NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF MATHEMATICS

ANA BEATRIZ FERREIRA BEJA

BSc in Mathematics

MULTI-ASSET OPTIONS AND THEIR ANALYSIS

MASTER IN MATHEMATICS AND APPLICATIONS

NOVA University Lisbon

September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
MATHEMATICS

MULTI-ASSET OPTIONS AND THEIR ANALYSIS

ANA BEATRIZ FERREIRA BEJA

BSc in Mathematics

Adviser: Pedro José dos Santos Palhinhas Mota
Assistant Professor, NOVA University Lisbon

Co-adviser: Raphael Poix
Market Valuation Control Officer, Bank BNP Paribas

Examination Committee

Chair: Marta Cristina Vieira Faias Mateus
Associate Professor, NOVA University Lisbon

Rapporteur: Manuel Leote Tavares Inglês Esquível
Associate Professor with Aggregation, NOVA University Lisbon

Member: Pedro José dos Santos Palhinhas Mota
Associate Professor, NOVA University Lisbon

MASTER IN MATHEMATICS AND APPLICATIONS

NOVA University Lisbon
September, 2024

Multi-asset options and their analysis

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.18) [17].

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Pedro Mota, for his invaluable guidance and support throughout the past two years. His insightful feedback and constant encouragement were crucial in shaping the direction of this thesis.

I would also like to extend my heartfelt thanks to Professor Marta Faias for her support in helping me secure an internship at BNP Paribas. This experience played a significant role in the development of my thesis.

I would like to express my sincere appreciation to Antoine Finot at BNP Paribas for providing me with the opportunity to develop my thesis in collaboration with the bank.

I would also like to extend my deepest thanks to Raphael Poix at BNP Paribas for his invaluable assistance throughout the development of my thesis. His extensive knowledge and thoughtful feedback helped me refine and improve my work significantly.

I want to thank my parents for their unwavering support and the strength they have given me when I wanted to give up. Their sacrifices, belief in my choices, and encouragement throughout my academic journey and life have been invaluable. I could not ask for better parents.

A heartfelt thank you to Carolina Pé-Curto and Diana Leonardo for being such incredible friends over these past five years. Your support, your willingness to listen, and the countless laughs we have shared have made all the difference. I am truly grateful for every bit of advice and every moment of friendship you have given me.

I want to express my heartfelt thanks to Beatriz Delgado, a wonderful friend I made during my internship at BNP Paribas. Your support, willingness to listen, and the thoughtful advice you offered when I needed it most have been incredibly meaningful to me.

I would also like to thank all the teachers who have accompanied me on my academic journey.

I would like to thank my colleagues for their support and collaboration throughout my academic journey.

I would like to thank the colleagues I worked with at BNP Paribas over the past 11 months for their valuable advice, shared knowledge, and the many laughs along the way.

Abstract

This thesis is divided into two main parts: the first part consists of the theoretical part of the multi-asset options dealt with in this thesis, i.e. basket options, outperformance options and best and worst of options; the second part presents and analyzes the results obtained for option prices and Greeks, with all outcomes derived from fictitious data.

In general, the main Greeks are studied for all multi-asset options, but the most important for this thesis is Cega, the Greek related to the correlation coefficient, since this parameter only exists when studying options on more than one asset.

To study the effects of correlation and other parameters that influence the price of an option, Python programming was used. The Python programming language can be used to program the pricing formulas for options that have them and their respective derivatives. For options that do not have explicit formulas to calculate their price, Monte Carlo simulation and other numerical methods were utilized to calculate their price and derivatives.

To analyse the results obtained for the price and the Greeks of the options, graphs were made in Python. The graphs made the analysis easier and it was possible to conclude that the different multi-asset options behave differently with respect to the correlation parameter. Additionally, it was observed that other methods for approximating the price and Greeks of options that do not have an explicit formula are more viable than Monte Carlo simulation, due to the latter's longer computational time.

Keywords: Multi-asset options, Greeks, Correlation, Cega, Monte Carlo simulation

Resumo

Esta tese está dividida em duas grandes partes: a primeira consiste na parte teórica das multi-asset options abordadas nesta tese, ou seja, das basket options, das outperformance options e das best e worst of options; a segunda parte resume-se à exposição e análise dos resultados obtidos sobre os preços e os Greeks das opções, onde todos os resultados obtidos provêm de dados fictícios.

Em geral, para todas as multi-asset options são estudados os Greeks mais importantes, mas ao que se vai dar mais importância é ao Cega, o Greek que está relacionado com o coeficiente de correlação, uma vez que este parâmetro só existe no estudo de opções sobre mais de um ativo.

Para se estudar os efeitos da correlação e dos outros parâmetros que influenciam o preço de uma opção, recorreu-se à programação em Python. Com a linguagem de programação Python pode-se programar as fórmulas de preço para as opções que as têm e as suas respetivas derivadas. Para as opções que não têm fórmulas explícitas para calcular o seu preço recorreu-se à simulação de Monte Carlo e outros métodos numéricos tanto para calcular o seu preço como as suas derivadas.

Para se analisar os resultados obtidos para o preço e os Greeks das opções fez-se gráficos em Python. Com os gráficos, a análise foi mais fácil e pode-se concluir que as diferentes multi-asset options têm comportamentos diferentes em relação ao parâmetro da correlação. Além disso, verificou-se que outros métodos para aproximar o preço e os Greeks das opções que não têm fórmula explícita são mais viáveis que a simulação de Monte Carlo, devido ao demorado tempo que esta última demora a ser executada.

Palavras-chave: Multi-asset options, Greeks, Correlação, Cega, Simulação de Monte Carlo

Contents

List of Figures

List of Tables

List of Listings

1

Introduction

The study of simple options is a fundamental tool in the world of financial derivatives. These instruments are linked to the price movements of a specific underlying asset, which could be a stock, a commodity or a currency. As noted in Wilmott [28], simple options provide investors with a direct and focused approach to gain exposure to, or protection against, price fluctuations in a particular market.

Simple options offer simplicity and specificity, so there is a need for more complex options, multi-asset options, which offer greater strategic scope.

Bouzoubaa and Osseiran [4] describe multi-asset options as financial derivatives that give investors the flexibility to trade and hedge against price movements in multiple underlying assets within a single contract. These options allow traders to manage risk and take advantage of opportunities in different markets simultaneously.

The aim of this thesis is to study and analyse multi-asset options, and more specifically to observe the effects of correlation on them. There is a huge variety of multi-asset options that can be explored, but since there are not many in-depth studies (for instance, see Stulz [25]) on sensitivity analysis, and more specifically on the effects of correlation on these options, the focus of this thesis is these three different types of multi-asset options: basket options, best and worst of options and outperformance options. With this new advance in the analysis of these options, it will be possible to gain a better understanding of their risks and the behaviour to adopt when deciding to invest in them.

Firstly, the specific nature of each of the three multi-asset options is explained and then the concept of Greeks is discussed, more specifically the Greek of the correlation variable, as Greeks are related to sensitivity analysis.

Basket options, the most common type of multi-asset option, are, according to Musiela and Rutkowski [22], a type of financial derivative that derives its value from the combined performance of a portfolio or a basket of underlying assets. The value of a basket option is determined by the weighted average of the prices of the individual assets. These options provide a strategic advantage by allowing investors to diversify their portfolios and manage risk across different markets.

Best and worst of options, also known as options on the maximum or minimum, are

described by Johnson [11] as options whose value is derived from the highest (maximum) or lowest (minimum) price among a set of underlying assets within a specified time frame. This financial tool provides a strategic advantage by allowing investors to take advantage of extreme price movements within a portfolio, offering both risk management and profit potential. This means that these options are ultimately more valuable in volatile markets and are also customisable, permitting adaptation of them to market conditions.

The final type of multi-asset option to be discussed is dispersion options, more specifically outperformance options. Weert [27] defines outperformance options, also known as relative performance options, as financial derivatives that enable investors to profit from the relative performance of one asset compared to another. These instruments allow traders to speculate on the outperformance or underperformance of a selected asset relative to a benchmark within a specified time frame. Thus, the value of an outperformance option is derived from the difference in returns between the selected asset and the reference benchmark. Outperformance options can be a valuable tool for those wishing to hedge against specific risks or take advantage of relative strength within a portfolio.

In summary, multi-asset options have grown in popularity because of their versatility and ability to adapt to dynamic market conditions. Investors can explore different combinations of assets and adjust their exposure to market trends and economic factors. While these options offer greater flexibility, they also require a thorough understanding of the interrelationships between different asset classes, making them a sophisticated tool suitable for experienced market participants.

When studying multi-asset options, it is inevitable to talk about sensitivity analysis, also known as Greeks. Björk [2] describes Greeks as a measure of the sensitivity of the price of the derivative product to changes in the various variables that influence the price of the product. One of the differences between simple options and multi-asset options is the existence of a correlation between the various underlying assets, which makes it possible to carry out a sensitivity analysis on this parameter. In financial markets, correlation specifically measures the relationship between the price movements of different assets, so understanding correlation is essential for investors to build a well-diversified portfolio. A portfolio that includes assets with low or negative correlations can potentially reduce overall risk because these assets may not move in tandem. Investors use correlation analysis to assess the interdependencies between stocks, bonds and other instruments and to make strategic decisions to manage risk exposure effectively. However, it is important to note that correlation can change over time due to various factors such as economic conditions or market sentiment. Therefore, continuous monitoring and adjustment of portfolio allocations based on changing correlation is essential for successful risk management.

This thesis is therefore structured as follows: in the second chapter there is a literature review of the various books/articles that were used in this thesis; the third chapter focus on concepts that were necessary in the detailed study of each option, these were: pricing models, Monte Carlo simulation, Greeks and correlation; the fourth chapter focus on the detailed study of each of the multi-asset options and their respective sensitivity analysis to

the various parameters; in the fifth chapter the results obtained from the pricing of these options and their respective sensitivity analysis are analysed; the final chapter highlights some of the advances made in the study of multi-asset options throughout this thesis and outlines potential areas for further exploration in future research.

2

Literature Review

In today's complex and interconnected financial markets, the limitations of simple options, such as those priced using the Black & Scholes model [3], have become more apparent. As a result, multi-asset options have emerged as a superior alternative for many investors and institutions. By offering diversification, dynamic risk management, customisation, and enhanced flexibility, multi-asset options have become the preferred choice for many investors seeking to navigate today's complex investment landscape.

Therefore, in line with the proposed objectives, in this chapter the studies and respective approaches of various authors in the field of multi-asset options are described and presented.

2.1 Best and Worst of Options

Having looked at some of the studies that have been done on multi-asset options, let's now look at the literature of one type of multi-asset option, the best and worst of options.

The concept of this type of option was first introduced by Johnson [12] and Stulz [25]. In his study, Stulz presents the assumptions and steps to obtain the formula for the price of a worst of call on two assets.

One of the assumptions maintained for this type of option is that the price process is still given by a geometric Brownian motion [23], where the Brownian movements are correlated and the interest rate is constant over time. The article also introduces a two-dimensional version of the Black & Scholes equation with its specified constraints.

Using similar techniques to those for a simple option, it is therefore possible to obtain the formula for the price of a worst of call on two assets. With this price formula, Stulz [25] is able to deduce parity formulas for the other types of best and worst of options.

Since there is an explicit formula, it is possible to calculate the Greeks of each parameter of a worst of call and to perform a sensitivity analysis of these. The article therefore presents the calculation of some Greeks and their analysis for the different parameters.

However, Johnson [11] extends the results obtained by Johnson [12] and Stulz [25] to have a formula for the price of a worst and best of call for the case where there are N

assets.

The Cox and Ross [6] strategy is used to obtain a formula for the general case of a best of call, showing what each variable means. It also presents the formula for the price of a worst of call, which could be demonstrated applying the same strategy as for the best of call. The formula for the worst and best of calls is also introduced for the case in where there are only two assets.

Veiga, Wystup, and Esquível [26] present an explicit pricing formula applicable to various types of multi-asset options. This formula can be used to determine the prices of both best and worst of options. Notably, it also serves as a generalized version of the formulas introduced by Johnson [11] and Stulz [25].

2.2 Basket Options

Another type of multi-asset options are the basket options.

Krekel, Kock, Korn and Man [14] define the price of a basket option as the weighted average of the prices of all the assets on the maturity date.

The article also describes the payoff function for both call and put option, the T-forward price of the asset i and the discount factor.

As in the case of the best and worst of options, the prices of each asset are modelled by a geometric Brownian motion [23].

Since there is not an exact formula for the price of these options due to their complexity, Krekel, Kock, Korn and Man [14] resort to Monte Carlo (see section 3.4) or Quasi-Monte Carlo simulation and other methods, which are presented below.

The first technique presented in the article is Beisser's conditional expectation technique. Beisser [1] adapts a concept introduced for pricing Asian options. His idea is to condition the random variable Z, which is defined as the product of a Brownian motion and a quotient between an appropriately chosen volatility and the square root of the maturity date, and use Jensen's inequality [1] so that the price of a basket call is estimated by the weighted sum of (artificial) European call prices.

The second technique detailed in the article is Gentle's approximation by geometric average. Gentle [9] approximates the arithmetic average in the basket payoff by a geometric average. This approximation is utilised because a geometric mean of log-normal random variables still has a log-normal distribution, which allows the Black & Scholes method to be used to approximate the payoff price.

The next technique introduced in the article is Levy's log-normal moment matching. Levy [15] and Levy and Turnbull [16] describe this technique in the same way as will be seen below. Both articles establish a type of "running average" and subsequently define the

[1] Let X be a random variable with expected value $\mathbb{E}[X]$, and let g(.) be a continuous convex function. Then

$$\mathbb{E}\left[g(X)\right] \geq g\left(\mathbb{E}[X]\right) [2].$$

payoff of the basket option in line with that. In addition, the asset price processes in both articles follow a geometric Brownian motion [23] and the basket option price function is identical as well. The idea behind these articles is to assume that a variable that depends on the "running average" is a geometric average. This allows to obtain the value of the parameters of the geometric average, i.e. the mean and the standard deviation, from the first two moments of the variable, and thus to derive an approximate formula for the price of an option that depends on these parameters.

The fourth strategy described in the article is Ju's Taylor expansion. As in the other strategies, Ju [13] defines the asset prices as being modelled by a geometric Brownian process [23]. The main steps of this method are as follows:

1. Consider a fictitious market where all volatilities are scaled by the same parameter z. This allows to define the price of a basket option and calculate the first two moments of the price of a basket option, that depends on the parameter z.

2. Suppose a normal random variable that depends on the parameter z in such a way that the first two moments of the exponential of this variable are equal to those of the price function of a basket option.

3. Obtain the mean and the variance of this random variable which depends on z, i.e. the mean and the variance also depend on z, and consider a new variable which is the logarithm of the price function of a basket option.

4. Find the density function of this new variable by looking at its characteristic function. After some manipulations, the characteristic function of this variable is found and it is the product of the characteristic function of the normal random variable defined earlier and a function that depends on the parameter z.

5. Calculate the Taylor expansion with polynomials up to degree six of the function that depends on the parameter z. Consider the logarithm of the price function of a basket option with $z = 1$ and approximate its characteristic function and its density function.

With these steps and approximations, it is possible to deduce a formula for the price of a basket option.

The penultimate technique presented in the article is the reciprocal gamma approximation by Milevsky and Posner. First, Milevsky and Posner [20] define a reciprocal gamma distribution, which turns out to be a difference between one and a gamma distribution with an inverse argument to the reciprocal gamma argument. As with all methods, the asset price is assumed to follow a geometric Brownian motion [23]. Additionally, the price of a basket option is defined as a weighted average of the prices of all assets on the maturity date. The goal of this strategy is to show that the sum of log-normal variables is close to a reciprocal gamma. It should be noted, however, that this method is only accurate when the correlation matrix is decaying, as only then does it resemble Asian options. First, the

"pseudo-forward" time T of a basket is fixed and then the price function of a basket option is divided by this variable. This new variable is normalised and its first two moments can be calculated. To obtain the value of the parameters of the reciprocal gamma, equate the moments of this log-normal variable with the reciprocal gamma and derive the values of the parameters as a function of the moments of the log-normal variable. By finding the value of the parameters of the reciprocal gamma, it is possible to deduce the price of a basket option under these conditions.

The final technique covered in this article is the Milevsky and Posner's approximation via higher moments. Milevsky and Posner [21] reduce the problem of pricing a basket option to the product of the discount factor and an integral, where the integrated function is the product between the payoff of the basket option and the state price density function. Since the objective of this method is to approximate the pricing formula using the first four moments of the state price density function to match the higher moments of the arithmetic mean distribution, these are calculated as a function of the mean and standard deviation of this function. The authors evaluate two types of transformation of the normal distribution with parameters to be determined in order to estimate the first four moments of the arithmetic mean. For each of the transformations, it is possible to obtain an approximate formula for the price of a basket option. The type of transformation is selected based on the kurtosis values.

As Monte Carlo simulation is utilised to compare the techniques described above, it was necessary to consult books/articles on the subject. Pagès [24] gives a theoretical description of Monte Carlo simulation for calculating the price of a basket option and for calculating Greeks. In addition to this book, other books [5, 10] were consulted in order to be able to program what was needed in Python. When calculating the Greeks, the finite difference method was also used to calculate the derivatives.

With all the techniques described, Krekel, Kock, Korn and Man [14] choose to test which methods are most effective for approximating the price of a basket option by comparing the results with those obtained through Monte Carlo simulation. They conclude that Ju's [13] method is the most effective. However, the Beisser [1] method is the only viable approach when homogeneous volatilities are lacking. The Milevsky and Posner [20] and Gentle [9] methods exhibit the poorest performance and the Milevsky and Posner [21] method is only recommended for assets with low volatilities.

The explicit formula proposed by Veiga, Wystup, and Esquível [26] is not applicable for pricing basket options, as it imposes a condition that such options do not satisfy. Consequently, it was necessary to rely on the alternative methods discussed earlier.

2.3 Dispersion Options

The next and final type of multi-asset option to be discussed in this thesis is dispersion options, or more specifically, outperformance options.

For this type of option, three articles will be analysed, Margrabe [18], Derman [7] and Veiga, Wystup, and Esquível [26].

Margrabe [18] first introduces the stochastic differential equation that the asset price process follows, which remains a geometric Brownian motion [23]. He then presents the payoff function of this option, which can be seen as a call option on one of the assets and with an exercise price equal to the price of the other asset, since this option is only on two assets, and can also be viewed as a put option where each of the assets is the opposite of what it was in a call option. Since this problem can be reduced to a simple call and put option, it is possible to obtain a two-dimensional Black & Scholes equation (see Equation 4.30) with an initial condition equal to the payoff function and a boundary condition stating that the price of this option is greater than or equal to 0 and less than or equal to the price of the asset. The price function of an outperformance option is deduced by solving the equation.

Derman [7] has a similar interpretation to Margrade [18] in order to derive an explicit formula for the price of these options. In the end, he makes the same assumptions as Margrade [18], but assumes currency changes in the price of this type of option.

The formula developed by Veiga, Wystup, and Esquível [26] is applicable for pricing outperformance options. Notably, the pricing expressions introduced by Margrade [18] and Derman [7] are specific cases of this more general formula.

3

Base Concepts

In this chapter, definitions for the following topics are provided: Black & Scholes model, correlation, Greeks and Monte Carlo simulation. The first subject to be covered is the Black & Scholes model.

3.1 Black & Scholes Model

According to Björk [2], the Black & Scholes model translates into the existence of a market formed by two assets with price dynamics given by:

$$dB(t) = rB(t)dt, \quad B(0) = 1 \tag{3.1}$$

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \quad S(0) = s \tag{3.2}$$

where r (interest rate), μ (instantaneous expected rate of return) and σ (instantaneous variance of the asset return) are constants and $W(t)$ is the Brownian motion process (Musiela and Rutkowski [22]).

The price process of a risk-free asset is represented in Equation 3.1 and the price process of a risky asset is defined in Equation 3.2. Note that the price process of a risk-free asset has a solution that is equal to

$$B(t) = B(0)e^{rt} \tag{3.3}$$

and the price process of a risky asset is a geometric Brownian motion [23] whose solution is given by

$$S(t) = S(0)e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W(t)}. \tag{3.4}$$

One of the most important aspects of this model is the defining equation, which is as follows:

Definition 1 Assume that the market is defined by the Black & Scholes model and is wanted to price a contingent claim $\Phi(S(T))$, which is defined in **Definition 2**. Then the only pricing function of the form $\Pi(t) = F(t, S(t))$, which is consistent with the absence

of arbitrage, is when F is the solution of the following boundary value problem in the domain $[0, T] \times \mathbb{R}_+$

$$\begin{aligned} \frac{\partial F(t,s)}{\partial t} + rs\frac{\partial F(t,s)}{\partial s} + \frac{1}{2}s^2\sigma^2\frac{\partial^2 F(t,s)}{\partial s^2} - rF(t,s) = 0, \\ F(T,s) = \Phi(s). \end{aligned} \tag{3.5}$$

The Equation 3.5 is known as the Black & Scholes Equation.

In the Black & Scholes model, the free arbitrage price of the contingent claim $\Phi(S(T))$ is given by $\Pi(t) = F(t, S(t))$, with

$$F(t,s) = e^{-r(T-t)}\mathbb{E}_{t,s}[\Phi(S(T))] \tag{3.6}$$

and the dynamic of the process S is given by Equation 3.2 with $\mu = r$, where r is the risk-free interest rate.

Having discussed the concept of a contingent claim, Björk [2] defines it as follows:

Definition 2 Consider a financial market with a vector price process S. A contingent claim with maturity date (exercise date) T, also called a T-claim, is an arbitrary random variable $\mathcal{X}$. A contingent claim $\mathcal{X}$ is called a simple claim if it is of the form

$$\mathcal{X} = \Phi(S(T)), \text{ the function } \Phi \text{ is called the contract function.}$$

3.2 Correlation

As multi-asset options are being studied, the topic of correlation should be addressed, as it is a variable that is taken into account when pricing these options.

3.2.1 Realized Correlation

Realized correlation is a statistical measure that expresses the degree to which two variables are linearly related (that is, they change together at a constant rate). It is a common tool for describing simple relationships without implying cause and effect.

There are several correlation coefficients, often referred to as ρ, which measure the degree of correlation. The best known correlation coefficient is Pearson's [4], defined as

$$\rho_{X,Y} = \text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X\sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}, \quad \text{with} \quad \sigma_X\sigma_Y > 0$$

with X and Y being two random variables with expected values μ_X and μ_Y, and standard deviations σ_X and σ_Y, respectively.

A statistical correlation calculated in this way will take values between -1 and $+1$. A negative correlation means that, historically, when one variable has moved up, the other has moved down. A positive correlation means that, historically, both variables have generally moved up or moved down. The cases $\rho = +1$ and $\rho = -1$ indicate perfect positive and perfect negative correlation, respectively. The case of zero correlation means that the two variables move in a generally random way.

3.2.2 Correlation Matrices

A correlation matrix M_ρ is a square matrix that describes the correlation among n variables. Let $S_1(t), S_2(t), \ldots, S_n(t)$ denote the prices at time t of n assets, and ρ_{ij} the correlation between assets i and j, then the correlation matrix M_ρ is given by:

$$M_\rho = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots \\ \rho_{21} & \rho_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} 1 & \rho_{12} & \cdots \\ \rho_{21} & 1 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Two things to note about this matrix are that it is symmetric because the correlation between asset i and asset j is the same as the correlation between asset j and asset i, and it is also necessarily positive definite.

The concept of realized correlation can be defined using the correlation matrix.

According to Bouzoubaa and Osseiran [4], given an index of n stocks or a basket of n assets $S_1, S_2, \ldots, S_n$ with respective weights $w_1, w_2, \ldots, w_n$, the realized index correlation (or realized basket correlation) is defined as the weighted average of the realized correlation matrix between the components, excluding the diagonal of 1's:

$$\rho_{\text{realized}} = \frac{\sum_{1 \le i < j \le n} w_i w_j \rho_{ij}}{\sum_{i<j}^{n} w_i w_j} \tag{3.7}$$

where the weights w_i have the following constraints:

$$0 \le w_i \le 1 \text{ (for all } i = 1, 2, \ldots, n) \text{ and } \sum_{i=0}^{n} w_i = 1$$

and ρ_{ij} is the realized correlation between components i and j.

3.2.3 Implied Correlation

Implied correlation consists of estimating the correlation between the underlyings through the correlation expected by the market for the underlyings over the period to maturity, i.e. it is calculated using the market price of another contingent claim on the same assets [4].

Let's look at an example of how to calculate the implicit correlation of a specific derivative product by using its definition.

Consider the case of an index for which there are both European options on the index itself and on each of the underlyings that make up the index, this case can be called as a call on index versus a call on basket. The market prices can be used to derive an implied correlation, which is a measure of the dependency between the components of the index. For the implied correlation of this example, Bouzoubaa and Osseiran [4] give the following formula:

$$\rho^{\text{index}}_{\text{implied}} = \frac{\sigma^2_{\text{index}} - \sum_{i=1}^{n} w_i^2 \sigma_i^2}{2 \sum_{1 \le i < j \le n} w_i w_j \sigma_i \sigma_j} \tag{3.8}$$

where n is the number of components, w_i is the weight of the ith component in the index, σ_{index} is the implied volatility of the index and σ_i is the implied volatility of the ith component of the index.

In order to demonstrate this result, it is necessary to define the formula for the variance of a portfolio. Markowitz [19] expresses the variance of a portfolio as equal to:

$$\sigma_P^2 = \sum_{1 \le i \le n} w_i^2 \sigma_i^2 + 2 \sum_{1 \le i < j \le n} w_i w_j \sigma_i \sigma_j \rho_{ij}. \tag{3.9}$$

Note that this expression is the formula for the variance of a linear combination of random variables. Going back to the variance of a portfolio (Equation 3.9) and considering the index as a portfolio of n assets with different weights, then the implied correlation of an index is denoted as the correlation ρ_{implied}, which, when substituted in place of the $n(n-1)$ individual correlations ρ_{ij}, gives the same portfolio variance:

$$\sigma_{\text{index}}^2 = \sum_{1 \le i \le n} w_i^2 \sigma_i^2 + 2 \sum_{1 \le i < j \le n} w_i w_j \sigma_i \sigma_j \rho_{\text{implied}}^{\text{index}}.$$

After some manipulations on the previous equality, the Equation 3.8 is obtained. One thing to point out is that this formula assumes that all weights are constant. However, in the case of an index, this is usually not the case as the weights vary as the components of the index vary, making the above formula inaccurate for an index.

