.

American Call Payoff

If the call is not exercised before the dividend is paid at date T, then its value at date T will be

‘Black_Scholes_Call(Z,K,r,sigma,0,Tprime-T)‘

where $\mathtt{Z} = Z(T)$. Hence, exercise is optimal when

‘Z‘ + ‘D‘ − ‘K‘ > ‘Black_Scholes_Call(Z,K,r,sigma,0,Tprime-T)‘ .

A lower bound for the Black-Scholes call value on the right-hand side is $Z(T) - \mathrm{e}^{-r(T'-T)}K$. If $Z(T) + D - K$ is less than or equal to this lower bound, then exercise cannot be optimal. Thus, if $D - K$ is less than or equal to $-\mathrm{e}^{-r(T'-T)}K$, then exercise will never be optimal. In this circumstance, the dividend is simply too small to offset the time value of money on the exercise price K, and the value of the American call written on the asset with price S is the same as the value of the European call written on the non-dividend-paying portfolio with price Z.

On the other hand, if $D - K > -\mathrm{e}^{-r(T'-T)}K$, then exercise will be optimal for sufficiently large $Z(T)$. In this case, there is some Z^* such that the owner of the call will be indifferent about exercise, and exercise will be optimal for all $Z(T) > Z^*$. This Z^* is defined by

$$\text{`Zstar`} + \text{`D`} - \text{`K`} = \text{`Black\_Scholes\_Call(Zstar,K,r,sigma,0,Tprime-T)`}\ .$$

As in the previous section, we can compute Z^* by bisection.

Define

$$x = \begin{cases} 1 & \text{if } Z(T) > Z^*\ , \\ 0 & \text{otherwise}\ , \end{cases}$$
$$y = \begin{cases} 1 & \text{if } Z(T) \le Z^* \text{ and } Z(T') > K\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

Then the American call option will pay $[Z(T)+D-K]x$ at date T (due to early exercise) and $[Z(T')-K]y$ at date T' (due to exercise at maturity), if $D-K > -\mathrm{e}^{-r(T'-T)}K$.

Numeraires

Assume for now that $D-K > -\mathrm{e}^{-r(T'-T)}K$. The payoff $(D-K)x$ is the payoff of $D-K$ digital options maturing at T, and the payoff $Z(T)x$ is the payoff of one share digital on the portfolio with price Z. Therefore, the value of receiving $[Z(T)+D-K]x$ at date T is

$$Z(0)\mathrm{N}(d_1) + \mathrm{e}^{-rT}(D-K)\mathrm{N}(d_2)\ ,$$

where

$$\begin{aligned} d_1 &= \frac{\log\left(\frac{Z(0)}{Z^*}\right) + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \\ &= \frac{\log\left(\frac{S(0)-\mathrm{e}^{-rT}D}{Z^*}\right) + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\ , \end{aligned} \tag{12}$$

$$d_2 = d_1 - \sigma\sqrt{T}\ . \tag{13}$$

As in the previous section,[2] the value of receiving $[Z(T)-K]y$ at date T' is

$$Z(0) \times \mathrm{prob}^Z(y=1) - \mathrm{e}^{-rT'}K \times \mathrm{prob}^R(y=1)\ .$$

### Calculating Probabilities The calculations are very similar to the calculations we did for a call option on a call. In fact, they are exactly the same as we would do for a put option on a call.

[2]The only difference is that here Z is the price of a non-dividend-paying portfolio, so, in the notation of the previous section, we have $V(t) = Z(t)$.

1. The event $y = 1$ is equivalent to

$$\log Z(0) + \left(r + \frac{1}{2}\sigma^2\right) T + \sigma B^*(T) \leq \log Z^*$$

and

$$\log Z(0) + \left(r + \frac{1}{2}\sigma^2\right) T' + \sigma B^*(T') > \log K \;,$$

where B^* is a Brownian motion when the underlying asset (with price Z) is used as the numeraire. We can write this as

$$\frac{B^*(T)}{\sqrt{T}} < -d_1 \quad \text{and} \quad -\frac{B^*(T')}{\sqrt{T'}} < d_1' \;, \tag{14}$$

where d_1 is defined in Equation 12,

$$\begin{aligned} d_1' &= \frac{\log\left(\frac{Z(0)}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right) T'}{\sigma\sqrt{T'}} \\ &= \frac{\log\left(\frac{S(0) - \mathrm{e}^{-rT} D}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right) T'}{\sigma\sqrt{T'}} \end{aligned} \tag{15}$$

$$d_2' = d_1' - \sigma\sqrt{T'} \;. \tag{16}$$

The two standard normal variables on the left-hand sides in Equation 14 have a covariance equal to

$$-\frac{1}{\sqrt{TT'}}\text{cov}(B(T), B(T')) = -\frac{1}{\sqrt{TT'}}\text{cov}(B(T), B(T)) = -\sqrt{\frac{T}{T'}} \;.$$

Hence, $\text{prob}^Z(y = 1)$ is the probability that $a \leq -d_1$ and $b \leq d_1'$, where a and b are standard normal random variables with covariance (= correlation coefficient) of $-\sqrt{T/T'}$. We are writing this probability as $\mathrm{M}\left(-d_1, d_1', -\sqrt{T/T'}\right)$.

2. The calculation for $\text{prob}^R(y = 1)$ is similar. The event $y = 1$ is equivalent to

$$\log Z(0) + \left(r + \frac{1}{2}\sigma^2\right) T + \sigma B^*(T) \leq \log Z^*$$

and

$$\log Z(0) + \left(r + \frac{1}{2}\sigma^2\right) T' + \sigma B^*(T') > \log K \;,$$

where B^* now denotes a Brownian motion under the risk-neutral measure. These are equivalent to

$$\frac{B^*(T)}{\sqrt{T}} \leq -d_2 \quad \text{and} \quad -\frac{B^*(T')}{\sqrt{T'}} < d_2' \;. \tag{17}$$

Hence, $\text{prob}^R(y = 1) = \mathrm{M}\left(-d_2, d_2', -\sqrt{T/T'}\right)$.

American Call Pricing Formula

Tip

Under our assumptions, the value of an American call option maturing at T' with a dividend payment of D at date $T < T'$ is as follows.
If

$$D - K \le -\mathrm{e}^{-r(T'-T)}K \;,$$

then the value of the call is given by the Black-Scholes formula

$$[S(0) - \mathrm{e}^{-rT}D]\mathrm{N}(d_1') - \mathrm{e}^{-rT}K\mathrm{N}(d_2') \;,$$

where d_1' and d_2' are defined in Equation 15 - Equation 16. On the other hand, if

$$D - K > -\mathrm{e}^{-r(T'-T)}K \;,$$

then the value of the call is

$$\begin{aligned}[S(0) - \mathrm{e}^{-rT}D]\mathrm{N}(d_1) + \mathrm{e}^{-rT}(D-K)\mathrm{N}(d_2) \\ + [S(0) - \mathrm{e}^{-rT}D]\mathrm{M}\Big(-d_1, d_1', -\sqrt{T/T'}\Big) \\ - \mathrm{e}^{-rT'}K\mathrm{M}\Big(-d_2, d_2', -\sqrt{T/T'}\Big) \;, \end{aligned} \tag{18}$$

{#eq-americancall}
where d_1 and d_2 are defined in Equation 12 - Equation 13 and d_1' and d_2' are defined in Equation 15 - Equation 16.

To value an American call when there is one dividend payment before the option matures, we input the initial asset price $S(0)$ and then compute $Z(0) = X(0) - \mathrm{e}^{-rT}D$. If $D - K \le -\mathrm{e}^{-r(T'-T)}K$, we return the Black-Scholes value of a European call written on Z. Otherwise, we need to compute Z^* and our bisection algorithm requires an upper bound for Z^*, which would be any value of $Z(T)$ such that exercise at T is optimal. It is not obvious what this should be, so we start with K and keep doubling this until we obtain a value of $Z(T)$ at which exercise would be optimal. Then, we use the bisection algorithm to compute Z^* and finally compute the option value **?@eq-americancall**.

```
def american_call_dividend(S, K, r, sigma, Div, TDiv, TCall):
    LessDiv = S - np.exp(-r * TDiv) * Div
    if Div / K <= 1 - np.exp(-r * (TCall - TDiv)):
        return black_scholes_call(LessDiv, K, r, sigma, 0, TCall)

    upper = K
    while upper + Div - K < black_scholes_call(upper, K, r, sigma, 0, TCall - TDiv):
        upper *= 2

    tol = 1e-6
    lower = 0
    flower = Div - K
    fupper = upper + Div - K - black_scholes_call(upper, K, r, sigma, 0, TCall - TDiv)
    guess = 0.5 * lower + 0.5 * upper
    fguess = guess + Div - K - black_scholes_call(guess, K, r, sigma, 0, TCall - TDiv)
    while upper - lower > tol:
        if fupper * fguess < 0:
            lower = guess
            flower = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = guess + Div - K - black_scholes_call(guess, K, r, sigma, 0, TCall - TDi
        else:
            upper = guess
            fupper = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = guess + Div - K - black_scholes_call(guess, K, r, sigma, 0, TCall - TDi
    LessDivStar = guess

    d1 = (np.log(LessDiv / LessDivStar) + (r + sigma ** 2 / 2) * TDiv) / (sigma * np.sqrt(TD
    d2 = d1 - sigma * np.sqrt(TDiv)
    d1prime = (np.log(LessDiv / K) + (r + sigma ** 2 / 2) * TCall) / (sigma * np.sqrt(TCall)
    d2prime = d1prime - sigma * np.sqrt(TCall)
    rho = -np.sqrt(TDiv / TCall)
    N1 = norm.cdf(d1)
    N2 = norm.cdf(d2)
    M1 = binormal_prob(-d1, d1prime, rho)
    M2 = binormal_prob(-d2, d2prime, rho)

    return LessDiv * N1 + np.exp(-r * TDiv) * (Div - K) * N2 + LessDiv * M1 - np.exp(-r * TC

# Example usage

S = 100
K = 90
r = 0.05
sigma = 0.2
Div = 5
TDiv = 0.5
```

```
TCall = 1
N = 10

print("American Call with Dividend:", american_call_dividend(S, K, r, sigma, Div, TDiv, TCal
```

```
American Call with Dividend: 13.983999456433198
```

Choosers

A chooser option allows the holder to choose whether the option will be a put or call at some fixed date before the option maturity. Let T denote the date at which the choice is made, T_c the date at which the call expires, T_p the date at which the put expires, K_c the exercise price of the call, and K_p the exercise price of the put, where $0 < T < T_c$ and $0 < T < T_p$. A simple chooser has $T_c = T_p$ and $K_c = K_p$. A chooser is similar in spirit to a straddle: it is a bet on volatility without making a bet on direction. A simple chooser must be cheaper than a straddle with the same exercise price and maturity $T' = T_c = T_p$, because a straddle is always in the money at maturity, whereas a simple chooser has the same value as the straddle if it is in the money but is only in the money at T' when the choice made at T turns out to have been the best one.

Chooser Payoff

The value of the chooser at date T will be the larger of the call and put prices. Let S^* denote the stock price at which the call and put have the same value. We can find S^* by solving `Black\_Scholes\_Call(Sstar,Kc,r,sigma,q,Tc-T)` \= `Black\_Scholes\_Put(Sstar,Kp,r,sigma,q,Tp-T)`.

For a simple chooser with $K_c = K_p = K$ and $T_c = T_p = T'$, we can find S^* from the put-call parity relation at T, leading to $S^* = \mathrm{e}^{(q-r)(T'-T)}K$.

The call will be chosen when $S(T) > S^*$ and it finishes in the money if $S(T_c) > K_c$ at date T_c, so the payoff of the chooser is $S(T_c) - K_c$ when

$$S(T) > S^* \quad \text{and} \quad S(T_c) > K_c \,.$$

The payoff is $K_p - S(T_p)$ at date T_p when

$$S(T) < S^* \quad \text{and} \quad S(T_p) < K_p \,.$$

Let

$$x = \begin{cases} 1 & \text{if } S(T) > S^* \text{ and } S(T_c) > K_c \;, \\ 0 & \text{otherwise .} \end{cases}$$
$$y = \begin{cases} 1 & \text{if } S(T) < S^* \text{ and } S(T_p) < K_p \;, \\ 0 & \text{otherwise .} \end{cases}$$

Then the payoff of the chooser is $xS(T_c) - xK_c$ at date T_c and $yK_p - yS(T_p)$ at date T_p.

Numeraires

As in the analysis of compound options, the value of the chooser at date 0 must be

$$\begin{aligned} \mathrm{e}^{-qT_c}S(0) \times \text{prob}^V(x=1) \;-\; \mathrm{e}^{-rT_c}K_c \times \text{prob}^R(x=1) \\ +\; \mathrm{e}^{-rT_p}K_p \times \text{prob}^R(y=1) \;-\; \mathrm{e}^{-qT_p}S(0) \times \text{prob}^V(y=1)\;, \end{aligned} \tag{19}$$

{#eq-chooser1}

where we use $V(t) = \mathrm{e}^{qt}S(t)$ and $R(t) = \mathrm{e}^{rt}$ as numeraires.

