

Exotic Options: Pricing a Rainbow Option of Two Assets via Monte Carlo and Variance Reduction Techniques

Pattas Panagiotis (MFT2510)

DelRosso Vasileios (MFT2509)

Lathiotakis Efthymios (MFT2504)

Agianogloy Nikolaos-Vasileios (MFT2501)

Instructor: Englezos Nikolaos — University of Piraeus

Department of Banking and Financial Management

February 13, 2026

Abstract

This study is about the pricing of an exotic class derivative, that of an Option named Rainbow under a two-dimensional Black and Scholes setting. Although a closed-form expression exists (Stulz), the pricing is computed with Monte Carlo numerical estimation and furthermore it is showed that classical variance reduction techniques—Antithetic Variates and Control Variates—improve accuracy and/or runtime relative to crude Monte Carlo.

Keywords : Exotic Options, Rainbow, Call on min, Monte Carlo Simulation, Variance Reduction

1 Introduction

1.1 Definition: Rainbow Options

First of all there must be a reference that the main theoretical blueprints for the rainbow options were established in the late seventies (1978) by Margrabe, who studied the option to exchange one asset for another one at maturity. That special class of derivative can be plainly described by a call where the strike price is the value of a second stochastic underlying asset.

Therefore we can generally define, that a Rainbow Option is a type of option whose value is dependant on multiple underlying assets and it can have many possible structure payoffs up until maturity. Based on this whole concept, slightly later in the early eighties (1982) Stulz enhanced this prior framework by introducing the concept of "Call on min". In that form of the derivative the buyer of the option now has the right to purchase the minimum of those two assets at a fixed strike price upon expiration. Last but not least, apart from the payoff form that will be examined here—"Call on min" some others include :

- Best of assets or cash : $\max(S_1, S_2 \dots S_i, K)$
- Call on max : $\max(\max(S_1, \dots S_n) - K, 0.0)$
- Call on min : $\max(\min(S_1, \dots S_n) - K, 0.0)$
- Put on max : $\max(K - \max(S_1, \dots S_n), 0.0)$
- Put on min : $\max(K - \min(S_1, \dots S_n), 0.0)$
- Put 2 and Call 1 : $\max(S_1 - S_2, 0.0)$

Thus, here the payoff's structure is : $C = \max(\min(S_1, \dots S_n) - K, 0.0)$ Therefore the rainbow options can be an advanced tool in the buyer portfolio because due to their nature they provide a natural diversification tool due to the multi asset exposure under one derivative contract, instead of holding those individualy. Also they might have enhanced returns and a better cost efficiency depending on their payoff structure.

1.2 Model Assumptions

Moving on to the main body of this analysis there is the need to highlight some important assumptions made a priori, in order to simplify the pricing modelling.

Here the Rainbow Option is in the form of a European Call thus it can only be exercised on its maturity date. Also the derivative is constructed by only two assets who both are subject to Geometrical Brownian Motion. Therefore the "call-on-min" formula is in the form of:

- $C = \max(\min(S_1, S_2) - K, 0.0)$

where each asset $S_i \quad i = 1, 2$ satisfies the following stochastic differential equation (SDE):

$$dS_i(t) = rS_i(t) dt + \sigma_i S_i(t) dZ_i(t), \quad i = 1, 2, \longleftrightarrow \frac{dS_i(t)}{S_i(t)} = rdt + \sigma_i dZ_i(t) \quad (1)$$

with each variable representing:

- Underlying asset's price $S_i(t)$
- risk-free rate r ,
- volatilities σ_i ,
- and Brownian motions $Z_i(t)$.

Given the above equation (1) there is another important assumption that was made. Here no asset will be distributing dividends so:

- Dividends $q_1 = q_2 = 0$

Last but an extreme important assumption is that in the presented model, the assets are taken to be **uncorrelated**:

- $\text{Corr}(dZ_1, dZ_2) = 0 \xrightarrow{\text{SDE} \quad (1)} \text{Corr}(S_1, S_2) = 0$,

Thus each Brownian Motion Z_i is independent.

1.3 Theoretical Pricing

Since there is already the Stochastic Differential Equation (1) there is the need to solve to obtain the value of the call's price at the maturity.

Applying Itô's lemma :

$$dG = \left(\frac{dG}{dt} + \frac{dG}{dS} \mu S + \frac{1}{2} \frac{\partial^2 G}{\partial S^2} \sigma^2 S^2 \right) dt + \frac{\partial G}{\partial S} \sigma S dZ_t \quad (2)$$

which implies that the stochastic process is followed by a function $G = G(t, S_i)$ of S_i, t :

A key aspect for proceeding with more algebraic operations is find the best possible replacement of $G(t, S)$ that solves the (2). Here by glimpsing the (1) it is observable that $\frac{dS}{dt}$ leads to the $\ln S_i$ being the best replacement for $G(t, S)$.

