In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps
import numpy.random as npr
```

1. Textbook version of two assets correlation call option price

In [ ]:
```
S_a = 52
S_b = 65
X_1 = 50
X_2 = 70
rf = 0.1
b_1 = 0.1
b_2 = 0.1
sigma_1 = 0.2
sigma_2 = 0.3
rho = 0.75
T = 0.5

y1 = (np.log(S_a / X_1) + (b_1 - sigma_1 ** 2 / 2) * T) / (sigma_1 * np.sqrt(T))
y2 = (np.log(S_b / X_2) + (b_2 - sigma_2 ** 2 / 2) * T) / (sigma_2 * np.sqrt(T))

def std_bi_nd_cdf(x, y, rho):
    return sps.multivariate_normal.cdf([x, y], mean=[0, 0], cov=np.array([[1, rho], [rho, 1]]))

m1 = std_bi_nd_cdf(y2 + sigma_2 * np.sqrt(T), y1 + rho * sigma_2 * np.sqrt(T), rho)
m2 = std_bi_nd_cdf(y2, y1, rho)

c = S_b * np.exp((b_2 - rf) * T) * m1 - X_2 * np.exp(- rf * T) * m2
print('The two assets correlation call option premium: ', c)
```

```
The two assets correlation call option premium:  4.707330012666844
```

2. The correct closed-form of two assets correlation call option

In [ ]:
```
# The correct pricing formula of two assets correlation call option

y1 = (np.log(S_a / X_1) + (rf - b_1 - sigma_1 ** 2 / 2) * T) / (sigma_1 * np.sqrt(T))
y2 = (np.log(S_b / X_2) + (rf - b_2 - sigma_2 ** 2 / 2) * T) / (sigma_2 * np.sqrt(T))

def std_bi_nd_cdf(x, y, rho):
    return sps.multivariate_normal.cdf([x, y], mean=[0, 0], cov=np.array([[1, rho], [rho, 1]]))

m1 = std_bi_nd_cdf(y2 + sigma_2 * np.sqrt(T), y1 + rho * sigma_2 * np.sqrt(T), rho)
m2 = std_bi_nd_cdf(y2, y1, rho)

c = S_b * np.exp((-b_2) * T) * m1 - X_2 * np.exp(- rf * T) * m2
print('The two assets correlation call option premium: ', c)
```