Chooser Pricing Formula

Equation **?@eq-chooser1** and calculations similar to those of the previous two sections lead us to:

Tip

The value of a chooser option is

$$\begin{aligned} \mathrm{e}^{-qT_c}S(0)\mathrm{M}\Big(d_1, d_{1c}, \sqrt{T/T_c}\Big) - \mathrm{e}^{-rT_c}K_c\mathrm{M}\Big(d_2, d_{2c}, \sqrt{T/T_c}\Big) \\ + \mathrm{e}^{-rT_p}K_p\mathrm{M}\Big(-d_2, -d_{2p}, \sqrt{T/T_p}\Big) \\ - \mathrm{e}^{-qT_p}S(0)\mathrm{M}\Big(-d_1, -d_{1p}, \sqrt{T/T_p}\Big)\;, \end{aligned} \tag{20}$$

{#eq-chooser2}

where

$$d_1 = \frac{\log\left(\frac{S(0)}{S^*}\right)+(r-q+\frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}\,, \qquad d_2 = d_1 - \sigma\sqrt{T}\,,$$

$$d_{1c} = \frac{\log\left(\frac{S(0)}{K_c}\right)+(r-q+\frac{1}{2}\sigma^2)T_c}{\sigma\sqrt{T_c}}\,, \qquad d_{2c} = d_{1c} - \sigma\sqrt{T_c}\,,$$

$$d_{1p} = \frac{\log\left(\frac{S(0)}{K_p}\right)+(r-q+\frac{1}{2}\sigma^2)T_p}{\sigma\sqrt{T_p}}\,, \qquad d_{2p} = d_{1p} - \sigma\sqrt{T_p}\,.$$

To implement the bisection to compute S^*, we can take zero as a lower bound and $K_c + K_p$ as an upper bound.^[We take the call value to be zero and the put value to be $e^{-r(T_p-T)}K_p$ at date T when the stock price is zero. To see why the upper bound works, note that when the stock price is S at date T, the call is worth at least $S^* - K_c$ and the put is worth no more than K_p; i.e, $C \geq S - K_c$ and $P \leq K_p$. Therefore, $C - P \geq S - K_c - K_p$. Hence when $S = K_c + K_p$, we have $C - P \geq 0$. }

```
def chooser(S, Kc, Kp, r, sigma, q, T, Tc, Tp):
    tol = 1e-6
    lower = 0
    upper = np.exp(q * Tc) * (Kc + Kp)
    guess = 0.5 * Kc + 0.5 * Kp
    flower = -np.exp(-r * (Tp - T)) * Kp
    fupper = black_scholes_call(upper, Kc, r, sigma, q, Tc - T) - black_scholes_put(upper, K
    fguess = black_scholes_call(guess, Kc, r, sigma, q, Tc - T) - black_scholes_put(guess, K
    while upper - lower > tol:
        if fupper * fguess < 0:
            lower = guess
            flower = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_call(guess, Kc, r, sigma, q, Tc - T) - black_scholes_put(
        else:
            upper = guess
            fupper = fguess
            guess = 0.5 * lower + 0.5 * upper
            fguess = black_scholes_call(guess, Kc, r, sigma, q, Tc - T) - black_scholes_put(
    Sstar = guess

    d1 = (np.log(S / Sstar) + (r - q + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    d1c = (np.log(S / Kc) + (r - q + sigma ** 2 / 2) * Tc) / (sigma * np.sqrt(Tc))
    d2c = d1c - sigma * np.sqrt(Tc)
    d1p = (np.log(S / Kp) + (r - q + sigma ** 2 / 2) * Tp) / (sigma * np.sqrt(Tp))
    d2p = d1p - sigma * np.sqrt(Tp)
    rhoc = np.sqrt(T / Tc)
    rhop = np.sqrt(T / Tp)
    M1c = binormal_prob(d1, d1c, rhoc)
    M2c = binormal_prob(d2, d2c, rhoc)
    M1p = binormal_prob(-d1, -d1p, rhop)
    M2p = binormal_prob(-d2, -d2p, rhop)

    return np.exp(-q * Tc) * S * M1c - np.exp(-r * Tc) * Kc * M2c + np.exp(-r * Tp) * Kp * M

# Example usage
S = 100
Kc = 80
Kp = 80
r = 0.05
sigma = 0.2
q = 0.02
Tc = 1.5
Tp = 1.5
T=1

print("Chooser Option:", chooser(S, Kc, Kp, r, sigma, q, T, Tc, Tp))
```

```
Chooser Option: 24.965320659540296
```

Options on the Max or Min

We will consider here an option written on the maximum or minimum of two asset prices; for example, a call on the maximum pays

$$\max(0, \max(S_1(T), S_2(T)) - K) = \max(0, S_1(T) - K, S_2(T) - K)$$

at maturity T. There are also call options on $\min(S_1(T), S_2(T))$ and put options on the maximum and minimum of two (or more) asset prices. Pricing formulas for these options are due to Stulz [@Stulz], who also discusses applications. We will assume the two assets have constant dividend yields q_i, constant volatilities σ_i, and a constant correlation ρ.

Call-on-the-Max Payoff

To value the above option, define the random variables:

$$
\begin{aligned}
x &= \begin{cases} 1 & \text{if } S_1(T) > S_2(T) \text{ and } S_1(T) > K\;, \\ 0 & \text{otherwise}\;, \end{cases} \\
y &= \begin{cases} 1 & \text{if } S_2(T) > S_1(T) \text{ and } S_2(T) > K\;, \\ 0 & \text{otherwise}\;, \end{cases} \\
z &= \begin{cases} 1 & \text{if } S_1(T) > K \text{ or } S_2(T) > K\;, \\ 0 & \text{otherwise}\;. \end{cases}
\end{aligned}
$$

Then the value of the option at maturity is

$$
xS_1(T) + yS_2(T) - zK\;.
$$

### Numeraires Consider numeraires $V_1(t) = \mathrm{e}^{q_1 t}S_1(t)$, $V_2(t) = \mathrm{e}^{q_2 t}S_2(t)$, and $R(t) = \mathrm{e}^{rt}$. By familiar arguments, the value of the option at date 0 is

$$
\begin{multline}
\mathrm{e}^{-q_1 T}S_1(0) \times \mathrm{prob}^{V_1}(x=1) + \mathrm{e}^{-q_2 T}S_2(0) \times \mathrm{prob}^{V_2}(y=1) \\
- \mathrm{e}^{-rT}K \times \mathrm{prob}^{R}(z=1)\;.
\end{multline}
$$

Calculating Probabilities

1. We will begin by calculating $\mathrm{prob}^{V_1}(x=1)$. From the second and third examples in **?@sec-s_girsanov**, the asset prices satisfy

$$
\begin{aligned}
\frac{\mathrm{d}S_1}{S_1} &= (r - q_1 + \sigma_1^2)\,\mathrm{d}t + \sigma_1\,\mathrm{d}B_1^*\;, \\
\frac{\mathrm{d}S_2}{S_2} &= (r - q_2 + \rho\sigma_1\sigma_2)\,\mathrm{d}t + \sigma_2\,\mathrm{d}B_2^*\;,
\end{aligned}
$$

where B_1^* and B_2^* are Brownian motions when we use V_1 as the numeraire. Thus,

$$
\begin{aligned}
\log S_1(T) &= \log S_1(0) + \left(r - q_1 + \frac{1}{2}\sigma_1^2\right)T + \sigma_1 B_1^*(T)\;, \\
\log S_2(T) &= \log S_2(0) + \left(r - q_2 + \rho\sigma_1\sigma_2 - \frac{1}{2}\sigma_2^2\right)T + \sigma_2 B_2^*(T)\;.
\end{aligned}
$$

The condition $\log S_1(T) > \log K$ is therefore equivalent to

$$-\frac{1}{\sqrt{T}} B_1^*(T) < d_{11} \ , \tag{21}$$

and the condition $\log S_1(T) > \log S_2(T)$ is equivalent to

$$\frac{\sigma_2 B_2^*(T) - \sigma_1 B_1^*(T)}{\sigma\sqrt{T}} < d_1 \ , \tag{22}$$

where

$$\sigma = \sqrt{\sigma_1^2 - 2\rho\sigma_1\sigma_2 + \sigma_2^2} \ , \tag{23}$$

and

$$d_1 = \frac{\log\left(\frac{S_1(0)}{S_2(0)}\right) + \left(q_2 - q_1 + \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \ , \qquad d_2 = d_1 - \sigma\sqrt{T} \ , \tag{24}$$

$$d_{11} = \frac{\log\left(\frac{S_1(0)}{K}\right) + \left(r - q_1 + \frac{1}{2}\sigma_1^2\right) T}{\sigma_1\sqrt{T}} \ , \qquad d_{12} = d_{11} - \sigma_1\sqrt{T} \ . \tag{25}$$

The random variables on the left-hand sides of Equation 21 - Equation 22 have standard normal distributions and their correlation is

$$\rho_1 = \frac{\sigma_1 - \rho\sigma_2}{\sigma} \ .$$

Therefore,

$$\text{prob}^{V_1}(x = 1) = \mathrm{M}(d_{11}, d_1, \rho_1) \ ,$$

where M again denotes the bivariate normal distribution function.

2. The probability $\text{prob}^{V_2}(y = 1)$ is exactly symmetric to $\text{prob}^{V_1}(x = 1)$, with the roles of S_1 and S_2 interchanged. Note that the mirror image of d_1 defined in Equation 24 is

$$\frac{\log\left(\frac{S_2(0)}{S_1(0)}\right) + \left(q_1 - q_2 + \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \ ,$$

   which equals $-d_2$. Therefore,

$$\text{prob}^{V_2}(y = 1) = \mathrm{M}(d_{21}, -d_2, \rho_2) \ ,$$

   where

$$d_{21} = \frac{\log\left(\frac{S_2(0)}{K}\right) + \left(r - q_2 + \frac{1}{2}\sigma_2^2\right) T}{\sigma_2\sqrt{T}}, \qquad d_{22} = d_{21} - \sigma_2\sqrt{T} \ , \tag{26}$$

and

$$\rho_2 = \frac{\sigma_2 - \rho\sigma_1}{\sigma} \;.$$

3. As usual, we have

$$\log S_1(T) = \log S_1(0) + \left(r - q_1 - \frac{1}{2}\sigma_1^2\right) T + \sigma_1 B_1^*(T) \;,$$
$$\log S_2(T) = \log S_2(0) + \left(r - q_2 - \frac{1}{2}\sigma_2^2\right) T + \sigma_2 B_2^*(T) \;,$$

where B_1^* and B_2^* now denote Brownian motions under the risk-neutral measure. The event $z = 1$ is the complement of the event

$$S_1(T) \le K \quad \text{and} \quad S_2(T) \le K \;,$$

which is equivalent to

$$\frac{1}{\sqrt{T}} B_1^*(T) < -d_{12} \;, \tag{27}$$

and

$$\frac{1}{\sqrt{T}} B_2^*(T) < -d_{22} \;. \tag{28}$$

The random variables on the left-hand sides of Equation 27 and Equation 28 are standard normals and have correlation ρ. Therefore,

$$\text{prob}^R(z = 1) = 1 - \text{M}(-d_{12}, -d_{22}, \rho) \;.$$

Call-on-the-Max Pricing Formula

Tip

The value of a call option on the maximum of two risky asset prices with volatilities σ_1 and σ_2 and correlation ρ is

$$\mathrm{e}^{-q_1 T} S_1(0) \text{M}\left(d_{11}, d_1, \frac{\sigma_1 - \rho\sigma_2}{\sigma}\right) + \mathrm{e}^{-q_2 T} S_2(0) \text{M}\left(d_{21}, -d_2, \frac{\sigma_2 - \rho\sigma_1}{\sigma}\right) + \mathrm{e}^{-rT} K \text{M}(-d_{12}, -d_{22}, \rho) - \mathrm{e}^{-rT} K \;, \tag{29}$$

{#eq-callonmaxformula}
where σ is defined in Equation 23 and d_1, d_2, d_{11}, d_{12}, d_{21} and d_{22} are defined in Equation 24 - Equation 25.