In order to understand why there is the need first for some algebra to be performed:

$$\begin{aligned} G &= \ln S \xrightarrow{(2)} d(\ln S) = \left(0 + \frac{1}{S} r S + \frac{1}{2} \frac{-1}{S^2} \sigma^2 S^2 \right) dt + \frac{1}{S} \sigma S dZ_t \Rightarrow \\ &\Rightarrow dG = \left(r - \frac{1}{2} \sigma^2 \right) dt + \sigma S dZ_t \end{aligned}$$

With, $(r - \frac{1}{2} \sigma^2)$ and σ being constant this shows that G follows a Generalized Wiener Process. So be continuing the analysis the next step is to follow a simple integration on both parts and since our call is only exercised to its maturity we have a closed time interval $t \in [0, T]$. Hence,

$$\int_0^T d(\ln S_i) = \int_0^T \left(r - \frac{1}{2} \sigma^2 dt \right) + \int_0^T \sigma dZ_{i,t} \Rightarrow \ln S_i(T) = \ln S_i(0) + \left(r - \frac{1}{2} \sigma^2 \right) T + \sigma Z_i(T)$$

Last step is to exponentiate both sides, yielding the solution below:

$$S_i(T) = S_i(0) \exp \left(\left(r - \frac{1}{2} \sigma_i^2 \right) T + \sigma_i \sqrt{T} \varepsilon_i \right), \quad \varepsilon_i \sim \mathcal{N}(0, 1). \quad (3)$$

Thus, $\ln S_i(T)$ is normal with mean $\ln S_i(0) + (r - \frac{1}{2} \sigma_i^2)T$ and variance $\sigma_i^2 T$.

$$\ln S_{i,T} \sim N[\ln S(0) + (r - \frac{1}{2}\sigma^2)T, \sigma_i\sqrt{T}]$$

$$E[S_t] = S_0 \exp(r \cdot T)$$

$$Var(S_T) = S_0^2 \exp(2r \cdot T)(\exp(\sigma^2 \cdot T) - 1)$$

Before we conclude this part a reference must be made again to the choice of $G = \ln S$. Apart from the option given by the dS/dt one more important reason that the discretization of the equation was made this way was that the Stochastic Differential Equation (1) in the right-hand part there was the form: $S_i(t) \cdot dZ_i(t)$ so if anyone went on to discretize directly the SDE then that part would change per each path up until maturity thus introducing a discretization error. So practically $\ln S$ is the most natural way to jump directly from $0 \rightarrow T$ without causing error inference.

1.4 Closed-Form Pricing Formula (Stulz)

In this section there will be presented a short but core disclaimer on the closed-form formula for the "call on min" provided by Stulz. Its importance is derived from the key aspect that since it exists in most cases it must be the preferred choice when trying to compute the exact payoff of the derivative. The nature of closed-form formulas in general removes any possible errors induced to the pricing model by numerical approximations as Monte Carlo.

Here, since the payoff depends on both assets jointly, there is the requirement of using the joint distribution of $(\ln S_1(T), \ln S_2(T))$. The main reason of that acknowledgement is the fact that : $E[f(S_1, S_2)] \neq f(E[S_1], E[S_2])$. So, let :

$$(Y_1, Y_2) = (\ln S_1(T), \ln S_2(T)) \sim N(\mu, \Sigma)$$

with the μ being a column matrix:

$$\mu = \begin{pmatrix} \ln S_{i=1,0} + (r - 1/2 \cdot \sigma_1^2) T \\ \ln S_{i=2,0} + (r - 1/2 \cdot \sigma_2^2) T \end{pmatrix}$$

and the covariance matrix being:

$$\Sigma = \begin{pmatrix} \sigma_1^2 \cdot T & \rho \cdot \sigma_1 \sigma_2 \cdot T \\ \rho \cdot \sigma_1 \sigma_2 \cdot T & \sigma_2^2 \cdot T \end{pmatrix}$$

But since in this case $\rho = 0$ then the positive definite covariance matrix becomes:

$$\Sigma = \begin{pmatrix} \sigma_1^2 \cdot T & 0 \\ 0 & \sigma_2^2 \cdot T \end{pmatrix}$$

with that implying that it is diagonal. Hence the (Y_1, Y_2) are independent normal variables. Following is the exact closed form-formula of Stulz with all the auxiliary quantities:

Final Closed-Form Formula

$$C_{min}(0) = S_{1,0} \cdot e^{-q_1 T} M(d_1, -d; \rho_1) + S_{1,0} \cdot e^{-q_2 T} M(d_2, d - \Sigma\sqrt{T}; -\rho_2) - K e^{-rT} M(d_1 - \sigma_1\sqrt{T}, d_2 - \sigma_2\sqrt{T}; \rho) \quad (4)$$

With the auxiliary variables being :

$$d_i = \frac{\ln(S_{i,0}/K) + (r - q_i + \frac{1}{2}\sigma_i^2)T}{\sigma_i\sqrt{T}} \quad i = 1, 2$$

$$\Sigma^2 = \sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2 \xrightarrow{\rho=0} \Sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$$

$$d_i = \frac{\ln(\frac{S_{1,0}}{S_{2,0}}) + (q_2 - q_1 + \frac{1}{2}\Sigma)T}{\Sigma\sqrt{T}} \quad i = 1, 2$$

and the $M(a, b; \lambda)$ denotes the Cumulative Distribution Function of the Standard Bivariate Normal with correlation λ :

$$M(a, b; \lambda) = \int_{-\infty}^a \int_{-\infty}^b \frac{1}{2\pi\sqrt{1-\lambda^2}} \exp\left[-\frac{-x^2 - 2\lambda xy + y^2}{2(1-\lambda^2)}\right] dy dx$$

where in this model :

- $M(d_1, -d; 0)$
- $M(d_2, d - \Sigma\sqrt{T}; -\sigma_1/\Sigma)$
- $M(d_1 - \sigma_1\sqrt{T}, d_2 - \sigma_2\sqrt{T}; -\sigma_2/\Sigma)$

Therefore a closed-form expression for a call on the minimum exists (attributed to Stulz), and is computed by finding the bivariate normal CDF. In principle, using that closed-form is preferable when available; nevertheless, Monte Carlo remains valuable for extensions where closed forms do not exist, for validation, and for building intuition about variance reduction methods.