Another way of calculating the implied correlation of an index is through a simple parameterisation involving a coefficient λ which relates the realized and implied correlations of the index, and in turn using this coefficient and also the realized correlation between the index components to derive a specific implied correlation. To do this, first the realized correlation of the index is computed with Equation 3.7 and the implied correlation using Equation 3.8, then solve in order to λ the following equation:

$$\rho_{\text{implied}}^{\text{index}} = \rho_{\text{realized}}^{\text{index}} + \lambda(1 - \rho_{\text{realized}}^{\text{index}}). \tag{3.10}$$

Having obtained the value of λ using Equation 3.10, use this value and the realized correlation between the stocks to estimate the implied correlation between them.

3.3 Greeks

Since a sensitivity analysis is performed for each multi-asset option, the concept of Greeks must be established.

The Greeks are defined as derivatives of the option value with respect to various variables and parameters. The purpose of calculating the Greeks is to see how sensitive the price function of a derivative product is to the variables on which it depends.

The three Greeks considered most important for sensitivity analysis are:

- The derivative of the price function in order to the price of the underlying asset, known as the Delta, is represented as a vector with n entries. Each entry i in the

Delta vector corresponds to the derivative of the price function with respect to the price of the i-th underlying asset. The formula for the i-th entry of Delta is given by:

$$\Delta_i = \frac{\partial P}{\partial S_i}.$$

- The derivative of the price function in order to the volatility of the underlying asset, known as the Vega, is represented as a vector with n entries. Each entry i in the Vega vector corresponds to the derivative of the price function with respect to the volatility of the i-th underlying asset. The formula for the i-th entry of Vega is given by:

$$\nu_i = \frac{\partial P}{\partial \sigma_i}.$$

- The derivative of the price function in order to the correlation between the underlying assets, known as the Cega, is represented as a square matrix of size n. Each entry (i, j) in the Cega matrix corresponds to the derivative of the price function with respect to the correlation between the i-th and j-th underlying assets. The formula for the (i, j)-th entry of the Cega is given by:

$$\text{Cega}_{ij} = \frac{\partial P}{\partial \rho_{ij}}.$$

3.4 Monte Carlo Simulation

Sometimes it is not possible to explicitly calculate the formulas for the price of derivative products, so Monte Carlo simulation is used as an alternative.

In order to implement the Monte Carlo method, three different concepts need to be addressed. The first concept is the law of large numbers, which is defined as follows: If $(X_k)_{k\geq 1}$ denotes a sequence, defined in the probability space $(\Omega, \mathcal{A}, \mathbb{P})$, of independent and integrable random variables with the same distribution as X, then

$$\mathbb{P}(d\omega) - a.s \quad \overline{X}_M(\omega) := \frac{X_1(\omega) + \cdots + X_M(\omega)}{M} \xrightarrow{M\to+\infty} m_X := \mathbb{E}[X].$$

The central limit theorem is another notion that is explained. This theorem states that if X is square integrable ($X \in L^2(\mathbb{P})$), then

$$\sqrt{M}\left(\overline{X}_M - m_X\right) \xrightarrow{\mathcal{L}} N\left(0, \sigma_X^2\right) \quad \text{as} \quad M \to +\infty.$$

Finally, it is possible to specify the definition of a confidence interval in the Monte Carlo simulation. Therefore, the confidence interval at a α level is equal to

$$I_M^\alpha = \left[\overline{X}_M - q_\alpha\sqrt{\frac{\overline{V}_M}{M}}, \overline{X}_M + q_\alpha\sqrt{\frac{\overline{V}_M}{M}}\right]$$

which will still satisfy (for large M)

$$\mathbb{P}(m_X \in I_M^\alpha) \simeq \mathbb{P}(|N(0,1)| \leq q_\alpha) = \alpha.$$

In the confidence interval expression, $\overline{X}_M$ is the mean estimator, $\overline{V}_M$ is the variance estimator and q_α is the two sided α-quantile.

In the context of this thesis, let's look at the use of Monte Carlo simulation for options. Knowing that the value of an option is given by Equation 3.6, the following steps must be taken to calculate the price of an option:

1. Simulate the price process using Equation 3.2, which represents the stochastic differential equation that models asset prices, utilising the current value of the assets.

2. Then calculate the payoff of the option at maturity based on the previously simulated price.

3. Generate more realizations, around 10000 price repetitions usually gives a good approximation [24].

4. Calculate the average payoff over all realizations.

5. Take the present value of this average, this is the Monte Carlo option value (Equation 3.6).

Monte Carlo simulation can also be used to calculate the Greeks of derivative products that do not have an explicit pricing formula. Let's see how to simulate the derivative of an option using Monte Carlo simulation.

1. Make a small perturbation in the value of the parameter to be calculated the derivative of. Let's denote this perturbation ϵ, where this value is added to and subtracted from the initial value of the parameter.

2. Then do item 1, item 2 and item 3 for both disturbances.

3. As the derivatives of the simulated prices are to be calculated, the finite differences method (see Pagès [24] for more details) is applied, with the simulated perturbed prices being used for the calculation.

4. Calculate the average of the finite differences, which will be the value of the Monte Carlo Greek.

4

Multi-Asset Options

In this chapter, the assumptions and the differential equation of each multi-asset option are analysed. Some characteristics of these options are also studied.

A sensitivity analysis is performed for each of the options, so the calculation of some derivatives are demonstrated. One of the parameters of interest in the analysis is the correlation, which was discussed earlier in section 3.2.

But first, it is important to consider some definitions and characteristics of simple options, as these are analogous to the multi-asset options.

4.1 Options

According to Bouzoubaa and Osseiran [4], options are contracts that give their holder the right, but not the obligation, to either buy (known as a call option) or sell (known as a put option) an amount of an underlying asset at a pre-determined price, known as the strike price, on or before a pre-determined date in the future, the maturity date. The term of an option can be as short as one day or as long as a few years. To buy an option, a premium has to be paid.

The most common types of options are European and American. The difference between European and American options is that European options can only be exercised on the maturity date. American options can be exercised at any time before the maturity date.

In the next subsections, the formula for the price of a call option is introduced, based on what was studied by Black & Scholes [3], and a property of parity is contemplated.

4.1.1 Price of an European Call Option Under the Black & Scholes Model

The European call option is an example of a simple contingent claim, for which the contract function is given by

$$\Phi(S(T)) = \max[S(T) - K, 0].$$

Knowing the contract function of a call option, the formula for the price of this option can be deduced.

Thus, Björk [2] shows that the price of an European call option with strike price K and maturity T is given by the formula $C(S(t), K, T)$, with,

$$C(S(t), K, T) = S(t)N(d_1(t, S(t))) - e^{-r(T-t)}KN(d_2(t, S(t))), \tag{4.1}$$

where N represents the normal distribution function $N(0, 1)$ and,

$$d_1(t, S(t)) = \frac{1}{\sigma\sqrt{T-t}}\left(\log\left(\frac{S(t)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right), \tag{4.2}$$

$$d_2(t, S(t)) = d_1(t, S(t)) - \sigma\sqrt{T-t}. \tag{4.3}$$

4.1.2 Put-Call Parity

Let's consider an European call and an European put, both with strike price K and time of maturity T. The corresponding price functions, denoted by $C(S(t), K, T)$ and $P(S(t), K, T)$, have the following relationship presented in Björk [2]:

$$P(S(t), K, T) = Ke^{-r(T-t)} + C(S(t), K, T) - S(t). \tag{4.4}$$

One of the advantages of this relationship is that the price of an European call option can be used to obtain the price of an European put option on the same terms. However, it is also possible to deduce a formula for the put option, analogous to the call option (Equation 4.1). Note that the parity relation is model-free, i.e. it is sufficient to use non-arbitrage arguments to justify it.

4.2 Basket Options

A basket option is a type of financial derivative where the underlying asset is a group or basket of an asset, such as commodities, stocks, securities or currencies [4]. Like other options, a basket option gives the holder the right, but not the obligation, to buy or sell the basket at a specified price on a specified date.

The price of a basket of underlyings is defined by the following expression

$$B(T) = \sum_{i=1}^{n} w_i S_i(T), \tag{4.5}$$

which means that the price of a basket of underlyings is the weighted average of the prices of n underlyings at maturity T. The weights w_i are normally positive and their sum is equal to 1.

Once the price expression of a basket is defined, its payoff can be determined. So the price of a call ($\theta = 1$) or put ($\theta = -1$) with strike K and maturity T on the basket has a payoff equal to

$$P_{\text{Basket}}(B(T), K, \theta) = \max[\theta(B(T) - K), 0] \tag{4.6}$$

with $B(T)$ defined in Equation 4.5.

4.2.1 The Pricing of a Basket Option

In this subsection, the assumptions made in an attempt to derive a formula for the price of these options are outlined. These are:

1. The prices of the assets are described by the following stochastic differential equation:

$$dS_i(t) = \mu_i S_i(t)dt + \sigma_i S_i(t)dW_i(t) \tag{4.7}$$

   where σ_i^2 are constant and are known as the instantaneous variances of the asset returns. The instantaneous expected rate of return, represented by μ_i, are also constant. The W_i are the Brownian motion processes and are correlated with each other, the coefficient of correlation between them is constant over time.

2. The current interest rate r is constant over time.

3. The portfolio whose value is P and replicates the derivative, satisfies the following partial differential equation:

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_i\sigma_j\rho_{ij}S_iS_j\frac{\partial^2 P}{\partial S_i\,\partial S_j} + r\sum_{i=1}^{n}S_i\frac{\partial P}{\partial S_i} - rP = 0 \tag{4.8}$$

   with a boundary condition equal to Equation 4.6.

In addition to these assumptions, the following variables are defined. The T-forward price of asset i is given by

$$F_i^T = S_i(0)e^{\int_0^T r ds} = S_i(0)e^{rT}. \tag{4.9}$$

Another variable is the resolution of Equation 4.7 with $\mu_i = r$, as it will be used in subsequent subsections, and the solution to the equation is equal to

$$\begin{aligned} S_i(T) &= F_i^T e^{-\int_0^T \frac{1}{2}\sigma_i^2 ds + \int_0^T \sigma_i dW_i(s)} \\ &= S_i(0)e^{rT}e^{-\frac{1}{2}\sigma_i^2 T + \sigma_i W_i(T)} \\ &= S_i(0)e^{\left(r-\frac{1}{2}\sigma_i^2\right)T + \sigma_i W_i(T)}. \end{aligned} \tag{4.10}$$

The last variable to be established is the discount factor, which is equal to

$$\mathrm{Df}(T) = e^{-\int_0^T r ds} = e^{-rT}. \tag{4.11}$$

Even with all these assumptions and variables, there is a problem with using the Black & Scholes model to arrive at a formula for the price of these options. The assets are modelled by a geometric Brownian motion, which means that the assets have a log-normal distribution, and the sum of all the assets does not have this property and so it is not possible to obtain an explicit formula for the price of these multi-asset options.

A way of dealing with this problem is to use Monte Carlo simulation to have a price for these options. The application of Monte Carlo simulation for pricing an option is detailed in section 3.4.

Another approach is to use methods based on analytical approximations. Therefore, the next subsection focus on the study of four methods that can be employed to approximate the price of basket options.

4.2.2 Methods to Approximate the Price of a Basket Option

The article by Krekel, Kock, Korn and Man [14] presents six techniques for approximating the price of a basket option, but only four of these are discussed. The first technique to be examined is **Levy's log-normal moment matching**.

1) Levy's log-normal moment matching

Levy [15] and Levy and Turnbull [16] begin by defining the variable $A(t)$ as a "running average". Thus, with $t_i = t_0 + ih$ where $i = 0, 1, \ldots, n$ and $h = \frac{(t_n - t_0)}{n}$ then

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

with $0 \leq m \leq n$, $t_m \leq t \leq t_{m+1}$ and $A(t) = 0$ if $t < t_0$. This means that $A(t_n)$ is the arithmetic mean of $n+1$ prices at equal time intervals between t_0 and t_n.

Then the payoff of this method at time t_n can be fixed, which is

$$\max[A(t_n) - K, 0]$$

in the case of a call and

$$\max[K - A(t_n), 0]$$

in the case of a put.

The price processes of the underlyings are still modelled by a geometric Brownian motion, i.e. the stochastic differential equation is equal to Equation 4.7.

In the article, the authors demonstrate how to derive the formula for the price of a call. They define the basket call price as

$$BC_{\text{Levy}}(t) = e^{-rT}\mathbb{E}^*[\max[M(t_n) - K^*, 0]]$$

where $\mathbb{E}^*$ is the expected value conditioned on $(A(t), S(t))$ at time t under the risk-adjusted density function f^*, $K^* = K - A(t)\frac{(m+1)}{(n+1)}$ and $M(t) = A(t_n) - A(t)\frac{(m+1)}{(n+1)}$. By definition,

$$\mathbb{E}^*[\max[M(t_n) - K^*, 0]] = \int_{K^*}^{\infty} [M(t_n) - K^*] f^*(w) dw.$$

The idea behind this method is to assume that $M(t)$ is a geometric mean $G(t) = [S(t_i) \times S(t_{i+1}) \times \cdots \times S(t_n)]^{\frac{1}{n+1}}$. Let's stipulate that m and v^2, which are calculated later, are the mean and the variance of $\log(G(t))$.

Therefore, with this assumption it is known that

$$\mathbb{E}^*[M(t)^n] = e^{nm + \frac{1}{2}n^2v^2},$$

so m and v^2 are determined using the first two moments of $M(t)$. This results in

$$m = 2\log(M) - \frac{1}{2}\log(V^2) \tag{4.12}$$

and

$$v^2 = \log(V^2) - 2\log(M) \tag{4.13}$$

with

$$M \equiv \mathbb{E}^*[M(t)] = \sum_{i=1}^{n} w_i F_i(T), \tag{4.14}$$

$$V^2 \equiv \mathbb{E}^*[M(t)^2] = \sum_{i,j=1}^{n} w_i w_j F_i^T F_j^T e^{\sigma_i \sigma_j \rho_{ij} T}. \tag{4.15}$$

This signifies that the price of a basket call option utilising this technique is equal to

$$BC_{\text{Levy}}(t) = \text{Df}(T)(MN(d_1) - KN(d_2)) \tag{4.16}$$

with

$$d_1 = \frac{m - \log(K) + v^2}{v}, \tag{4.17}$$

$$d_2 = d_1 - v. \tag{4.18}$$

For the put, a similar reasoning can be used to arrive at an approximate price formula.

2) Ju's Taylor expansion

Ju [13] states that the Equation 4.7 is still the stochastic differential equation that the price process follows.

Let's consider that the correlation coefficient between the Brownian processes of assets i and j is equal to ρ_{ij}.

Since the Taylor expansion method is to be applied around zero volatility, a fictitious market must be considered where all volatilities are scaled by the same parameter z, so that the following result is obtained

$$S_i(z,t) = S_i e^{\left(r - \frac{z^2\sigma_i^2}{2}\right)t + z\sigma_i W_i(t)}, \quad i = 1, 2, \ldots, n.$$

If $z = 1$, the Equation 4.10 is obtained with $T = t$.

Consider the variable

$$A(z) = \sum_{i=1}^{n} w_i S_i(z,T)$$

where w_i is the weight of the asset i and $S_i(z,T)$ is equal to the variable previously defined with $t = T$. Thus, the payoff of a basket call option is equal to

$$BC_{\text{Ju}}(T) = \max[A(1) - K, 0].$$

To simplify, the following variables $\bar{S_i} = w_i S_i e^{rT}$ and $\bar{\rho_{ij}} = \rho_{ij}\sigma_i\sigma_j T$ are introduced.

The first two moments of $A(z)$ are equal to

$$M_1 = \sum_{i=1}^{n} \bar{S}_i = A(0),$$

$$M_2(z^2) = \sum_{ij=1}^{n} \bar{S}_i \bar{S}_j e^{z^2 \bar{\rho}_{ij}}.$$

Observe that M_1 is equal to Equation 4.14 and if $z^2 = 1$, then $M_2(z^2)$ is equal to Equation 4.15.

Let $Y(z)$ be a normal random variable with mean $m(z^2)$ and variance $v(z^2)$. In order for the first two moments of $e^{Y(z)}$ to be equal to those of $A(z)$, it follows that

$$m(z^2) = 2\log(M_1) - \frac{1}{2}\log(M_2(z^2)),$$

$$v(z^2) = \log(M_2(z^2)) - 2\log(M_1).$$

If $z^2 = 1$, then $m(z^2)$ and $v(z^2)$ are equal to Equation 4.12 and Equation 4.13, respectively.

Consider the variable $X(z) = \log(A(z))$. The objective is to find the density function of $X(z)$, and to achieve this the characteristic function of this variable is considered, which is equal to

$$\mathbb{E}\left[e^{i\phi X(z)}\right] = \mathbb{E}\left[e^{i\phi Y(z)}\right] \frac{\mathbb{E}\left[e^{i\phi X(z)}\right]}{\mathbb{E}\left[e^{i\phi Y(z)}\right]} = \mathbb{E}\left[e^{i\phi Y(z)}\right] f(z)$$

where

$$\mathbb{E}\left[e^{i\phi Y(z)}\right] = e^{i\phi m(z^2) - \frac{\phi^2 v(z^2)}{2}}$$

is the characteristic function of the normal random variable $Y(z)$ and

$$f(z) = \frac{\mathbb{E}\left[e^{i\phi X(z)}\right]}{\mathbb{E}\left[e^{i\phi Y(z)}\right]} = \mathbb{E}\left[e^{i\phi X(z)}\right] e^{-i\phi m(z^2) + \frac{\phi^2 v(z^2)}{2}}$$

is the ratio of the characteristic function of $X(z)$ to that of $Y(z)$.

The idea is to expand $f(z)$ around $z = 0$ up to z^6 and to do that the term $e^{-i\phi m(z^2) + \frac{\phi^2 v(z^2)}{2}}$ needs to be expanded first. The detailed calculations for this term are in Ju's article [13].

The next step is to approximate the term $\mathbb{E}[e^{i\phi X(z)}]$ by deriving it in order to the second, fourth and sixth derivatives. The calculus of these derivatives can be reviewed in the article. With the derivatives calculated, the term can be approximated using the Taylor expansion.

Then obtain the Taylor expansion of $f(z)$ which is equal to

$$f(z) \approx 1 - i\phi d_1(z) - \phi^2 d_2(z) + i\phi^3 d_3(z) + \phi^4 d_4(z)$$

where $d_i(z)$ are polynomials of z up to degree 6, higher degrees are ignored.

To complete the calculations and get the approximate formula for the basket call price, it is necessary to calculate $\mathbb{E}[e^{i\phi X(z)}]$ and its density function at the point $z = 1$. As always, the detailed calculations can be consulted in the article.

The price of a basket call using this method is therefore equal to

$$\begin{aligned} BC_{\text{Ju}}(t) &= \text{Df}(T)\mathbb{E}[e^{X(1)} - K]^+ \\ &= (M_1\text{Df}(T)N(y_1) - K\text{Df}(T)N(y_2)) \\ &+ \left(\text{Df}(T)K\left(z_1 p(y) + z_2\frac{\text{d}p(y)}{\text{d}y} + z_3\frac{\text{d}^2p(y)}{\text{d}y^2}\right)\right) \end{aligned} \tag{4.19}$$

where

$$y = \log K, \quad y_1 = \frac{m(1) - y}{\sqrt{v(1)}} + \sqrt{v(1)} \quad \text{and} \quad y_2 = y_1 - \sqrt{v(1)}.$$

Furthermore, $z_1 = d_2(1) - d_3(1) + d_4(1)$, $z_2 = d_3(1) - d_4(1)$ and $z_3 = d_4(1)$, where the values of $d_i(1)$ are derived from the article, and $p(y)$ is the density function of a normal distribution with mean $m(1)$ and variance $v(1)$.

3) The reciprocal gamma approximation by Milevsky and Posner

Milevsky and Posner [20] begin by defining the cumulative distribution function of a reciprocal gamma, which is equal to

$$G_R(y, \alpha, \beta) = 1 - G\left(\frac{1}{y}, \alpha, \beta\right)$$

where $G(\dots)$ is the cumulative distribution function of a gamma. Therefore, the probability density function of a reciprocal gamma is equal to

$$g_r(y, \alpha, \beta) = \frac{g\left(\frac{1}{y}, \alpha, \beta\right)}{y^2}, \quad y \geq 0, \quad \alpha, \beta > 0$$

where $g(\dots)$ is the probability density function of a gamma.

If a random variable Y is reciprocal gamma distributed, then

$$\mathbb{E}[Y^i] = \frac{1}{\beta^i(\alpha - 1)(\alpha - 2)\dots(\alpha_i)}, \quad i = 1, 2, 3, \dots \quad .$$

The price of the underlying assets is still modelled by the Equation 4.7. The price of a basket is given by Equation 4.5. The value of a basket call option is consequently equal to

$$BC_{\text{MP-RG}}(t) = e^{-rT}\mathbb{E}^*[\max(B(T) - K, 0)] = e^{-rT}\int_0^\infty \max(B(T) - K, 0)d\Psi[B(T)]$$

where $\Psi[B(T)]$ is the state-price density function.

Milevsky and Posner [20] say that it can be demonstrated that the sum of log-normal variables is almost approximately a reciprocal gamma. Furthermore, they emphasise that this method is only valid if the correlation matrix is decaying, as this is the only way that basket options resemble Asian options.

Initially, the first two moments of the variable $B(T)$ are defined. This variable has a mean equal to Equation 4.14 and a second moment M_2 equal to Equation 4.15.

In the article it is explained how to approximate a basket call formula using a log-normal approximation. It also shows how to approximate the formula using reciprocal gamma as the distribution.

Since $M_1 = M$, $\mathbb{E}[Y] = \frac{1}{\beta(\alpha-1)}$ and $\mathbb{E}[Y^2] = \frac{1}{\beta(\alpha-1)(\alpha-2)}$, then

$$\beta = \frac{1}{M_1(\alpha-1)}$$

$$\text{and } M_2 = \frac{1}{\beta^2(\alpha-1)(\alpha-2)}.$$

This means that

$$\alpha = \frac{2M_2 - M_1^2}{M_2 - M_1^2} \tag{4.20}$$

$$\beta = \frac{M_2 - M_1^2}{M_2 M_1}. \tag{4.21}$$

Thus, the price formula utilising the reciprocal gamma approximation is equal to

$$BC_{\text{MP-RG}}(t) = \text{Df}(T)\left(M_1 G\left(\frac{1}{K}, \alpha - 1, \beta\right) - KG\left(\frac{1}{K}, \alpha, \beta\right)\right). \tag{4.22}$$

The detailed calculations of the Equation 4.22 can be found in [20].

4) Milevsky and Posner's approximation via higher moments

Milevsky and Posner [21] begin by defining the variable X_T as a stochastic element of the time-T basket of underlyings. The aim is to price a basket call with a payoff equal to $\max[X_T - K, 0]$.

The generalised forward price of the stochastic element X_T, denoted $F(X_T)$, is basically the first moment of X_T at time T, i.e,

$$F(X_T) := E(X_T) := \mu_x := \int_0^\infty x h(x) dx.$$

The other moments are defined as being equal to:

$$\zeta := \sqrt{\mathbb{E}[(X_T - F(X_T))^2]},$$

$$\eta := \frac{\mathbb{E}[(X_T - F(X_T))^3]}{\zeta^3},$$

$$\kappa := \frac{\mathbb{E}[(X_T - F(X_T))^4]}{\zeta^4}.$$

The idea of this method is to consider two types of transformations involving the standard normal distribution, as these serve as an approximation to the true state-price density function. The type I transformation is equal to

$$X = c + de^{\frac{Z-a}{b}}$$

and the type II transformation is equal to

$$X = c + d \sinh\left(\frac{Z - a}{b}\right)$$

with a, b, c, d to be determined.

The pricing problem is then equal to

$$e^{-rT}\left[\int_K^\infty (x - K)h(x)dx\right]$$

where the cumulative distribution function of $h(x)$ is one of the transformations mentioned above.

The price formulas for each type of transformation can be found in the article [21]. However, the Type I transformation is the only focus to obtain a price formula, as it is the one that most closely resembles a log-normal distribution. The price formula for Type I is therefore

$$BC_{\text{MP-4M}}(t) = \text{Df}(T)\left[F(X_T) - K + (K - c)N(Q) - de^{\frac{1-2ab}{2b^2}} N\left(Q - \frac{1}{b}\right)\right] \tag{4.23}$$

with $F(X_T)$ equal to Equation 4.14,

$$b = \frac{1}{\sqrt{\log(w)}},$$

$$a = \frac{1}{2} b \log\left(\frac{w(w-1)}{\zeta^2}\right),$$

$$d = \text{sign}(\eta),$$

$$c = dF(X_T) - e^{\frac{\left(\frac{1}{2b} - a\right)}{b}},$$

$$Q = a + b \log\left(\frac{K - c}{d}\right),$$

$$\text{and } w = \frac{1}{2}\sqrt[3]{8 + 4\eta^2 + 4\sqrt{4\eta^2 + \eta^4}} + \frac{2}{\sqrt[3]{8 + 4\eta^2 + 4\sqrt{4\eta^2 + \eta^4}}} - 1.$$

This pricing formula (Equation 4.23) and the parameters that define it are demonstrated in the article [21].

4.2.3 Sensitivity Analysis

In this subsection, the Greeks of the basket options are calculated using Monte Carlo simulation and through the derivatives of the approximate pricing formulas from the methods discussed in subsection 4.2.2. Only three of the various existing Greeks are calculated for this option, namely Δ, ν and Cega.

The process of calculating the Greeks using Monte Carlo simulation is described in section 3.4. Due to the complexity of the pricing formulas of some of the methods, it

was decided to calculate the derivatives of the functions corresponding to only two of the methods described in subsection 4.2.2.