The following code shows how to compute the price of a Call on the Max.

```
def call_on_max(S1, S2, K, r, sig1, sig2, rho, q1, q2, T):
    sigma = np.sqrt(sig2 ** 2 - 2 * rho * sig1 * sig2 + sig1 ** 2)
    d1 = (np.log(S1 / S2) + (q2 - q1 + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    d11 = (np.log(S1 / K) + (r - q1 + sig1 ** 2 / 2) * T) / (sig1 * np.sqrt(T))
    d12 = d11 - sig1 * np.sqrt(T)
    d21 = (np.log(S2 / K) + (r - q2 + sig2 ** 2 / 2) * T) / (sig2 * np.sqrt(T))
    d22 = d21 - sig2 * np.sqrt(T)
    rho1 = (sig1 - rho * sig2) / sigma
    rho2 = (sig2 - rho * sig1) / sigma
    M1 = binormal_prob(d11, d1, rho1)
    M2 = binormal_prob(d21, -d2, rho2)
    M3 = binormal_prob(-d12, -d22, rho)

    return np.exp(-q1 * T) * S1 * M1 + np.exp(-q2 * T) * S2 * M2 + np.exp(-r * T) * K * M3 -
# Example usage

S1 = 100
S2= 100
K = 100
r = 0.05
sigma1 = 0.2
sigma2 = 0.2
q1 = 0.02
q2= 0.01
T=1
rho= 0.1

print("Call on Max:", call_on_max(S1, S2, K, r, sigma1, sigma2, rho, q1, q2, T))
```

```
Call on Max: 15.890779319201613
```

Barrier Options

A down-and-out call pays the usual call value at maturity if and only if the stock price does not hit a specified lower bound during the lifetime of the option. If it does breach the lower barrier, then it is out. Conversely, a down-and-in call pays off only if the stock price *does* hit the lower bound. Up-and-out and up-and-in calls are defined similarly, and there are also put options of this sort. The out versions are called knock-outs and the in versions are called knock-ins.

Knock-ins can be priced from knock-outs and vice-versa. For example, the combination of a down-and-out call and a down-and-in call creates a standard European call, so the value of a down-and-in can be obtained by subtracting the value of a down-and-out from the value of a standard European call. Likewise, up-and-in calls can be valued by subtracting the value of an up-and-out from the value of a standard European call. Both knock-outs and knock-ins are of course less expensive than comparable standard options.

We will describe the pricing of a down-and-out call. The pricing of up-and-out calls and knock-out puts is similar. Often there are rebates associated with the knocking-out of a barrier option, but we will not include that feature here (see **?@sec-s_finitedifferencebarriers** however).

A down-and-out call provides a hedge against an increase in an asset price, just as does a standard call, for someone who is short the asset. The difference is that the down-and-out is knocked out when the asset price falls sufficiently. Presumably this is acceptable to the buyer because the need to hedge against high prices diminishes when the price falls. In fact, in this circumstance the buyer may want to establish a new hedge at a lower strike. However, absent re-hedging at a lower strike, the buyer of a knock-out call obviously faces the risk that the price may reverse course after falling to the knock-out boundary, leading to regret that the option was knocked out. The rationale for accepting this risk is that the knock-out is cheaper than a standard call. Thus, compared to a standard call, a down-and-out call provides cheaper but incomplete insurance.

The combination of a knock-out call and a knock-in call (or knock-out puts) with the same barrier and different strikes creates an option with a strike that is reset when the barrier is hit. This is a hedge that adjusts automatically to the market. An example is given in Probs.~**??** and~**??**.

Down-and-Out Call Payoff

Let L denote the lower barrier for the down-and-out call and assume it has not yet been breached at the valuation date, which we are calling date 0. Denote the minimum stock price realized during the remaining life of the contract by $z = \min_{0\leq t\leq T} S(t)$. In practice, this minimum is calculated at discrete dates (for example, based on daily closing prices), but we will assume here that the stock price is monitored continuously for the purpose of calculating the minimum.

The down-and-out call will pay $\max(0, S(T) - K)$ if $z > L$ and 0 otherwise, at its maturity T. Let

$$x = \begin{cases} 1 & \text{if } S(T) > K \text{ and } z > L\;, \\ 0 & \text{otherwise}\;. \end{cases}$$

Then the value of the down-and-out call at maturity is

$$xS(T) - xK\;.$$

### Numeraires As in other cases, the value at date 0 can be written as

$$\mathrm{e}^{-qT}S(0) \times \mathrm{prob}^V(x=1) - \mathrm{e}^{-rT}K \times \mathrm{prob}^R(x=1)\ ,$$

where $V(t) = \mathrm{e}^{qt}S(t)$ and $R(t) = \mathrm{e}^{rt}$.

Calculating Probabilities

To calculate $\mathrm{prob}^V(x=1)$ and $\mathrm{prob}^R(x=1)$, we consider two cases.

1. Suppose $K > L$. Define

$$y = \begin{cases} 1 & \text{if } S(T) > K \text{ and } z \leq L \\ 0 & \text{otherwise .} \end{cases}$$

The event $S(T) > K$ is equal to the union of the disjoint events $x = 1$ and $y = 1$. Therefore,

$$\begin{aligned} \mathrm{prob}^V(x=1) &= \mathrm{prob}^V(S(T)>K) - \mathrm{prob}^V(y=1)\ , \\ \mathrm{prob}^R(x=1) &= \mathrm{prob}^R(S(T)>K) - \mathrm{prob}^R(y=1)\ . \end{aligned}$$

As in the derivation of the Black-Scholes formula, we have

$$\mathrm{prob}^V(S(T)>K) = \mathrm{N}(d_1) \quad \text{and} \quad \mathrm{prob}^R(S(T)>K) = \mathrm{N}(d_2)\ , \tag{30}$$

where

$$d_1 = \frac{\log\left(\frac{S(0)}{K}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\ , \qquad d_2 = d_1 - \sigma\sqrt{T}\ . \tag{31}$$

Furthermore , defining

$$d_1' = \frac{\log\left(\frac{L^2}{KS(0)}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\ , \qquad d_2' = d_1' - \sigma\sqrt{T}\ , \tag{32}$$

it can be shown (see Appendix~**??**) that

$$\mathrm{prob}^V(y=1) = \left(\frac{L}{S(0)}\right)^{2\left(r-q+\frac{1}{2}\sigma^2\right)/\sigma^2} \mathrm{N}(d_1')\ , \tag{33}$$

$$\mathrm{prob}^R(y=1) = \left(\frac{L}{S(0)}\right)^{2\left(r-q-\frac{1}{2}\sigma^2\right)/\sigma^2} \mathrm{N}(d_2')\ . \tag{34}$$

2. Suppose $K \leq L$. Then the condition $S(T) > K$ in the definition of the event $x = 1$ is redundant: if $z > L \geq K$, then it is necessarily true that $S(T) > K$. Therefore, the probability (under either numeraire) of the event $x = 1$ is the probability that $z > L$. Define

$$y = \begin{cases} 1 & \text{if } S(T) > L \text{ and } z \leq L \,, \\ 0 & \text{otherwise} \,. \end{cases}$$

The event $S(T) > L$ is the union of the disjoint events $x = 1$ and $y = 1$. Therefore, as in the previous case (but now with K replaced by L),

$$\begin{aligned} \text{prob}^V(x=1) &= \text{prob}^V(S(T)>L) - \text{prob}^V(y=1) \,, \\ \text{prob}^R(x=1) &= \text{prob}^R(S(T)>L) - \text{prob}^R(y=1) \,. \end{aligned}$$

Also as before, we know that

$$\text{prob}^V(S(T)>L) = \mathrm{N}(d_1) \quad \text{and} \quad \text{prob}^R(S(T)>L) = \mathrm{N}(d_2) \,, \tag{35}$$

where now

$$d_1 = \frac{\log\left(\frac{S(0)}{L}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \,, \qquad d_2 = d_1 - \sigma\sqrt{T} \,. \tag{36}$$

Moreover, $\text{prob}^V(y=1)$ and $\text{prob}^R(y=1)$ are given by Equation 33 - Equation 34 but with K replaced by L, which means that

$$d_1' = \frac{\log\left(\frac{L}{S(0)}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \,, \qquad d_2' = d_1' - \sigma\sqrt{T} \,. \tag{37}$$

Down-and-Out Call Pricing Formula

Tip

The value of a continuously-sampled down-and-out call option with barrier L is

$$\mathrm{e}^{-qT}S(0)\left[\mathrm{N}(d_1) - \left(\frac{L}{S(0)}\right)^{2(r-q+\frac{1}{2}\sigma^2)/\sigma^2}\mathrm{N}(d_1')\right] - \mathrm{e}^{-rT}K\left[\mathrm{N}(d_2) - \left(\frac{L}{S(0)}\right)^{2(r-q-\frac{1}{2}\sigma^2)/\sigma^2}\mathrm{N}(d_2')\right] \,, \tag{38}$$

{#eq-downout100}

where

1. if $K > L$, d_1, d_2 , d_1' and d_2' are defined in Equation 31 - Equation 32,
2. if $K \leq L$, d_1, d_2, d_1' and d_2' are defined in Equation 36 - Equation 37.

The following code computes the price of a down and out call option.

```
def down_and_out_call(S, K, r, sigma, q, T, Barrier):
    if K > Barrier:
        a = S / K
        b = Barrier * Barrier / (K * S)
    else:
        a = S / Barrier
        b = Barrier / S

    d1 = (np.log(a) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    d1prime = (np.log(b) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2prime = d1prime - sigma * np.sqrt(T)
    N1 = norm.cdf(d1)
    N2 = norm.cdf(d2)
    N1prime = norm.cdf(d1prime)
    N2prime = norm.cdf(d2prime)
    x = 1 + 2 * (r - q) / (sigma ** 2)
    y = x - 2
    q1 = N1 - (Barrier / S) ** x * N1prime
    q2 = N2 - (Barrier / S) ** y * N2prime

    return np.exp(-q * T) * S * q1 - np.exp(-r * T) * K * q2
print("Down and Out Call:", down_and_out_call(S, K, r, sigma, q, T, 80))
```

```
Down and Out Call: 9.133306436498117
```

Lookbacks

A floating-strike lookback call pays the difference between the asset price at maturity and the minimum price realized during the life of the contract. A floating-strike lookback put pays the difference between the maximum price over the life of the contract and the price at maturity. Thus, the floating-strike lookback call allows one to buy the asset at its minimum price, and the floating-strike lookback put allows one to sell the asset at its maximum price. Of course, one pays upfront for this opportunity to time the market. These options were first discussed by Goldman, Sosin and Gatto [@GSG].

A fixed-strike lookback put pays the difference between a fixed strike price and the minimum price during the lifetime of the contract. Thus, a fixed-strike lookback put and a floating-strike lookback call are similar in one respect: both enable one to buy the asset at its minimum price. However, the put allows one to sell the asset at a fixed price whereas the call allows one to sell it at the terminal asset price. A fixed-strike lookback call pays the difference between the maximum price and a fixed strike price and is similar to a floating-strike lookback put in the sense that both enable one to sell the asset at its maximum price. Fixed-strike lookback options were first discussed by Conze and Viswanathan [@CV]. We will discuss the valuation of floating-strike lookback calls. As in the discussion of barrier options, we will assume that the price is continuously sampled for the purpose of computing the minimum.

Floating-Strike Lookback Call Payoff

As in the previous section, let z denote the minimum stock price realized over the *remaining lifetime of the contract.* This is not necessarily the minimum stock price realized during the entire lifetime of the contract. Let $S_{\min}$ denote the minimum stock price realized during the lifetime of the contract up to and including date 0, which is the date at which we are valuing the contract. The minimum stock price during the *entire lifetime of the contract* will be the smaller of z and $S_{\min}$. The payoff of the floating strike lookback call is $S(T) - \min(z, S_{\min})$.

Calculations

The value at date 0 of the piece $S(T)$ is simply $\mathrm{e}^{-qT}S(0)$. Using the result in Appendix~**??** on the distribution of z, it can be shown (see, e.g., Musiela and Rutkowski [@MR] for the details) that the value at date 0 of receiving

$$\min(z, S_{\min})$$

at date T is

$$\mathrm{e}^{-rT}S_{\min}\mathrm{N}(d_2) - \frac{\sigma^2}{2(r-q)}\left(\frac{S_{\min}}{S(0)}\right)^{2(r-q)/\sigma^2}\mathrm{e}^{-rT}S(0)\mathrm{N}(d_2') + \left(1+\frac{\sigma^2}{2(r-q)}\right)\mathrm{e}^{-qT}S(0)\mathrm{N}(-d_1)\ .$$

where

$$d_1 = \frac{\log\left(\frac{S(0)}{S_{\min}}\right) + \left(r-q+\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\ , \qquad d_2 = d_1 - \sigma\sqrt{T}\ , \tag{39}$$

$$d_1' = \frac{\log\left(\frac{S_{\min}}{S(0)}\right) + \left(r - q + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\,, \qquad d_2' = d_1' - \sigma\sqrt{T}\,. \tag{40}$$

Using the fact that $[1 - \mathrm{N}(-d_1)]\mathrm{e}^{-qT}S(0) = \mathrm{e}^{-qT}S(0)\mathrm{N}(d_1)$, this implies:

Floating-Strike Lookback Call Pricing Formula

Tip

The value at date 0 of a continuously-sampled floating-strike lookback call, given that the minimum price during the lifetime of the contract through date 0 is $S_{\min}$ and the remaining time to maturity is T, is

$$\begin{aligned} &\mathrm{e}^{-qT}S(0)\mathrm{N}(d_1) - \mathrm{e}^{-rT}S_{\min}\mathrm{N}(d_2) \\ &\qquad + \frac{\sigma^2}{2(r-q)}\left(\frac{S_{\min}}{S(0)}\right)^{2(r-q)/\sigma^2} \mathrm{e}^{-rT}S(0)\mathrm{N}(d_2') \\ &\qquad\qquad - \frac{\sigma^2}{2(r-q)}\mathrm{e}^{-qT}S(0)\mathrm{N}(-d_1)\,, \end{aligned} \tag{41}$$

{#eq-fslc100}
where d_1, d_2, and d_2' are defined in Equation 39 - Equation 40.