2 Monte Carlo Pricing

Having concluded the base framework of the theoretical pricing, now it is time to proceed with the implementation of Monte Carlo simulations to try and estimate the fair value of a European Rainbow Call on min option. Since though plain Monte Carlo requires a lot of computational power especially when there is a high amount of simulations to be generated there is the need to support it with additional techniques.

These two are variance reductions methods names respectively Antithetic Variates and Control Variates and they can largely enhance the efficiency and the accuracy of our estimates. The way that they achieve this is by reducing the standard error of the simulation without increasing the number of simulation and therefore the computational cost.

Simulating Asset Prices

Since the correlation in our modelling is set to zero, $\rho = 0$ then the two assets can be simulated independently. Recalling the solution of the Stochastic Differential Equation (1) we got the form of (3). So now the algorithm for simulating those independent asset paths is straightforward. The main key is to simulate up to N times, $j = 1 : N$, the independent asset paths with the payoff be given by:

$$\text{Payoff}_j = \max(\min(S_{1,j}(T), S_{2,j}(T)) - K, 0.0)$$

and then the generic Monte Carlo estimate of the option's present value will be :

$$V_0 \approx e^{-rT} \max(\min(S_{1,j}(T), S_{2,j}(T)) - K, 0.0)$$

2.1 Crude Monte Carlo Estimator

Given the previous blueprint and the fact that our random numbers are independent $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1)$, from $S_1(T), S_2(T)$, we compute the payoff, with the discounted factor as:

$$\hat{V}_{\text{MC}} = e^{-rT} \frac{1}{N} \sum_{j=1}^N \max(\min(S_{1,j}(T), S_{2,j}(T)) - K, 0).$$

By the Law of Large Numbers, the estimator of our plain Monte Carlo $\hat{V}_{\text{MC}} \rightarrow V$, where V is the true expected value of the option as $N \rightarrow \infty$. The importance is also that its standard error scales like $\mathcal{O}(1/\sqrt{N})$, which is also the convergence rate. Below there is a summary of the whole Crude Monte Carlo(MC) pricing Algorithm

Crude Monte Carlo (MC) Pricing Algorithm

Input: $S_1, S_2, K, r, \sigma_1, \sigma_2, T, N$ (and correlation ρ if used)

set sum = 0

for $j = 1, \dots, N$ **do**

Generate $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1)$

(If correlated) set $\varepsilon_2 \leftarrow \rho \varepsilon_1 + \sqrt{1 - \rho^2} \varepsilon_2$

 set $S_1(T) = S_1 \exp\left((r - \frac{1}{2}\sigma_1^2)T + \sigma_1 \sqrt{T} \varepsilon_1\right)$

 set $S_2(T) = S_2 \exp\left((r - \frac{1}{2}\sigma_2^2)T + \sigma_2 \sqrt{T} \varepsilon_2\right)$

 set payoff = max (min($S_1(T), S_2(T)$) - K , 0)

 set sum \leftarrow sum + payoff

end for

set $\hat{V} = e^{-rT} \frac{\text{sum}}{N}$

Output: \hat{V} (Crude MC price estimate)

3 Antithetic Variates (AV)

Now with the Antithetic Variates reduce variance of the crude Monte Carlo by pairing each normal draw ε with its negation $-\varepsilon$ in the equation (3).

$$S_{i,j}^+(T) \text{ uses } \varepsilon_{i,j}, \quad S_{i,j}^-(T) \text{ uses } -\varepsilon_{i,j}.$$

The key idea behind this is that by applying this transformation there is a high negative correlated outcome which reduces the standard error through the reduction of the variance without having to increase the number of simulations.

Furthermore that technique allows to achieve such a variance reduction by computing the two payoffs per pair and then average across pairs and discount them back to their present value. This often helps when the payoff is (approximately) monotone in the driving randomness, by balancing high and low outcomes in a symmetric structure.

A key aspect here is to observe how exactly the noise is being cancelled with the Antithetic Variates. In our setting we price the discounted payoff

$$X = e^{-rT} \max(\min(S_1(T), S_2(T)) - K, 0),$$

where under the risk-neutral measure

$$S_i(T) = S_i(0) \exp\left((r - \frac{1}{2}\sigma_i^2)T + \sigma_i \sqrt{T} \varepsilon_i\right), \quad \varepsilon_i \sim \mathcal{N}(0, 1), \quad i = 1, 2.$$

Antithetic pairing. Antithetic Variates (AV) use paired normal draws

$$(\varepsilon_1, \varepsilon_2) \quad \text{and} \quad (-\varepsilon_1, -\varepsilon_2).$$

Later the definition of the corresponding discounted payoffs becomes,

$$X^+ = e^{-rT} \max(\min(S_1^+(T), S_2^+(T)) - K, 0), \quad X^- = e^{-rT} \max(\min(S_1^-(T), S_2^-(T)) - K, 0),$$

with

$$S_i^+(T) = S_i(0) \exp\left((r - \frac{1}{2}\sigma_i^2)T + \sigma_i \sqrt{T} \varepsilon_i\right), \quad S_i^-(T) = S_i(0) \exp\left((r - \frac{1}{2}\sigma_i^2)T - \sigma_i \sqrt{T} \varepsilon_i\right).$$

The AV pair-estimator is

$$A = \frac{X^+ + X^-}{2}.$$

Unbiasedness. Since $(\varepsilon_1, \varepsilon_2) \stackrel{d}{=} (-\varepsilon_1, -\varepsilon_2)$, they have $\mathbb{E}[X^+] = \mathbb{E}[X^-]$, hence

$$\mathbb{E}[A] = \frac{\mathbb{E}[X^+] + \mathbb{E}[X^-]}{2} = \mathbb{E}[X^+],$$

so AV does not change the target price of the payoff and it only changes the variance.