1) Levy's log-normal moment matching

Note that in each of the Greeks that will be calculated later, the formulas are for generic cases. The first Greek to be calculated is Δ:

$$\Delta_i = \frac{\partial P}{\partial S_i} = e^{-rT}\left(w_i e^{rT} N(d_1) + M\varphi(d_1)\frac{\partial d_1}{\partial S_i} - K\varphi(d_2)\frac{\partial d_2}{\partial S_i}\right)$$

where $\varphi(.)$ represents the density function of $N(0,1)$.

From the definition of d_2 (see Equation 4.18) it can be noted that

$$\frac{\partial d_2}{\partial S_i} = \frac{\partial d_1}{\partial S_i} - \frac{\partial v}{\partial S_i}.$$

Furthermore the term $\varphi(d_2)$ is equal to

$$\begin{aligned}
\varphi(d_2) &= \frac{e^{-\frac{d_2^2}{2}}}{\sqrt{2\pi}} = \frac{e^{-\frac{(d_1-v)^2}{2}}}{\sqrt{2\pi}} = \frac{e^{-\frac{d_1^2-2d_1v+v^2}{2}}}{\sqrt{2\pi}} \\
&= e^{d_1 v - \frac{v^2}{2}}\varphi(d_1) = e^{m-\log(K)+v^2-\frac{v^2}{2}}\varphi(d_1) \\
&= e^m K^{-1} e^{\frac{v^2}{2}}\varphi(d_1) = e^{2\log(M)-0.5\log(V^2)}\frac{1}{K}e^{0.5\log(V^2)-\log(M)}\varphi(d_1) \\
&= e^{\log(M)}\frac{1}{K}\varphi(d_1) = \frac{M}{K}\varphi(d_1).
\end{aligned}$$

That means,

$$\Delta_i = \frac{\partial P}{\partial S_i} = e^{-rT}\left(w_i e^{rT} N(d_1) + M\varphi(d_1)\frac{\partial v}{\partial S_i}\right). \tag{4.24}$$

All that remains is to calculate the value of $\frac{\partial v}{\partial S_i}$. That is equal to

$$\frac{\partial v}{\partial S_i} = \frac{1}{2}\frac{1}{v}\left(\frac{2\sum_{j=1,i\neq j}^{n} w_i w_j e^{2rT} S_j(0) e^{\sigma_i\sigma_j\rho_{ij}T} + 2w_i^2 e^{2rT} S_i(0) e^{\sigma_i^2 T}}{V^2} - \frac{2w_i e^{rT}}{M}\right).$$

Next, the derivative in order to the correlation is calculated.

$$\text{Cega}_{ij} = \frac{\partial P}{\partial \rho_{ij}} = e^{-rT} M\varphi(d_1)\frac{\partial v}{\partial \rho_{ij}}. \tag{4.25}$$

The calculation of $\frac{\partial v}{\partial \rho_{ij}}$ is as follows:

$$\frac{\partial v}{\partial \rho_{ij}} = \frac{1}{2}\frac{1}{v}\left(\frac{2w_i w_j F_i^T F_j^T \sigma_i\sigma_j T e^{\sigma_i\sigma_j\rho_{ij}T}}{V^2}\right) = \frac{w_i w_j F_i^T F_j^T \sigma_i\sigma_j T e^{\sigma_i\sigma_j\rho_{ij}T}}{vV^2}.$$

To complete the calculation of the Greeks using this method, let's calculate ν.

$$\nu_i = \frac{\partial P}{\partial \sigma_i} = e^{-rT} M \varphi(d_1) \frac{\partial v}{\partial \sigma_i}. \tag{4.26}$$

The value of $\frac{\partial v}{\partial \sigma_i}$ is

$$\frac{\partial v}{\partial \sigma_i} = \frac{1}{2}\frac{1}{v}\left(\frac{2\sum_{j=1, j\neq i}^{n} w_i w_j F_i^T F_j^T \rho_{ij}\sigma_j T e^{\sigma_i \sigma_j \rho_{ij} T} + 2 w_i^2 (F_i^T)^2 \sigma_i T e^{\sigma_i^2 T}}{V^2} \right).$$

3) The reciprocal gamma approximation by Milevsky and Posner

The Greeks that are calculated for this method are for generic cases. The first Greek to be calculated is Δ.

$$\begin{aligned} \Delta_i = \frac{\partial P}{\partial S_i} = e^{-rT}\left(w_i e^{rT} G\left(\frac{1}{K}, \alpha - 1, \beta\right) + M \frac{\partial G}{\partial S_i}\left(\frac{1}{K}; \alpha - 1, \beta\right)\right) \\ - e^{-rT} K \frac{\partial G}{\partial S_i}\left(\frac{1}{K}; \alpha, \beta\right). \end{aligned} \tag{4.27}$$

Let's calculate the term $\frac{\partial G(x;\alpha-1,\beta)}{\partial S_i}$. If $\alpha = \alpha(S_1, S_2, ..., S_n)$ and $\beta = \beta(S_1, S_2, ..., S_n)$, then:

$$\frac{\partial G}{\partial S_i}(x; \alpha - 1, \beta) = \frac{\partial}{\partial S_i}\left(\int_0^x g(u; \alpha - 1, \beta) du \right) = \int_0^x \frac{\partial g}{\partial S_i}(u; \alpha - 1, \beta) du =$$
$$= \int_0^x \frac{\partial g}{\partial \alpha}\frac{\partial \alpha}{\partial S_i} + \frac{\partial g}{\partial \beta}\frac{\partial \beta}{\partial S_i} du = \int_0^x \frac{\partial g}{\partial \alpha}(u; \alpha - 1, \beta) du \frac{\partial \alpha}{\partial S_i} + \int_0^x \frac{\partial g}{\partial \beta}(u; \alpha - 1, \beta) du \frac{\partial \beta}{\partial S_i}.$$

The last integral is equal to:

$$\begin{aligned} \frac{\partial g}{\partial \beta}(u; \alpha - 1, \beta) &= \frac{\partial}{\partial \beta}\left(\frac{u^{\alpha-2} e^{-\frac{u}{\beta}} \left(\frac{1}{\beta}\right)^{\alpha-1}}{\Gamma(\alpha - 1)} \right) \\ &= \frac{u^{\alpha-2}}{\Gamma(\alpha-1)}\left(\frac{u}{\beta^2} e^{-\frac{u}{\beta}} \left(\frac{1}{\beta}\right)^{\alpha-1} + e^{-\frac{u}{\beta}} (\alpha - 1)\left(\frac{1}{\beta}\right)^{\alpha-2} \frac{-1}{\beta^2} \right) \\ &= \frac{u^{\alpha-2}}{\Gamma(\alpha-1)} e^{-\frac{u}{\beta}} \left(\frac{1}{\beta}\right)^{\alpha-1} \left(\frac{u}{\beta^2} - \frac{\alpha - 1}{\beta} \right) \\ &= -g(u; \alpha - 1, \beta)\frac{\alpha - 1}{\beta} + g(u; \alpha, \beta)\frac{\Gamma(\alpha)}{\Gamma(\alpha-1)\beta} \end{aligned}$$

$$\int_0^x \frac{\partial g}{\partial \beta}(u; \alpha - 1, \beta) du = -G(x; \alpha - 1, \beta)\frac{\alpha - 1}{\beta} + G(x; \alpha, \beta)\frac{\Gamma(\alpha)}{\Gamma(\alpha-1)\beta}.$$

While the first integral is equal to:

$$\frac{\partial g}{\partial \alpha}(u; \alpha - 1, \beta) = \frac{\partial}{\partial \alpha}\left(\frac{u^{\alpha-2} e^{-\frac{u}{\beta}} \left(\frac{1}{\beta}\right)^{\alpha-1}}{\Gamma(\alpha - 1)} \right) = \frac{e^{-\frac{u}{\beta}}}{\beta}\frac{\partial}{\partial \alpha}\left(\frac{e^{(\alpha-2)\ln\left(\frac{u}{\beta}\right)}}{\Gamma(\alpha - 1)} \right) =$$

$$= \frac{e^{-\frac{u}{\beta}} \left(\frac{u}{\beta}\right)^{\alpha-2}}{\beta\Gamma^2(\alpha-1)}\left(\ln\left(\frac{u}{\beta}\right)\Gamma(\alpha - 1) - \Gamma'(\alpha - 1) \right) = g(u; \alpha - 1, \beta)\left(\ln\left(\frac{u}{\beta}\right) - \frac{\Gamma'(\alpha - 1)}{\Gamma(\alpha - 1)} \right)$$

$$\int_0^x \frac{\partial g}{\partial \alpha}(u;\alpha-1,\beta)du = \int_0^x \ln(u)g(u;\alpha-1,\beta)du + G(x;\alpha-1,\beta)\left(\ln\left(\frac{1}{\beta}\right) - \frac{\Gamma'(\alpha-1)}{\Gamma(\alpha-1)}\right).$$

leaving an integral to be calculated which can be approximated numerically.

The only terms left to calculate are $\frac{\partial \alpha}{\partial S_i}$ and $\frac{\partial \beta}{\partial S_i}$. It can be seen that:

$$\alpha = \frac{V^2}{V^2 - M^2} + 1 \quad \text{and} \quad \beta = \frac{1}{M} - \frac{M}{V^2}.$$

Thus,

$$\begin{aligned}
\frac{\partial \alpha}{\partial S_i} &= \frac{\frac{\partial V^2}{\partial S_i}(V^2-M^2) - V^2\frac{\partial}{\partial S_i}(V^2-M^2)}{(V^2-M^2)^2} \\
&= \frac{\frac{\partial V^2}{\partial S_i}V^2 - \frac{\partial V^2}{\partial S_i}M^2 - V^2\frac{\partial V^2}{\partial S_i} + V^2\frac{\partial M^2}{\partial S_i}}{(V^2-M^2)^2} \\
&= \frac{2V^2M\frac{\partial M}{\partial S_i} - M^2\frac{\partial V^2}{\partial S_i}}{(V^2-M^2)^2} \\
&= \frac{2V^2Mw_ie^{rT} - M^2\left(2\sum_{j\neq i,j=1}^n w_jw_iS_j(0)e^{2rT}e^{\sigma_i\sigma_j\rho_{ij}T} + 2w_i^2S_i(0)e^{2rT}e^{\sigma_i^2T}\right)}{(V^2-M^2)^2},
\end{aligned}$$

$$\frac{\partial \beta}{\partial S_i} = -\frac{w_ie^{rT}}{M^2} - \frac{w_ie^{rT}V^2 - M\left(2\sum_{j\neq i,j=1}^n w_jw_iS_j(0)e^{2rT}e^{\sigma_i\sigma_j\rho_{ij}T} + 2w_i^2S_i(0)e^{2rT}e^{\sigma_i^2T}\right)}{(V^2)^2}.$$

The term $\frac{\partial G(x;\alpha,\beta)}{\partial S_i}$ can be calculated in a similar way as before. Therefore,

$$\begin{aligned}
&\frac{\partial G}{\partial S_i}(x;\alpha,\beta) = \frac{\partial}{\partial S_i}\left(\int_0^x g(u;\alpha,\beta)du\right) = \int_0^x \frac{\partial g}{\partial S_i}(u;\alpha,\beta)du = \\
&= \int_0^x \frac{\partial g}{\partial \alpha}\frac{\partial \alpha}{\partial S_i} + \frac{\partial g}{\partial \beta}\frac{\partial \beta}{\partial S_i}du = \int_0^x \frac{\partial g}{\partial \alpha}(u;\alpha,\beta)du\frac{\partial \alpha}{\partial S_i} + \int_0^x \frac{\partial g}{\partial \beta}(u;\alpha,\beta)du\frac{\partial \beta}{\partial S_i}
\end{aligned}$$

with

$$\int_0^x \frac{\partial g}{\partial \beta}(u;\alpha,\beta)du = -G(x;\alpha,\beta)\frac{\alpha}{\beta} + G(x;\alpha+1,\beta)\frac{\Gamma(\alpha+1)}{\Gamma(\alpha)\beta},$$

$$\int_0^x \frac{\partial g}{\partial \alpha}(u;\alpha,\beta)du = \int_0^x \ln(u)g(u;\alpha,\beta)du + G(x;\alpha,\beta)\left(\ln\left(\frac{1}{\beta}\right) - \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}\right).$$

Note that $\frac{\partial \alpha}{\partial S_i}$ and $\frac{\partial \beta}{\partial S_i}$ have been calculated previously.

The next Greek to be computed is the Cega.

$$\begin{aligned}\text{Cega}_{ij} = \frac{\partial P}{\partial \rho_{ij}} &= e^{-rT} M \frac{\partial G}{\partial \rho_{ij}} \left(\frac{1}{K}; \alpha - 1, \beta\right) \\ &\quad - e^{-rT} K \frac{\partial G}{\partial \rho_{ij}} \left(\frac{1}{K}; \alpha, \beta\right).\end{aligned} \tag{4.28}$$

The reasoning for calculating the terms $\frac{\partial G(x;\alpha-1,\beta)}{\partial \rho_{ij}}$ and $\frac{\partial G(x;\alpha,\beta)}{\partial \rho_{ij}}$ is similar to the calculation of Δ, so the only terms that need to be calculated are $\frac{\partial \alpha}{\partial \rho_{ij}}$ and $\frac{\partial \beta}{\partial \rho_{ij}}$, since the others have already been demonstrated. This signifies that

$$\begin{aligned}\frac{\partial \alpha}{\partial \rho_{ij}} &= \frac{\frac{\partial V^2}{\partial \rho_{ij}}(V^2 - M^2) - V^2 \frac{\partial V^2}{\partial \rho_{ij}}}{(V^2 - M^2)^2} \\ &= -\frac{M^2 \frac{\partial V^2}{\partial \rho_{ij}}}{(V^2 - M^2)^2} \\ &= -\frac{M^2}{(V^2 - M^2)^2} \left(2 w_i w_j S_i(0) S_j(0) e^{2rT} \sigma_i \sigma_j T e^{\sigma_i \sigma_j \rho_{ij} T}\right),\end{aligned}$$

$$\frac{\partial \beta}{\partial \rho_{ij}} = M \left(\frac{2 w_i w_j S_i(0) S_j(0) e^{2rT} \sigma_i \sigma_j T e^{\sigma_i \sigma_j \rho_{ij} T}}{(V^2)^2}\right).$$

The last Greek to be calculated is ν.

$$\begin{aligned}\nu_i = \frac{\partial P}{\partial \sigma_i} &= e^{-rT} M \frac{\partial G}{\partial \sigma_i} \left(\frac{1}{K}; \alpha - 1, \beta\right) \\ &\quad - e^{-rT} K \frac{\partial G}{\partial \sigma_i} \left(\frac{1}{K}; \alpha, \beta\right).\end{aligned} \tag{4.29}$$

Analogous to the above, it is only necessary to calculate $\frac{\partial \alpha}{\partial \sigma_i}$ and $\frac{\partial \beta}{\partial \sigma_i}$, resulting in

$$\begin{aligned}\frac{\partial \alpha}{\partial \sigma_i} &= \frac{\frac{\partial V^2}{\partial \sigma_i}(V^2 - M^2) - V^2 \frac{\partial V^2}{\partial \sigma_i}}{(V^2 - M^2)^2} \\ &= -\frac{M^2 \frac{\partial V^2}{\partial \sigma_i}}{(V^2 - M^2)^2} \\ &= -\frac{M^2 \left(2 \sum_{j \neq i, j=1}^{n} w_j w_i S_i(0) S_j(0) e^{2rT} \sigma_j \rho_{ij} T e^{\sigma_i \sigma_j \rho_{ij} T} + 2 w_i^2 S_i(0)^2 e^{2rT} \sigma_i T e^{\sigma_i^2 T}\right)}{(V^2 - M^2)^2},\end{aligned}$$

$$\frac{\partial \beta}{\partial \sigma_i} = M \left(\frac{2 \sum_{j \neq i, j=1}^{n} w_j w_i S_i(0) S_j(0) e^{2rT} \sigma_j \rho_{ij} T e^{\sigma_i \sigma_j \rho_{ij} T} + 2 w_i^2 S_i(0)^2 e^{2rT} \sigma_i T e^{\sigma_i^2 T}}{(V^2)^2}\right).$$

4.3 Outperformance Options

The outperformance option measures the relative performance of one stock over another, whether up or down, and pays the difference.

Suppose there are 2 assets with prices equal to S_1 and S_2, then the payoff of an outperformance option at maturity T is equal to

$$\text{OUT}_{\text{payoff}} = \max[S_1 - S_2, 0].$$

4.3.1 The Pricing of an Outperformance Option

In this subsection, the outperformance option pricing formula is illustrated. The following assumptions are made:

1. The following stochastic differential equations describe the asset prices:

$$dS_1(t) = \mu_1 S_1(t)dt + \sigma_1 S_1(t)dW_1(t),$$

$$dS_2(t) = \mu_2 S_2(t)dt + \sigma_2 S_2(t)dW_2(t),$$

   where σ_1^2 and σ_2^2 are constant and are known as the instantaneous variances of the asset returns. The instantaneous expected rate of return, represented by μ_1 and μ_2, are also constant. The W_1 and W_2 are the Brownian motion processes and are correlated with each other, the coefficient of correlation between them is ρ_{12} and is constant over time.

2. The hedging portfolio, the value of which is P, satisfies the following partial differential equation:

$$\frac{\partial P}{\partial t} + \frac{1}{2}\left(\frac{\partial^2 P}{\partial (S_1)^2}\sigma_1^2 S_1^2 + 2\frac{\partial^2 P}{\partial S_1 \partial S_2}\sigma_1\sigma_2\rho_{12}S_1S_2 + \frac{\partial^2 P}{\partial (S_2)^2}\sigma_2^2 S_2^2\right) = 0 \tag{4.30}$$

   with a boundary condition equal to

$$0 \leq \text{OUT}_{\text{price}} \leq S_1$$

   and an initial condition equal to

$$P(S_1, S_2) = \max[S_1 - S_2, 0].$$

It can be observed that in order to obtain the previous initial condition, known as the payoff of this option, a change in the probability measure must be made in order to end up in a case similar to that of the Black & Scholes model for a simple option. This is possible because an outperformance option is considered to be an exchange option. The article by German, Karoui and Rochet [8] discusses this topic in detail and can be demonstrated that the price of this option at time 0 is equal to

$$\frac{P(0)}{S_2(0)} = \mathbb{E}_{Q_{S_2}}\left[\left(\frac{S_1(T)}{S_2(T)} - 1\right)^+\right].$$

Using these assumptions and this observation, it is possible to arrive at the option pricing formula that satisfies the Equation 4.30. This implies that the pricing formula for an outperformance option is equal to

$$\text{OUT}_{\text{price}} = S_1 N(d_1) - S_2 N(d_2) \tag{4.31}$$

where $N(.)$ is the cumulative standard Normal distribution and

$$d_1 = \frac{\log\left(\frac{S_1}{S_2}\right) + \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \tag{4.32}$$

$$d_2 = d_1 - \sigma\sqrt{T}, \tag{4.33}$$

$$\sigma^2 = \sigma_1^2 - 2\rho_{12}\sigma_1\sigma_2 + \sigma_2^2. \tag{4.34}$$

As discussed in section 2.3, this formula represents a specific case of the more general pricing formula introduced by Veiga, Wystup, and Esquível [26].

4.3.2 Sensitivity Analysis

In this subsection, the Greeks of the outperformance option are calculated. Since the formula for the price of an outperformance option closely resembles that of a call option, this means that some of the Greeks are analogous and so in this case they are not calculated and they are taken from Björk's [2].

The first derivative to be computed is in order to the price of asset 1, which is equal to

$$\Delta_1 = \frac{\partial P}{\partial S_1} = N(d_1). \tag{4.35}$$

As this derivative is quite simple, it can be seen that Δ_1 is positive and has values between 0 and 1, due to the properties of the cumulative function of the standard normal distribution. It also grows as d_1 increases, i.e. it grows as S_1 increases.

Since the derivative of the price of asset 1 was analysed, it is logical to also calculate the derivative of the price of asset 2, which behaves similarly to the derivative of the strike price of a call option. So Δ_2 is equal to

$$\Delta_2 = \frac{\partial P}{\partial S_2} = -N(d_2). \tag{4.36}$$

On the other hand, Δ_2 is negative and takes values between -1 and 0, the opposite of Δ_1. It also increases as d_2 decreases, i.e. it increases as S_2 increases.

The next Greek to be calculated is Θ, the derivative with respect to the maturity date, which is equal to

$$\Theta = \frac{\partial P}{\partial T} = S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial T} - S_2 \times \varphi(d_2) \times \frac{\partial d_2}{\partial T}.$$

From the d_2 expression, it can be immediately verified that

$$\frac{\partial d_2}{\partial T} = \frac{\partial d_1}{\partial T} - \sigma \frac{1}{2} \frac{1}{\sqrt{T}}.$$

Let's try to simplify the Greek expression. The term $S_2\varphi(d_2)$ is equal to

$$\begin{aligned} S_2\varphi(d_2) &= S_2 \frac{e^{-\frac{(d_1-\sigma\sqrt{T})^2}{2}}}{\sqrt{2\pi}} = e^{d_1\sigma\sqrt{T}-\frac{\sigma^2 T}{2}} S_2\varphi(d_1) \\ &= e^{\log\left(\frac{S_1}{S_2}\right)+\frac{1}{2}\sigma^2 T-\frac{\sigma^2 T}{2}} S_2\varphi(d_1) = e^{\log\left(\frac{S_1}{S_2}\right)} S_2\varphi(d_1) \\ &= \frac{S_1}{S_2} S_2\varphi(d_1) = S_1\varphi(d_1). \end{aligned}$$

That means,

$$\begin{aligned} \Theta &= S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial T} - S_1 \times \varphi(d_1) \times \left(\frac{\partial d_1}{\partial T} - \sigma \frac{1}{2} \frac{1}{\sqrt{T}}\right) \\ &= S_1\varphi(d_1) \frac{\sigma}{2\sqrt{T}}. \end{aligned} \tag{4.37}$$

Next, the derivative of the correlation variable is computed. This is equal to

$$\begin{aligned} \text{Cega} = \frac{\partial P}{\partial \rho_{12}} &= S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial \rho_{12}} - S_2 \times \varphi(d_2) \times \frac{\partial d_2}{\partial \rho_{12}} \\ &= S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial \rho_{12}} - S_1 \times \varphi(d_1) \times \left(\frac{\partial d_1}{\partial \rho_{12}} - \sqrt{T} \frac{\partial \sigma}{\partial \rho_{12}}\right) \\ &= S_1\varphi(d_1)\sqrt{T} \frac{\partial \sigma}{\partial \rho_{12}}. \end{aligned}$$

To complete the calculation of this derivative, let's determine what $\frac{\partial \sigma}{\partial \rho_{12}}$ is equal

$$\frac{\partial \sigma}{\partial \rho_{12}} = \frac{1}{2} \times \frac{1}{\sigma} \times (-2\sigma_1\sigma_2) = -\frac{\sigma_1\sigma_2}{\sigma}.$$

So,

$$\text{Cega} = -S_1\varphi(d_1) \frac{\sigma_1\sigma_2\sqrt{T}}{\sigma}. \tag{4.38}$$

The next derivative to be calculated is the derivative in order to the volatility of asset 1, which has the following expression

$$\begin{aligned} \nu_1 = \frac{\partial P}{\partial \sigma_1} &= S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial \sigma_1} - S_2 \times \varphi(d_2) \times \frac{\partial d_2}{\partial \sigma_1} \\ &= S_1 \times \varphi(d_1) \times \frac{\partial d_1}{\partial \sigma_1} - S_1 \times \varphi(d_1) \times \left(\frac{\partial d_1}{\partial \sigma_1} - \sqrt{T} \frac{\partial \sigma}{\partial \sigma_1}\right) \\ &= S_1\varphi(d_1)\sqrt{T} \frac{\partial \sigma}{\partial \sigma_1}. \end{aligned}$$

In order to finish calculating the derivative, $\frac{\partial\sigma}{\partial\sigma_1}$ is deduced

$$\frac{\partial\sigma}{\partial\sigma_1} = \frac{1}{2} \times \frac{1}{\sigma} \times (2\sigma_1 - 2\rho\sigma_2) = \frac{\sigma_1 - \rho\sigma_2}{\sigma}.$$

Thus, ν_1 is equal to

$$\nu_1 = S_1\varphi(d_1)\frac{(\sigma_1 - \rho\sigma_2)\sqrt{T}}{\sigma}. \tag{4.39}$$

By similar reasoning, it follows that ν_2 is equal to

$$\nu_2 = S_1\varphi(d_1)\frac{(\sigma_2 - \rho\sigma_1)\sqrt{T}}{\sigma}. \tag{4.40}$$

4.4 Best and Worst of Options

A best of option, also known as a maximum option, is, as the name suggests, an option on the best performer in a basket of assets. A worst of option, also known as a minimum option, is, as the name implies, an option on the worst performer in a basket of assets.

Suppose that given n assets with prices equal to $S_1, S_2, \ldots, S_n$ then a best of call option has a payoff at maturity T given by

$$\text{BO Call}_{\text{payoff}} = \max[\max(S_1(T), S_2(T), \ldots, S_n(T)) - K, 0],$$

a best of put option has a payoff at maturity T given by

$$\text{BO Put}_{\text{payoff}} = \max[K - \max(S_1(T), S_2(T), \ldots, S_n(T)), 0],$$

a worst of call option has a payoff at maturity T given by

$$\text{WO Call}_{\text{payoff}} = \max[\min(S_1(T), S_2(T), \ldots, S_n(T)) - K, 0],$$

and a worst of put option has a payoff at maturity T given by

$$\text{WO Put}_{\text{payoff}} = \max[K - \min(S_1(T), S_2(T), \ldots, S_n(T)), 0]$$

where K is the strike price.