The following program calculates the price of a floating strike lookback option.

```python
def floating_strike_call(S, r, sigma, q, T, SMin):
    d1 = (np.log(S / SMin) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    d2prime = (np.log(SMin / S) + (r - q - 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    N1 = norm.cdf(d1)
    N2 = norm.cdf(d2)
    N2prime = norm.cdf(d2prime)
    x = 2 * (r - q) / (sigma ** 2)
    return np.exp(-q * T) * S * N1 - np.exp(-r * T) * SMin * N2 + (1 / x) * (SMin / S) ** x *

# Example usage

S = 100
r = 0.05
sigma = 0.2
```

```python
q = 0.02
T=1


print("Floating Strike Call:", floating_strike_call(S, r, sigma, q, T, 90))
```

```
Floating Strike Call: 16.27191861732918
```

Basket and Spread Options

A spread option is a call or a put written on the difference of two asset prices. For example, a spread call will pay at maturity T the larger of zero and $S_1(T) - S_2(T) - K$, where the S_i are the asset prices and K is the strike price of the call. Spread options can be used by producers to hedge the difference between an input price and an output price. They are also useful for hedging basis risk. For example, someone may want to hedge an asset by selling a futures contract on a closely related but not identical asset. This exposes the hedger to basis risk: the difference in value between the asset and the underlying asset on the futures contract. A spread call can hedge the basis risk: take S_1 to be the value of the asset underlying the futures contract and S_2 the value of the asset being hedged.

A spread option is actually an exchange option. Assuming constant dividend yields q_1 and q_2, we can take the assets underlying the exchange option to be as follows

1. At date 0, purchase $\mathrm{e}^{-q_1 T}$ units of the asset with price S_1 and reinvest dividends, leading to a value of $S_1(T)$ at date T,
2. At date 0, purchase $\mathrm{e}^{-q_2 T}$ units of the asset with price S_2 and invest $\mathrm{e}^{-rT}K$ in the risk-free asset. Reinvesting dividends and accumulating interest means that we will have $S_2(T) + K$ dollars at date T.

However, we cannot apply Margrabe's formula to price spread options, because the second portfolio described above will have a stochastic volatility. To see this, note that if the price $S_2(t)$ falls to a low level, then the portfolio will consist primarily of the risk-free asset, so the portfolio volatility will be near the volatility of the risk-free asset, which is zero. On the other hand, if $S_2(t)$ becomes very high, then the portfolio will be weighted very heavily on the stock investment, and its volatility will approach the volatility of S_2.

A basket option is an option written on a portfolio of assets. For example, someone may want to hedge the change in the value of the dollar relative to a basket of currencies. A basket option is an alternative to purchasing separate options on each currency. Generally, the basket option would have a lower premium than the separate options, because an option on a portfolio is cheaper (and pays less at maturity) than a portfolio of options.

Letting S_1, ..., S_n denote the asset prices and w_1, ..., w_n the weights specified by the contract, a basket call would pay

$$\max\left(0,\ \sum_{i=1}^{n} w_i S_i(T) - K\right)$$

at maturity T. A spread option is actually a special case of a basket option, with $n = 2$, $w_1 = 1$, and $w_2 = -1$. The difficulty in valuing basket options is the same as that encountered in valuing spread options. The volatility of the basket price $\sum_{i=1}^{n} w_i S_i(t)$ will vary over time, depending on the relative volatilities of the assets and the price changes in the assets. For example, consider the case $n = 2$ and write $S(t)$ for the basket price $w_1 S_1(t) + w_2 S_2(t)$. Then

$$\begin{aligned}\frac{\mathrm{d}S}{S} &= \frac{w_1\,\mathrm{d}S_1}{S} + \frac{w_2\,\mathrm{d}S_2}{S}\\ &= \frac{w_1 S_1}{S} \times \frac{\mathrm{d}S_1}{S_1} + \frac{w_2 S_2}{S} \times \frac{\mathrm{d}S_2}{S_2}\ .\end{aligned}$$

Let $x_i(t) = w_i S_i(t)/S(t)$. This is the fraction of the portfolio value that the i–th asset contributes. It will vary randomly over time as the prices change. Letting σ_i denote the volatilities of the individual assets and ρ their correlation, the formula just given for $\mathrm{d}S/S$ shows that the instantaneous volatility of the basket price at any date t is

$$\sqrt{x_1^2(t)\sigma_1^2 + 2x_1(t)x_2(t)\rho\sigma_1\sigma_2 + x_2^2(t)\sigma_2^2}\ .$$

Hence, the volatility will vary randomly over time as the x_i change. As in the case of spread options, there is no simple closed-form solution for the value of a basket option.

Asian Options

An Asian option is an option the value of which depends on the average underlying asset price during the lifetime of the option. Average-price calls and puts are defined like standard calls and puts but with the final asset price replaced by the average price. Average-strike calls and puts are defined like standard calls and puts but with the exercise price replaced by the average asset price. A firm that must purchase an input at frequent intervals or will sell a product in a foreign currency at frequent intervals can use an average price option as an alternative to buying multiple options with different maturity dates. The average-price option will generally be both less expensive and a better hedge than purchasing multiple options.

In practice, the average price is computed by averaging over the prices sampled at a finite number of discrete dates. First, we consider the case of continuous sampling. With continuous sampling, the average price at date T for an option written at date 0 will be denoted by $A(T)$ and is defined as

$$A(T) = \frac{1}{T}\int_0^T S(t)\,\mathrm{d}t\ .$$

To obtain a closed-form solution for the value of an option on the average price, we face essentially the same problem as for basket and spread options: a sum of lognormally distributed variables is not itself lognormally distributed. In this case, the integral, which is essentially a continuous sum of the prices at different dates, is not lognormally distributed.

An alternative contract would replace the average price with the geometric average. This is defined as the exponential of the average logarithm. We will denote this by $A_{\mathrm{g}}(T)$. The average logarithm is

$$\frac{1}{T}\int_0^T \log S(t)\,\mathrm{d}t\;,$$

and the geometric average is

$$A_{\mathrm{g}}(T) = \exp\left(\frac{1}{T}\int_0^T \log S(t)\,\mathrm{d}t\right)\;.$$

The concavity of the logarithm function guarantees that

$$\log\frac{1}{T}\int_0^T S(t) > \frac{1}{T}\int_0^T \log S(t)\,\mathrm{d}t\;.$$

Therefore,

$$\begin{aligned}
A(T) &= \exp\left(\log\frac{1}{T}\int_0^T S(t)\right)\\
&> \exp\left(\frac{1}{T}\int_0^T \log S(t)\,\mathrm{d}t\right)\\
&= A_{\mathrm{g}}(T)\;.
\end{aligned}$$

Consequently, approximating the value of an average-price or average-strike option by substituting $A_{\mathrm{g}}(T)$ for $A(T)$ will produce a biased estimate of the value. Nevertheless, the geometric average$A_{\mathrm{g}}(T)$ and the arithmetic average $A(T)$ will be highly correlated, so $A_{\mathrm{g}}(T)$ forms a very useful control variate for Monte-Carlo valuation of average-price and average-strike options, as will be discussed in **?@sec-c_montecarlo**. To implement the idea, we need a valuation formula for options written on $A_{\mathrm{g}}(T)$. We will derive this for an average-price call, in which $A_{\mathrm{g}}(T)$ substitutes for $A(T)$.

Specifically, consider a contract that pays

$$\max(0, A_{\mathrm{g}}(T) - K)$$

at its maturity T. This is a geometric-average-price call,, and we will analyze it in the same way that we analyzed quanto options in **?@sec-c_foreignexchange**. Let $V(t)$ denote the value at date t of receiving $A_{\mathrm{g}}(T)$ at date T. This can be calculated, and the result will be given below. $V(t)$ is the value of a non-dividend-paying portfolio, and, by definition, $V(T) =$

$A_{\mathrm{g}}(T)$, so the geometric-average-price call is equivalent to a standard call with V being the price of the underlying. We will show that V has a time-varying but non-random volatility. Therefore, we can apply the Black-Scholes formula, inputting the average volatility as described in **?@sec-s__timevaryingvolatility**, to value the geometric-average-price call. We could attempt the same route to price average-price options, but we would find that the volatility of the corresponding value V would vary randomly, just as we found the basket portfolio to have a random volatility in the previous section.

The value $V(t)$ can be calculated as

$$V(t) = \mathrm{e}^{-r(T-t)} E_t^R[A_{\mathrm{g}}(T)] \;.$$

Define

$$A_g(t) = \exp\left(\frac{1}{t}\int_0^t \log S(u)\,du\right) \;.$$

We will verify at the end of this section that

$$V(t) = \mathrm{e}^{-r(T-t)} A_g(t)^{\frac{t}{T}} S(t)^{\frac{T-t}{T}} \exp\left(\frac{(r-q-\sigma^2/2)(T-t)^2}{2T} + \frac{\sigma^2(T-t)^3}{6T^2}\right) \;. \tag{42}$$

Two points are noteworthy. First, the value at date 0 is

$$\begin{aligned} V(0) &= \mathrm{e}^{-rT} S(0) \exp\left(\frac{(r-q-\sigma^2/2)T}{2} + \frac{\sigma^2 T}{6}\right) \\ &= \exp\left(-\frac{6r+6q+\sigma^2}{12}T\right) S(0) \;. \end{aligned} \tag{43}$$

Second, the volatility comes from the factor

$$S(t)^{\frac{T-t}{T}} \;,$$

and, by Ito's formula,

$$\frac{\mathrm{d}S^{\frac{T-t}{T}}}{S^{\frac{T-t}{T}}} = \text{something}\;\mathrm{d}t + \left(\frac{T-t}{T}\right)\sigma\,\mathrm{d}B \;.$$

This implies that the average volatility, in the sense of **?@sec-s__timevaryingvolatility**, is

$$\sigma_{\mathrm{avg}} = \sqrt{\frac{1}{T}\int_0^T \left(\frac{T-t}{T}\right)^2 \sigma^2\,\mathrm{d}t} = \frac{\sigma}{\sqrt{3}} \;.$$

Applying the Black-Scholes formula yields:

Tip

The value at date 0 of a continuously-sampled geometric-average-price call written at date~0 and having T years to maturity is

$$V(0)\mathrm{N}(d_1) - \mathrm{e}^{-rT} K \mathrm{N}(d_2) \;,$$

where

$$d_1 = \frac{\log\left(\frac{V(0)}{K}\right) + \left(r + \frac{1}{2}\sigma_{\text{avg}}^2\right)T}{\sigma_{\text{avg}}\sqrt{T}}, \qquad d_2 = d_1 - \sigma_{\text{avg}}\sqrt{T} \;,$$

$V(0)$ is defined in Equation 43, and $\sigma_{\text{avg}} = \sigma/\sqrt{3}$.