Variance reduction and “noise cancellation”. The variance of the pair-average is

$$\text{Var}(A) = \text{Var}\left(\frac{X^+ + X^-}{2}\right) = \frac{1}{4}(\text{Var}(X^+) + \text{Var}(X^-) + 2\text{Cov}(X^+, X^-)).$$

By symmetry $\text{Var}(X^+) = \text{Var}(X^-) = \sigma_X^2$, therefore

$$\text{Var}(A) = \frac{\sigma_X^2}{2}(1 + \rho), \quad \rho = \text{Corr}(X^+, X^-).$$

Thus Antithetic Variates improves the efficiency over crude Monte Carlo whenever $\rho < 0$, i.e. whenever the paired outcomes are negatively correlated. In the extreme case $\rho = -1$ the variance would vanish (perfect cancellation).

Why ρ is typically negative in our payoff. For each asset, $S_i(T)$ is strictly increasing in ε_i because of the exponential mapping. Moreover, the payoff $\max(\min(S_1(T), S_2(T)) - K, 0)$ is increasing in each argument $S_1(T)$ and $S_2(T)$. Hence, when $(\varepsilon_1, \varepsilon_2)$ are large, both terminal prices tend to be large and X^+ tends to be large; the antithetic pair $(-\varepsilon_1, -\varepsilon_2)$ tends to produce small terminal prices and therefore a small payoff X^- . This creates negative dependence between X^+ and X^- and leads to partial cancellation of random fluctuations when averaging $(X^+ + X^-)/2$.

A local cancellation view (odd terms vanish). If we view the discounted payoff as a function of the Gaussian vector, $X = g(\varepsilon_1, \varepsilon_2)$, then AV replaces it by

$$\frac{g(\varepsilon_1, \varepsilon_2) + g(-\varepsilon_1, -\varepsilon_2)}{2}.$$

In a Taylor expansion around the origin, all terms that are odd in $(\varepsilon_1, \varepsilon_2)$ cancel out in this symmetrized average, so the dominant linear “noise” component is removed, explaining the observed reduction in variance.

So the algorithm is :

Antithetic Variates (AV) Pricing Algorithm

Input: $S_1, S_2, K, r, \sigma_1, \sigma_2, T, N$ (use even N)

set $\text{sum} = 0$

for $j = 1, \dots, N/2$ **do**

- Generate** $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1)$
- set** $S_1^+(T) = S_1 \exp\left((r - \frac{1}{2}\sigma_1^2)T + \sigma_1\sqrt{T}\varepsilon_1\right)$
- set** $S_2^+(T) = S_2 \exp\left((r - \frac{1}{2}\sigma_2^2)T + \sigma_2\sqrt{T}\varepsilon_2\right)$
- set** $S_1^-(T) = S_1 \exp\left((r - \frac{1}{2}\sigma_1^2)T - \sigma_1\sqrt{T}(-\varepsilon_1)\right)$
- set** $S_2^-(T) = S_2 \exp\left((r - \frac{1}{2}\sigma_2^2)T - \sigma_2\sqrt{T}(-\varepsilon_2)\right)$
- set** $\text{payoff}^+ = \max(\min(S_1^+(T), S_2^+(T)) - K, 0)$
- set** $\text{payoff}^- = \max(\min(S_1^-(T), S_2^-(T)) - K, 0)$
- set** $\text{sum} \leftarrow \text{sum} + \frac{\text{payoff}^+ + \text{payoff}^-}{2}$

end for

set $\hat{V}_{\text{AV}} = e^{-rT} \frac{\text{sum}}{N/2}$

Output: \hat{V}_{AV} (AV price estimate)

Therefore the essential part is to recognize that the effectiveness of the antithetic variates depends significantly on the nature of the payoff’s function. Highly rising monotonic payoffs, such as in our case, benefit from this approach due to the strongly negative correlated paired outcomes.

4 Control Variates (CV)

Control variates is method in which the reduction of the variance occurs by exploiting a correlated random variable, with known expectation, to the original which is being estimated. Thus the newly incorporated variables have added extra information capable of reducing the variance of the desired estimator.

Hence, when using controls Y_1, Y_2 with known $\nu_1 = \mathbb{E}[Y_1]$ and $\nu_2 = \mathbb{E}[Y_2]$, define:

$$X_c = X + c_1(Y_1 - \nu_1) + c_2(Y_2 - \nu_2),$$

where X is the discounted payoff. In this linear equation the new added variables Y_1, Y_2 shall be directly correlated with the desired estimator X

Now the best way to search for that variance minimization, is by getting the First and Second Order Conditions with respect to the coefficients of the previous equation:

$$\text{Var}(X_c) = \text{Var}(X + c_1(Y_1 - \nu_1) + c_2(Y_2 - \nu_2))$$

$$\text{First Order Condition(FOC)} : \frac{d\text{Var}(X_c)}{dc_i}|_{c_i*} = 0$$

$$\text{Second Order Condition(SOC)} : \frac{d^2\text{Var}(X_c)}{dc_i^2}|_{c_i*} > 0$$

This way there is minimization of the Variance for the given coefficients c_i^* . Below there is the analytical algebraic operations in which the closed form of those optimal coefficients is obtained:

Analytical Methodology of the Variance Minimization

$$\text{Var}(X_c) = \text{Var}(X) + c_1^2\text{Var}(Y_1) + c_2^2\text{Var}(Y_2) + 2c_1\text{Cov}(X, Y_1) + 2c_2\text{Cov}(X, Y_2) + 2c_1c_2\text{Cov}(Y_1, Y_2)$$

but in this model the $\text{Cov}(Y_1, Y_2) = 0$ so :

$$\text{Var}(X_c) = \text{Var}(X) + c_1^2\text{Var}(Y_1) + c_2^2\text{Var}(Y_2) + 2c_1\text{Cov}(X, Y_1) + 2c_2\text{Cov}(X, Y_2)$$

II) FOC:

$$\frac{d\text{Var}(X_c)}{dc_i}|_{c_i*} = 0 \xrightarrow{i=1} 2c_1\text{Var}(Y_1) + 2\text{Cov}(X, Y_1) = 0$$

Same for the $i = 2$:

$$\frac{d\text{Var}(X_c)}{dc_i}|_{c_i*} = 0 \xrightarrow{i=2} 2c_2\text{Var}(Y_2) + 2\text{Cov}(X, Y_2) = 0$$

We have a two unknown coefficients with two given equations. We express this linear system in form of matrices:

$$\begin{pmatrix} \text{Var}(Y_1) & 0 \\ 0 & \text{Var}(Y_2) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = -\begin{pmatrix} \text{Cov}(X, Y_1) \\ \text{Cov}(X, Y_2) \end{pmatrix}$$

which gives in return :

$$\text{Var}(Y_1) \cdot c_1 = -\text{Cov}(X, Y_1)$$

and

$$\text{Var}(Y_2) \cdot c_2 = -\text{Cov}(X, Y_2)$$

Thus Optimal coefficients satisfy:

$$c_i^* = -\frac{\text{Cov}(X, Y_i)}{\text{Var}(Y_i)}.$$

Therefore, by returing to the analytical formula of (4) it is straightforward that :

$$\text{Var}(X_c) = \text{Var}(X) - \frac{\text{Cov}^2(X, Y_1)}{\text{Var}(Y_1)} - \frac{\text{Cov}^2(X, Y_2)}{\text{Var}(Y_2)}$$

So by observing that :

$$\rho_{X, Y_i} = \frac{\text{Cov}(X, Y_i)}{\sqrt{\text{Var}(Y_1)} \sqrt{\text{Var}(Y_2)}}$$

Then the controlled estimators variance is in the form of :

$$\text{Var}(X_c) = \text{Var}(X) \cdot [1 - \rho_{X, Y_1}^2 - \rho_{X, Y_2}^2]$$

II) SOC: Last but not least the second order condition must be calculated in order to ensure the rightness of our coefficients which are trully minimizing the variance:

4.0.1 Second Order Condition (SOC) for the CV Coefficient

From the expansion (with one control Y_i):

$$\text{Var}(X_c) = \text{Var}(X) + c_i^2 \text{Var}(Y_i) + 2c_i \text{Cov}(X, Y_i).$$

Differentiate w.r.t. c_i :

$$\frac{d}{dc_i} \text{Var}(X_c) = 2c_i \text{Var}(Y_i) + 2 \text{Cov}(X, Y_i).$$

And by the FOC the coefficients are:

$$c_i^* = -\frac{\text{Cov}(X, Y_i)}{\text{Var}(Y_i)}.$$

So now the second derivative is:

$$\frac{d^2}{dc_i^2} \text{Var}(X_c) = 2\text{Var}(Y_i) > 0$$

The above holds whenever $\text{Var}(Y_i) > 0$, hence the c_i^* indeed minimize the $\text{Var}(X_c)$.

In the model the best variance reduction is achieved by finding either positive or negative and highly correlated related variables to the desired estimated one. Also now the optimal variance-minimizing control estimator obtained through the monte carlo simulations is of the form :

$$\hat{V}_{CV-optimal} = \frac{1}{N} \sum_{j=0}^N [X^{(j)} - \frac{\text{Cov}(X, Y_1)}{\text{Var}(Y_1)}(Y_1^{(j)} - \nu_1) - \frac{\text{Cov}(X, Y_2)}{\text{Var}(Y_2)}(Y_2^{(j)} - \nu_1)]$$

In practice, covariances/variances are estimated via a pilot simulations. Here, a natural choice of control variables is $Y_1 = S_1(T)$ and $Y_2 = S_2(T)$, since their risk-neutral expectations are known analytically.

But an importance notice is, to actually examine the use of pilot simulations. Their main benefit is that the computation of the optimal c_i^* is produced through the prior estimating of the $\hat{\text{Cov}}(X, Y_i)$ & $\hat{\text{Var}}(X, Y_i)$. Then the \hat{c}_i^* is computed and later on assigned as $c_i^* \approx \hat{c}_i^*$ where it is used in the main pricing for loop.

Reason of this whole implementation is that by reusing the same data to both fit \hat{c}_i^* and evaluate the final mean can introduce small finite-sample quirks (often tiny, but separations keep things clean and textbook-correct). So when addressing models where aspects and parameters are usually unknown and need to be estimated prior to their main functionality, the implementation where pilot simulations precede main code are for computed for exactly that reason.