4.4.1 The Pricing of a Worst of Call Option

In this subsection, the formula for the price of a worst of call option on n assets is presented. The following assumptions are made:

1. Asset prices are described by the following stochastic differential equation:

$$dS_i(t) = \mu_i S_i(t)dt + \sigma_i S_i(t)dW_i(t) \tag{4.41}$$

   where σ_i^2 are constant and are known as the instantaneous variances of the asset returns. The instantaneous expected rate of return, represented by μ_i, are also constant. The W_i are the Brownian motion processes and are correlated with each other, the coefficient of correlation between them is constant over time.

2. The instantaneous interest rate r is constant over time.

3. The replicating portfolio whose value is P satisfies the following partial differential equation:

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_i\sigma_j\rho_{ij}S_iS_j\frac{\partial^2 P}{\partial S_i\,\partial S_j} + r\sum_{i=1}^{n}S_i\frac{\partial P}{\partial S_i} - rP = 0 \tag{4.42}$$

with a boundary condition equal to

$$P(S_1, S_2, \ldots, S_n, 0) = \max[\min(S_1(T), S_2(T), \ldots, S_n(T)) - K, 0]. \tag{4.43}$$

With all these assumptions, it is therefore possible to obtain a formula for the price of this option that satisfies the Equation 4.42. Thus, using the Cox and Ross [6] strategy proposed by Johnson [11], the formula for the price of a worst of call option on n assets is obtained and equal to

$$\begin{aligned}
\text{WO Call}_n = {} & S_1N_n\left(d_1\left(S_1, K, \sigma_1^2\right), -d_1'\left(S_1, S_2, \sigma_{12}^2\right), \ldots,\right. \\
& \left. -d_1'\left(S_1, S_n, \sigma_{1n}^2\right), -\rho_{112}, -\rho_{113}, \ldots, \rho_{123}, \ldots\right) \\
& + S_2N_n\left(d_1\left(S_2, K, \sigma_2^2\right), -d_1'\left(S_2, S_1, \sigma_{12}^2\right), \ldots,\right. \\
& \left. -d_1'\left(S_2, S_n, \sigma_{2n}^2\right), -\rho_{212}, -\rho_{223}, \ldots, \rho_{213}, \ldots\right) \\
& + \ldots \\
& + S_nN_n\left(d_1\left(S_n, K, \sigma_n^2\right), -d_1'\left(S_n, S_1, \sigma_{1n}^2\right), \ldots,\right. \\
& \left. -d_1'\left(S_n, S_{n-1}, \sigma_{n(n-1)}^2\right), -\rho_{n1n}, -\rho_{n2n}, \ldots, \rho_{n12}, \ldots\right) \\
& - Ke^{-rT}N_n\left(d_2\left(S_1, K, \sigma_1^2\right), d_2\left(S_2, K, \sigma_2^2\right), \ldots,\right. \\
& \left. d_2\left(S_n, K, \sigma_n^2\right), \rho_{12}, \rho_{13}, \ldots\right),
\end{aligned} \tag{4.44}$$

where d_1 is equal to Equation 4.2,

$$d_1'\left(S_i, S_j, \sigma_{ij}^2\right) = \frac{\log\left(\frac{S_i}{S_j}\right) + \frac{1}{2}\sigma_{ij}^2T}{\sigma_{ij}\sqrt{T}},$$

$$\rho_{iij} = \frac{\sigma_i - \rho_{ij}\sigma_j}{\sigma_{ij}},$$

and

$$\rho_{ijk} = \frac{\sigma_i^2 - \rho_{ij}\sigma_i\sigma_j - \rho_{ik}\sigma_i\sigma_k + \rho_{jk}\sigma_j\sigma_k}{\sigma_{ij}\sigma_{ik}}.$$

As the previous formula is complex, this thesis studies this option for the case where $n = 2$. Thus, the formula for the worst of call option on two assets is equal to

$$\begin{aligned}\text{WO Call}_{\text{price}} = S_1 N_2\left(\gamma_1 + \sigma_1\sqrt{T}, \frac{\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \frac{\rho_{12}\sigma_2 - \sigma_1}{\sigma}\right) \\ + S_2 N_2\left(\gamma_2 + \sigma_2\sqrt{T}, \frac{\log\left(\frac{S_1}{S_2}\right) - \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \frac{\rho_{12}\sigma_1 - \sigma_2}{\sigma}\right) \\ - Ke^{-rT} N_2(\gamma_1, \gamma_2, \rho_{12})\end{aligned} \tag{4.45}$$

where $N_2(\dots)$ is the bivariate cumulative standard normal distribution, ρ_{12} is the correlation between the two underlying assets, σ_1 and σ_2 are the respective volatilities of S_1 and S_2, and

$$\gamma_1 = \frac{\log\left(\frac{S_1}{K}\right) + \left(r - \frac{1}{2}\sigma_1^2\right)T}{\sigma_1\sqrt{T}}, \tag{4.46}$$

$$\gamma_2 = \frac{\log\left(\frac{S_2}{K}\right) + \left(r - \frac{1}{2}\sigma_2^2\right)T}{\sigma_2\sqrt{T}}, \tag{4.47}$$

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 - 2\rho_{12}\sigma_1\sigma_2. \tag{4.48}$$

When $K = 0$ the Equation 4.45 is simplified and equal to

$$\text{WO Call}_{\text{price}}(0) = S_2 - S_2 N(d_1) + S_1(d_2) \tag{4.49}$$

where

$$d_1 = \frac{\log\left(\frac{S_2}{S_1}\right) + \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}, \quad \sigma^2 \text{ is equal to Equation 4.48.}$$

To demonstrate the Equation 4.49, two approaches can be employed. The first is to set K to tend to 0 in Equation 4.45, and after some manipulations the desired result is achieved. The second approach is to see that $\max[\min(S_1, S_2) - 0, 0] = \min(S_1, S_2)$ and that $\min(S_1, S_2) = S_2 - \max(S_2 - S_1, 0)$, then note that the expression $\max(S_2 - S_1, 0)$ corresponds to the payoff of an outperformance option and thus arrive at the expected outcome.

As noted in section 2.1, the formulas presented in this section are specific cases of the pricing formula introduced by Veiga, Wystup, and Esquível [26].

4.4.2 Parity Relationships

As some of these options do not have an explicit price formula deduced, the aim of this subsection is to deduce parity relationships, similary to those for simple options (see Equation 4.4), between these options and options for which an explicit formula is already known.

Although the best of call option on two assets has an explicit formula in Johnson's article [11], a parity relationship exists between this option and a worst of call option on two assets. This parity relationship is found in Stulz [25] and is as follows:

$$\text{BO Call}_{\text{price}}(K) = C(S_1, K, T) + C(S_2, K, T) - \text{WO Call}_{\text{price}}(K) \tag{4.50}$$

which means that a best of call on two assets is equal to the sum of two standard call options on the two assets minus a worst of call option on two assets. It should be emphasised that the left and right sides of the equation are equivalent, regardless of the position of S_1 and S_2 in relation to each other and to the strike price K.

To verify this result, consider the following two investments:

- **Portfolio A**: Buy a call on the maximum of S_1 and S_2 struck at K.
- **Portfolio B**: Buy a call option on asset S_1 with strike price K, buy another call option with the same strike price but now on the asset S_2 and sell a call on the minimum of S_1 and S_2 struck at K.

If $\max(S_1, S_2) = S_1 > K$, there are two cases to examine:

- The first is when $S_2 > K$, then Portfolio A pays

  $$S_1 - K$$

  whereas Portfolio B pays

  $$S_1 - K + S_2 - K - S_2 + K = S_1 - K.$$

- The second is when $S_2 < K$, then Portfolio A pays

  $$S_1 - K$$

  and Portfolio B pays

  $$S_1 - K + 0 - \max[0, S_2 - K] = S_1 - K + 0 - 0 = S_1 - K.$$

The case $\max(S_1, S_2) = S_2 > K$, follows a similar logic as before, so A and B are equivalent at maturity. In the case where $\max(S_1, S_2) < K$, the prices of both assets are less than K and therefore Portfolio A is worth 0 and Portfolio B is also worth 0, as both portfolios consist of call options. For it to be arbitrage free[1], at any date $t < T$, both portfolios must also have the same value, making Equation 4.50 true.

With this equality, the price of a best of call on two assets can be calculated, since the formulas for simple call options (Equation 4.1) and for the worst of call on two assets (Equation 4.45) are known.

[1]Means that there is no possibility of a sure profit with zero investment [28], i.e two assets that have the same value on the maturity date must have the same value on any date.

Johnson's article [11] also gives the pricing formula for a best of call option on n assets, but since there is a parity relationship when $n = 2$, the goal is to find one for a generic n. After a few attempts, the following parity relationship was discovered:

$$\begin{aligned}\text{BO Call}(S_{1,t}, S_{2,t}, ..., S_{n,t}) &= \sum_{i=1}^{n} \text{WO Call}(S_{i,t}) - \sum_{\substack{i,j=1 \\ i<j}}^{n} \text{WO Call}(S_{i,t}, S_{j,t}) \\ &+ \sum_{\substack{i,j,k=1 \\ i<j<k}}^{n} \text{WO Call}(S_{i,t}, S_{j,t}, S_{k,t}) - \sum_{\substack{i,j,k,l=1 \\ i<j<k<l}}^{n} \text{WO Call}(S_{i,t}, S_{j,t}, S_{k,t}, S_{l,t}) \\ &+ ... + (-1)^{n-1}\text{WO Call}(S_{1,t}, S_{2,t}, ..., S_{n,t}).\end{aligned} \tag{4.51}$$

To verify this result, consider, in full generality, that $S_{1,T} \leq S_{2,T}, ..., \leq S_{n,T}$ at maturity date $t = T$. Removing the date subscript for simplicity, Equation 4.51 can be rewritten as:

$$\begin{aligned}&\text{BO Call}(S_1, S_2, ..., S_n) = \\ &\sum_{i=1}^{n} \text{WO Call}(S_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{WO Call}(S_i, S_j) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \text{WO Call}(S_i, S_j, S_k) \\ &- \sum_{i=1}^{n-3} \sum_{j=i+1}^{n-2} \sum_{k=j+1}^{n-1} \sum_{l=k+1}^{n} \text{WO Call}(S_i, S_j, S_k, S_l) + ... + (-1)^{n-1}\text{WO Call}(S_1, ..., S_n) \\ &= \sum_{i=1}^{n} \text{WO Call}(S_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{WO Call}(S_i) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \text{WO Call}(S_i) \\ &- \sum_{i=1}^{n-3} \sum_{j=i+1}^{n-2} \sum_{k=j+1}^{n-1} \sum_{l=k+1}^{n} \text{WO Call}(S_i) + ... + (-1)^{n-1}\text{WO Call}(S_1)\end{aligned}$$

Note that, given $S_1 \leq S_2, ..., \leq S_n$, the left side of the equation simplifies to

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

Regarding the right side, each term of the form WO Call(S_k), $k = 1, \ldots, n$, appears a number of times given by:

$$N_k = \sum_{j=0}^{n-k} (-1)^j \binom{n-k}{j}$$

however, using the binomial theorem, it can be written as:

$$N_k = \sum_{j=0}^{n-k} (-1)^j \binom{n-k}{j} = \sum_{j=0}^{n-k} \binom{n-k}{j} (-1)^j 1^{(n-k)-j} = (-1+1)^{n-k}, \quad k = 1, 2, \ldots, n-1$$

obtaining $N_k = 0, k = 1, \ldots, n-1$, and $N_n = 1$, which means that all terms are canceled except for WO Call(S_n), which appears only once in the first sum. Next, it is sufficient to write

$$\text{WO Call}(S_n) = \max(\min(S_n) - K, 0) = \max(S_n - K, 0)$$

to conclude that the left and right side portfolios of Equation 4.51 are equal at maturity. For the prices to be arbitrage free[1], both portfolios must also have the same value at any time $t < T$, so Equation 4.51 is true.

The next parity relationship to check is between the worst of put and the worst of call for the general case, i.e. when there are n assets. This is equal to:

$$\text{WO Put}_{\text{price}}(K) = e^{-rT}K - \text{WO Call}_{\text{price}}(0) + \text{WO Call}_{\text{price}}(K). \tag{4.52}$$

To get the expression for $\text{WO Call}_{\text{price}}(0)$, when the option is on n assets, is enough to make K tend to 0 in Equation 4.44. If the option is on two assets, $\text{WO Call}_{\text{price}}(0)$ is equal to Equation 4.49.

To verify this result (Equation 4.52), contemplate the following two investments:

- **Portfolio A**: Buy a put on the minimum of n assets with a strike price of K.
- **Portfolio B**: Buy a bond that pays K at maturity T, sell a call on the minimum of n assets with a strike price of 0 and buy a call on the minimum of n assets with a strike price of K.

Let's consider that $\min(S_1, S_2, \ldots, S_n) = S_i$ for $i = 1, 2, \ldots, n$. Therefore, only two cases need to be analysed, since for any S_j, with $j \neq i$, less than or greater than K, the parity expression only depends on the worst performing asset, i.e. S_i. The two cases are:

- First case is when $S_i < K$, then Portfolio A pays

  $$K - S_i$$

  whereas Portfolio B pays

  $$K - S_i + 0 = K - S_i.$$

- Second case is when $S_i > K$, then Portfolio A pays 0 whereas Portfolio B pays

  $$K - S_i + (S_i - K) = 0.$$

So A and B are equivalent at maturity. For it to be arbitrage free[1], both portfolios must have the same value at any time $t < T$, so Equation 4.52 is true.

The last parity relationship to verify is between the best of put and the best of call for the general case, i.e. when there are n assets. This is equal to:

$$\text{BO Put}_{\text{price}}(K) = e^{-rT}K - \text{BO Call}_{\text{price}}(0) + \text{BO Call}_{\text{price}}(K). \tag{4.53}$$

To get the expression for $\text{BO Call}_{\text{price}}(0)$, when the option is on n assets, is enough to make K tend to 0 in Equation 4.51. If the option is on two assets, $\text{BO Call}_{\text{price}}(0)$ is equal to

$$\text{BO Call}_{\text{price}}(0) = S_1 + S_2 - \text{WO Call}_{\text{price}}(0).$$

To verify this result (Equation 4.53), consider the following two investments:

- **Portfolio A**: Buy a put on the maximum of n assets with a strike price of K.
- **Portfolio B**: Buy a bond that pays K at maturity T, sell a call on the maximum of n assets with a strike price of 0 and buy a call on the maximum of n assets with a strike price of K.

Suppose that $\max(S_1, S_2, \ldots, S_n) = S_i$ for $i = 1, 2, \ldots, n$. Therefore, only two cases need to be analysed, since for any S_j, with $j \neq i$, less than or greater than K, the parity expression only depends on the best performing asset, i.e. S_i. The two cases are:

- First case is when $S_i > K$, then Portfolio A pays 0 whereas Portfolio B pays

$$K - S_i + (S_i - K) = 0.$$

- Second case is when $S_i < K$, then Portfolio A pays

$$K - S_i$$

whereas Portfolio B pays

$$K - S_i + 0 = K - S_i.$$

So A and B are equivalent at maturity. For it to be arbitrage free[1], both portfolios must also have the same value at any time $t < T$, so Equation 4.53 is true. Note that from the parity relationships of Equation 4.50 and Equation 4.53, it is possible to establish a relationship between the best and worst of put options.

Therefore, using these parity relationships, the price and the Greeks of each of these options can be calculated.

4.4.3 Sensitivity Analysis

Throughout this subsection, the Greeks of the worst option on two assets are calculated, in other words, the derivatives of the price function with respect to the various variables it depends on are computed.

First, let's define some of the notation that is used when writing the derivatives. So:

$$\begin{aligned}
\alpha_1 &= \gamma_1 + \sigma_1\sqrt{T}, \quad \alpha_2 = \frac{\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \\
\beta_1 &= \gamma_2 + \sigma_2\sqrt{T}, \quad \beta_2 = \frac{\log\left(\frac{S_1}{S_2}\right) - \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \\
\rho_c &= \frac{\rho_{12}\sigma_2 - \sigma_1}{\sigma}, \quad \rho_C = \frac{\rho_{12}\sigma_1 - \sigma_2}{\sigma}.
\end{aligned}$$

With this new notation it is possible to write the option price formula of the worst of call option (Equation 4.45) in a simplified way:

$$\text{WO Call}_{\text{price}} = S_1 N_2\left(\alpha_1, \alpha_2, \rho_c\right) + S_2 N_2\left(\beta_1, \beta_2, \rho_C\right) - K e^{-rT} N_2(\gamma_1, \gamma_2, \rho_{12}) \tag{4.54}$$

The first three derivatives to be presented were already calculated in the article by Stulz [25]. These derivatives are the interest rate derivative, the strike price derivative and the correlation between the two assets derivative, respectively. So these derivatives are equal to:

$$\rho = \frac{\partial P}{\partial r} = K \times T \times e^{-rT} N_2(\gamma_1, \gamma_2, \rho_{12}), \tag{4.55}$$

$$\frac{\partial P}{\partial K} = -e^{-rT} N_2(\gamma_1, \gamma_2, \rho_{12}), \tag{4.56}$$

$$\text{Cega} = \frac{\partial P}{\partial \rho} = S_2 N\left(\frac{\beta_1 - \rho_C \beta_2}{\sqrt{1-\rho_C^2}}\right) \frac{e^{-\frac{1}{2}\beta_2^2}}{\sqrt{2\pi}} \frac{\sigma_1 \sigma_2 \sqrt{T}}{\sigma}. \tag{4.57}$$

From now on, the calculations made to obtain the derivatives of the other variables are demonstrated. The first derivative to look at is the derivative in order to the maturity, which is equal to:

$$\begin{aligned} \Theta = \frac{\partial P}{\partial T} = S_1 \left(\frac{\partial N_2}{\partial \alpha_1}\frac{\partial \alpha_1}{\partial T} + \frac{\partial N_2}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial T}\right) + S_2 \left(\frac{\partial N_2}{\partial \beta_1}\frac{\partial \beta_1}{\partial T} + \frac{\partial N_2}{\partial \beta_2}\frac{\partial \beta_2}{\partial T}\right) \\ + K r e^{-rT} N_2(\gamma_1, \gamma_2, \rho) - K e^{-rT}\left(\frac{\partial N_2}{\partial \gamma_1}\frac{\partial \gamma_1}{\partial T} + \frac{\partial N_2}{\partial \gamma_2}\frac{\partial \gamma_2}{\partial T}\right). \end{aligned} \tag{4.58}$$

There are several partial derivatives in the Equation 4.58 that need to be calculated. So let's start by calculating the partial derivatives of the alphas, the betas and the gammas in order to the maturity:

$$\begin{aligned} \frac{\partial \gamma_1}{\partial T} &= \frac{\sqrt{T}\left(r - \frac{1}{2}\sigma_1^2\right) - \frac{1}{2\sqrt{T}}\left(\log\left(\frac{S_1}{K}\right) + \left(r - \frac{1}{2}\sigma_1^2\right)T\right)}{\sigma_1 T} \\ &= \frac{-\frac{\log\left(\frac{S_1}{K}\right)}{2\sqrt{T}} + \sqrt{T}\left(r - \frac{1}{2}\sigma_1^2\right) - \frac{\sqrt{T}}{2}\left(r - \frac{1}{2}\sigma_1^2\right)}{\sigma_1 T} \\ &= \frac{\log\left(\frac{K}{S_1}\right) + \left(r - \frac{1}{2}\sigma_1^2\right)T}{2\sigma_1 T\sqrt{T}}. \end{aligned}$$

The partial derivative $\frac{\partial \gamma_2}{\partial T}$ has a similar reasoning, since the two variables are almost equal, which means that

$$\frac{\partial \gamma_2}{\partial T} = \frac{\log\left(\frac{K}{S_2}\right) + \left(r - \frac{1}{2}\sigma_2^2\right)T}{2\sigma_2 T\sqrt{T}}.$$

Next,

$$\begin{aligned}\frac{\partial \alpha_1}{\partial T} &= \frac{\partial \gamma_1}{\partial T} + \sigma_1 \frac{1}{2\sqrt{T}} \\ &= \frac{\log\left(\frac{K}{S_1}\right) + \left(r - \frac{1}{2}\sigma_1^2\right) T + \sigma_1^2 T}{2\sigma_1 T\sqrt{T}} \\ &= \frac{\log\left(\frac{K}{S_1}\right) + \left(r + \frac{1}{2}\sigma_1^2\right) T}{2\sigma_1 T\sqrt{T}}.\end{aligned}$$

The partial derivative $\frac{\partial \beta_1}{\partial T}$ has an analogous logic, due to the two variables being almost identical, which means that

$$\frac{\partial \beta_1}{\partial T} = \frac{\log\left(\frac{K}{S_2}\right) + \left(r + \frac{1}{2}\sigma_2^2\right) T}{2\sigma_2 T\sqrt{T}}.$$

Then,

$$\begin{aligned}\frac{\partial \alpha_2}{\partial T} &= \frac{\sqrt{T}\left(-\frac{1}{2}\sigma^2\right) - \frac{1}{2\sqrt{T}}\left(\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T\right)}{\sigma T} \\ &= \frac{-\frac{\log\left(\frac{S_2}{S_1}\right)}{2\sqrt{T}} + \frac{1}{4}\sigma^2\sqrt{T} - \frac{1}{2}\sigma^2\sqrt{T}}{\sigma T} \\ &= \frac{\log\left(\frac{S_1}{S_2}\right) - \frac{1}{2}\sigma^2 T}{2\sigma T\sqrt{T}}.\end{aligned}$$

The partial derivative $\frac{\partial \beta_2}{\partial T}$ can be understood similarly, given that the two variables are nearly the same, which implies that

$$\frac{\partial \beta_2}{\partial T} = \frac{\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T}{2\sigma T\sqrt{T}}.$$

Now let's study the partial derivatives of the bivariate cumulative standard normal distribution in order to the alphas, the betas and the gammas. Looking at the partial derivative of $\frac{\partial N_2}{\partial \alpha_1}$

$$\begin{aligned}\frac{\partial N_2}{\partial \alpha_1} &= \frac{\partial}{\partial \alpha_1}\left(\int_{-\infty}^{\alpha_1}\int_{-\infty}^{\alpha_2} f(x, y; \rho_c)\, dy\, dx\right) \\ &= \int_{-\infty}^{\alpha_2} \frac{\partial}{\partial \alpha_1}\left(\int_{-\infty}^{\alpha_1} f(x, y; \rho_c)\, dx\right) dy \\ &= \int_{-\infty}^{\alpha_2} f(\alpha_1, y; \rho_c)\, dy \\ &= \int_{-\infty}^{\alpha_2} \frac{1}{2\pi\sqrt{1-\rho_c^2}} e^{-\frac{\alpha_1^2 - 2\rho_c \alpha_1 y + y^2}{2(1-\rho_c^2)}}\, dy \\ &= e^{-\frac{\alpha_1^2}{2}} \int_{-\infty}^{\alpha_2} \frac{1}{2\pi\sqrt{1-\rho_c^2}} e^{-\frac{(y-\rho_c \alpha_1)^2}{2(1-\rho_c^2)}}\, dy\end{aligned}$$

with $f(x, y; \rho_c)$ being the bivariate normal density function.

To solve the integral, the following change of variable is necessary

$$\frac{y - \rho_c \alpha_1}{\sqrt{1 - \rho_c^2}} = u \Leftrightarrow y = \sqrt{1 - \rho_c^2} u + \rho_c \alpha_1$$

as a result,

$$\frac{\partial N_2}{\partial \alpha_1} = \frac{e^{-\frac{\alpha_1^2}{2}}}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\alpha_2 - \rho_c \alpha_1}{\sqrt{1-\rho_c^2}}} \frac{1}{\sqrt{2\pi}\sqrt{1-\rho_c^2}} e^{-\frac{u^2}{2}} \sqrt{1-\rho_c^2}\, du = \frac{e^{-\frac{\alpha_1^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\alpha_2 - \rho_c \alpha_1}{\sqrt{1-\rho_c^2}}\right).$$

The idea for the other bivariate normals is very similar to what was done earlier. Therefore, the objective is to use pairs of greek letters with each other, i.e. α_1 with α_2, β_1 with β_2 and γ_1 with γ_2, since each bivariate normal is a function of each of these pairs. In addition, the correlation coefficients corresponding to each bivariate normal are taken into account in the expressions for the density functions of the bivariate normals. Finally, regarding the variable to be changed to solve the integral, the variable that is derived is the one that is always multiplied by the correlation coefficient corresponding to each respective bivariate normal. The other partial derivatives are therefore equal to:

$$\frac{\partial N_2}{\partial \alpha_2} = \frac{e^{-\frac{\alpha_2^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\alpha_1 - \rho_c \alpha_2}{\sqrt{1-\rho_c^2}}\right),$$

$$\frac{\partial N_2}{\partial \beta_1} = \frac{e^{-\frac{\beta_1^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\beta_2 - \rho_C \beta_1}{\sqrt{1-\rho_C^2}}\right),$$

$$\frac{\partial N_2}{\partial \beta_2} = \frac{e^{-\frac{\beta_2^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\beta_1 - \rho_C \beta_2}{\sqrt{1-\rho_C^2}}\right),$$

$$\frac{\partial N_2}{\partial \gamma_1} = \frac{e^{-\frac{\gamma_1^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\gamma_2 - \rho \gamma_1}{\sqrt{1-\rho^2}}\right),$$

$$\frac{\partial N_2}{\partial \gamma_2} = \frac{e^{-\frac{\gamma_2^2}{2}}}{\sqrt{2\pi}} N\left(\frac{\gamma_1 - \rho \gamma_2}{\sqrt{1-\rho^2}}\right).$$

From now on, only the partial derivatives of the alphas, betas and gammas in order to the variables are computed, since the partial derivatives of the bivariate normals are always the same and have therefore already been calculated.