We can also value a discretely-sampled geometric-average-price call by the same arguments. Consider dates $0 < t_0 < t_1 < \cdots t_N = T$, where $t_i - t_{i-1} = \Delta t$ for each i and suppose the price is to be sampled at the dates $t_1, \ldots, t_N$. Now let $V(t)$ denote the value at date t of the contract that pays

$$\exp\left(\frac{1}{N}\sum_{i=1}^{N} \log S(t_i)\right) = \left(\prod_{i=1}^{N} S(t_i)\right)^{1/N} \tag{44}$$

at date T. The call option will pay $\max(0, V(T) - K)$ at date T. Let k denote the integer such that $t_{N-k-1} \le t < t_{N-k}$. This means that we have already observed the prices $S(t_1), \ldots, S(t_{N-k-1})$ and we have yet to observe the $k+1$ prices $S(t_{N-k}), \ldots, S(t_N)$. Define $\varepsilon = (t_{N-k} - t)/\Delta t$, which is fraction of the interval Δt that must pass before we reach the next sampling date t_{N-k}. We will show at the end of this section that

$$V(t) = \mathrm{e}^{-r(T-t)} S(t)^{\frac{k+1}{N}} \prod_{i=1}^{N-k-1} S(t_i)^{\frac{1}{N}}$$
$$\times \exp\left(\left[\frac{(k+1)\varepsilon\nu}{N} + \frac{k(k+1)\nu}{2N} + \frac{(k+1)^2\sigma^2\varepsilon}{2N^2} + \frac{k(k+1)(2k+1)\sigma^2}{12N^2}\right]\Delta t\right) \;, \tag{45}$$

{#eq-geometricaveragev2}

where $\nu = r - q - \sigma^2/2$

Again, two points are noteworthy. Assume the call was written at date 0 and the first observation date t_1 is Δt years away. Then, we have $k+1 = N$ and $\varepsilon = 1$ so

$$V(0) = \mathrm{e}^{-rT} S(0) \exp\left(\frac{(N+1)\nu\Delta t}{2} + \frac{(N+1)(2N+1)\sigma^2\Delta t}{12N}\right) \;. \tag{46}$$

Second, the volatility of $V(t)$ comes from the factor $S(t)^{(k+1)/N}$, and

$$\frac{\mathrm{d}S^{\frac{k+1}{N}}}{S^{\frac{k+1}{N}}} = \text{something } \mathrm{d}t + \left(\frac{k+1}{N}\right)\sigma\,\mathrm{d}B\,.$$

This implies that the average volatility, in the sense of **?@sec-s_timevaryingvolatility**, is

$$\begin{aligned}\sigma_{\text{avg}} &= \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}\left(\frac{k+1}{N}\right)^2\sigma^2\,\mathrm{d}t} \\ &= \frac{\sigma}{N^{3/2}}\sqrt{\frac{N(N+1)(2N+1)}{6}}\,, \end{aligned} \tag{47}$$

where we have used the fact that $\sum_{i=1}^N i^2 = N(N+1)(2N+1)/6$ to obtain the second equality. Thus, the Black-Scholes formula implies:

Tip

The value at date 0 of a discretely-sampled geometric-average-price call written at date~0 and having T years to maturity is

$$V(0)\mathrm{N}(d_1) - \mathrm{e}^{-rT}K\mathrm{N}(d_2)\,, \tag{48}$$

where

$$d_1 = \frac{\log\left(\frac{V(0)}{K}\right) + \left(r + \frac{1}{2}\sigma_{\text{avg}}^2\right)T}{\sigma_{\text{avg}}\sqrt{T}}, \qquad d_2 = d_1 - \sigma_{\text{avg}}\sqrt{T}\,,$$

$V(0)$ is defined in Equation 46, and σ_{avg} is defined in Equation 47.

This formula will be used in **?@sec-s_controlvariates** as a control variate for pricing discretely-sampled average-price calls (even average-price calls that were written before the date of valuation).

Note

##
We will now derive equations Equation 42 and **?@eq-geometricaveragev2**. We will begin with Equation 42. The random variable $A_g(T)$ is normally distributed under the risk-neutral measure given information at time t. To establish this, and to calculate the mean and variance of $A_g(T)$, the key is to change the order of integration in the integral

in the second line below to obtain the third line:

$$\begin{aligned}
\int_t^T \log S(u)\,du &= \int_t^T \left\{\log S(t) + \left(r - q - \frac{1}{2}\sigma^2\right)(u-t) + \sigma[B(u)-B(t)]\right\} du \\
&= (T-t)\log S(t) + \left(r-q-\frac{1}{2}\sigma^2\right)\frac{(T-t)^2}{2} + \sigma\int_t^T\int_t^u \mathrm{d}B(s)\,du \\
&= (T-t)\log S(t) + \left(r-q-\frac{1}{2}\sigma^2\right)\frac{(T-t)^2}{2} + \sigma\int_t^T\int_s^T du\,\mathrm{d}B(s) \\
&= (T-t)\log S(t) + \left(r-q-\frac{1}{2}\sigma^2\right)\frac{(T-t)^2}{2} + \sigma\int_t^T (T-s)\,\mathrm{d}B(s)
\end{aligned}$$

and then to note that $\int_t^T (T-s)\,\mathrm{d}B(s)$ is normally distributed with mean zero and variance equal to

$$\int_t^T (T-s)^2\,ds = \frac{(T-t)^3}{3}\ .$$

Therefore $E_t^R\left[A_g(T)\right]$ is the expectation of the exponential of a normally distributed random variable. Equation~Equation 42 now follows from the fact that if x is normally distributed with mean μ and variance σ^2 then $E\left[\mathrm{e}^x\right] = \mathrm{e}^{\mu+\sigma^2/2}$.
To establish **?@eq-geometricaveragev2**, note that the discounted risk-neutral expectation of Equation 44, conditional on having observed $S(t_1), \ldots, S(t_{N-k-1})$, is

$$\begin{aligned}
V(t) &= \mathrm{e}^{-r(T-t)}E_t^R\left[\exp\left(\frac{1}{N}\sum_{i=1}^N \log S(t_i)\right)\right] \\
&= \mathrm{e}^{-r(T-t)}\exp\left(\frac{1}{N}\sum_{i=1}^{N-k-1}\log S(t_i)\right) \times E_t^R\left[\exp\left(\frac{1}{N}\sum_{i=N-k}^N \log S(t_i)\right)\right] \\
&= \left(\prod_{i=1}^{N-k-1} S(t_i)^{\frac{1}{N}}\right) \times \mathrm{e}^{-r(T-t)}E_t^R\left[\exp\left(\frac{1}{N}\sum_{i=N-k}^N \log S(t_i)\right)\right]\ . \qquad (49)
\end{aligned}$$

Let $\Delta_0 B = B(t_{N-k}) - B(t)$ and $\Delta_i B = B(t_{N-k+i}) - B(t_{N-k+i-1})$ for $i \geq 1$. We can write the sum of logarithms inside the expectation above as

$$\begin{aligned}
&\sum_{i=0}^k \left\{[\log S(t) + (t_{N-k+i} - t)\nu + \sigma[B(t_{N-k+i}) - B(t)]\right\} \\
&\quad = (k+1)\log S(t) + \sum_{i=0}^k (\varepsilon + i)\nu\Delta t + \sigma\sum_{i=0}^k[\Delta_0 B + \Delta_1 B + \cdots + \Delta_i B] \\
&\quad = (k+1)\log S(t) + (k+1)\varepsilon\nu\Delta t + \frac{k(k+1)}{2}\nu\Delta t + \sigma\sum_{i=0}^k (k+1-i)\Delta_i B\ ,
\end{aligned}$$

where to obtain the last equality we used the fact that $\sum_{i=0}^{k} i = k(k+1)/2$. The random variables $\Delta_i B$ are normally distributed with mean zero and variance Δt (the variance is $\varepsilon\Delta t$ for $i = 0$). Thus, the sum of logarithms is a normally distributed random variable with mean

$$(k+1)\log S(t) + (k+1)\varepsilon\nu\Delta t + \frac{k(k+1)}{2}\nu\Delta t$$

and variance

$$(k+1)^2\sigma^2\varepsilon\Delta t + \sigma^2\sum_{i=1}^{k}(k+1-i)^2\Delta t = (k+1)^2\sigma^2\varepsilon\Delta t + \frac{k(k+1)(2k+1)\sigma^2}{6} \ ,$$

using the fact that $\sum_{i=1}^{k} i^2 = k(k+1)(2k+1)/6$. The expectation of the exponential of a normally distributed random variable equals the exponential of its mean plus one-half of its variance, and the exponential of $(k+1)\log S(t)/N$ is $S(t)^{(k+1)/N}$. Therefore the conditional expectation in Equation 49 is

$$S(t)^{\frac{k+1}{N}} \exp\left(\left[\frac{(k+1)\varepsilon\nu}{N} + \frac{k(k+1)\nu}{2N} + \frac{(k+1)^2\sigma^2\varepsilon}{2N^2} + \frac{k(k+1)(2k+1)\sigma^2}{12N^2}\right]\Delta t\right) \ ,$$

which implies **?@eq-geometricaveragev2**.

The following code computes the price of a geometric-average-price call.

```
def discrete_geom_average_price_call(S, K, r, sigma, q, T, N):
    dt = T / N
    nu = r - q - 0.5 * sigma ** 2
    a = N * (N + 1) * (2 * N + 1) / 6
    V = np.exp(-r * T) * S * np.exp(((N + 1) * nu / 2 + sigma ** 2 * a / (2 * N ** 2)) * dt)
    sigavg = sigma * np.sqrt(a) / (N ** 1.5)
    return black_scholes_call(V, K, r, sigavg, q, T)

print("Discrete Geometric Average Price Call:", discrete_geom_average_price_call(S, K, r, si
```

```
Discrete Geometric Average Price Call: 4.375035678935298
```

Monte Carlo Models for Path-Dependent Options

A derivative is said to be path dependent if its value depends on the path of the underlying asset price rather than just on the price at the time of exercise. Examples of path-dependent options are lookbacks, barrier options, and Asians. To value a path-dependent option by Monte Carlo, we need to simulate an approximate path of the stock price. We do this by considering time periods of length $\Delta t = T/N$ for some integer N. Under the risk-neutral measure, the logarithm of the stock price changes over such a time period by

$$\Delta \log S = \nu\, \Delta t + \sigma \sqrt{\Delta t}\, z\ , \tag{50}$$

where $\nu = r - q - \sigma^2/2$ and z is a standard normal. Given that there are N time periods of length Δt, we need to generate N standard normals to generate a stock price path. If we generate M paths to obtain a sample of M option values, then we will need to generate MN standard normals.

Consider for example a floating-strike lookback call. The formula for this option given in Section assumes the minimum stock price is computed over the entire path of the stock price, i.e., with continuous sampling of the stock price. In practice, the minimum will be computed by recording the price at a discrete number of dates. We can value the discretely sampled lookback using Monte-Carlo by choosing Δt to be the interval of time (e.g., a day or week) at which the price is recorded. For example, if the contract calls for weekly observation, we will attain maximum precision by setting N to be the number of weeks before the option matures.

For most path dependent options, a possible starting point is to generate an array of n paths but since we want the entire path we choose the number of time steps that is appropriate for our application. We can use the same code as in Section @#sec-s_mc_europeans if we are working in a Black Scholes setting.

```
# Simulate Geometric Brownian Motion
import numpy as np
import matplotlib.pyplot as plt
# number of paths
n = 1000
#number of divisions
m = 1000
# Interest rate (We set the drift equal to the interest rate for the risk neutral measure)
r = 0.1
# Dividend yield
q=0.0
# Volatility
sig = 0.2
# Initial Stock Price
S0 = 42
# Maturity
T = 0.5
#Strike Price
K=40
# Delta t
```

```
dt = T/m
# Drift
drift = (r-q-0.5*sig**2)
# Volatility
vol = sig * np.sqrt(dt)

t = np.array(range(0,m + 1,1)) * dt

# seed for random generator
seed= 2024
# define a random generator
np.random.seed(seed)
inc = np.zeros(shape = (m + 1, n))
inc[1:] = np.transpose(np.random.normal(loc = 0, scale = vol,size = (n,m)))
St = np.zeros(shape = (m + 1, n))
St = S0 * np.exp(np.cumsum(inc,axis=0) + (drift * t[0:m + 1])[:,None])
St1 = S0 * np.exp(-np.cumsum(inc,axis=0) + (drift * t[0:m + 1])[:,None])
```

As before this code generates two samples the original and the antithetic. The output is an array of n sample paths with m time steps. The sample can also be used to find the value of a floating strike lookback call.

```
Stmin=St[m:]-np.minimum(np.min(St,axis=0),S0)
St1min=St1[m:]-np.minimum(np.min(St1,axis=0),S0)
floatlkbk=np.exp(-r*T)*np.mean(Stmin)
floatlkbk1=np.exp(-r*T)*np.mean(St1min)

print('The first estimate is=',floatlkbk)
print('The second estimate is=',floatlkbk1)
print('The average estimate is=',(floatlkbk+floatlkbk1)/2)
print('The exact formula is=',floating_strike_call(S0, r, sigma, 0, T, S0))
```

```
The first estimate is= 5.45244209168367
The second estimate is= 5.468030697490039
The average estimate is= 5.460236394586854
The exact formula is= 5.5418069098303775
```

To value the fixed strike lookback call option with time T payoff $\max(\max_{0\leq t\leq T} S_t.0)$, we simply add the following

```
Stmax=np.maximum(np.max(St,axis=0)-K,0)
St1max=np.maximum(np.max(St1,axis=0)-K,0)
lookbck = np.exp(-r*T) *np.mean(Stmax)
lookbck1=np.exp(-r * T)*np.mean(St1max)
print('The first estimate is=',lookbck)
print('The second estimate is=',lookbck1)
print('The average estimate is=', (lookbck + lookbck1)/2)
```

```
The first estimate is= 7.716467940991822
The second estimate is= 7.769558237964809
The average estimate is= 7.743013089478316
```

Asian and barrier options are also subject to discrete rather than continuous sampling and can be valued by Monte-Carlo in the same way as lookbacks.