The analytical algorithm is provided in the code below:

Control Variates (CV) Pricing Algorithm

Input: $S_1, S_2, K, r, \sigma_1, \sigma_2, T, N$; pilot size N_p

Controls: $Y_1 = S_1(T)$, $Y_2 = S_2(T)$

Known means: $\nu_1 = \mathbb{E}[S_1(T)] = S_1 e^{rT}$, $\nu_2 = \mathbb{E}[S_2(T)] = S_2 e^{rT}$

(1) Pilot run to estimate coefficients

set $\bar{X} = 0$, $\bar{Y}_1 = 0$, $\bar{Y}_2 = 0$

set $S_{XY_1} = 0$, $S_{XY_2} = 0$, $S_{Y_1 Y_1} = 0$, $S_{Y_2 Y_2} = 0$

for $j = 1, \dots, N_p$ do

 Generate $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1)$

 set $S_1(T), S_2(T)$ as in Crude MC

 set $X = e^{-rT} \max(\min(S_1(T), S_2(T)) - K, 0)$

 set $Y_1 = S_1(T)$, $Y_2 = S_2(T)$

 update running sums to estimate $\text{Cov}(X, Y_i)$, $\text{Var}(Y_i)$

 note: if known $\text{Var}(Y_i)$ do not estimate it through pilot simulations

end for

set $c_1 = -\frac{\text{Cov}(X, Y_1)}{\text{Var}(Y_1)}$, $c_2 = -\frac{\text{Cov}(X, Y_2)}{\text{Var}(Y_2)}$

(2) Main run using the CV estimator

set sum = 0

for $j = 1, \dots, N$ do

 Generate $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 1)$

 set $S_1(T), S_2(T)$ as in Crude MC

 set $X = e^{-rT} \max(\min(S_1(T), S_2(T)) - K, 0)$

 set $Y_1 = S_1(T)$, $Y_2 = S_2(T)$

 set $X_c = X + c_1(Y_1 - \nu_1) + c_2(Y_2 - \nu_2)$

 set sum \leftarrow sum + X_c

end for

set $\hat{V}_{\text{CV}} = \frac{\text{sum}}{N}$

Output: \hat{V}_{CV} (CV price estimate)

5 Numerical Results

By accumulating all the results from the coding implementation there is the report that, for input parameters of $N = 4 \times 10^6$ simulations per random number, with additional 300 pilot sims again per random number included for CV technique, under a symmetric setup $S_1 = S_2$ and $\sigma_1 = \sigma_2$, and $\rho = 0$ the numerical results are represented below.

5.1 Performance Summary (Numerical Results)

Method	\hat{V} (Price)	Rel. Error (95%)	Runtime (s)
Crude MC	3.297 000	0.002 081	3.150 000
Antithetic Variates (AV)	3.297 000	0.001 299	2.400 000
Control Variates (CV)	3.295 000	0.001 530	1.550 000

5.2 Interpretation

Method	Notes / Interpretation
Crude MC	Baseline Monte Carlo; largest uncertainty for the same N .
Antithetic Variates (AV)	Best variance cancellation here due to payoff monotonicity (paired negatives).
Control Variates (CV)	Strong variance reduction and fastest runtime; min-payoff limits correlation and pilot-estimation adds some noise.
Consistency check: all estimates are within $\mathcal{O}(10^{-3})$ of each other \Rightarrow no evidence of bias; differences consistent with MC sampling error.	

Initial parameters: $N = 4 \times 10^6$, $S_1 = S_2 = 100$, $K = 100$, $r = 0.05$, $\sigma_1 = \sigma_2 = 0.2$, $T = 1$, $\rho = 0$.

6 Conclusion

Crude Monte Carlo provides a consistent estimate but with larger confidence intervals for a fixed N . Antithetic Variates deliver the best precision in this experiment, consistent with the payoff's monotonic structure under exponentiation. Control Variates achieve the fastest runtime and strong variance reduction overall, though performance depends on correlation strength between payoff and controls and on the stability of pilot estimates.

References

- [1] *Computational Finance C++ - Monte Carlo.pdf* -Courses Slides, Englezos Nicholaos, M.Sc. in FinTech 2025-2026.
- [2] *Computational Finance C++ -Black and Scholes.pdf* -Courses Slides, Englezos Nicholaos, M.Sc. in FinTech 2025-2026.
- [3] *Options on the minimum or the maximum of two risky assets*, Rene M. Stulz, Journal of Financial Economics 10(1982) 161-185. North-Holland Publishing Company

7 Appendix: Main Codes for the Monte Carlo implementations in C++

Listing 1: Uniform RNG + Inverse Normal CDF + Standard Normal RNG (as in slides)

```

1 // First lets set the uniform distribution number generator:
2 double UnifRand(){
3     return rand() / double(RAND_MAX);
4     // Scaling has been performed by dividing with RAND_MAX
5 }
6
7 /* We take now the Inverse CDF of the Normal distribution N(mu,sigma^2)
8 that we need in order to generate the Normal Random Numbers that we
9 need
10 for the MC simulation */
11 double inverse_of_normal_cdf(const double p, const double mu, const
12     double sigma)
13 {
14     if (p <= 0.0 || p >= 1.0)
15     {
16         std::stringstream os;
17         os << "Invalid input argument (" << p
18             << "); must be larger than 0 but less than 1.";
```

```

17     throw std::invalid_argument(os.str());
18 }
19
20 double r, val;
21 const double q = p - 0.5;
22
23 if (std::abs(q) <= .425) {
24     r = .180625 - q * q;
25     val =
26         q * (((((r * 2509.0809287301226727 +
27             33430.575583588128105) * r + 67265.770927008700853) * r
28             +
29             45921.953931549871457) * r + 13731.693765509461125) * r
30             +
31             1971.5909503065514427) * r + 133.14166789178437745) * r
32             +
33             3.387132872796366608)
34     / (((((r * 5226.495278852854561 +
35         28729.085735721942674) * r + 39307.89580009271061) * r
36         +
37         21213.794301586595867) * r + 5394.1960214247511077) * r
38         +
39         687.1870074920579083) * r + 42.313330701600911252) * r
40         +
41         1);
42 }
43 else {
44     if (q > 0) {
45         r = 1 - p;
46     }
47     else {
48         r = p;
49     }
50
51     r = std::sqrt(-std::log(r));
52
53     if (r <= 5)
54     {
55         r += -1.6;
56         val = (((((r * 7.7454501427834140764e-4 +
57             .0227238449892691845833) * r + .24178072517745061177) * r
58             +
59             1.27045825245236838258) * r +
60             3.64784832476320460504) * r + 5.7694972214606914055) *
61             r + 4.6303378461565452959) * r +
62             1.42343711074968357734)
63         / (((((r *
64             1.05075007164441684324e-9 + 5.475938084995344946e
65             -4) *
66             r + .0151986665636164571966) * r +
67             .14810397642748007459) * r + .68976733498510000455) *
68             *
69             r + 1.6763848301838038494) * r +
70             2.05319162663775882187) * r + 1);
71 }
72 else /* very close to 0 or 1 */
73 {
74     r += -5;
75     val = (((((r * 2.01033439929228813265e-7 +
76         2.71155556874348757815e-5) * r +