The next derivative to be determined is the derivative in order to the volatility of the asset 1, which is equal to:

$$\begin{aligned}\nu_1 = \frac{\partial P}{\partial \sigma_1} = S_1 \left(\frac{\partial N_2}{\partial \alpha_1} \frac{\partial \alpha_1}{\partial \sigma_1} + \frac{\partial N_2}{\partial \alpha_2} \frac{\partial \alpha_2}{\partial \sigma_1} \right) + S_2 \frac{\partial N_2}{\partial \beta_2} \frac{\partial \beta_2}{\partial \sigma_1} \\ - K e^{-rT} \frac{\partial N_2}{\partial \gamma_1} \frac{\partial \gamma_1}{\partial \sigma_1}.\end{aligned} \tag{4.59}$$

The partial derivatives in order to σ_1 are equal to

$$\begin{aligned}\frac{\partial \gamma_1}{\partial \sigma_1} &= \frac{\sigma_1 \left(-\frac{1}{2} 2\sigma_1 T\right) - \left(\log\left(\frac{S_1}{K}\right) + \left(r - \frac{1}{2}\sigma_1^2\right) T \right)}{\sigma_1^2 \sqrt{T}} \\ &= \frac{\log\left(\frac{K}{S_1}\right) - \left(r + \frac{1}{2}\sigma_1^2\right) T}{\sigma_1^2 \sqrt{T}},\end{aligned}$$

$$\begin{aligned}\frac{\partial \alpha_1}{\partial \sigma_1} &= \frac{\log\left(\frac{K}{S_1}\right) - \left(r + \frac{1}{2}\sigma_1^2\right) T}{\sigma_1^2 \sqrt{T}} + \sqrt{T} \\ &= \frac{\log\left(\frac{K}{S_1}\right) - \left(r - \frac{1}{2}\sigma_1^2\right) T}{\sigma_1^2 \sqrt{T}},\end{aligned}$$

$$\begin{aligned}\frac{\partial \alpha_2}{\partial \sigma_1} &= \frac{\sigma \left(-\frac{1}{2} T(2\sigma_1 - 2\rho\sigma_2)\right) - \frac{1}{2\sigma}(2\sigma_1 - 2\rho\sigma_2) \left(\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T \right)}{\sigma^2 \sqrt{T}} \\ &= \frac{-\sigma T(\sigma_1 - \rho\sigma_2) - \frac{1}{\sigma}(\sigma_1 - \rho\sigma_2) \left(\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T \right)}{\sigma^2 \sqrt{T}} \\ &= (\sigma_1 - \rho\sigma_2) \left(\frac{\log\left(\frac{S_1}{S_2}\right) - \frac{1}{2}\sigma^2 T}{\sigma^3 \sqrt{T}} \right).\end{aligned}$$

Since β_2 is similar to α_2, so

$$\frac{\partial \beta_2}{\partial \sigma_1} = (\sigma_1 - \rho\sigma_2) \left(\frac{\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T}{\sigma^3 \sqrt{T}} \right).$$

Now that the derivative for the volatility of asset 1 has been calculated, let's look at the derivative for the volatility of asset 2, which is equal to:

$$\nu_2 = \frac{\partial P}{\partial \sigma_2} = S_1 \frac{\partial N_2}{\partial \alpha_2} \frac{\partial \alpha_2}{\partial \sigma_2} + S_2 \left(\frac{\partial N_2}{\partial \beta_1} \frac{\partial \beta_1}{\partial \sigma_2} + \frac{\partial N_2}{\partial \beta_2} \frac{\partial \beta_2}{\partial \sigma_2} \right) - K e^{-rT} \frac{\partial N_2}{\partial \gamma_2} \frac{\partial \gamma_2}{\partial \sigma_2}. \tag{4.60}$$

Due to the fact that the derivatives of the variables in order to σ_1 have already been computed, and they are similar in order to σ_2, the calculations can be simplified, resulting in the outcome that

$$\frac{\partial \alpha_2}{\partial \sigma_2} = (\sigma_2 - \rho\sigma_1)\left(\frac{\log\left(\frac{S_1}{S_2}\right) - \frac{1}{2}\sigma^2 T}{\sigma^3\sqrt{T}}\right),$$

$$\frac{\partial \beta_2}{\partial \sigma_2} = (\sigma_2 - \rho\sigma_1)\left(\frac{\log\left(\frac{S_2}{S_1}\right) - \frac{1}{2}\sigma^2 T}{\sigma^3\sqrt{T}}\right),$$

$$\frac{\partial \gamma_2}{\partial \sigma_2} = \frac{\log\left(\frac{K}{S_2}\right) - \left(r + \frac{1}{2}\sigma_2^2\right)T}{\sigma_2^2\sqrt{T}},$$

$$\frac{\partial \beta_1}{\partial \sigma_2} = \frac{\log\left(\frac{K}{S_2}\right) - \left(r - \frac{1}{2}\sigma_2^2\right)T}{\sigma_2^2\sqrt{T}}.$$

The derivative with respect to the price of asset 1 is now to be determined. Although the article provides this calculation, the presented expression was challenging to interpret, leading to independent computation. The result is:

$$\Delta_1 = \frac{\partial P}{\partial S_1} = N_2(\alpha_1, \alpha_2) + S_1\left(\frac{\partial N_2}{\partial \alpha_1}\frac{\partial \alpha_1}{\partial S_1} + \frac{\partial N_2}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial S_1}\right) + S_2\frac{\partial N_2}{\partial \beta_2}\frac{\partial \beta_2}{\partial S_1} - Ke^{-rT}\frac{\partial N_2}{\partial \gamma_1}\frac{\partial \gamma_1}{\partial S_1}. \tag{4.61}$$

The partial derivatives in order to S_1 are equal to

$$\frac{\partial \alpha_1}{\partial S_1} = \frac{\partial \gamma_1}{\partial S_1} = \frac{1}{\sigma_1\sqrt{T}S_1},$$

$$\frac{\partial \alpha_2}{\partial S_1} = -\frac{1}{\sigma\sqrt{T}S_1},$$

$$\frac{\partial \beta_2}{\partial S_1} = \frac{1}{\sigma\sqrt{T}S_1}.$$

Since the derivative in order to the price of asset 1 has been calculated, the derivative with respect to the price of asset 2 is also considered, and is as follows:

$$\begin{aligned}\Delta_2 = \frac{\partial P}{\partial S_2} &= S_1\frac{\partial N_2}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial S_2} + N_2(\beta_1, \beta_2) \\ &+ S_2\left(\frac{\partial N_2}{\partial \beta_1}\frac{\partial \beta_1}{\partial S_2} + \frac{\partial N_2}{\partial \beta_2}\frac{\partial \beta_2}{\partial S_2}\right) - Ke^{-rT}\frac{\partial N_2}{\partial \gamma_2}\frac{\partial \gamma_2}{\partial S_2}.\end{aligned} \tag{4.62}$$

The partial derivatives in order to S_2 are equal to

$$\frac{\partial \beta_1}{\partial S_2} = \frac{\partial \gamma_2}{\partial S_2} = \frac{1}{\sigma_2\sqrt{T}S_2},$$

$$\frac{\partial \alpha_2}{\partial S_2} = \frac{1}{\sigma\sqrt{T}S_2},$$

$$\frac{\partial \beta_2}{\partial S_2} = -\frac{1}{\sigma\sqrt{T}S_2}.$$

The final partial derivative to be determined is the derivative of Δ_1 with respect to the price of asset 2, which is equal to:

$$\begin{aligned}\frac{\partial \Delta_1}{\partial S_2} = \frac{\partial N_2}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial S_2} + \left(\frac{1}{\sigma_1\sqrt{T}}\frac{\partial \frac{\partial N_2}{\partial \alpha_1}}{\partial S_2} - \frac{1}{\sigma\sqrt{T}}\frac{\partial \frac{\partial N_2}{\partial \alpha_2}}{\partial S_2}\right) \\ + \frac{\partial N_2}{\partial \beta_2}\frac{1}{\sigma\sqrt{T}S_1} + \frac{S_2}{\sigma\sqrt{T}S_1}\frac{\partial \frac{\partial N_2}{\partial \beta_2}}{\partial S_2} - Ke^{-rT}\frac{1}{\sigma_1\sqrt{T}S_1}\frac{\partial \frac{\partial N_2}{\partial \gamma_1}}{\partial S_2}.\end{aligned} \tag{4.63}$$

Many of these partial derivatives have already been calculated, except for the derivatives in order to the price of asset 2 of the bivariate normal derivative in the order to the greek letters. Thus, let's compute them:

$$\frac{\partial \frac{\partial N_2}{\partial \alpha_1}}{\partial S_2} = \frac{1}{\sqrt{2\pi}}e^{-\frac{\alpha_1^2}{2}}\varphi\left(\frac{\alpha_2 - \alpha_1\rho_c}{\sqrt{1-\rho_c^2}}\right)\frac{1}{\sigma\sqrt{T}S_2\sqrt{1-\rho_c^2}},$$

$$\frac{\partial \frac{\partial N_2}{\partial \gamma_1}}{\partial S_2} = \frac{1}{\sqrt{2\pi}}e^{-\frac{\gamma_1^2}{2}}\varphi\left(\frac{\gamma_2 - \gamma_1\rho}{\sqrt{1-\rho^2}}\right)\frac{1}{\sigma_2\sqrt{T}S_2\sqrt{1-\rho^2}},$$

$$\begin{aligned}\frac{\partial \frac{\partial N_2}{\partial \alpha_2}}{\partial S_2} &= \frac{1}{\sqrt{2\pi}}\left(-\frac{1}{2}2\alpha_2\frac{1}{\sigma\sqrt{T}S_2}\right)e^{-\frac{\alpha_2^2}{2}}N\left(\frac{\alpha_1 - \alpha_2\rho_c}{\sqrt{1-\rho_c^2}}\right) \\ &+ \frac{1}{\sqrt{2\pi}}e^{-\frac{\alpha_2^2}{2}}\varphi\left(\frac{\alpha_1 - \alpha_2\rho_c}{\sqrt{1-\rho_c^2}}\right)\frac{-\rho_c}{\sigma\sqrt{T}S_2\sqrt{1-\rho_c^2}} \\ &= \frac{1}{\sigma S_2\sqrt{2\pi T}}e^{-\frac{\alpha_2^2}{2}}\left(-\alpha_2 N\left(\frac{\alpha_1 - \alpha_2\rho_c}{\sqrt{1-\rho_c^2}}\right) - \frac{\rho_c}{\sqrt{1-\rho_c^2}}\varphi\left(\frac{\alpha_1 - \alpha_2\rho_c}{\sqrt{1-\rho_c^2}}\right)\right),\end{aligned}$$

$$\begin{aligned}\frac{\partial \frac{\partial N_2}{\partial \beta_2}}{\partial S_2} &= \frac{1}{\sqrt{2\pi}}\left(-\frac{1}{2}2\beta_2\frac{-1}{\sigma\sqrt{T}S_2}\right)e^{-\frac{\beta_2^2}{2}}N\left(\frac{\beta_1 - \beta_2\rho_C}{\sqrt{1-\rho_C^2}}\right) \\ &+ \frac{1}{\sqrt{2\pi}}e^{-\frac{\beta_2^2}{2}}\varphi\left(\frac{\beta_1 - \beta_2\rho_C}{\sqrt{1-\rho_C^2}}\right)\left(\frac{1}{\sqrt{1-\rho_C^2}}\left(\frac{1}{\sigma_2\sqrt{T}S_2} - \rho_C\frac{-1}{\sigma\sqrt{T}S_2}\right)\right) \\ &= \frac{1}{\sigma S_2\sqrt{2\pi T}}e^{-\frac{\beta_2^2}{2}}\left(\beta_2 N\left(\frac{\beta_1 - \beta_2\rho_C}{\sqrt{1-\rho_C^2}}\right) + \frac{\sigma + \rho_C\sigma_2}{\sigma_2\sqrt{1-\rho_C^2}}\varphi\left(\frac{\beta_1 - \beta_2\rho_C}{\sqrt{1-\rho_C^2}}\right)\right).\end{aligned}$$

With all the derivatives calculated for the worst of call, it is therefore possible to calculate the Greeks for the other options due to the parity relationships shown in subsection 4.4.2.

The deduction starts with the Greeks for the best of call. The Greeks of the simple call option are taken from Björk's book [2].

The first Greek to be computed is ρ, which is the derivative of the interest rate:

$$\rho_{(\text{BO Call})} = K \times Te^{-rT}N(d_2(t, S_1)) + K \times Te^{-rT}N(d_2(t, S_2)) - \rho_{(\text{WO Call})} \tag{4.64}$$

with $d_2(t, S_i)$ given by Equation 4.3.

The derivative of the best of call function with respect to the strike price is equal to

$$\frac{\partial P_{(\text{BO Call})}}{\partial K} = -e^{-rT}N(d_2(t, S_1)) - e^{-rT}N(d_2(t, S_2)) - \frac{\partial P_{(\text{WO Call})}}{\partial K}. \tag{4.65}$$

The next derivative to be calculated is the derivative in order to the correlation, known as Cega, which is equal to

$$\text{Cega}_{(\text{BO Call})} = -\text{Cega}_{(\text{WO Call})} \tag{4.66}$$

since there is no correlation term on simple options, there is no derivative on it.

The derivative in order to the maturity date is as follows

$$\begin{aligned}\Theta_{(\text{BO Call})} = &\frac{S_1\varphi(d_1(t, S_1))\sigma_1}{2\sqrt{T}} + rKe^{-rT}N(d_2(t, S_1)) \\ &+ \frac{S_2\varphi(d_1(t, S_2))\sigma_2}{2\sqrt{T}} + rKe^{-rT}N(d_2(t, S_2)) - \Theta_{(\text{WO Call})}\end{aligned} \tag{4.67}$$

with $d_1(t, S_i)$ given by Equation 4.2.

The following derivatives to be determined are in order to the volatilities of each asset, and are equal to

$$\nu_{1(\text{BO Call})} = S_1\varphi(d_1(t, S_1))\sqrt{T} - \nu_{1(\text{WO Call})}, \tag{4.68}$$

$$\nu_{2(\text{BO Call})} = S_2\varphi(d_1(t, S_2))\sqrt{T} - \nu_{2(\text{WO Call})}, \tag{4.69}$$

there is no derivative in ν_1 for the option on asset 2 because the volatility of asset 1 is not one of the parameters of the price function. The same happens in ν_2, since there is no volatility of asset 2 in the price function of the option on asset 1.

The derivatives in order to the prices of each asset, are equal to

$$\Delta_{1(\text{BO Call})} = N(d_1(t, S_1)) - \Delta_{1(\text{WO Call})}, \tag{4.70}$$

$$\Delta_{2(\text{BO Call})} = N(d_1(t, S_2)) - \Delta_{2(\text{WO Call})}, \tag{4.71}$$

the same situation described above when calculating the Vega also occurs when calculating the derivatives in order to the price.

To finish calculating the Greeks for the best of call option, let's calculate the derivative of Δ_1 with respect to the price of the asset 2.

$$\frac{\partial \Delta_{1(\text{BO Call})}}{\partial S_2} = -\frac{\partial \Delta_{1(\text{WO Call})}}{\partial S_2}. \tag{4.72}$$

This derivative ultimately depends only on the crossgamma of the worst of call, as there are no cross effects in simple call options.

The next set of Greeks that are calculated are for the worst of put options. Begin by calculating the derivative of the interest rate

$$\rho_{(\text{WO Put})} = -Te^{-rT}K + \rho_{(\text{WO Call})}, \tag{4.73}$$

the term WO Call(0) does not depend on the interest rate variable, so there is no derivative for this term.

The next Greek to be computed is the derivative in order to the strike price, which is as follows:

$$\frac{\partial P_{(\text{WO Put})}}{\partial K} = e^{-rT} + \frac{\partial P_{(\text{WO Call})}}{\partial K}, \tag{4.74}$$

the term WO Call(0) does not depend on the strike price variable, so there is no derivative for this term.

The correlation derivative is as follows

$$\text{Cega}_{(\text{WO Put})} = -\frac{S_2\varphi(d_1)\sigma_1\sigma_2\sqrt{T}}{\sigma} + \text{Cega}_{(\text{WO Call})}, \tag{4.75}$$

the WO Call(0) price expression is different from the usual one because the strike price is 0, resulting in a different Cega expression. These differences happen in the next few derivatives.

The next Greek to be calculated is Θ, which is equal to

$$\Theta_{(\text{WO Put})} = -re^{-rT}K + \frac{S_2\varphi(d_1)\sigma}{2\sqrt{T}} + \Theta_{(\text{WO Call})}. \tag{4.76}$$

The next derivatives to be deduced are in order to the volatilities of each asset, these are equal to

$$\nu_{1(\text{WO Put})} = \frac{S_2\varphi(d_1)\sqrt{T}(\sigma_1 - \rho\sigma_2)}{\sigma} + \nu_{1(\text{WO Call})}, \tag{4.77}$$

$$\nu_{2(\text{WO Put})} = \frac{S_2\varphi(d_1)\sqrt{T}(\sigma_2 - \rho\sigma_1)}{\sigma} + \nu_{2(\text{WO Call})}. \tag{4.78}$$

Now calculating the Δ of each asset

$$\Delta_{1(\text{WO Put})} = -N(d_2) + \Delta_{1(\text{WO Call})}, \tag{4.79}$$

$$\Delta_{2(\text{WO Put})} = -1 + N(d_1) + \Delta_{2(\text{WO Call})}. \tag{4.80}$$

To complete the calculation of the Greeks for the worst of put option, the derivative of Δ_1 with respect to the price of asset 2 is computed.

$$\frac{\partial \Delta_{1(\text{WO Put})}}{\partial S_2} = -\varphi(d_2)\frac{1}{S_2\sigma\sqrt{T}} + \frac{\partial \Delta_{1(\text{WO Call})}}{\partial S_2}. \tag{4.81}$$

To finalize the computation of the Greeks for these options, the derivatives for the best of put option are calculated. Calculations start with the determination of the derivative of the interest rate

$$\rho_{(\text{BO Put})} = -Te^{-rT}K + \rho_{(\text{BO Call})} \tag{4.82}$$

the term BO Call(0) does not depend on the interest rate variable, so there is no derivative for this term.

The next Greek to be determined is the derivative in order to the strike price, which is the following:

$$\frac{\partial P_{(\text{BO Put})}}{\partial K} = e^{-rT} + \frac{\partial P_{(\text{BO Call})}}{\partial K} \tag{4.83}$$

the term BO Call(0) does not depend on the strike price variable, so there is no derivative for this term.

The derivative of the correlation is the following

$$\text{Cega}_{(\text{BO Put})} = \frac{S_2\varphi(d_1)\sigma_1\sigma_2\sqrt{T}}{\sigma} + \text{Cega}_{(\text{BO Call})} \tag{4.84}$$

the price expression of BO Call(0) is different from the usual one because the strike price is 0, which leads to a different Cega expression. These differences happen in the next few derivatives.

The next Greek to be calculated is Θ. This is equal to

$$\Theta_{(\text{BO Put})} = -re^{-rT}K - \frac{S_2\varphi(d_1)\sigma}{2\sqrt{T}} + \Theta_{(\text{BO Call})}. \tag{4.85}$$

The next Greeks to be deduced are the Vegas, these are equal to

$$\nu_{1(\text{BO Put})} = -\frac{S_2\varphi(d_1)\sqrt{T}(\sigma_1 - \rho\sigma_2)}{\sigma} + \nu_{1(\text{BO Call})}, \tag{4.86}$$

$$\nu_{2(\text{BO Put})} = -\frac{S_2\varphi(d_1)\sqrt{T}(\sigma_2 - \rho\sigma_1)}{\sigma} + \nu_{2(\text{BO Call})}. \tag{4.87}$$

Calculating Δ for each asset follows.

$$\Delta_{1(\text{BO Put})} = -1 + N(d_2) + \Delta_{1(\text{BO Call})}, \tag{4.88}$$

$$\Delta_{2(\text{BO Put})} = -N(d_1) + \Delta_{2(\text{BO Call})}. \tag{4.89}$$

To conclude the Greeks analysis for the best of put option, the derivative of Δ_1 with respect to the price of asset 2 is computed

$$\frac{\partial \Delta_{1(\text{BO Put})}}{\partial S_2} = \varphi(d_2)\frac{1}{S_2\sigma\sqrt{T}} + \frac{\partial \Delta_{1(\text{BO Call})}}{\partial S_2}. \tag{4.90}$$

5

Results and Analysis

In this chapter, the focus is on presenting and analysing the results obtained from the pricing and Greeks computation of each multi-asset option.

The first type of multi-asset option to be discussed is the basket option.

5.1 Basket Options

In order to analyse which is the best method for approximating the price of a basket option compared to the Monte Carlo simulation, it was decided to use a fictitious example of a basket with 4 underlying assets and with the parameters that define the pricing of the option equal to

$$S_i(0) = 100$$

$$\sigma_i = 40\%$$

$$r = 0$$

$$T = 5$$

$$w_i = 0.25$$

$$\rho_{ij} = 0.5 \quad (\text{for} \quad i \neq j) \quad \text{and}$$

$$K = 100$$

Therefore, the results achieved for this example are equal to

Methods	**Levy**	**MP-RG**	**MP-4M**	**Monte Carlo**
Price	28.05	24.50	27.99	27.61

Table 5.1: Price of a Basket Option for the different methods

Ju's method, addressed in subsection 4.2.2, is excluded from Table 5.1 because, despite multiple attempts to implement it in Python, it was not possible to replicate the results

from the article by Krekel, Kock, Korn and Man [14]. Consequently, this technique is not included in the discussion of the subsequent results.

Looking at the values presented in Table 5.1, it can be concluded that, for this example, the MP-4M method is the closest to the Monte Carlo simulation, while the MP-RG method is the furthest. However, it is important to emphasise that, to draw more precise conclusions about which methods yield results closest to the Monte Carlo simulation, it is necessary to vary the various parameters that define the price function of a basket option. This allows for an evaluation of which methods provide the most reliable results under various fluctuations. Only then the method that best approximates the price of a basket option can be selected. Thus, this approach is implemented.

All varied parameters are adjusted to the same values. For instance, if the price of one asset is changed, all the others are also set to that same value. However, due to the completeness of the programmed code, it is not necessary for the above situation to occur, i.e. it is possible, for example, for all assets to have different prices.

The first parameter to vary is the asset price. The variation in the asset price values for the different methods is shown in Figure 5.1.

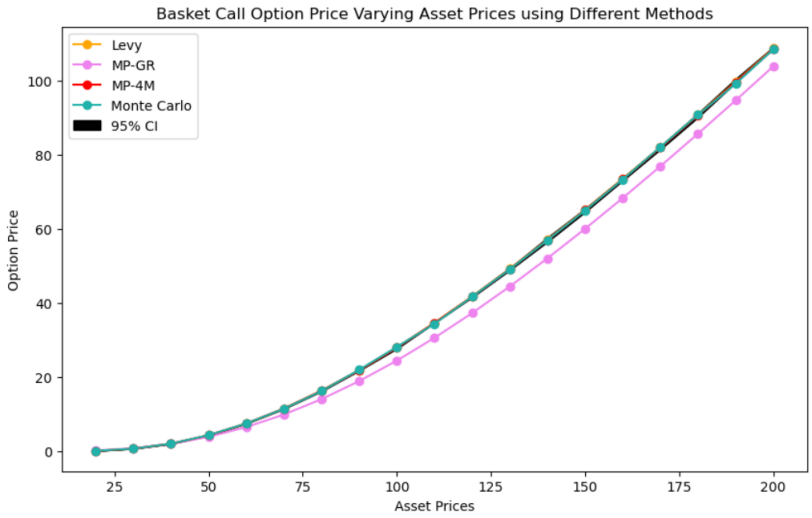

Figure 5.1: Basket Call Option Price Varying Asset Prices using Different Methods

Analysing the graph of Figure 5.1, it can be seen that all methods, except the MP-GR, perform well. The Levy and the MP-4M methods have prices that are virtually the same as the Monte Carlo simulation. This observation is reinforced by the confidence interval of the Monte Carlo simulation in the graph, as the values of the Levy and the MP-4M methods are within the confidence interval.

The next parameter to be varied is the correlation. Since all assets have the same correlation with each other, and the correlation matrix must be positive definite (see subsection 3.2.2), this parameter can only be successfully varied if the correlation is equal to or greater than -0.3. The graph illustrating these variations is shown in Figure 5.2.

Examining the graph of Figure 5.2, it is evident that, aside from the MP-GR method,

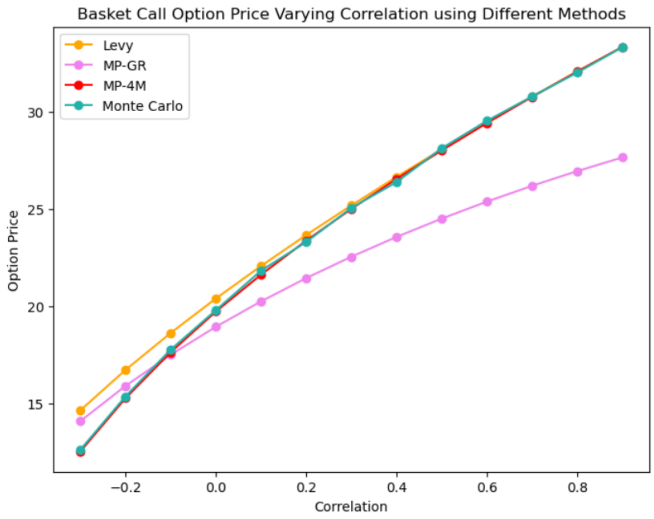

Figure 5.2: Basket Call Option Price Varying Correlation using Different Methods

the other methods perform well. For negative and small positive correlations, the Levy method slightly overprices the basket option compared to the Monte Carlo simulation values. However, for correlation values above 0.5, the option price becomes nearly identical to both the Monte Carlo simulation and the MP-4M method. In contrast, the MP-4M method consistently produces option prices that are nearly identical to the Monte Carlo simulation, regardless of the correlation value.

The Figure 5.3 shows the effect of varying the volatility parameter in the different methods.