As another example, consider the classic case of estimating the value of a discretely-sampled average-price call, using a discretely-sampled geometric-average-price call as a control variate. Let τ denote the amount of time that has elapsed since the call was issued and T the amount of time remaining before maturity, so the total maturity of the call is $T+\tau$. To simplify somewhat, assume date 0 is the beginning of a period between observations. Let $t_1,\ldots,t_N$ denote the remaining sampling dates, with $t_1=\Delta t$, $t_i-t_{i-1}=\Delta t=T/N$ for each i, and $t_N=T$. We will input the average price $A(0)$ computed up to date 0, assuming this average includes the price $S(0)$ at date 0. The average price at date T will be

$$A(T)=\frac{\tau}{T+\tau}A(0)+\frac{T}{T+\tau}\left(\frac{\sum_{i=1}^N S(t_i)}{N}\right)\,.$$

The average-price call pays $\max(0,A(T)-K)$ at its maturity T, and we can write this as

$$\begin{aligned}\max(A(T)-K,0)&=\max\left(\frac{T}{T+\tau}\left(\frac{\sum_{i=1}^N S(t_i)}{N}\right)-\left(K-\frac{\tau}{T+\tau}A(0)\right),0\right)\\&=\frac{T}{T+\tau}\max\left(\frac{\sum_{i=1}^N S(t_i)}{N}-K^*,0\right)\,,\end{aligned}$$

where

$$K^*=\frac{T+\tau}{T}K-\frac{\tau}{T}A(0)\,.$$

Therefore, the value at date 0 of the discretely-sampled average-price call is

$$\frac{T}{T+\tau}\,\mathrm{e}^{-rT}E^R\left[\max\left(\frac{\sum_{i=1}^N S(t_i)}{N}-K^*,0\right)\right]\,.$$

In terms of the discussion above, the random variable the mean of which we want to estimate is

$$x = \mathrm{e}^{-rT} \max\left(\frac{\sum_{i=1}^{N} S(t_i)}{N} - K^*, 0\right) .$$

A random variable y that will be closely correlated to x is

$$y = \mathrm{e}^{-rT} \max\left(\mathrm{e}^{\sum_{i=1}^{N} \log S(t_i)/N} - K^*, 0\right) .$$

The mean ϕ of y under the risk-neutral measure is given in the pricing Equation 48. We can use the sample mean of y and its known mean ϕ to adjust the sample mean of x as an estimator of the value of the average-price call. Generally, the estimated adjustment coefficient $\widehat{\beta}$ will be quite close to 1.

Again we can get a sample of payoffs using our stock price samples.

```
average = np.mean(St,axis=0)
average1 = np.mean(St1,axis=0)
dpayoff=np.exp(-r*T)*np.mean(np.maximum(average-K,0))
dpayoff1=np.exp(-r*T)*np.mean(np.maximum(average1-K,0))
print('The first estimate is=',dpayoff)
print('The second estimate is=',dpayoff1)
print('The average of the estimates=',(dpayoff+dpayoff1)/2)
```

```
The first estimate is= 3.231154315344899
The second estimate is= 3.296876297130776
The average of the estimates= 3.2640153062378374
```

We now construct a control variate, the geometric asian option which has a known formula for its value.

```
geom=np.exp((np.mean(np.log(St),axis=0)))
geom1=np.exp((np.mean(np.log(St1),axis=0)))
geomavgpo=np.maximum(geom-K,0)
geomavg1po=np.maximum(geom1-K,0)
value=np.mean(geomavgpo)*np.exp(-r*T)
value1=np.mean(geomavg1po)*np.exp(-r*T)
tga=discrete_geom_average_price_call(S0, K, r, sigma, q, T, m)
error =tga-value
error1=tga-value1
print('The estimate from the first sample=',value)
print('The estimate from the second sample=',value1)
print('The average of the two estimates is=',(value+value1)/2)
print('The value from the exact formula=',tga)
print('The error in the first estimate=',error)
print('The error in the second estimate=',error1)
```

```
The estimate from the first sample= 3.172076049997697
The estimate from the second sample= 3.236566795172543
The average of the two estimates is= 3.20432142258512
The value from the exact formula= 3.1804170589762464
The error in the first estimate= 0.008341008897854917
The error in the second estimate= -0.05614973619629682
```

Next we estimate the beta. As discussed before, we could simply set beta=1. Alternatively, if we estimate beta from the simulated sample, then our update could be biased. Instead we compute an independent sample from which we estimate beta. We then estimate the updated estimate for both samples from the formula

$$\text{new estimate} = \text{original estimate} + \beta * \text{error}$$

```
incpre = np.zeros(shape = (m + 1, n))
incpre[1:] = np.transpose(np.random.normal(loc = 0, scale = vol,size = (n,m)))
Stpre = np.zeros(shape = (m + 1, n))
St1pre=np.zeros(shape = (m + 1, n))
Stpre = S0 * np.exp(np.cumsum(inc,axis=0) + (drift * t[0:m + 1])[:,None])
St1pre = S0 * np.exp(-np.cumsum(inc,axis=0) + (drift * t[0:m + 1])[:,None])

amean=np.mean(Stpre,axis=0)
amean1=np.mean(St1pre,axis=0)
apo = np.maximum(amean-K,0)
a1po=np.maximum(amean1-K,0)
gmean=np.exp(np.mean(np.log(St),axis=0))
g1mean=np.exp(np.mean(np.log(St1),axis=0))
gpo=np.maximum(gmean-K,0)
g1po=np.maximum(g1mean-K,0)
beta=np.cov(gpo,apo)[0,1]/np.cov(gpo,apo)[1,1]
beta1=np.cov(g1po,a1po)[0,1]/np.cov(g1po,a1po)[1,1]
update=dpayoff +beta*error
update1=dpayoff1+beta1*error1

print('The updated estimate for the first sample=',update)
print('The updated value for the second sample=',update1)
print('The average of the updated values is=',(update +update1)/2)
```

```
The updated estimate for the first sample= 3.239338189254849
The updated value for the second sample= 3.24174435394373
The average of the updated values is= 3.2405412715992896
```

Monte Carlo Valuation of Basket and Spread Options

In this section, we will consider the valuation of European spread and basket options by the Monte Carlo method. As noted in Section , there are no simple formulas for these options. In each simulation, we will generate a terminal price for each of the underlying assets and compute the value of the option at its maturity. Discounting the average terminal value gives the estimate of the option value as usual.

The difference between binomial and Monte Carlo methods for options written on multiple assets can be understood as follows. Both methods attempt to estimate the discounted expected value of the option (under the risk-neutral measure). In an N–period model, the binomial model produces $N+1$ values for the terminal price of each underlying asset. Letting k denote the number of underlying assets, this produces $(N+1)^k$ combinations of asset prices. Of course, each combination has an associated probability. In contrast, the Monte Carlo method produces M combinations of terminal prices, where M is the number of simulations. Each combination is given the same weight $(1/M)$ when estimating the expected value.

With a single underlying asset, the binomial model is more efficient, as discussed in **?@sec-s_introbinomial**, because the specifically chosen terminal prices in the binomial model sample the set of possible terminal prices more efficiently than randomly generated terminal prices. However, this advantage disappears, and the ranking of the methods can be reversed, when there are several underlying assets. The reason is that many of the $(N+1)^k$ combinations of prices in the binomial model will have very low probabilities. For example, with two assets that are positively correlated, it is very unlikely that one asset will be at its highest value in the binomial model and the other asset simultaneously at its lowest. It is computationally wasteful to evaluate the option for such a combination, because the probability-weighted value will be very small and hence contribute little to the estimate of the expected value. On the other hand, each set of terminal prices generated by the Monte Carlo method will be generated from a distribution having the assumed correlation. Thus, only relatively likely combinations will typically be generated, and time is not wasted on evaluating unlikely combinations. However, it should not be concluded that Monte Carlo valuation of a derivative on multiple assets will be quick and easy—even though the computation time required for more underlying assets does not increase as much with Monte Carlo as for binomial models, it can nevertheless be substantial.

To implement Monte Carlo valuation of options on multiple assets, we must first explain how to simulate correlated asset prices. As observed in **?@sec-s_stochasticvolatility**, we can simulate the changes in two Brownian motions B_1 and B_2 that have correlation ρ by generating two independent standard normals Z_1 and Z_2 and defining

$$\Delta B_1 = \sqrt{\Delta t}\, Z_1 \,, \qquad \text{and} \qquad \Delta B_2 = \sqrt{\Delta t}\, Z \,,$$

where Z is defined as

$$Z = \rho Z_1 + \sqrt{1-\rho^2}\, Z_2 \,.$$

The random variable Z is also a standard normal, and the correlation between Z_1 and Z is ρ.

```
# Simulate 2 Geometric Brownian Motions
import numpy as np
import matplotlib.pyplot as plt
# number of paths
n = 1000
#number of divisions
m = 1000
# Interest rate (We set the drift equal to the interest rate for the risk neutral measure)
r = 0.1
# Dividend yield
q1=0.0
q2=0
# Volatility
sig1 = 0.2
sig2=.3
# correlation
rho=0.5
# Initial Stock Price
S0 = 42
V0 = 50
# Maturity
T = 0.5

# Delta t
dt = T/m
# Drift
drift1 = (r-q1-0.5*sig1**2)
drift2 = (r-q2-0.5*sig2**2)
# Volatility
vol = np.sqrt(dt)

t = np.array(range(0,m + 1,1)) * dt

# seed for random generator
```

```
seed= 2024
# define a random generator
np.random.seed(seed)
inc = np.zeros(shape = (m + 1, n))
inc[1:] = np.transpose(np.random.normal(loc = 0, scale = vol,size = (n,m)))
inc1 = np.zeros(shape = (m + 1, n))
inc1[1:] = np.transpose(np.random.normal(loc = 0, scale = vol,size = (n,m)))
incr = np.zeros(shape = (m + 1, n))
incr = rho*inc + np.sqrt(1-rho**2)*inc1
```

Thus, we can simulate the changes in the logarithms of two correlated asset prices as

$$\Delta \log S_1 = \nu_1 \Delta t + \sigma_1 \sqrt{\Delta t} Z_1 \ ,$$
$$\Delta \log S_2 = \nu_2 \Delta t + \sigma_2 \rho \sqrt{\Delta t} Z_1 + \sigma_2 \sqrt{1-\rho^2} \sqrt{\Delta t} Z_2 \ ,$$

where $\nu_i = r - q_1 - \sigma_i^2/2$ and the Z_i are independent standard normals.

```
St1 = np.zeros(shape = (m + 1, n))
St2 = np.zeros(shape = (m + 1, n))
St1 = S0 * np.exp(sig1*np.cumsum(inc,axis=0) + (drift1 * t[0:m + 1])[:,None])
St2 = V0 * np.exp(sig2*np.cumsum(incr,axis=0) + (drift2 * t[0:m + 1])[:,None])
```

We can also construct antithetic variables.

```
St1a = np.zeros(shape = (m + 1, n))
St2a = np.zeros(shape = (m + 1, n))
St1a = S0 * np.exp(-sig1*np.cumsum(inc,axis=0) + (drift1 * t[0:m + 1])[:,None])
St2a = V0 * np.exp(-sig2*np.cumsum(incr,axis=0) + (drift2 * t[0:m + 1])[:,None])
```

Given this sample, we can estimate the value of a best of 2 option with payoff $\max(S_{1T}, S_{2T})$.

```
payoff = np.maximum(St1[m,:],St2[m,:])
payoffa = np.maximum(St1a[m,:],St2a[m,:])
value= np.exp(-r*T)*np.mean(payoff)
valuea= np.exp(-r*T)*np.mean(payoffa)

print('The first estmate is =',value)
print('The second estimate is =',valuea)
print('The avergae of the estimates is=',(value+valuea)/2)
```

```
The first estmate is = 50.11374784508302
The second estimate is = 51.46596027225067
The avergae of the estimates is= 50.78985405866685
```

To generalize this idea to more than two assets, we introduce some additional notation. The simulation for the case of two assets can be written as

$$\Delta \log S_1 = \nu_1 \Delta t + a_{11}\sqrt{\Delta t} Z_1 + a_{12}\sqrt{\Delta t} Z_2 \ , \tag{51}$$

$$\Delta \log S_2 = \nu_2 \Delta t + a_{21}\sqrt{\Delta t} Z_1 + a_{22}\sqrt{\Delta t} Z_2 \ , \tag{52}$$

where

$$\begin{aligned} a_{11} &= \sigma_1 \ , & a_{12} &= 0 \ , \\ a_{21} &= \sigma_2 \rho \ , & a_{22} &= \sigma_2 \sqrt{1-\rho^2} \ . \end{aligned}$$

These are not the only possible choices for the constants a_{ij}. Given that Z_1 and Z_2 are independent standard normals, the conditions the a_{ij} must satisfy in order to match the variances $\sigma_i^2 \Delta t$ and correlation ρ of the changes in the logarithms are

$$a_{11}^2 + a_{12}^2 = \sigma_1^2 \ , \tag{53}$$

$$a_{21}^2 + a_{22}^2 = \sigma_2^2 \ , \tag{54}$$

$$a_{11}a_{21} + a_{12}a_{22} = \sigma_1 \sigma_2 \rho \ . \tag{55}$$

These three equations in the four coefficients a_{ij} leave one degree of freedom. We choose to take $a_{12} = 0$ and then solve for the other three.