```
The two assets correlation call option premium:  3.2425278975519625
```

3. The derivation of closed-form of two assets correlation call option

$$
\begin{aligned}
S_T^{(2)} > X_2 &\Rightarrow S_0^{(2)} e^{\sigma_2\sqrt{T}x+(r-b_2-\frac{\sigma_2^2}{2})T} > X_2 \\
&\Rightarrow x > -\frac{ln(\frac{S_0^{(2)}}{X_2}) + (r-b_2-\frac{\sigma_2^2}{2}T)}{\sigma_2\sqrt{T}} = -y_2 \\
S_T^{(1)} > X_1 &\Rightarrow S_0^{(1)} e^{\sigma_1\sqrt{T}y+(r-b_1-\frac{\sigma_1^2}{2})T} > X_1 \\
&\Rightarrow y > -\frac{ln(\frac{S_0^{(1)}}{X_1}) + (r-b_1-\frac{\sigma_1^2}{2}T)}{\sigma_1\sqrt{T}} = -y_1
\end{aligned}
$$

$$
\begin{aligned}
e^{-rT}\mathbb{E}^Q[1_{\{S_T^{(1)}>X_1\}}(S_T^{(2)}-X_2)^+] &= e^{-rT}\int_{-y_1}^{\infty}\int_{-y_2}^{\infty}(S_0^{(2)}e^{\sigma_2\sqrt{T}x+(r-b_2-\frac{\sigma_2^2}{2})T}-X_2)\frac{1}{2\pi\sqrt{1-\rho}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y \\
&= e^{-rT}\int_{-y_1}^{\infty}\int_{-y_2}^{\infty}(S_0^{(2)}e^{\sigma_2\sqrt{T}x+(r-b_2-\frac{\sigma_2^2}{2})T})\frac{1}{2\pi\sqrt{1-\rho}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y \\
&\quad - e^{-rT}\int_{-y_1}^{\infty}\int_{-y_2}^{\infty}X_2\frac{1}{2\pi\sqrt{1-\rho}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y
\end{aligned}
$$

For the first term:

$$e^{-rT}\int_{-y_1}^{\infty}\int_{-y_2}^{\infty}(S_0^{(2)}e^{\sigma_2\sqrt{T}x+(r-b_2-\frac{\sigma_2^2}{2})T})\frac{1}{2\pi\sqrt{1-\rho^2}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y$$

$$z_1 = y-\rho\sigma_2\sqrt{T}$$
$$z_2 = x-\sigma_2\sqrt{T}$$

$$= S_0^{(2)}e^{-b_2T}\int_{-y_1-\rho\sigma_2\sqrt{T}}^{\infty}\int_{-y_2-\sigma_2\sqrt{T}}^{\infty}\frac{1}{2\pi\sqrt{1-\rho^2}}e^{\frac{\sigma_2^2}{2}T+\sigma^2\sqrt{T}z_2-\frac{1}{2\pi\sqrt{1-\rho^2}}((z_2+\sigma_2\sqrt{T})^2-2\rho(z_2+\sigma_2\sqrt{T})(z_1+\rho\sigma_2\sqrt{T})+(z_1+\rho\sigma_2\sqrt{T})^2))}\mathrm{d}z_2\mathrm{d}z_1$$
$$= S_0^{(2)}e^{-b_2T}\int_{-y_1-\rho\sigma_2\sqrt{T}}^{\infty}\int_{-y_2-\sigma_2\sqrt{T}}^{\infty}\frac{1}{2\pi\sqrt{1-\rho^2}}e^{(z_1^2-2\rho z_1z_2+z_2^2)}\mathrm{d}z_2\mathrm{d}z_1$$
$$= S_0^{(2)}e^{-b_2T}\int_{-\infty}^{y_1+\rho\sigma_2\sqrt{T}}\int_{-\infty}^{y_2+\sigma_2\sqrt{T}}\frac{1}{2\pi\sqrt{1-\rho^2}}e^{(z_1^2-2\rho z_1z_2+z_2^2)}\mathrm{d}z_2\mathrm{d}z_1$$
$$= S_0^{(2)}e^{-b_2T}M(y_2+\sigma_2\sqrt{T}, y_1+\rho\sigma_2\sqrt{T}, \rho)$$

For the second term:

$$e^{-rT}\int_{-y_1}^{\infty}\int_{-y_2}^{\infty}X_2\frac{1}{2\pi\sqrt{1-\rho}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y$$
$$= e^{-rT}\int_{-\infty}^{y_1}\int_{-\infty}^{y_2}X_2\frac{1}{2\pi\sqrt{1-\rho}}e^{-\frac{1}{2(1-\rho^2)}(x^2-\rho xy+y^2)}\mathrm{d}x\mathrm{d}y$$
$$= X_2e^{-rT}M(y_2, y_1, \rho)$$

$$\Rightarrow e^{-rT}\mathbb{E}^Q[1_{\{S_T^{(1)})>X_1\}}(S_T^{(2)}-X_2)^+] = S_0^{(2)}e^{-b_2T}M(y_2+\sigma_2\sqrt{T}, y_1+\rho\sigma_2\sqrt{T}, \rho) - X_2e^{-rT}M(y_2, y_1, \rho)$$

$$, where \quad y_1 = \frac{ln(\frac{S_0^{(1)}}{X_1})+(r-b_1-\frac{\sigma_1^2}{2}T)}{\sigma_1\sqrt{T}}$$
$$y_2 = \frac{ln(\frac{S_0^{(2)}}{X_2})+(r-b_2-\frac{\sigma_2^2}{2}T)}{\sigma_2\sqrt{T}}$$

4. Using Monte Carlo Mehtod to price the two assets correlation call option

In [ ]:
```
# Use Cholesky decomposition to generate samples from bivariate normal distribution
corr = np.array([[1, rho], [rho, 1]])
print(f"Input correlation matrix: \n{corr}\n")

chol = np.linalg.cholesky(corr)
z = np.matmul(chol, npr.normal(size=(2, 10000000)))
```