```

```

66     .0012426609473880784386) * r + .026532189526576123093)
67     *
68     r + .29656057182850489123) * r +
69     1.7848265399172913358) * r + 5.4637849111641143699) *
70     r + 6.6579046435011037772)
71 / (((((r *
72     2.04426310338993978564e-15 + 1.4215117583164458887e
73     -7) *
74     r + 1.8463183175100546818e-5) * r +
75     7.868691311456132591e-4) * r +
76     .0148753612908506148525)
77     * r + .13692988092273580531) * r +
78     .59983220655588793769) * r + 1);
79 }
80
81     if (q < 0.0) {
82         val = -val;
83     }
84 }
85
86     return mu + sigma * val;
87 }
88
89 // Now we generate X_i random var. from the N(0,1)
90 double Stand_Normal_Rand(){
91     double u = UnifRand(); // Step 1: Gen u_i from Unif(0,1)
92
93     while(u <= 0.0 || u >= 1.0){
94         u = UnifRand();
95     }
96     return inverse_of_normal_cdf(u, 0.0, 1.0);
97 }
```

7.1 Crude Monte Carlo (Call on min)

Listing 2: Plain (Crude) Monte Carlo: Call on the minimum of two assets

```

1  double MC_Rainbow_Call_on_min(double S1, double S2, double K,
2                               double r, double v1, double v2,
3                               double T, int num_sims) {
4
5     // Two assets with zero correlation (rho = 0)
6     // Si(T) = Si(0) * exp( (r - 0.5*sigma_i^2)T + sigma_i*sqrt(T)*
7     // epsilon_i )
8
9     double nu_T1 = (r - 0.5 * v1 * v1) * T;
10    double nu_T2 = (r - 0.5 * v2 * v2) * T;
11
12    double v_T1 = v1 * sqrt(T);
13    double v_T2 = v2 * sqrt(T);
14
15    double disc_payoff_sum = 0.0;
16    double disc_payoff_squared_sum = 0.0;
17
18    for (int i = 0; i < num_sims; i++) {
19
20        // Independent standard normal shocks (rho = 0)
21
22        double Z1 = Stand_Normal_Rand();
23        double Z2 = Stand_Normal_Rand();
24
25        double S1_t = S1 * exp((r - 0.5 * v1 * v1) * T + v1 * Z1);
26        double S2_t = S2 * exp((r - 0.5 * v2 * v2) * T + v2 * Z2);
27
28        double min_S = min(S1_t, S2_t);
29
30        disc_payoff_sum += disc_min(S, min_S);
31        disc_payoff_squared_sum += disc_min(S, min_S) * disc_min(S, min_S);
32
33    }
34
35    double call_value = disc_min(S, disc_payoff_sum / num_sims);
36    double call_std = sqrt(disc_min(S, disc_payoff_squared_sum / num_sims));
37
38    return call_value;
39 }
```

```

20     double epsilon1 = Standard_Normal_Rand();
21     double epsilon2 = Standard_Normal_Rand();
22
23     // Terminal prices under Black-Scholes
24     double S_T1 = S1 * exp(nu_T1 + v_T1 * epsilon1);
25     double S_T2 = S2 * exp(nu_T2 + v_T2 * epsilon2);
26
27     // Payoff: call on min
28     double minn = min(S_T1, S_T2);
29     double disc_payoff = max(minn - K, 0.0) * exp(-r * T);
30
31     disc_payoff_sum += disc_payoff;
32     disc_payoff_squared_sum += disc_payoff * disc_payoff;
33 }
34
35 // Monte Carlo price estimator (already discounted)
36 double disc_payoff_average = disc_payoff_sum / num_sims;
37
38 // Unbiased sample variance of discounted payoff
39 sample_var = (disc_payoff_squared_sum
40                 - num_sims * disc_payoff_average *
41                 disc_payoff_average)
42                 / (num_sims - 1);
43
44 return disc_payoff_average;
}

```

7.2 Antithetic Variates (AV)

Listing 3: Monte Carlo with Antithetic Variates (AV)

```

1 double MC_Call_Price_Rainbow_AV(double S1, double S2, double K,
2                                 double r, double v1, double v2,
3                                 double T, int num_sims) {
4
5     double nu_T1 = (r - 0.5 * v1 * v1) * T;
6     double nu_T2 = (r - 0.5 * v2 * v2) * T;
7
8     double v_T1 = v1 * sqrt(T);
9     double v_T2 = v2 * sqrt(T);
10
11    double anti_payoff_sum = 0.0;
12    double anti_payoff_squared_sum = 0.0;
13
14    for (int i = 0; i < num_sims; i++) {
15
16        double epsilon1 = Standard_Normal_Rand();
17        double epsilon2 = Standard_Normal_Rand();
18
19        // "Plus" path
20        double S_T1 = S1 * exp(nu_T1 + v_T1 * epsilon1);
21        double S_T2 = S2 * exp(nu_T2 + v_T2 * epsilon2);
22        double payoff_plus = max(min(S_T1, S_T2) - K, 0.0);
23
24        // Antithetic "minus" path
25        double S_T1_a = S1 * exp(nu_T1 + v_T1 * (-epsilon1));
26        double S_T2_a = S2 * exp(nu_T2 + v_T2 * (-epsilon2));