In matrix notation, the system Equation 53 - Equation 55 plus the condition $a_{12} = 0$ can be written as the equation

$$\begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix}^{\top} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \ ,$$

where $^{\top}$ denotes the matrix transpose. The matrix on the right hand side is the covariance matrix of the continuously-compounded annual returns (changes in log asset prices). Choosing the a_{ij} so that the lower triangular matrix

$$A \equiv \begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix}$$

satisfies

$$AA^\top = \text{covariance matrix}$$

is called the the Cholesky decomposition of the covariance matrix. Given any number L of assets, provided none of the assets is redundant (perfectly correlated with a portfolio of the others), the Cholesky decomposition of the $L \times L$ covariance matrix always exists. An algorithm for computing the Cholesky decomposition in numpy is np.linalg.cholesky.

We can use the Cholesky decomposition to perform Monte-Carlo valuation of a basket or spread option.[3] If there were some path dependency in the option value, we would simulate the paths of the asset prices as in Equation 51 - Equation 52. However a standard basket option is not path dependent, so we only need to simulate the asset prices at the option maturity date T, as in **?@sec-s__mc__europeans**. The value of a basket call option at its maturity T is

$$\max\left(0,\; \sum_{i=1}^{L} w_i S_i(T) - K\right)\,,$$

where L is the number of assets in the basket (portfolio) and w_i is the weight of the i–th asset in the basket. The logarithm of the i–th asset price at maturity is simulated as

$$\log S_i(T) = \log S_i(0) + \nu_i T + \sqrt{T}\sum_{j=1}^{L} a_{ij} Z_j\,,$$

where the Z_j are independent standard normals. Given the simulated values of the $\log S_i(T)$, the value at maturity of the basket option is readily computed. The estimate of the date–0 value is then computed as the discounted average of the simulated values at maturity.

For our two asset example we compute the value of a call opttion on an equally weighted porfotlio.

```
w=0.5
K=45
basketpo=np.maximum(w*St1[m,:]+(1-w)*St2[m,:]-K,0)
basketpoa=np.maximum(w*St1a[m,:]+(1-w)*St2a[m,:]-K,0)
estimate=np.exp(-r*T)*np.mean(basketpo)
estimatea=np.exp(-r*T)*np.mean(basketpoa)
print('The first estimate is =',estimate)
print('The second estimate is =',estimatea)
print('The average of the estimates=',(estimate+estimatea)/2)
```

```
The first estimate is = 4.4030641748675095
The second estimate is = 4.945665333101932
The average of the estimates= 4.674364753984721
```

[3]For a spread option, take $L = 2$, $w_1 = 1$ and $w_2 = -1$.

Below is a three asset basket option whihc uses the numpy cholesky decomposition. In contrast to the above routine, this routine is does not have the option to generate the entire path, although this can be easily modeified.

```
import numpy as np
#risk free rate
r=0.1
# number of assets
k=3

# number of paths
n=100000
# Horizon
T=0.5

# Initial price

S0=[42,50,45]

# Basket Weights
w=[.25,.5,.25]

#Strike Price

K=45

#  put in volatilities
sig1=.2
sig2=.3
sig3=.4

#create diagonal
sig=[sig1,sig2,sig3]

S=np.diag(sig)

# drift of log returns

drift= r*np.ones(k) -0.5*np.dot(S@S,np.ones(k))


# correlation matrix
```

```
rho=np.array([[1.0, 0.5, 0.3],
                  [0.5, 1.0, 0.2],
                  [0.3, 0.2, 1.0]])
# covariance matrix

V = S@rho@S

# generate uniform n*k normal uncorrelated random variables

seed=2024
np.random.seed(seed)

inc1=np.transpose(np.random.normal(loc = 0, scale = np.sqrt(T),size = (n,k)))



# create correlated random variables
Z=np.linalg.cholesky(V)
incr=np.dot(Z,inc1)

print('The sample correlation matrix =',np.corrcoef(incr))
print('The input correlation matrix =',rho)




St = S0 * np.exp(drift *T + np.transpose(incr))
#antithetic sample
St1 = S0 * np.exp( drift * T - np.transpose(incr))

estimate=np.mean(St,axis=0)*np.exp(-r*T)
estimate1=np.mean(St1,axis=0)*np.exp(-r*T)
print('The average discounted stock price averaged over both samples=',(estimate+estimate1)/
print('The initial Srock Price input =',S0)

basketpo=np.maximum(w@np.transpose(St)-K,0)
basketpo1=np.maximum(w@np.transpose(St1)-K,0)
value=np.mean(basketpo)*np.exp(-r*T)
value1=np.mean(basketpo1)*np.exp(-r*T)
print('The first sample estimate of the basket option value=',value)
print('The second sample estimate of the basket option value=',value1)
print('The average estimate of the basket option value=',(value+value1)/2)
```

```
The sample correlation matrix = [[1.         0.50356746 0.30497838]
 [0.50356746 1.         0.20383175]
 [0.30497838 0.20383175 1.        ]]
The input correlation matrix = [[1.  0.5 0.3]
 [0.5 1.  0.2]
 [0.3 0.2 1. ]]
The average discounted stock price averaged over both samples= [42.00560021 50.00484037 45.0
The initial Srock Price input = [42, 50, 45]
The first sample estimate of the basket option value= 5.335894164783821
The second sample estimate of the basket option value= 5.289353051200051
The average estimate of the basket option value= 5.312623607991936
```

We can generate the entire path of multiple assets to value, for example, lookback options on a basket. The code below values the same European basket option as above only it calculates $m = 2$ time steps; a lookback can be created by changing the payoffs and increasing m.

```python
import numpy as np
#number of time steps
m=2
# number of assets
k=3
# number of sample paths
n=100000

#risk free rate
r=0.1




# Horizon
T=0.5
# delta t
dt=T/m
# Initial price

S0=[42,50,45]

#Strike Price

K=45

#  put in volatilities
```

```
sig1=.2
sig2=.3
sig3=.4

#create diagonal
sig=[sig1,sig2,sig3]

S=np.diag(sig)


# correlation matrix

rho=np.array([[1.0, 0.5, 0.3],
                  [0.5, 1.0, 0.2],
                  [0.3, 0.2, 1.0]])

# covariance matrix

V = S@rho@S

# drift of log returns
drift= np.array(r*np.ones(k) -0.5*np.dot(S@S,np.ones(k)))*dt

# times vector
t=np.array(range(1,m + 1,1))


driftv = np.transpose(np.kron(drift,t).reshape(3,m))

# generate uniform n(paths)*k(assets)*m(time steps) normal uncorrelated random variables

seed=2024
np.random.seed(seed)

inc=np.random.normal(loc = 0, scale = np.sqrt(dt),size = (n,k,m))

#create correlated random increments
Z=np.linalg.cholesky(V)
# numpy matmul assumes last two define matrix multiplication
incr=np.matmul(Z,inc)
```

```
SSt=S0*np.exp(driftv)

# generate returns along path and antithetic path
# first e^{cumsum(increments)} gives e^sigma B_t for different t

Stb = np.exp(np.cumsum(incr[:,:,],axis=2))
Stb1 = np.exp(-np.cumsum(incr[:,:,],axis=2))


#Multiply by S0 e^drift for each t
St=np.multiply(Stb,np.transpose(SSt))
St1=np.multiply(Stb1,np.transpose(SSt))

#Last date returns
Stm=St[:,:,m-1]
Stm1=St1[:,:,m-1]

#define payoff
# Basket Weights
w=[.25,.5,.25]


payoff= np.maximum(np.matmul(Stm,np.transpose(w))-K,0)
payoff1= np.maximum(np.matmul(Stm1,np.transpose(w))-K,0)


value= np.exp(-r*T)*np.mean(payoff)
value1= np.exp(-r*T)*np.mean(payoff1)
print('The estimate for the first sample value=',value)
print('The estimate for the second sample value=',value1)
print('The average estimate for the value=',(value+value1)/2)
```

```
The estimate for the first sample value= 5.310173600480473
The estimate for the second sample value= 5.287928240443306
The average estimate for the value= 5.299050920461889
```

Binomial Valuation of Basket and Spread Options

By combining binomial models, we can value options or other derivatives on multiple assets. We will illustrate for an option on two assets. This is the most important case, and the extension to more than two assets is straightforward.

Consider two stocks with constant dividend yields q_i and constant volatilities σ_i. Suppose the two Brownian motions driving the two stocks have a constant correlation coefficient ρ. We will denote the price of stock i ($i = 1, 2$) in the up state in each period by $u_i S_i$ and the price in the down state by $d_i S_i$, where S_i is the price at the beginning of the period, and u_i and d_i are parameters to be specified. In each period, there are four possible combinations of returns on the two stocks: up for both stocks, up for stock 1 and down for stock~2, down for stock 1 and up for stock 2, and down for both stocks. Denote the probabilities of these four combinations by p_{uu}, p_{ud}, p_{du}, and p_{dd} respectively. Thus, there are eight parameters in the binomial model: the number N of periods (which defines the length of each period as $\Delta t = T/N$ where T is the option maturity), the up and down parameters u_i and d_i for each stock, and three probabilities (the fourth probability being determined by the condition that the probabilities sum to one).

Given the period length Δt, we want to choose the up and down parameters and the probabilities to match (or approximately match in an appropriate sense) the means, variances and covariances of the returns $\Delta S_i/S_i$ or the continuously-compounded returns $\Delta \log S_i$. There are two means, two variances and one covariance, so there are five restrictions to be satisfied and seven parameters. As in **?@sec-c_introcomputation**, it is convenient to take $d_i = 1/u_i$, leaving five restrictions and five free parameters.

As discussed in **?@sec-s_binomialparameters**, there are multiple ways to define the binomial model so that it converges to the continuous-time model as the number of periods is increased. As an example, we will describe here the suggestion of Trigeorgis [@Trigeorgis], which matches the means, variances and covariance of the continuously-compounded returns. Letting p_i denote the probability of the up state for stock i, matching the means and variances implies, as in **?@sec-s_binomialparameters**,

$$\begin{aligned}\log u_i &= \sqrt{\sigma_i^2 \Delta t + \nu_i^2 (\Delta t)^2}\;,\\ p_i &= \frac{1}{2} + \frac{\nu_i \Delta t}{2 \log u_i}\;.\end{aligned}$$

where $\nu_i = r - q_i - \sigma_i^2/2$. In terms of the notation p_{uu}, p_{ud}, p_{du}, and p_{dd}, the probability of the up state for stock 1 is $p_1 = p_{uu} + p_{ud}$ and the probability of the up state for stock 2 is $p_2 = p_{uu} + p_{du}$. Therefore,

$$p_{uu} + p_{ud} = \frac{1}{2} + \frac{\nu_1 \Delta t}{2 \log u_1}\;, \tag{56}$$

$$p_{uu} + p_{du} = \frac{1}{2} + \frac{\nu_2 \Delta t}{2 \log u_2}\;. \tag{57}$$

In the continuous time model, over a discrete time period Δt, the covariance of $\Delta \log S_1$ and $\Delta \log S_2$ is $\rho \sigma_1 \sigma_2 \Delta t$. In the binomial model, with $d_i = 1/u_i$, we have

$$E[\Delta \log S_1 \times \Delta \log S_2] = (p_{uu} - p_{ud} - p_{du} + p_{dd}) \log u_1 \log u_2\;.$$

Given that $E[\Delta \log S_i] = \nu_i \Delta t$, this implies a covariance of

$$(p_{uu} - p_{ud} - p_{du} + p_{dd}) \log u_1 \log u_2 - \nu_1 \nu_2 (\Delta t)^2 \ .$$

Matching the covariance in the binomial model to the covariance in the continuous-time model therefore implies

$$p_{uu} - p_{ud} - p_{du} + p_{dd} = \frac{\rho \sigma_1 \sigma_2 \Delta t + \nu_1 \nu_2 (\Delta t)^2}{\log u_1 \log u_2} \ . \tag{58}$$

We can solve the system Equation 56 - Equation 58, together with the condition that the probabilities sum to one, to obtain the probabilities p_{uu}, p_{ud}, p_{du}, and p_{dd}. This solution and a Python function for valuing an American spread call option are given in **?@sec-s_montecarlo_matlab**. This function operates much like the binomial valuation of American options described in **?@sec-c_introcomputation**. The primary difference is that the value of the option at maturity depends on both stock prices, so we have to consider each possible combination of stock prices. In an N–period model, there are $N+1$ nodes at the final date for each of the two stocks, and hence $(N+1)^2$ possible combinations of nodes. In fact, at each date n ($n = 0, \dots, N$) there are $(n+1)^2$ combinations of nodes to be considered. The computation time required for a spread call option is therefore roughly the square of the time required for a standard call.

Likewise, in an N–period model for a basket option written on three assets, there are $(n+1)^3$ combinations of nodes to be considered at date n; if there are five assets, there are $(n+1)^5$ combinations, etc. Thus, the computation time required increases exponentially with the number of assets. This can be a serious problem. For example, with five assets and $N = 99$, we would have 100^5 (10 billion) combinations. As this suggests, problems with multiple assets quickly become intractable in a binomial framework. This is called the curse of dimensionality.