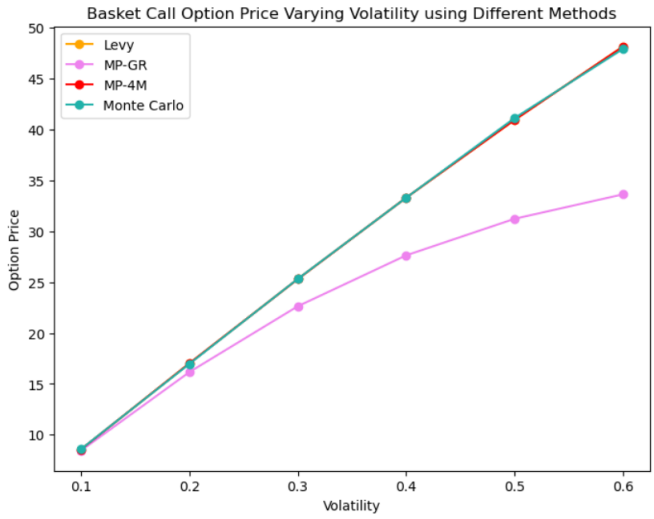

Figure 5.3: Basket Call Option Price Varying Volatility using Different Methods

Observing the graph of Figure 5.3, it is clear that for low volatilities, the methods closely match the values of the Monte Carlo simulation. Yet, at volatilities of 0.2, the MP-GR method begins to diverge from the Monte Carlo simulation. Additionally, the rate of increase of the MP-GR method appears to slow down as volatility rises. The

rate of increase for both the MP-4M and Levy methods appears to follow a linear trend, resembling a straight line as volatility increases.

Finally, the strike price parameter is varied. The graph corresponding to this variation for the different methods can be seen at Figure 5.4.

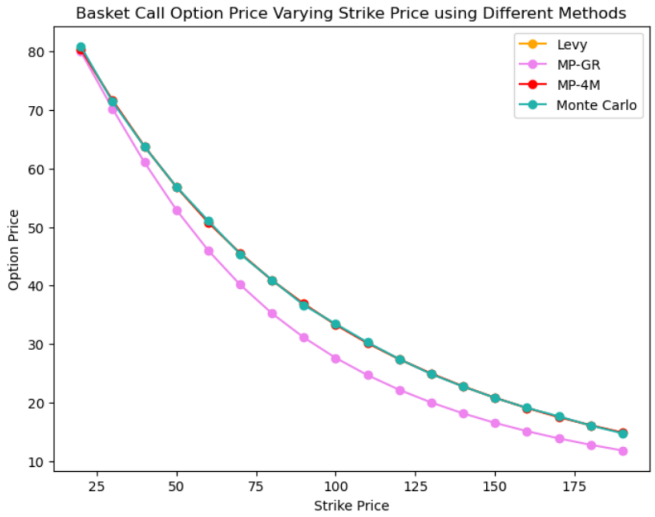

Figure 5.4: Basket Call Option Price Varying Strike Price using Different Methods

Analysing the graph of Figure 5.4, reveals that, for very small strike prices, the methods yield values nearly identical to those of the Monte Carlo simulation. However, as the strike price increases, starting from a strike price of 40, the MP-GR method begins to diverge from the others and ends up undervaluing the basket option compared to the Monte Carlo simulation.

Thus, with these variations in the parameters, it can be concluded that the MP-4M method is the most consistent and produces the most similar results to the Monte Carlo simulation. The MP-GR method, on the other hand, produces the poorest results compared to the Monte Carlo simulation.

The article by Krekel, Kock, Korn and Man [14] explores the idea of varying different parameters and concludes that Ju's method is the most effective for approximating the price of a basket option. This is because it produces results that most closely match those obtained from Monte Carlo simulation. Conversely, the MP-RG method demonstrates the poorest pricing performance among all the methods analyzed, as its results deviate the most from the Monte Carlo benchmarks. In addition, while the MP-4M method shows strong results in the calculations, the authors recommend its use only when the assets exhibit low volatilities.

Now, let's analyse the values obtained for the Greeks using Monte Carlo simulation and the two methods mentioned in subsection 4.2.3. The example used for these calculations is the same as that for pricing the basket option.

Due to the completeness of the code, i.e. all codes allow to choose the asset for which is wanted to calculate the derivatives, it was decided to calculate the derivatives in

order to the first asset, and in the case of Cega, in order to the first and the second asset. Consequently, the following results will reflect this.

The first Greek presented is Δ, with the results shown in Table 5.2.

Methods	Levy	MP-RG	Monte Carlo
Δ	0.16	0.15	0.16

Table 5.2: Delta of a Basket Option for the different methods

Examining the results in Table 5.2, it is evident that of the two methods used, the Levy method produces results most similar to those of the Monte Carlo simulation, with both yielding identical outcomes in this case.

Calculating the value of the derivative for just one asset price is not entirely reliable. Therefore, a graph has been created that illustrates various asset price values for the different methods, as shown in Figure 5.5.

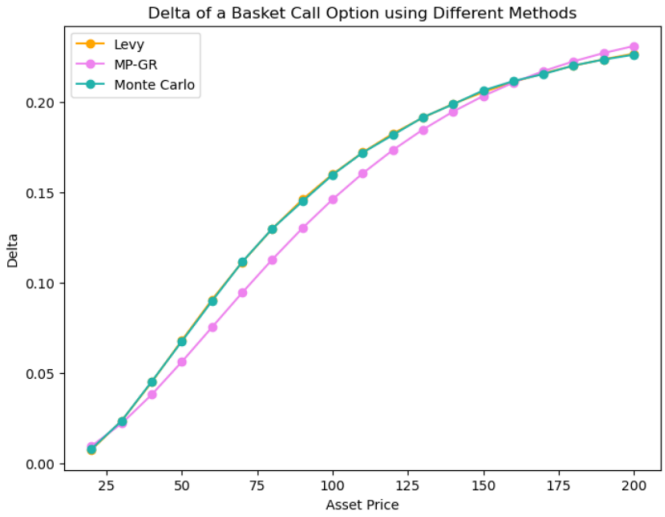

Figure 5.5: Delta of a Basket Call Option using Different Methods

Evaluating the graph of Figure 5.5, it can be seen that the conclusions drawn earlier remain valid across various asset price values, confirming that Levy's method produces results most similar to the Monte Carlo simulation. The results of the MP-GR method for extremely low and high asset prices align closely with those from the Monte Carlo simulation. Furthermore, the graph shows that Δ is positive and increasing, indicating that a one-unit increase in the price of any of the assets in the basket results in an increase in the price of the basket. This behavior is similar to that of call options, which is understandable, as the basket can be viewed as a call option on a single underlying asset, where the underlying asset is the basket itself. Another characteristic observed is that the values of the derivative for the different asset prices range between 0 and 1, a feature common to call options.

Next, the results for the derivative with respect to the correlation can be seen at Table 5.3.

Methods	Levy	MP-RG	Monte Carlo
Cega	2.32	1.51	2.38

Table 5.3: Cega of a Basket Option for the different methods

The results of Table 5.3, reveal that neither of the two methods produces results similar to the Monte Carlo simulation. However, Levy's method is the one with the closest results to those obtained by Monte Carlo simulation, making it a more reliable method than the MP-RG for calculating the Cega of a basket option. Given that the Monte Carlo simulation took approximately 16 minutes to obtain a single Cega result, due to the extensive number of repetitions required to achieve stability, Levy's method should be favored for determining this Greek, as it delivers the most satisfactory results between the two methods.

Similar to the case of Δ, a graph has been created displaying various correlation values for the different methods. This graph is presented in Figure 5.6.

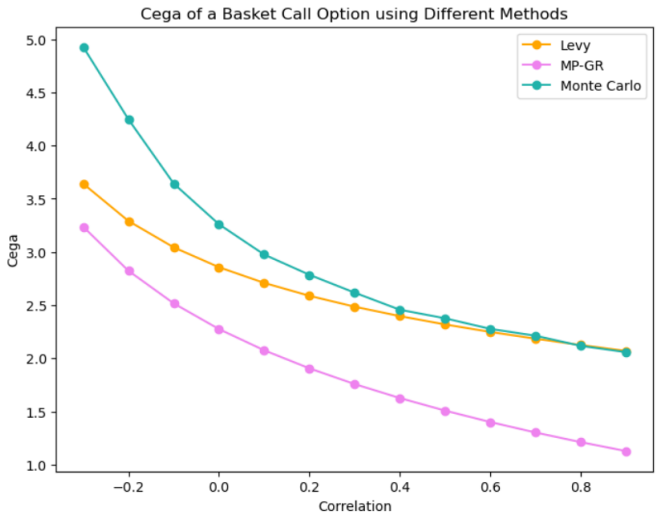

Figure 5.6: Cega of a Basket Call Option using Different Methods

The graphic of Figure 5.6, confirms the earlier observations made while analyzing Table 5.3, meaning that none of the methods yield results that are entirely similar to those of the Monte Carlo simulation. Yet, for correlation values exceeding 0.5, Levy's method starts to produce results that closely align with the Monte Carlo simulation.

The Cega is a decreasing positive function, as expected, indicating that an increase of one unit in the correlation causes an increase in the price of the basket. This means that the highest price of a basket occurs when the correlation is more positive, while the lowest price is when the correlation is more negative, practically close to -1. This observation is logical, since a negative correlation between the assets leads to greater dispersion in the asset prices, which can result in a smaller increase in the basket price. In contrast, a strong positive correlation between the assets makes the asset prices to be very similar to each other, potentially leading to a larger increase in the basket price.

Lastly, the results obtained for the ν are in Table 5.4.

Methods	**Levy**	**MP-RG**	**Monte Carlo**
ν	17.34	11.28	17.21

Table 5.4: Vega of a Basket Option for the different methods

As in the case of the Cega, the results of Table 5.4 indicate that none of the methods produce results similar to the Monte Carlo simulation. Nonetheless, Levy's method is once again the only one that provides results closer to those of the Monte Carlo simulation, establishing it as a more reliable approximation technique compared to the MP-RG method.

Now let's make a graph with different volatility values for the assets for the different methods. The graph can be seen at Figure 5.7.

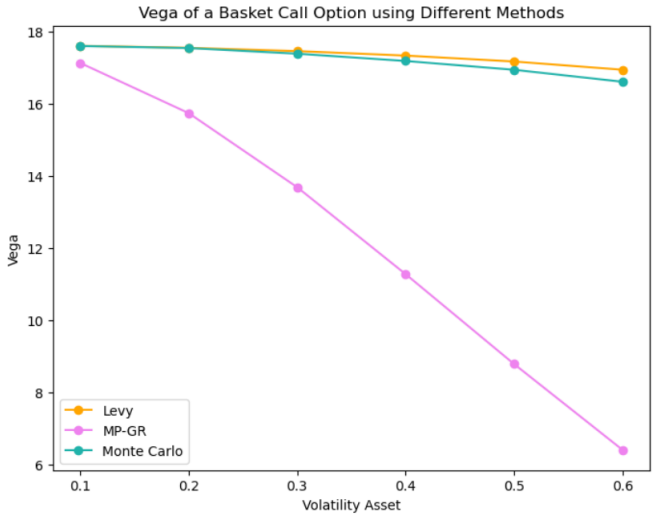

Figure 5.7: Vega of a Basket Call Option using Different Methods

Looking at the graph of Figure 5.7, it confirms what was said earlier when analysing Table 5.4, indicating that none of the methods produce results entirely similar to those of the Monte Carlo simulation. However, for low values of volatility, up to 0.25, Levy's method has ν values that are practically identical to those of the Monte Carlo simulation.

A closer examination of the graph reveals that the MP-GR method decreases at a faster rate than both the Levy method and the Monte Carlo simulation.

The ν graph is a decreasing positive function, meaning that a one unit increase in volatility causes an increase in the price of a basket. Although the Δ of a basket call option behaves similarly to a call option, the same cannot be said for the ν, as the sigma parameter now depends on the volatility of each asset. Consequently, the graph does not have the shape of a parabola with the concavity pointing downwards. It can also be observed that the lowest Vega value occurs when the volatility of the assets is 0.6, whereas the highest is found at a volatility of 0.1.

5.2 Outperformance Options

In order to analyse the Greeks of the outperformance option, the price function of this option was first programmed to help construct the graphs later.

As the pricing formula for these options is similar to that of an European call option, the Greeks will likewise behave similarly. However, let's analyse each of the Greeks in detail to confirm this.

To calculate the Greeks, a fictitious example has been used, where the parameters defining the option price function are

$$S_1(0) = 100$$

$$S_2(0) = 95$$

$$\sigma_1 = 20\%$$

$$\sigma_2 = 25\%$$

$$T = 1$$

$$\rho_{12} = 0.5$$

It should be noted that these fixed values for each parameter will change when calculating the Greeks for their respective parameters.

As with the basket options, graphs have been created to help analyse the results. The first Greek to be analysed is Δ_1, which is the derivative with respect to the price of the asset 1. The graph for this Greek is shown in Figure 5.8.

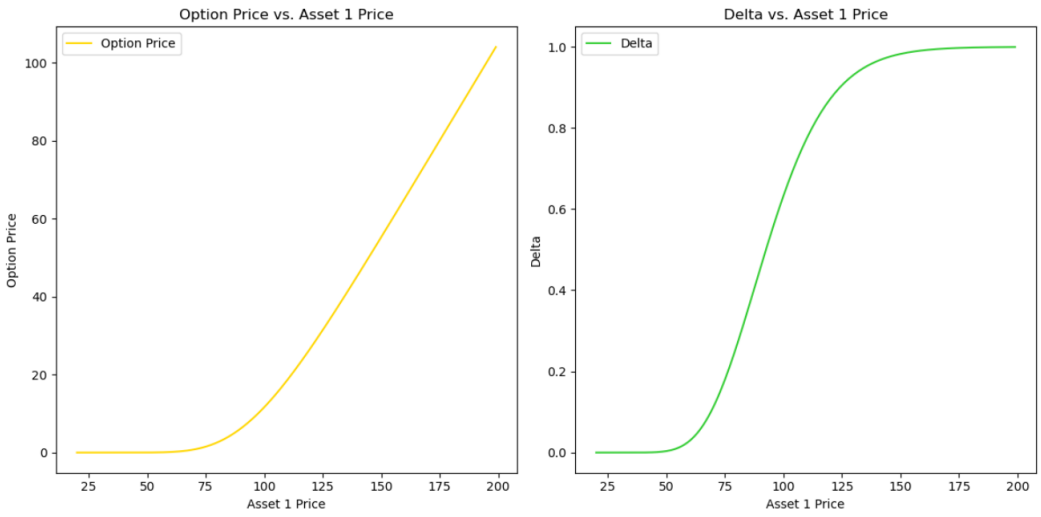

Figure 5.8: Graphic of Asset 1 Price Greek for Outperformance Option Function

Examining the first graph of Figure 5.8, it is evident that from values close to 95, an increase in the price of the asset 1 leads to an increase in the option's price. When the price of asset 1 is below 95 (the price of asset 2), the option is worthless, as asset 1 cannot

outperform asset 2. In the second graph, the behavior of Δ_1 mirrors the Δ behavior of European call options. Notably, for asset 1 prices between 20 and 60 (or 75), Δ_1 remains at zero, indicating no change in the option price. When the price of asset 1 varies between 60 and 140, the function grows at a positive rate, which means that an increase in the price of asset 1 during this price interval causes the option price to increase positively. Finally, for prices above 140, Δ_1 stabilises at 1, which is the maximum value that this Greek can take, meaning that the option's value will continue to increase at a constant rate.

Now let's analyse Δ_2, which represents the derivative with respect to the price of asset 2. The graph illustrating this Greek can be found in Figure 5.9.

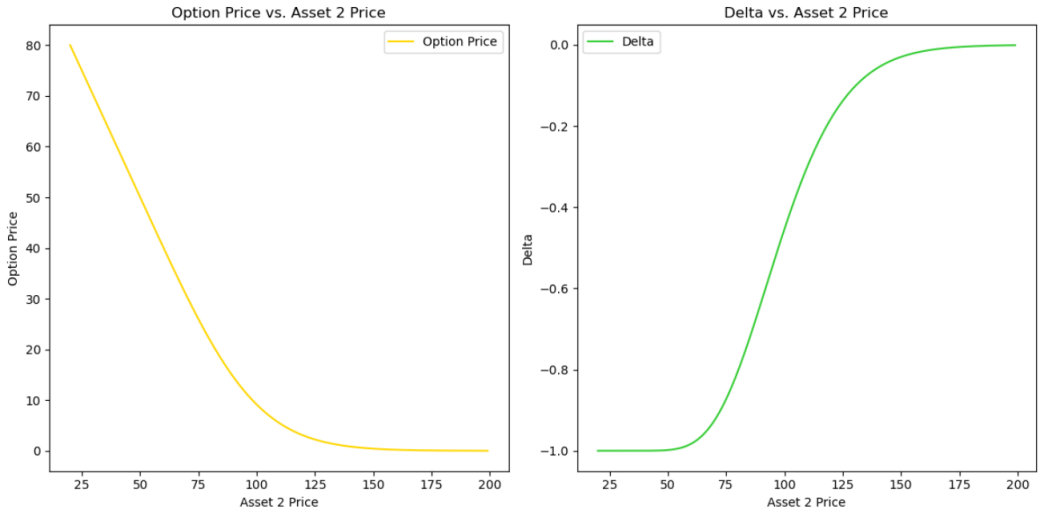

Figure 5.9: Graphic of Asset 2 Price Greek for Outperformance Option Function

Studying the graphs of Figure 5.9, its clear that the behaviour of Δ_2 is the opposite of Δ_1. The first graph shows that an increase in the price of the asset 2 causes a decrease in the price of the option. Once the price of asset 2 reaches 140, the option loses all value, as asset 2 outperforms asset 1, rendering the option worthless. The second graph shows that, initially, up to a price of 60, Δ_2 is equal to -1, which means that an increase in the price of the asset 2 will consistently cause the option price to decline at the same rate, which is reflected in the first graph by a decreasing straight line up to this point. Between the price range of 60 and 140, the value of Δ_2 increases, although it remains negative, indicating that while the option price continues to fall, it does so at a much slower rate. When the price of the asset 2 exceeds 140, the Δ_2 stabilises at 0, signifying that the option price no longer varies with changes in the price of asset 2.

Now, let's examine the Θ. Its graph is shown in Figure 5.10.

The first graph of Figure 5.10 shows that extending the maturity date of the option increases its price, with the growth being more rapid for shorter maturities and slowing as the maturity date extends. The second graph illustrates that increasing the maturity date causes the value of Θ to decrease positively, initially at a faster rate, but the decline becomes more gradual for longer maturity periods, which aligns with the observations

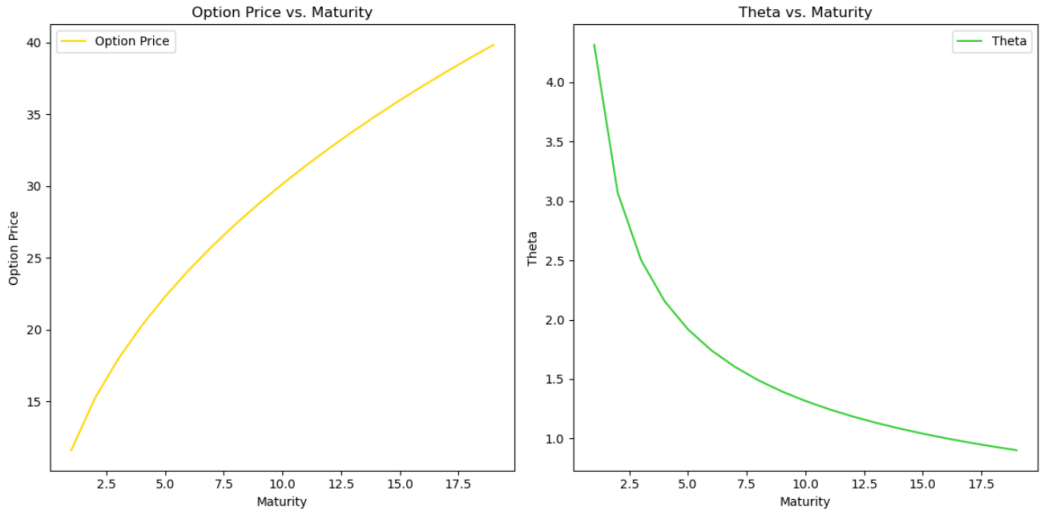

Figure 5.10: Graphic of Maturity Time Greek for Outperformance Option Function

from the first graph.

The next Greek to be analysed is Cega. The graph for this Greek can be viewed in Figure 5.11.

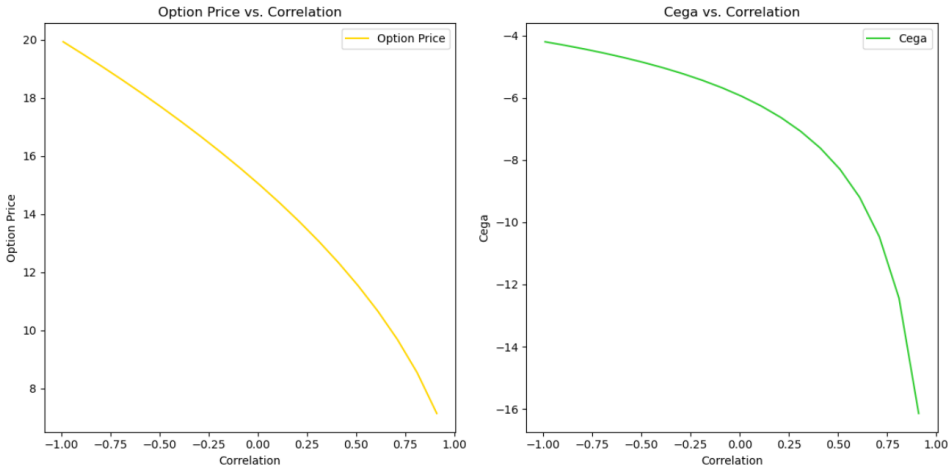

Figure 5.11: Graphic of Correlation Greek for Outperformance Option Function

Interpreting the first graph of Figure 5.11, it can be seen that as the correlation between the two assets increases, the option price decreases. This is consistent with expectations, as σ is equal to $\sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$. When the correlation is -1, σ is at its highest value, leading to the highest option price. As the correlation increases, σ decreases, resulting in a lower option price. The lowest price is observed when the correlation is 1. The second graph shows that as the correlation rises, Cega becomes more negative, indicating that the option price drops faster with increasing correlation, as was already evident from the first graph. Therefore, this type of option is more attractive when the two assets are negatively correlated.

Finally, the ν for each of the assets is analysed. The graphs of these derivatives are

presented in Figure 5.12 and Figure 5.13.

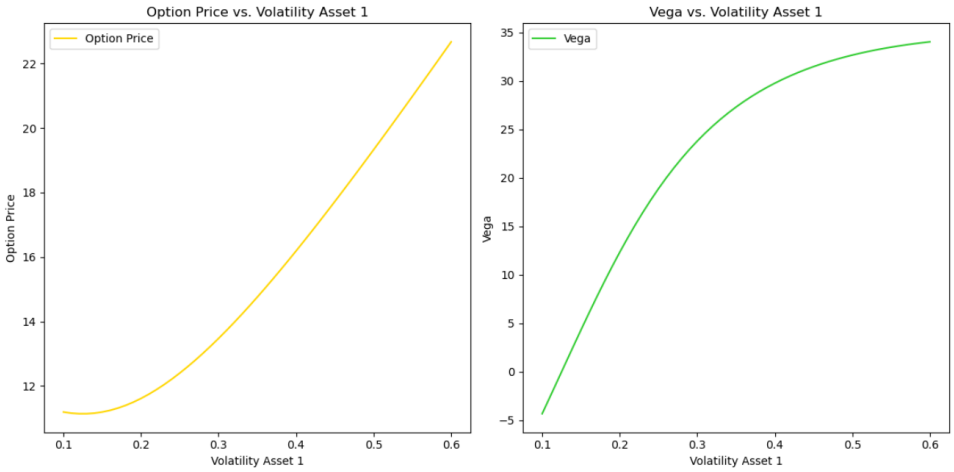

Figure 5.12: Graphic of Volatility Asset 1 Greek for Outperformance Option Function

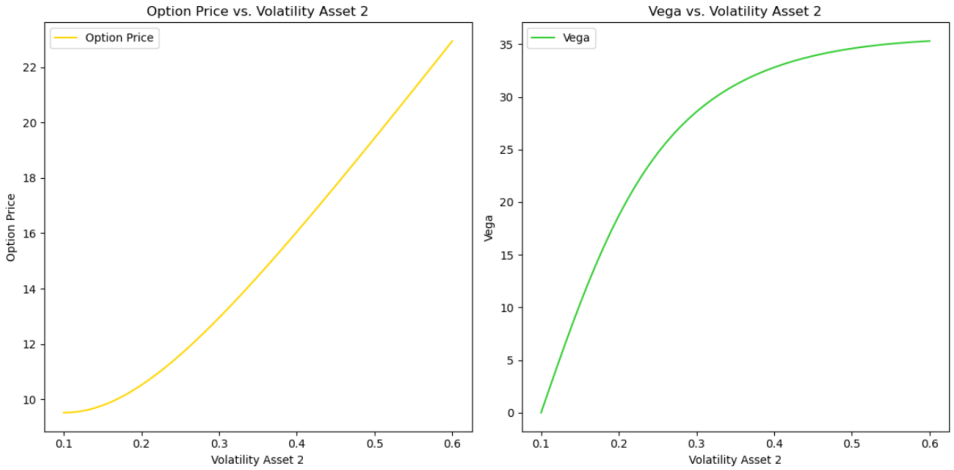

Figure 5.13: Graphic of Volatility Asset 2 Greek for Outperformance Option Function

The first graphs of Figure 5.12 and Figure 5.13 indicate that as the volatility of each asset rises, the price of the option also increases. This aligns with the definition of σ, where the volatilities of each asset are squared in the formula for σ, resulting in a higher value of σ and, consequently, an increase in the option price. The second graph in each figure shows that as the volatility of the assets increases, the value of ν rises and becomes positive, signifying that the option price increases at a faster rate. At certain higher volatility values, ν starts to stabilize, meaning the rate of price increase levels off.

5.3 Best and Worst of Options

As with the outperformance options, it was necessary to program the pricing formulas for these options, using either the explicit formula or the parity relationships discussed earlier (subsection 4.4.2), in order to generate graphs for analysing their Greeks.