```
Input correlation matrix:
[[1.   0.75]
 [0.75 1.  ]]
```

In [ ]:
```
# The GBM formula
def gbm(S_0, rf, q, sigma, delta_t, z):
    return S_0 * np.exp((rf - q - sigma**2 / 2) * delta_t + sigma * np.sqrt(delta_t) * z)

p1 = gbm(S_a, rf, b_1, sigma_1, T, z[0, :])
p2 = gbm(S_b, rf, b_2, sigma_2, T, z[1, :])

# Calculate the correlation between the two price paths' return
print(np.corrcoef(p1 / S_a - 1, p2 / S_b - 1))

callpayoff = np.maximum(p2 - X_2, 0) * (p1 > X_1)
print('The two asset correlation call option price from the Monte Carlo method: ', np.exp(- rf * T) * callpayoff.mean())
```

```
[[1.         0.74606586]
 [0.74606586 1.        ]]
The two asset correlation call option price from the Monte Carlo method:  3.2394208619516367
```

5. Use expectation method to get call price

$$callprice = e^{-rT}\mathbb{E}^Q[1_{\{S_1>X_1\}}(S_2-X_2)^+]$$

Monte Carlo Integration

Monte Carlo integration is a technique that uses random sampling to approximate definite integrals. For a function $q(x)$ over the interval $[a, b]$, the integral can be estimated as:

$$\int_a^b q(x)\mathrm{d}x = (b-a)\int_a^b q(x)\frac{1}{b-a}\mathrm{d}x = (b-a)\int_a^b q(x)f_U(x)\mathrm{d}x = (b-a)\mathbb{E}(q(U))$$

Here, U is a continuous uniform random variable with density function $f_U(x) = \frac{1}{b-a}$.

Moving to high-dimension

- One dimension

$$\int_c^d q(x)\mathrm{d}x = (b-a)\mathbb{E}(q(U_x))$$

- Two and higher Dimension

$$\int_c^d \int_a^b q(x,y)\mathrm{d}x\mathrm{d}y = (d-c)(b-a)\mathbb{E}(q(U_x, U_y))$$

Reference: FN6815 Lecture written by Dr. Yang Ye

In [ ]:
```
def mc_int_2d(func, rg1, rg2, N=int(1e6)):
    nr1 = npr.uniform(rg1[0], rg1[1], size=N)
    nr2 = npr.uniform(rg2[0], rg2[1], size=N)
    sums = func(nr1, nr2)
    # print(sums.shape)
    res = np.mean(sums) * (rg1[1] - rg1[0]) * (rg2[1] - rg2[0])
    std = np.std(sums) * (rg1[1] - rg1[0]) * (rg2[1] - rg2[0]) / np.sqrt(N)
    return res, std
```

In [ ]:
```
def call_payoff(S1, S2, K1, K2):
    return np.maximum(S2 - K2, 0) * (S1 > K1)

# def put_payoff(S, strike):
#     return np.maximum(strike - S, 0)

def option_payoff(S_a, S_b, K1, K2, T, rf, sigma_1, sigma_2, payoff_func, z1, z2):
    ST_a = gbm(S_a, rf, b_1, sigma_1, T, z1)
    ST_b = gbm(S_b, rf, b_2, sigma_2, T, z2)
    return payoff_func(ST_a, ST_b, K1, K2)
```

In [ ]:
```
def rainbow_opt_integrand(S_a, S_b, K1, K2, T, rf, sigma_1, sigma_2, rho, payoff_func):
    def _inner(z1, z2, payoff_func=payoff_func):
        return option_payoff(
            S_a, S_b, K1, K2, T, rf, sigma_1, sigma_2, payoff_func, z1, z2
        ) * sps.multivariate_normal.pdf(np.vstack((z1, z2)).T, mean=[0, 0], cov=np.array([[1, rho], [rho, 1]]))

    return _inner
```

In [ ]:
```
integral = mc_int_2d(rainbow_opt_integrand(S_a, S_b, X_1, X_2, T, rf, sigma_1, sigma_2, rho, call_payoff), (-10, 10), (-10, 10), int(10000000))
c = np.exp(-rf * T) * integral[0]
print('call price: ', c, 'standard deviation: ', integral[1])
```

```
call price:  3.250256075516205 standard deviation:  0.010092872609807464
```