Variance Swaps

Variance swaps are a type of futures contracts that allows investors to trade future realized volatility against current option implied variance. Unlike traditional futures, which provide a payoff based on the difference between the underlying asset's price and the futures price, variance swaps provide a payoff based on the difference between the realized variance of the underlying asset and the predetermined variance level (termed "Strike'' in practice), which is the futures price in variance unit.

Variance swaps have gained popularity due to their ability to provide pure exposure to the volatility of an underlying asset, independent of its price movements. Unlike traditional options, variance swaps allow investors to speculate on or hedge against changes in volatility without the need for constant delta hedging. This makes them an effective tool for managing volatility risk and for executing volatility arbitrage strategies, as they simplify the trading of variance and offer a more straightforward payoff structure based on realized versus implied volatility.

Payoff of a Variance Swap

A variance swap is a future contract on future realized variance. Its main components include:

- **Notional Amount**: Specifies the amount of money to be paid for each unit of variance difference.
- **Strike**: The predetermined level of variance agreed upon at the inception of the swap.
- **Realized Variance**: Calculated from the returns of the underlying asset over the life of the swap.

The payoff of a variance swap is given by:

$$\text{Payoff} = \text{Notional} \times (\text{Realized Variance} - \text{Strike})$$

Where:

- **Realized Variance** is typically calculated using the formula:

$$\text{Realized Variance} = \frac{252}{N} \sum_{i=1}^{N} \left(\log \left(\frac{S_i}{S_{i-1}} \right) \right)^2$$

Here, S_i represents the price of the underlying asset at time i, and N is the number of trading days over the contract period.

Example: Suppose an investor enters into a variance swap with a notional amount of $100,000 and a strike of 0.04 (implying a volatility strike of 20%). If the realized variance over the swap's life is 0.06 (implying a realized volatility of 24.5%), the payoff would be:

$$\text{Payoff} = 100,000 \times (0.06 - 0.04) = 100,000 \times 0.02 = 2,000$$

This means the investor would receive $2,000 at the end of the swap period.

As all futures constracts, variance swaps are marked to market on each trading day.

Variance swaps can be used to hedge against volatility risk. For example, a portfolio manager concerned about increasing market volatility can buy variance swaps to protect the portfolio's value.

Traders can also use variance swaps to speculate or arbitrage on future volatility. For example, If a trader believes that future volatility will be higher than the current implied volatility, they can enter a variance swap to profit from this view.

Pricing of a Variance Swap for Stocks with Geometric Brownian Motion Prices

Similor the pricing of a futures, we need find the fair swap strike such that the initial value of the swap is zero. For simplicity, consider a stock whose price S_t follows a geometric Brownian motion process:

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dZ_t$$

Applying Ito's Lemma to $\log(S_t)$:

$$d(\log S_t) = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma\, dZ_t$$

Rearranging terms, we get:

$$\frac{dS_t}{S_t} - d(\log S_t) = \frac{\sigma^2}{2}\, dt$$

Integrating over the life of the swap ([0, T]):

$$\text{Total Variance} = \frac{1}{T}\int_0^T \sigma^2\, dt = \frac{2}{T}\left(\int_0^T \frac{dS_t}{S_t} - \log\left(\frac{S_T}{S_0}\right)\right),$$

which implies that the total variance can be replicated by continuously rebalancing the stock position with weight $1/S_t$ and shorting one unit of the security which pays the log return of the stock. However, this log return secutiry is not traded in the market. Fortunately, we can synthesize this security with a forward and options, because

$$-ln(\frac{S_T}{S^*}) = -\frac{S - S^*}{S^*} + \int_0^{S^*} \frac{(K-S)^+}{K^2} dK + \int_{S^*}^{\infty} \frac{(S-K)^+}{K^2} dK,$$

where S^* is an arbitrary cut-off strike for calls and puts, the first term on the right hand side represents a short position in the forward contract on the stock, the second term represents a continuum of put options with strike prices from 0 to S^*, and the third term represents a continuum of call options with strike prices above S^*. Taking expectation under the risk neutral measure, we have that the fair swap strike K_{var} is equal to:

$$K_{\text{var}} = \frac{2}{T}\left(rT - \left(\frac{S_0}{S^*}e^{rT} - 1\right) - \log\left(\frac{S^*}{S_0}\right) + e^{rT}\int_0^{S^*} \frac{1}{K^2}P(K)\, dK + e^{rT}\int_{S^*}^{\infty} \frac{1}{K^2}C(K)\, dK\right)$$

Choosing S^* to be the current forward price $F_0 = S_0 e^{rT}$:

$$K_{\text{var}} = \frac{2e^{rT}}{T}\left(\int_0^{F_0} \frac{P(K)}{K^2}\,dK + \int_{F_0}^{\infty} \frac{C(K)}{K^2}\,dK\right)$$

This formula allows us to compute the fair strike price of a variance swap using the prices of European call and put options. The following code provides an example of this computation using data from Yahoo finance for Apple.

```python
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d
from scipy.stats import norm
import yfinance as yf

def indicator_function(condition):
    return 1 if condition else 0

def calculate_variance_swap_strike(S, r, T, options_data):
    # Extract call and put prices
    call_data = options_data[options_data['Type'] == 'call'].copy()
    put_data = options_data[options_data['Type'] == 'put'].copy()

    # Interpolate call and put prices
    strikes = np.unique(options_data['Strike'])
    call_interp = interp1d(call_data['Strike'], call_data['Price'], fill_value="extrapolate"
    put_interp = interp1d(put_data['Strike'], put_data['Price'], fill_value="extrapolate")

    # Integrate using numerical methods (trapezoidal rule)

    K_min, K_max = strikes.min(), strikes.max()
    K = np.linspace(K_min, S*np.exp(r*T), 500)
    integral1 = np.trapezoid(put_interp(K) / K**2, K)
    K = np.linspace(S*np.exp(r*T),K_max, 500)
    integral2 = np.trapezoid(call_interp(K) / K**2, K)

    # Calculate the variance swap strike
    variance_swap_strike = np.sqrt(2 * (integral1+integral2) / T)
    return variance_swap_strike

# Example usage
ticker = "AAPL"
```

```python
S = 150.0  # Current stock price
r = 0.01  # Risk-free rate
T = 0.5  # Time to maturity (6 months)

# Fetch options data from Yahoo Finance
stock = yf.Ticker(ticker)
expiry = stock.options[0]
opt_chain = stock.option_chain(expiry)
calls = opt_chain.calls[['strike', 'lastPrice']].copy()
puts = opt_chain.puts[['strike', 'lastPrice']].copy()
calls.columns = ['Strike', 'Price']
puts.columns = ['Strike', 'Price']
calls['Type'] = 'call'
puts['Type'] = 'put'
options_data = pd.concat([calls, puts])

# Calculate the variance swap strike
variance_swap_strike = calculate_variance_swap_strike(S, r, T, options_data)
print(f"Variance Swap Strike: {variance_swap_strike:.4f}")
```

```
Variance Swap Strike: 0.6310
```

Exercises

Exercise 0.1. Intuitively, the value of a forward-start call option should be lower the closer is the date T at which the strike is set to the date T' at which the option matures, because then the option has less time to maturity after being created at T. Create an Excel worksheet to confirm this. Allow the user to input S, r, σ, q, and T'. Compute and plot the value of the option for $T = 0.1T'$, $T = 0.2T'$, ..., $T = 0.9T'$.

Exercise 0.2. Create an Excel worksheet to demonstrate the additional leverage of a call-on-a-call relative to a standard call. Allow the user to input S, r, σ, q, and T'. Use the `Black-Scholes_Call` function to compute and output the value C of a European call with strike $K' = S$ (i.e., the call is at the money) and maturity T'. Use the `Call_on_Call` function to compute and output the value of a call option on the call with strike $K = C$ (i.e., the call-on-a-call is at the money) and maturity $T = 0.5T'$. Compute the percentage returns the standard European call and the call-on-a-call would experience if the stock price S instantaneously increased by 10%.

Exercise 0.3. Create an Excel worksheet to illustrate the early exercise premium for an American call on a stock paying a discrete dividend. Allow the user to input S, r, σ, and

T'. Take the date of the dividend payment to be $T = 0.5T'$ and take the strike price to be $K = S$. As discussed in Section , the value of a European call is given by the Black-Scholes formula with $S - \mathrm{e}^{-rT}D$ being the initial asset price and $q = 0$ being the constant dividend yield. Use the function `American_Call_Dividend` to compute the value of an American call for dividends $D = .1S$, ...$D = .9S$. Subtract the value of the European call with the same dividend to obtain the early exercise premium. Plot the early exercise premium against the dividend D.

Exercise 0.4. Create a Python function to value a simple chooser (a chooser option in which $K_c = K_p$ and $T_c = T_p$) using put-call parity to compute S^* as mentioned in Section . Verify that the function gives the same result as the function `Chooser`.

Exercise 0.5. Create a Python code to compare the cost of a simple chooser to that of a straddle (straddle = call + put with same strike and maturity). Allow the user to input S, r, σ, q, and T'. Take the time to maturity of the underlying call and put to be T' for both the chooser and the straddle. Take the strike prices to be $K = S$. Take the time the choice must be made for the chooser to be $T = 0.5T'$. Compute the cost of the chooser and the cost of the straddle.

Exercise 0.6. A stock has fallen in price and you are attempting to persuade a client that it is now a good buy. The client believes it may fall further before bouncing back and hence is inclined to postpone a decision. To convince the client to buy now, you offer to deliver the stock to him at the end of two months at which time he will pay you the lowest price it trades during the two months plus a fee for your costs. The stock is not expected to pay a dividend during the next two months. Assuming the stock actually satisfies the Black-Scholes assumptions, find a formula for the minimum fee that you would require. (Hint: It is almost in Section .) Create an Excel worksheet allowing the user to input S, r, and σ and computing the minimum fee.

Exercise 0.7. Suppose you must purchase 100 units of an asset at the end of a year. Create an Excel worksheet simulating the asset price and comparing the quality of the following hedges (assuming 100 contracts of each):

1. a standard European call,
2. a down-and-out call in which the knock-out barrier is 10% below the current price of the asset.

Take both options to be at the money at the beginning of the year. Allow the user to input S, r, σ and q. Generate 500 simulated end-of-year costs (net of the option values at maturity) for each hedging strategy and create histogram charts to visually compare the hedges. Note: to create histograms, you will need the Data Analysis add-in, which may be need to be loaded (click Tools/Add Ins).

Exercise 0.8. Compute the prices of the options in the previous exercise. Modify the simulations to compare the end-of-year costs including the costs of the options, adding interest on the option prices to put everything on an end-of-year basis.

Exercise 0.9. Modify Exercise 0.7 by including a third hedge: a combination of a down-and-out call as in part (b) of Exercise 0.7 and a down-and-in call with knockout barrier and strike 10% below the current price of the asset. Note that this combination forms a call option with strike that is reset when the underlying asset price hits a barrier.

Exercise 0.10. Modify Exercise 0.8 by including the hedge in Exercise 0.9. Value the down-and-in call using the function `Down_And_Out_Call` and the fact that a down-and-out and down-and-in with the same strikes and barriers form a standard option.

Exercise 0.11. Each week you purchase 100 units of an asset, and you want to hedge your total quarterly (13-week) cost. Create an Excel worksheet simulating the asset price and comparing the quality of the following hedges:

1. a standard European call maturing at the end of the quarter ($T = 0.25$) on 1300 units of the asset,
2. 13 call options maturing at the end of each week of the quarter, each written on 100 units of the asset, and
3. a discretely sampled average-price call with maturity $T = 0.25$ written on 1300 units of the asset, where the sampling is at the end of each week.

4. a discretely sampled geometric-average-price call with maturity $T = 0.25$ written on 1300 units of the asset, where the sampling is at the end of each week.

Allow the user to input S, r, σ and q. Assume all of the options are at the money at the beginning of the quarter ($K = S$). Compare the hedges as in Exercise 0.7.

Exercise 0.12. In the setting of the previous problem, compute the prices of the options in parts (a), (b) and (d). Modify the simulations in the previous problem to compare the end-of-quarter costs including the costs of the options (adding interest on the option prices to put everything on an end-of-quarter basis).

Exercise 0.13. Using the put-call parity relation, derive a formula for the value of a forward-start put.

Exercise 0.14. Derive **?@eq-callonaput** for the value of a call on a put.

Exercise 0.15. Complete the derivation of **?@eq-chooser2** for the value of a chooser option.

Exercise 0.16. Derive a formula for the value of a put option on the maximum of two risky asset prices.

Exercise 0.17. Using the result of the preceding exercise and Margrabe's formula, verify that calls and puts (having the same strike K and maturity T) on the maximum of two risky asset prices satisfy the following put-call parity relation:

$$\begin{aligned}
&\mathrm{e}^{-rT}K + \text{Value of call on max} \\
&\qquad = \mathrm{e}^{-q_2T}S_2(0) + \text{Value of option to exchange asset 2 for asset 1} \\
&\qquad\qquad + \text{Value of put on max}\,.
\end{aligned}$$