To calculate the Greeks of these options, a fictitious example has been used where the parameters defining the price function are

$$S_1(0) = 100$$

$$S_2(0) = 95$$

$$\sigma_1 = 20\%$$

$$\sigma_2 = 25\%$$

$$K = 110$$

$$r = 5\%$$

$$T = 1$$

$$\rho_{12} = 0.5$$

Note that the fixed values for each parameter will vary when calculating the Greeks for each respective parameter.

5.3.1 Worst of Call Option

The first Greek to be studied is ρ, the derivative in order to the risk-free interest rate, whose graph can be seen in Figure 5.14.

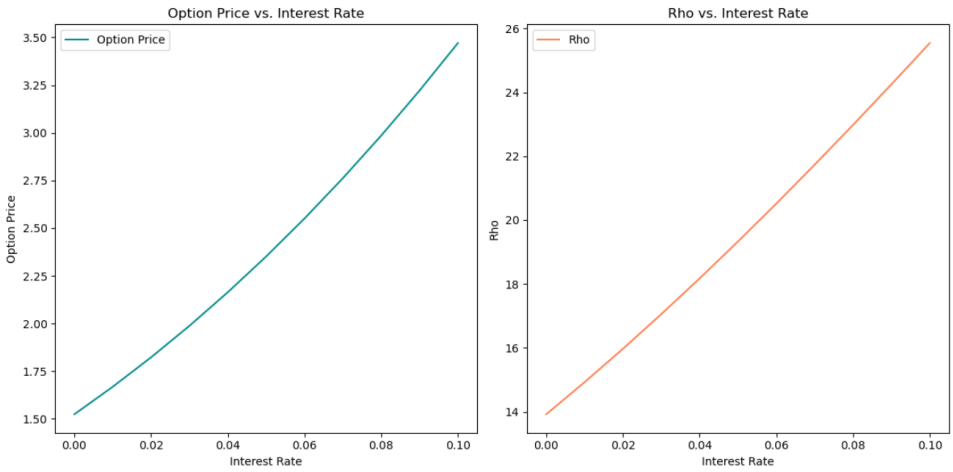

Figure 5.14: Graphic of Interest Rate Greek for Worst of Call Option Function

Examining the first graph of Figure 5.14, it can be observed that the price of a worst of call is a positive increasing function of the interest rate, similar to that of an European call option. The second graph indicates that ρ is an increasing positive function of the interest rate, meaning that a one-unit increase in the interest rate results in an increase in the option price, consistent with the observations made in the first graph.

The next Greek to analyse is the derivative of the option price function with respect to the strike price. The graph of this Greek can be found in Figure 5.15.

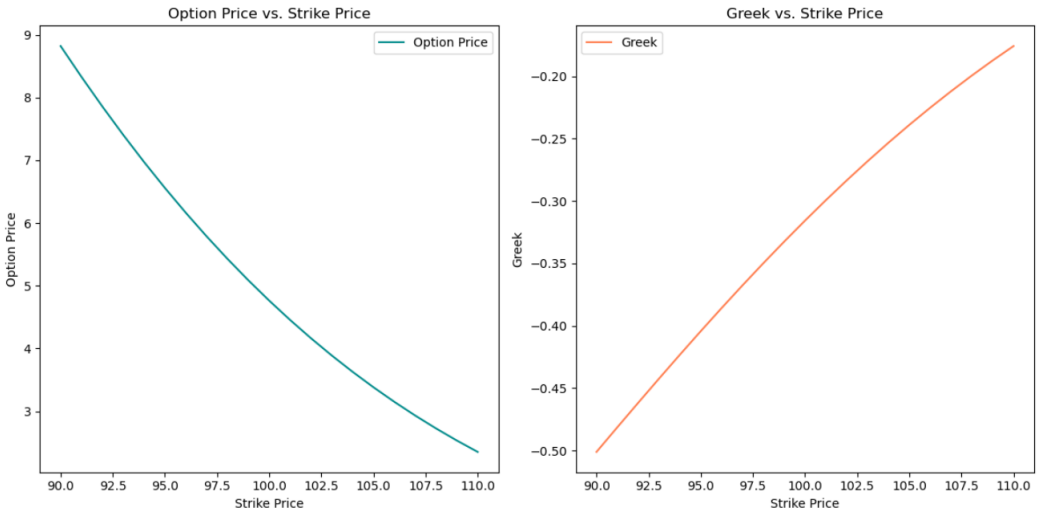

Figure 5.15: Graphic of Strike Price Greek for Worst of Call Option Function

Interpreting the first graph of Figure 5.15, it is evident that the price of a worst of call is a decreasing function as the strike price increases. This is consistent with the nature of call options, where an increase in the strike price leads to a decrease in value and may eventually render the option worthless. This behaviour is similar to that of an European call option. The second graph illustrates that this Greek is a negatively increasing function of the strike price, meaning that a one-unit increase in the strike price results in a decrease in the option price, but at a slower rate as the strike price rises. This observation further supports the findings from the first graph.

The next Greek to be analysed is Cega, and its graph can be viewed in Figure 5.16.

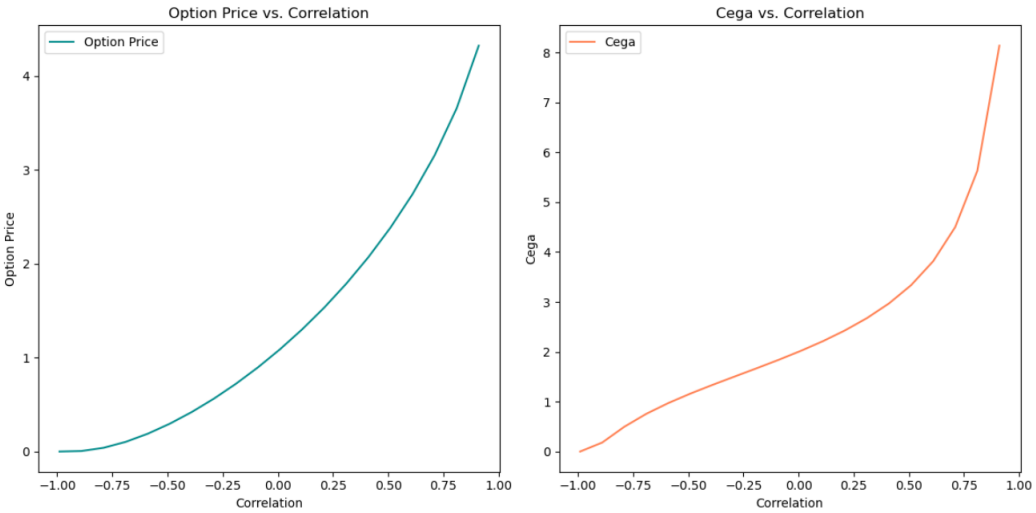

Figure 5.16: Graphic of Correlation Greek for Worst of Call Option Function

Looking at the first graph of Figure 5.16, it can be observed that at high negative correlations, the price of the option grows insignificantly. However, from a correlation of -0.75 onward, the price of the option increases significantly as the correlation value

rises. This behaviour is logical because the option is based on the worst performing asset, therefore, a smaller spread between asset prices is desirable to ensure that even the worst performing asset retains significant value. Turning to the second graph, it can be seen that Cega is a positively increasing function of the correlation coefficient, meaning that a one-unit increase in the correlation leads to an increase in the option price. These conclusions align with the observations made for the first graph. Additionally, a closer look at the second graph reveals that for correlation values below 0.5, the graph approximates a straight line, whereas from a correlation value of 0.5, the graph resembles an exponential function.

The next Greek to be studied is Θ, whose graph is shown in Figure 5.17.

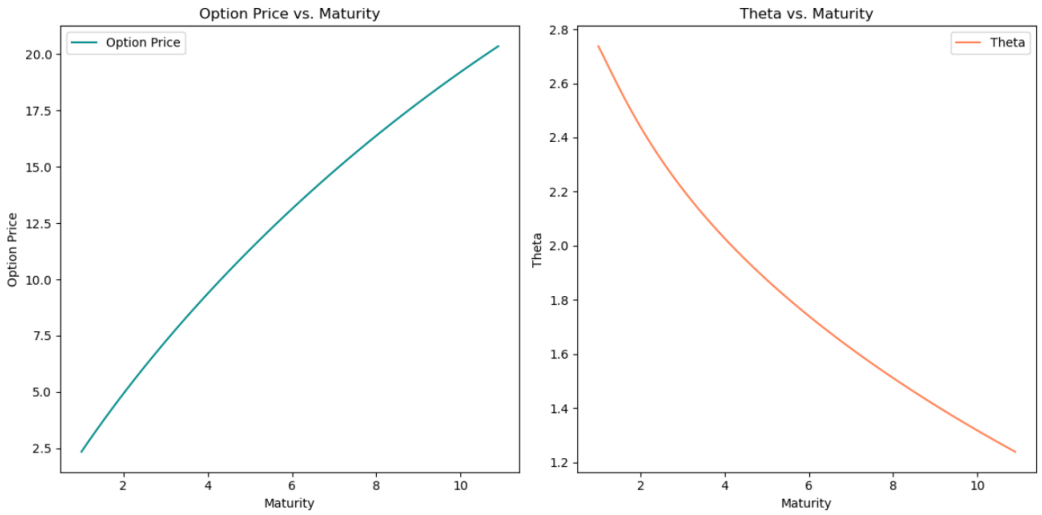

Figure 5.17: Graphic of Maturity Time Greek for Worst of Call Option Function

Analysing the first graph of Figure 5.17, it can be observed that the price of the worst of call is a positively increasing function of the maturity date, a behaviour similar to that of call options. The second graph indicates that Θ is a decreasing positive function of the maturity date, meaning that a one-unit increase in the maturity date leads to an increase in the option price.

The next Greek to examine is ν for the volatility of each asset. The two graphs can be seen in Figure 5.18 and Figure 5.19.

The first graphs in Figure 5.18 and Figure 5.19, show that the option price is an increasing positive function of each asset's volatility, with the price stabilising at higher volatility levels. In the second graphs, it can be seen that ν is a decreasing positive function of each asset's volatility. This means that a one-unit increase in the volatility of one of the assets will cause the option price to rise. It appears that the values of ν approach zero as the volatility increases, as observed in the graph.

The next Greek to analyse is the Δ for the price of each assets. The graphs can be viewed in Figure 5.20 and Figure 5.21.

From the first graphs of Figure 5.20 and Figure 5.21, it can be observed that the option

Figure 5.18: Graphic of Volatility Asset 1 Greek for Worst of Call Option Function

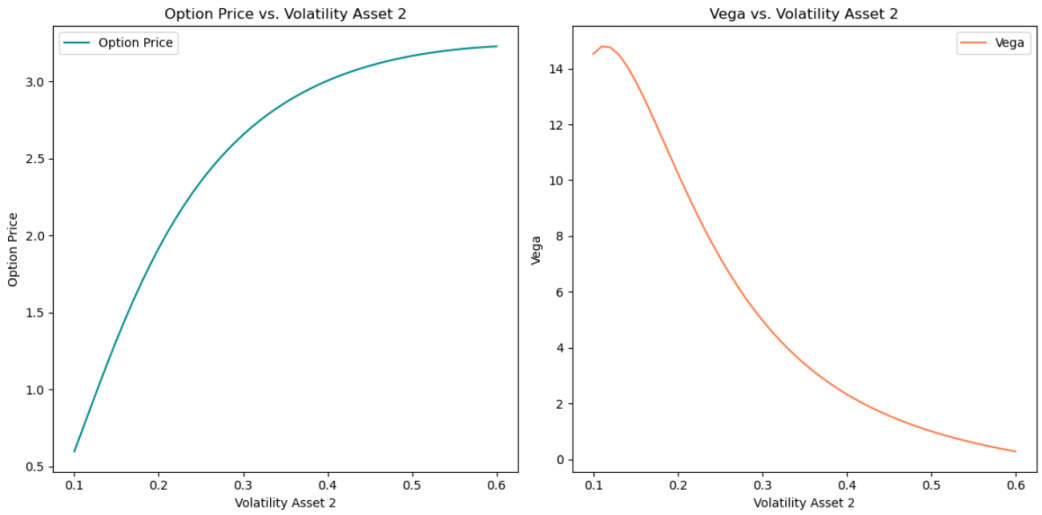

Figure 5.19: Graphic of Volatility Asset 2 Greek for Worst of Call Option Function

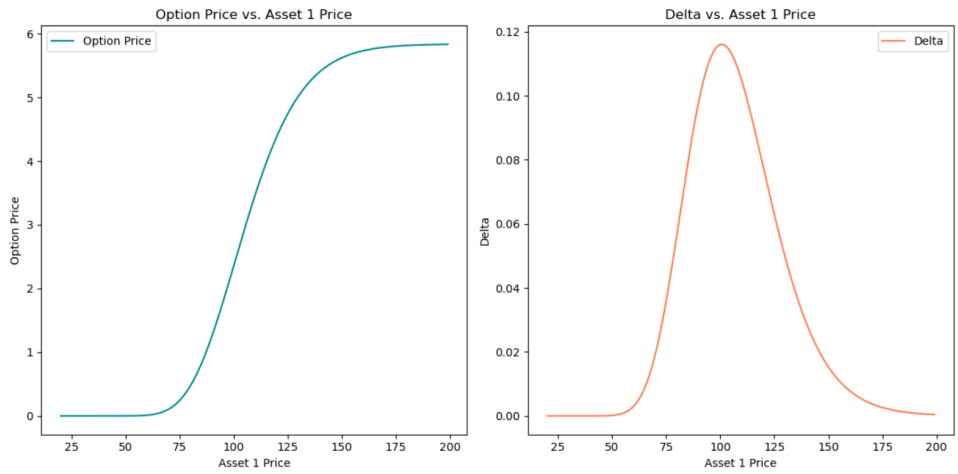

Figure 5.20: Graphic of Price Asset 1 Greek for Worst of Call Option Function

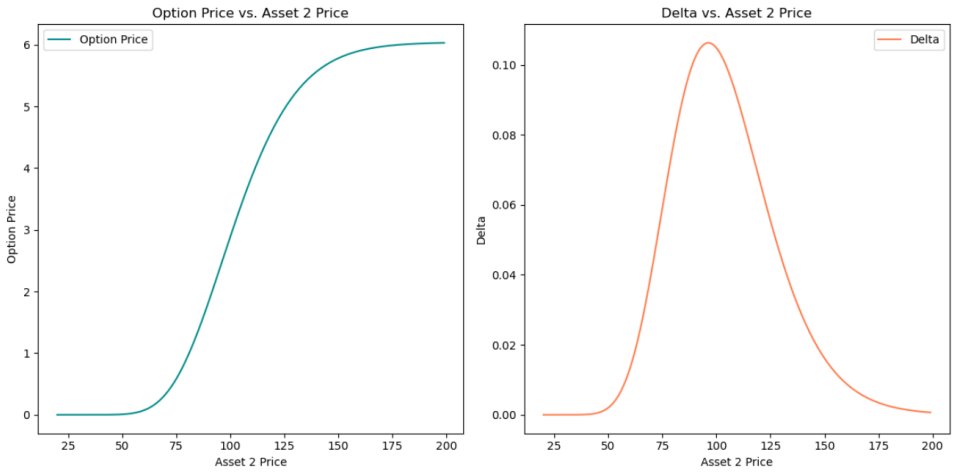

Figure 5.21: Graphic of Price Asset 2 Greek for Worst of Call Option Function

price is zero for asset prices up to 75. In this range, the worst performing asset has a value much lower than the strike price, and since the option is a call, it holds no value. For prices above 75, the option's value begins to increase as the price of the asset rises. Once asset prices exceed 150, the option value starts to stagnate. The second graphs in both figures show that for asset prices up to 75, Δ is zero, which aligns with the observation in the first graph, since the option price is zero, so is Δ. For asset prices between 75 and the point where the two assets reach equal prices, Δ is an increasing function, indicating that as the asset price rises, the option price increases as well. The maximum value of Δ occurs when both assets are priced equally, and beyond this point, Δ decreases, giving the impression that it returns to zero. This is consistent with the first graph's conclusion, as the option price becomes constant, its derivative (Δ) also approaches zero at these values. Since this is a worst of option, it makes sense that for asset prices above 100, Δ drops, as the worst performing asset is likely to be the other one at that point.

To conclude the study of the worst of call option Greeks, let's examine the crossgamma, which is the derivative of Δ_1 with respect to the price of asset 2. The graph of this Greek is displayed in Figure 5.22.

Interpreting Figure 5.22, it can be seen that the crossgamma remains at zero for asset 2 prices up to 50. This is logical because Δ_1 is zero at these values, and since crossgamma is its derivative, it follows that it would also be zero. As the price of asset 2 increases from 50 to a point where it is approximately equal to the price of asset 1, the crossgamma rises, indicating that an increase in asset 2's price leads to an increase in Δ_1. The crossgamma reaches its peak when both assets have the same price, after which it begins to decline, suggesting it may approach zero. This aligns with the behavior seen in the Δ_1 graph. Notably, the magnitude of crossgamma remains close to zero throughout, indicating that asset 2 has little influence on Δ_1.

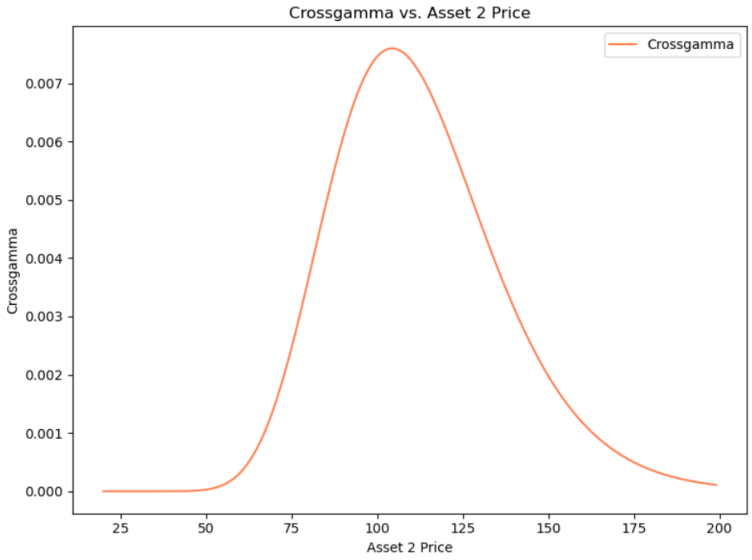

Figure 5.22: Crossgamma for Worst of Call Option Graphic

5.3.2 Best of Call Option

For these options, let's only analyse the most important Greeks, which are Cega, ν and Δ, in that order. The Figure Figure 5.23 illustrates the behavior of Cega.

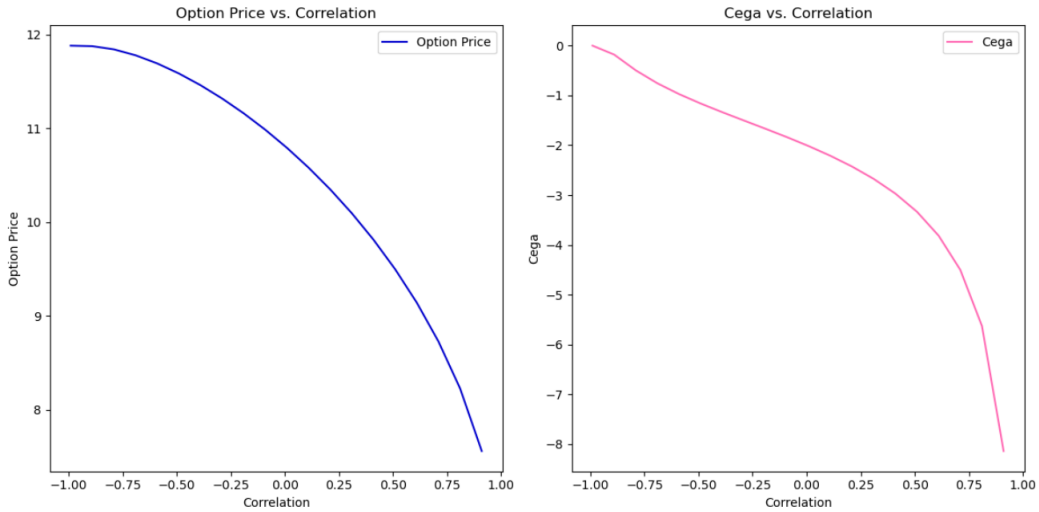

Figure 5.23: Graphic of Correlation Greek for Best of Call Option Function

Examining the first graph of Figure 5.23, it is clear that for a high negative correlation, the option price does not decrease significantly. However, starting from a correlation of -0.75, the option price begins to drop significantly as the correlation value increases. This behaviour is the opposite of what is observed with the worst of call, which is understandable since a more negative correlation benefits the option by increasing the dispersion between asset prices, allowing the best performing asset to reach a higher value. Turning to the second graph, it shows that Cega is a negatively decreasing function of the correlation coefficient. This means that a one-unit increase in the correlation results

in a decrease in the option price. These conclusions align with the insights from the first graph, and this graph is symmetrical to that of the Cega of the worst of call option.

The Figure 5.24 represents the ν graph of the best of call.

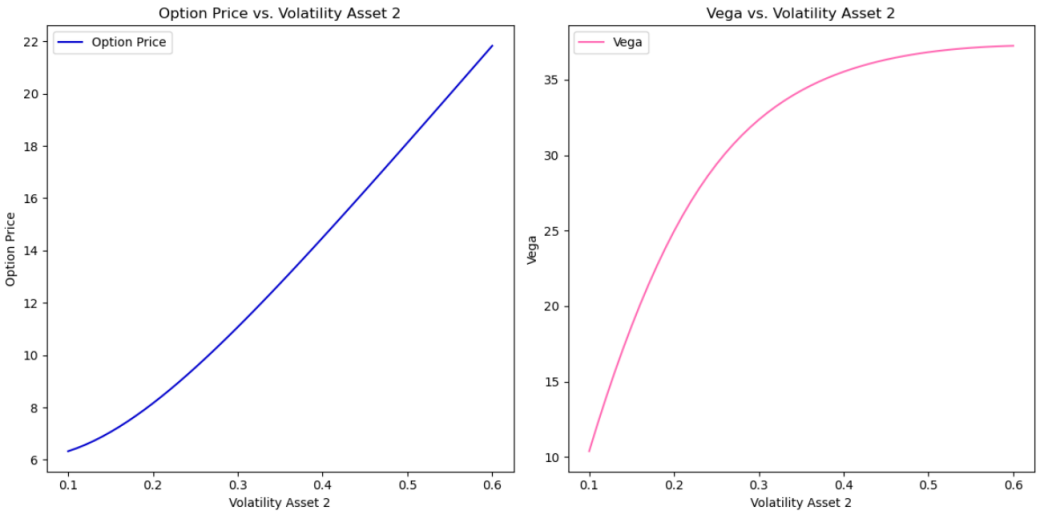

Figure 5.24: Graphic of Volatility Asset 2 Greek for Best of Call Option Function

The first graph of Figure 5.24 illustrates that the option price is an increasing and positive function of the volatility of each asset. The second graph indicates that ν is a positively increasing function of the volatility of each asset. This means that an one-unit increase in the volatility of any asset will lead to a rise in the option price. The graph suggests that if volatility continues to increase, ν is likely to stabilise and reach a constant value.

Finally, the Figure 5.25 contains the Δ of the best of call.

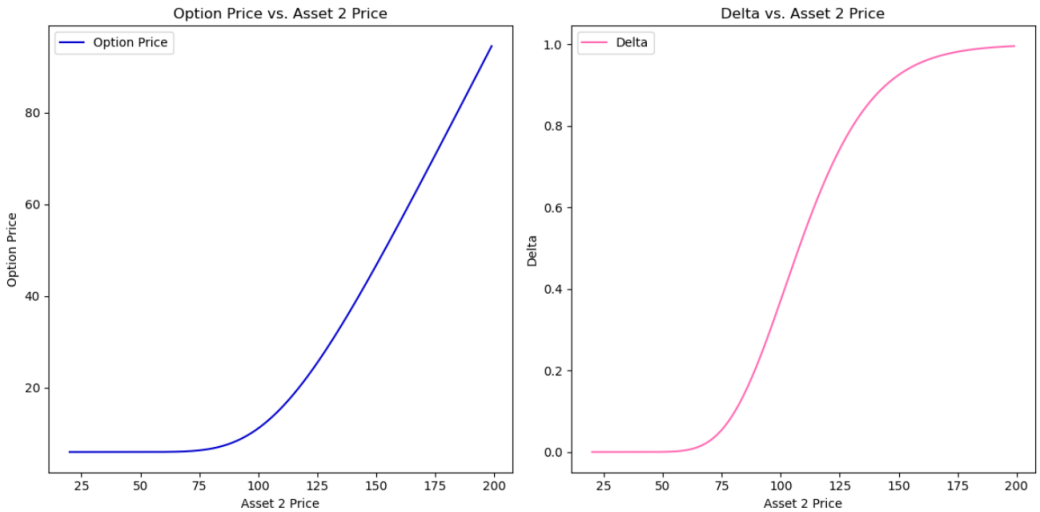

Figure 5.25: Graphic of Price Asset 2 Greek for Best of Call Option Function

From the first graph of Figure 5.25, it can be observed that for asset prices up to 75, the price of the option is zero. In this situation, the best performing asset, asset 1, has a

value lower than the strike price, meaning the call option has no value. For prices above 75, the option's value begins to increase as the price of the asset rises. The second graph shows that for asset prices up to 75, Δ is zero, which aligns with the observation in the first graph, as a zero option price corresponds to a zero Δ. For asset prices between 75 and 150, Δ is an increasing function, i.e. as the asset price increases, the option price also increases. At asset prices of 150 or more, Δ starts to stabilise and reaches a value of 1. This behaviour of this Greek is identical to the Δ of an European call option.

5.3.3 Worst of Put Option

First, let's analyse Cega, whose graph can be viewed in Figure 5.26.