```

```

27     double payoff_minus = max(min(S_T1_a, S_T2_a) - K, 0.0);
28
29     // Pair-average payoff
30     double anti_mean = 0.5 * (payoff_plus + payoff_minus);
31
32     anti_payoff_sum += anti_mean;
33     anti_payoff_squared_sum += anti_mean * anti_mean;
34 }
35
36     double anti_payoff_average = anti_payoff_sum / num_sims;
37
38     // Discount at the end (matches slide structure)
39     double price_anti = anti_payoff_average * exp(-r * T);
40
41     // Variance of discounted estimator (optional, as in slide)
42     double var_payoff = (anti_payoff_squared_sum
43                           - num_sims * anti_payoff_average *
44                               anti_payoff_average)
45                           / (num_sims - 1);
46     sample_var = var_payoff * exp(-2.0 * r * T);
47
48     return price_anti;
}

```

7.3 Control Variates (CV) with Pilot Estimation

Listing 4: Monte Carlo with Control Variates (CV) using pilot run

```

1 double MC_Call_Price_Rainbow_CV(double S1, double S2, double K,
2                                 double r, double v1, double v2,
3                                 double T, int num_sims, int num_pilot) {
4
5     // Correlation assumed rho = 0
6     double nu_1 = (r - 0.5 * v1 * v1) * T;
7     double nu_2 = (r - 0.5 * v2 * v2) * T;
8
9     double v_1 = v1 * sqrt(T);
10    double v_2 = v2 * sqrt(T);
11
12    // Pilot sums for sample covariance estimation
13    double S1_T_sum = 0.0;
14    double S2_T_sum = 0.0;
15    double disc_payoff_sum = 0.0;
16
17    double product1_sum = 0.0; // sum of (S1_T * disc_payoff)
18    double product2_sum = 0.0; // sum of (S2_T * disc_payoff)
19
20    for (int i = 0; i < num_pilot; i++) {
21
22        double epsilon1 = Standard_Normal_Rand();
23        double epsilon2 = Standard_Normal_Rand();
24
25        double S1_T = S1 * exp(nu_1 + v_1 * epsilon1);
26        double S2_T = S2 * exp(nu_2 + v_2 * epsilon2);
27
28        double minimum = min(S1_T, S2_T);
29        double disc_payoff = max(minimum - K, 0.0) * exp(-r * T);
}

```

```

30
31     // Accumulate pilot moments
32     S1_T_sum += S1_T;
33     S2_T_sum += S2_T;
34     disc_payoff_sum += disc_payoff;
35
36     product1_sum += S1_T * disc_payoff;
37     product2_sum += S2_T * disc_payoff;
38 }
39
40 // Sample covariance estimators from pilot
41 double sample1_cov = (product1_sum - (S1_T_sum * disc_payoff_sum) /
42                         num_pilot)
43                         / (num_pilot - 1);
44 double sample2_cov = (product2_sum - (S2_T_sum * disc_payoff_sum) /
45                         num_pilot)
46                         / (num_pilot - 1);
47
48 // Control variates: Y1 = S1(T), Y2 = S2(T)
49 // E[S(T)] = S0 e^{rT}
50 double ExpY_1 = S1 * exp(r * T);
51 double ExpY_2 = S2 * exp(r * T);
52
53 // Var(S(T)) = S0^2 e^{2rT} (e^{\sigma^2 T} - 1)
54 double VarY1 = S1 * S1 * exp(2.0 * r * T) * (exp(v1 * v1 * T) -
55                               1.0);
56 double VarY2 = S2 * S2 * exp(2.0 * r * T) * (exp(v2 * v2 * T) -
57                               1.0);
58
59 // Optimal coefficients (slides: c_i^* = -Cov(X, Y_i)/Var(Y_i))
60 double c1 = -sample1_cov / VarY1;
61 double c2 = -sample2_cov / VarY2;
62
63 // Main run with control variates
64 double control_var_sum = 0.0;
65 double control_var_squared_sum = 0.0;
66
67 for (int i = 0; i < num_sims; i++) {
68
69     double epsilon1 = Standard_Normal_Rand();
70     double epsilon2 = Standard_Normal_Rand();
71
72     double new_S1_T = S1 * exp(nu_1 + v_1 * epsilon1);
73     double new_S2_T = S2 * exp(nu_2 + v_2 * epsilon2);
74
75     double new_minimum = min(new_S1_T, new_S2_T);
76     double new_disc_payoff = exp(-r * T) * max(new_minimum - K,
77                                               0.0);
78
79     // Controlled estimator
80     double control_var = new_disc_payoff
81                         + c1 * (new_S1_T - ExpY_1)
82                         + c2 * (new_S2_T - ExpY_2);
83
84     control_var_sum += control_var;
85     control_var_squared_sum += control_var * control_var;
86 }
87
88

```

```
83     double control_var_average = control_var_sum / num_sims;
84
85     sample_var = (control_var_squared_sum
86                   - num_sims * control_var_average *
87                     control_var_average)
88                   / (num_sims - 1);
89
89     return control_var_average;
90 }
```