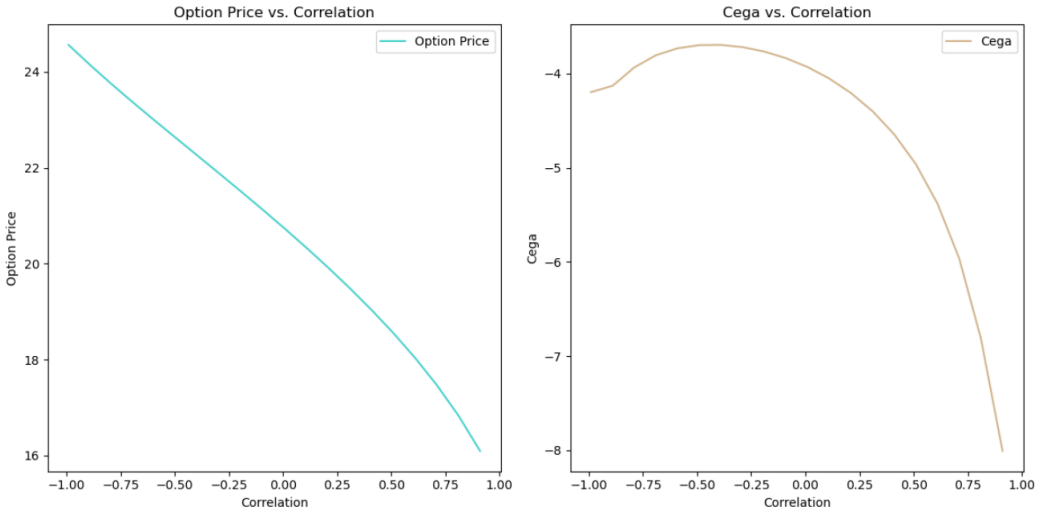

Figure 5.26: Graphic of Correlation Greek for Worst of Put Option Function

Looking at the first graph of Figure 5.26, it is evident that the price of the option decreases almost proportionally as the correlation value increases. This behaviour is comprehensible, as a lower worst performing asset price relative to the strike price is advantageous in this case, since the option is a put. In the second graph, it can be observed that Cega is a negatively increasing function for high negative correlation values, up to approximately a correlation coefficient of -0.5. Beyond this point, the function begins to decrease as the correlation coefficient increases, indicating that a one-unit increase in the correlation value may lead to either a smaller or a larger decrease in the option price.

The Figure 5.27 illustrates the ν of a worst of put option.

The first graph of Figure 5.27 shows that the option price is an increasing and positive function of the volatility of each asset. The second graph indicates that ν is a positive increasing function, meaning that a one-unit increase in the volatility of one of the assets will generally lead to an increase in the option price. Furthermore, it can be observed that for volatility values equal to or greater than 0.35, the value of ν begins to stabilise at approximately 35.

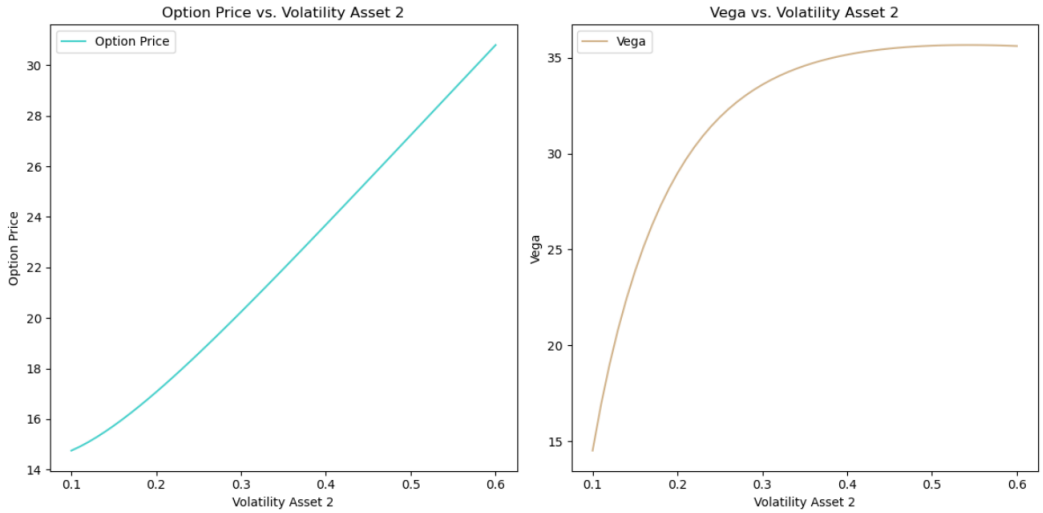

Figure 5.27: Graphic of Volatility Asset 2 Greek for Worst of Put Option Function

The last Greek of the worst of put options to be studied is the Δ, represented in Figure 5.28.

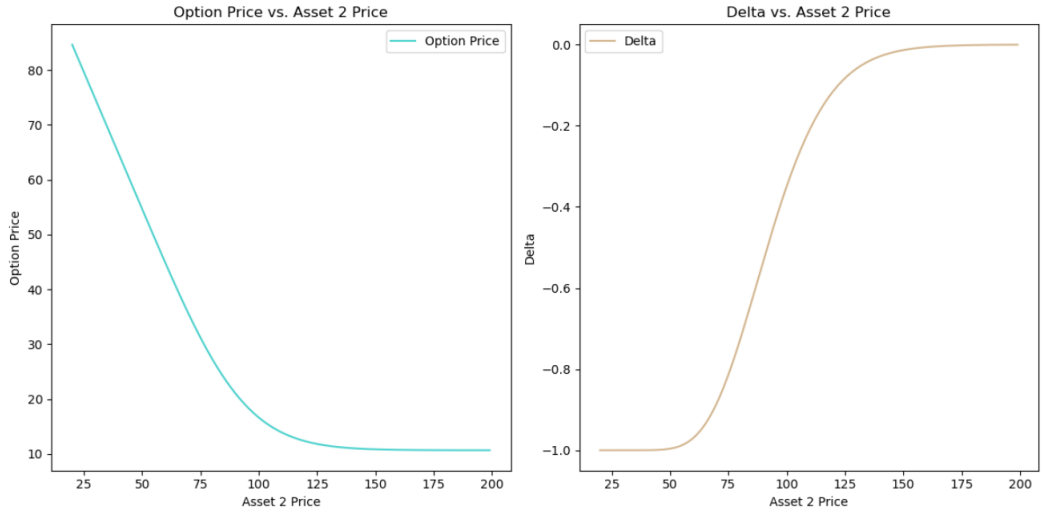

Figure 5.28: Graphic of Price Asset 2 Greek for Worst of Put Option Function

The first graph of Figure 5.28 illustrates that for asset prices up to 125, the option price decreases as the asset price increases. For values above 125, the option price begins to stabilise, remaining constant. The second graph shows that for asset prices up to 50, the Δ is -1. For asset prices between 50 and 150, Δ is a negative increasing function, i.e. as the asset price increases, the option price decreases at a slower rate. For asset prices of 150 or more, Δ begins to stabilise and reaches 0.

5.3.4 Best of Put Option

The Cega of this option is illustrated in Figure 5.29.

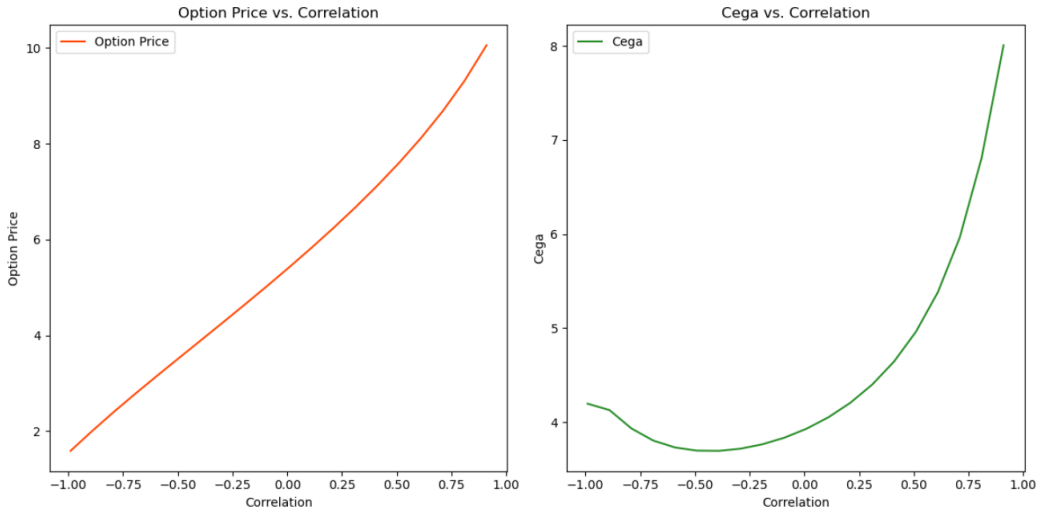

Figure 5.29: Graphic of Correlation Greek for Best of Put Option Function

Analysing the first graph of Figure 5.29, it is evident that that the option price increases almost proportionally as the correlation value rises. The option price reaches its maximum at perfect positive correlation, where asset prices are not dispersed, which is advantageous for a put option. In the second graph, Cega is shown to be a decreasing positive function for high values of negative correlation, up to approximately a correlation coefficient of -0.5. Beyond this point, the function begins to increase as the correlation coefficient rises, indicating that a one-unit increase in correlation can lead to either a smaller or larger increase in the option price.

The graph of the ν can be found in Figure 5.30.

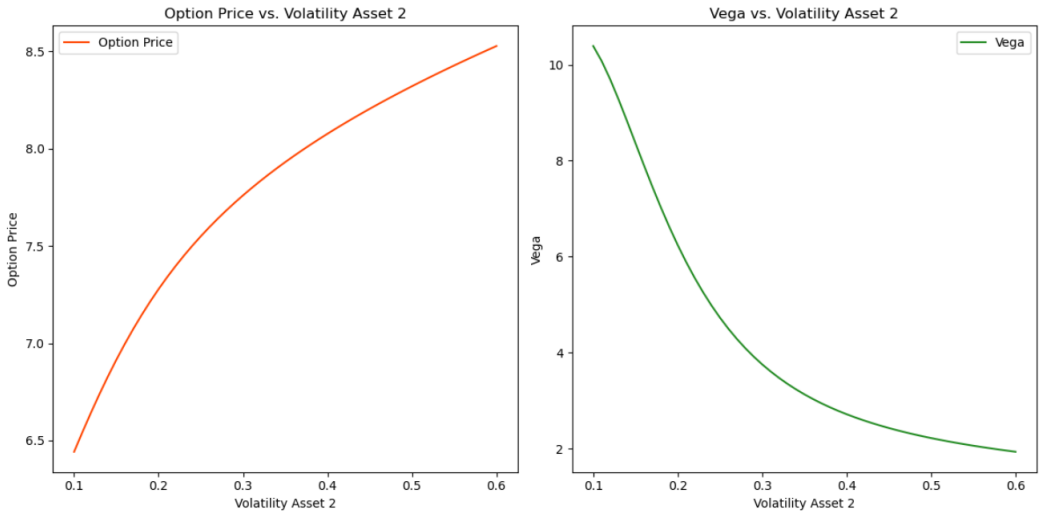

Figure 5.30: Graphic of Volatility Asset 2 Greek for Best of Put Option Function

The first graph of Figure 5.30 shows that the option price is a positive increasing function of the volatility of each asset. The second graph indicates that ν is a positive decreasing function, meaning that a one-unit increase in the volatility of one of the assets

will cause the option price to rise more rapidly at lower volatilities and more slowly at higher volatilities.

To finish the analysis of the Greeks of these options, let's examine the Δ of the best of put option. The graph for this Greek can be viewed in Figure 5.31.

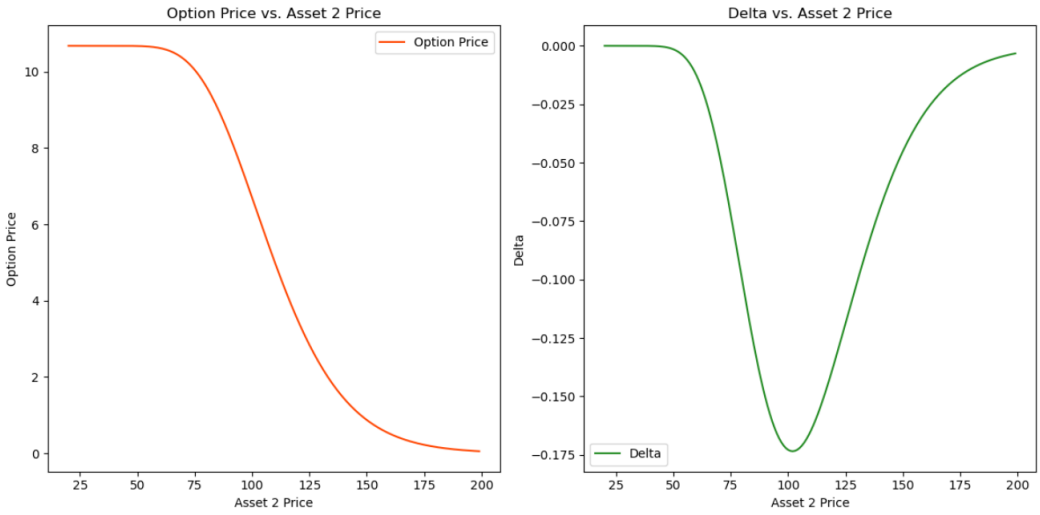

Figure 5.31: Graphic of Price Asset 2 Greek for Best of Put Option Function

The first graph of Figure 5.31 indicates that the option price remains constant for asset prices up to 75. Beyond this threshold, the option value begins to decrease, eventually reaching zero as the asset price continues to rise. The second graph shows that for asset prices up to 75, Δ is zero, aligning with the first graph's observations, if the option price is constant, Δ must be zero. For asset prices between 75 and 100, Δ behaves as a negatively decreasing function, i.e. as the asset price increases, the option price decreases. When the asset price ranges from 100 to 200, the function remains negative but starts to increase, meaning that a one-unit rise in the asset price results in a smaller decrease in the option price. It is also noteworthy that Δ approaches zero as the asset price nears 200.

6

Conclusion

The aim of this thesis was to study and analyse multi-asset options. While the effects of various parameters on these options were examined, particular emphasis was placed on the impact of correlation. This focus provided deeper insights into how correlation influences the behavior and pricing of multi-asset options.

In general, the three types of multi-asset options exhibited distinct behaviors regarding the increase in the correlation coefficient.

In the study of basket options, prior to examining the impact of correlation on these options, an effort was made to determine which of the four methods analysed yielded results most similar to those obtained from the Monte Carlo simulation. One of the four methods could not be implemented in the Python programming language, so it was excluded from consideration when drawing conclusions about which method was the best. By varying most of the parameters that define the price of a basket option, except for the maturity date and the interest rate, it was found that the method producing the most consistent and similar results to the Monte Carlo simulation is the MP-4M technique. After determining which method best approximates the price of a basket option, the effect of correlation on these options was analysed using Monte Carlo simulation. This method serves as the benchmark due to its high accuracy in approximating both the price and the Greeks of a basket option. Additionally, two of the four methods studied in this thesis were utilised in this analysis. The idea of using these methods to analyse the Greeks was innovative, as, to the best of available knowledge, none of the reviewed articles on this topic have addressed this aspect. Since these methods are used to approximate the price of a basket option, they could also be utilised to estimate the Greeks of these options. Consequently, varying the correlation coefficient between -0.3 and 0.9 revealed that none of the methods yielded results comparable to the Cega derived from the Monte Carlo simulation. However, for high positive correlation coefficients, Levy's method produces results nearly identical to those of the Monte Carlo simulation. Analysing the behaviour of Cega through simulation, reveals that this Greek is a positive decreasing function of the correlation coefficient, indicating that as the correlation increases, the option price continues to rise, but at a progressively lower rate. If the option price increases with rising

correlation, it implies that the maximum option price is attained when the correlation reaches the perfectly positive value. Monte Carlo simulation may be the best method for obtaining both the price and the Greeks of a basket option, serving as a benchmark due to its high accuracy. However, achieving reliable values requires running numerous repetitions, which can make it computationally intensive. Therefore, these alternative methods should be considered, as they can offer a faster computational approach while approximating the benchmark results.

The Cega of outperformance options is the opposite of that of basket options. The Cega curve of these options is a negative decreasing function of the correlation coefficient, indicating that as the correlation increases, the price of the option decreases at a greater rate. This relationship implies that the maximum price the option can reach occurs when the correlation is perfectly negative.

Finally, the behaviour of the Cega was examined for the best and the worst of options. Starting with the Cega of the worst of call, it can be seen that it is a positive increasing function of the correlation coefficient. This means that as the correlation increases, the price of the option rises at an increasing rate. Since the price of the option increases with the correlation, the maximum price it can achieve occurs when the correlation coefficient is 1.

On the other hand, the Cega of the best of call exhibits the opposite behaviour. The graph of this Greek is a negative decreasing function, which means that as the correlation coefficient increases, the price of the option decreases at an accelerating rate. Therefore, the highest price that the option can achieve occurs when the correlation coefficient is equal to -1.

The Cega of the worst of put option behaves similar to that of the best call option, primarily being a negative decreasing function. There are certain correlation values, particularly at highly negative correlation coefficients, where the function may show an increase. Generally, however, an increase in correlation leads to a decrease in the option price, initially at a slower rate and then at an accelerated rate as the correlation continues to rise. Consequently, this option reaches its maximum price with negative perfect correlation.

The Cega of the best of put demonstrates behavior similar to that of the worst call option, generally being a positive increasing function. At very negative correlation values, however, the function exhibits a decreasing trend. This indicates that as the correlation coefficient increases, the price of the option rise, initially at a slower pace and then accelerating. Thus, the maximum price for the best of put options occurs when the correlation value is perfectly positive.

As observed, the various multi-asset options exhibit distinct behaviors in relation to the correlation parameter, highlighting the diversity these options offer to investors.

One of the strengths of this thesis was the calculation of the derivatives of two of the four methods studied to approximate the price of a basket option, as no developments in this area were found in any of the literature reviewed. Additionally, all the code developed

for pricing and calculating the Greeks of basket options is general, it allows for differing parameters among assets, or for all assets to share the same parameters. In the case of the Greeks, users can choose the specific asset for which the derivatives are to be calculated. Another strong point was the extension of the parity formulas for the best and worst of options to n assets.

One area for improvement was the inability to program the Ju method in Python due to the lack of clarity in the available articles. To address this, the literature review could have been expanded to include more sources, in hopes of finding clearer explanations or alternative presentations of the method that would facilitate its implementation.

For future work on this topic, the Greeks of the MP-4M method could be calculated, as it is one of the most reliable techniques for approximating the price of a basket option. Another potential avenue for future research could involve repeating the analysis performed for basket options while considering the errors that arise from using these methods to approximate their price and Greeks. Another area for development is addressing the hedging problem for these options, since their Greeks have already been analysed. Future research could also focus on utilising basket option approximation methods to estimate implicit correlation. Additionally, researchers could extend the scope of this thesis by studying more complex multi-asset options and applying the same analytical approaches that were used throughout this dissertation.

Bibliography

[1] J. Beisser. "Another Way to Value Basket Options". Working Paper, Johannes Gutenberg-Universität Mainz. 1999 (cit. on pp. 6, 8).

[2] T. Björk. *Arbitrage Theory in Continuous Time*. Vol. 3. Oxford University Press, 2009 (cit. on pp. 2, 6, 11, 12, 18, 31, 46).

[3] F. Black and M. Scholes. "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3 (1973), pp. 637–654. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831029 (visited on 2024-01-18) (cit. on pp. 5, 17).

[4] M. Bouzoubaa and A. Osseiran. *Exotic Options and Hybrids - A Guide to Structuring Pricing and Trading*. Vol. 1. John Wiley & Sons, 2010 (cit. on pp. 1, 12, 13, 17, 18).

[5] P. Brandimarte. *Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics*. John Wiley & Sons, 2014 (cit. on p. 8).

[6] J. C. Cox and S. A. Ross. "The valuation of options for alternative stochastic processes". In: *Journal of Financial Economics* 3.1 (1976), pp. 145–166. ISSN: 0304-405X. URL: https://www.sciencedirect.com/science/article/pii/0304405X76900234 (visited on 2024-03-23) (cit. on pp. 6, 34).

[7] E. Derman. "Outperformance Options". Goldman Sachs. 1992 (cit. on p. 9).

[8] H. Geman, N. E. Karoui, and J.-C. Rochet. "Changes of Numéraire, Changes of Probability Measure and Option Pricing". In: *Journal of Applied Probability* 32.2 (1995), pp. 443–458. ISSN: 00219002. URL: http://www.jstor.org/stable/3215299 (visited on 2024-06-27) (cit. on p. 30).

[9] D. Gentle. "Basket Weaving". In: *RISK* (1993), pp. 51–52 (cit. on pp. 6, 8).

[10] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004 (cit. on p. 8).

[11] H. Johnson. "Options on the Maximum or the Minimum of Several Assets". In: *The Journal of Financial and Quantitative Analysis* 22.3 (1987), pp. 277–283. ISSN: 00221090, 17566916. URL: http://www.jstor.org/stable/2330963 (visited on 2024-01-27) (cit. on pp. 2, 5, 6, 34, 36, 37).

[12] H. Johnson. "The Pricing of Complex Options". In: *Unpubl. manuscript* (Aug 1981) (cit. on p. 5).

[13] N. Ju. "Pricing Asian and Basket Options Via Taylor Expansion". In: *Journal of Computational Finance* 5.3 (1992), pp. 79–103. (Visited on 2024-03-23) (cit. on pp. 7, 8, 21, 22).

[14] M. Krekel, J. De Kock, R. Korn, and T.-K. Man. "An analysis of pricing methods for baskets options". In: *Wilmott* 3 (2004), pp. 82–89. (Visited on 2024-01-31) (cit. on pp. 6, 8, 20, 52, 54).

[15] E. Levy. "Pricing European average rate currency options". In: *Journal of International Money and Finance* 11.5 (1992), pp. 474–491. ISSN: 0261-5606. URL: https://www.sciencedirect.com/science/article/pii/026156069290013N (visited on 2024-03-23) (cit. on pp. 6, 20).

[16] E. Levy and S. Turnbull. "Average Intelligence". In: *RISK* 6.2 (1992), pp. 5–9. (Visited on 2024-03-23) (cit. on pp. 6, 20).

[17] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[18] W. Margrabe. "The Value of an Option to Exchange One Asset for Another". In: *The Journal of Finance* 33.1 (1978), pp. 177–186. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2326358 (visited on 2024-03-28) (cit. on p. 9).

[19] H. M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press, 1959. ISBN: 9780300013726. URL: http://www.jstor.org/stable/j.ctt1bh4c8h (visited on 2024-01-18) (cit. on p. 14).

[20] M. Milevsky and S. Posner. "A Closed-Form Approximation for Valuing Basket Options". In: *Journal of Derivatives* 5 (1998), pp. 54–61. (Visited on 2024-03-23) (cit. on pp. 7, 8, 23, 24).

[21] M. Milevsky and S. Posner. "Valuing Exotic Options by Approximating the SPD with Higher Moments". In: *Journal of Financial Engineering* 7.2 (1998), pp. 54–61. (Visited on 2024-03-23) (cit. on pp. 8, 24, 25).

[22] M. Musiela and M. Rutkowski. *Martingale Methods in Financial Modelling*. Vol. 2. Springer, 2005 (cit. on pp. 1, 11).

[23] B. Øksendal. *Stochastic Differential Equations*. 6th ed. Springer, 2003 (cit. on pp. 5–7, 9, 11).

[24] G. Pagès. *Numerical Probability: An Introduction With Applications to Finance*. Springer, 2018 (cit. on pp. 8, 16).

[25] R. Stulz. "Options on the minimum or the maximum of two risky assets: Analysis and applications". In: *Journal of Financial Economics* 10.2 (1982), pp. 161–185. ISSN: 0304-405X. URL: https://www.sciencedirect.com/science/article/pii/0304405X82900113 (visited on 2024-03-23) (cit. on pp. 1, 5, 6, 36, 40).

[26] C. Veiga, U. Wystup, and M. L. Esquível. "Unifying exotic option closed formulas". In: *Review of Derivatives Research* 15 (2012), pp. 99–128. URL: https://link.springer.com/article/10.1007/s11147-011-9071-8 (visited on 2024-02-15) (cit. on pp. 6, 8, 9, 31, 35).

[27] F. de Weert. *Exotic Options Trading*. Vol. 1. John Wiley & Sons, 2008 (cit. on p. 2).

[28] P. Wilmott. *Introduces Quantitative Finance*. Vol. 2. John Wiley & Sons, 2007 (cit. on pp. 1, 36).

I

Code

```python
from math import log, sqrt, exp
from scipy.stats import norm, multivariate_normal
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from itertools import product

#Call Option Function with Black Scholes

def black_scholes_call(S, K, T, r, sigma):
    d1 = (log(S / K) + (r + (sigma**2) / 2) * T) / (sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    call_price = S * norm.cdf(d1) - K * exp(-r * T) * norm.cdf(d2)
    return call_price

# Example usage:
S0 = 100  # Current stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility

call_option_price = black_scholes_call(S0, K, T, r, sigma)
print(f"The price of the call option is: {call_option_price:.2f}")
The price of the call option is: 6.04
```

Listing I.1: Call Option Function and an Example

```python
def worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2

```

```
    # Define the parameters for the the bivariate cumulative standard normal distribution
    mean = [0, 0]
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    theta1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    theta2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)

    worst_call_price = S1 * multivariate_normal.cdf([alpha1, alpha2], mean=mean, cov=[[1,theta1], [theta1, 1]]) + S2 * multivariate_normal.cdf([beta1, beta2], mean=mean, cov=[[1, theta2], [theta2, 1]]) - K * exp(-r * T) * multivariate_normal.cdf([gamma1, gamma2], mean=mean, cov=[[1, rho], [rho, 1]])
    return worst_call_price

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

worst_call_option_price = worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho)
print(f"The price of the worst of call option is: {worst_call_option_price:.2f}")
The price of the worst of call option is: 2.35
```

Listing I.2: Worst of Call Option Function and an Example

```
def best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho):
    best_call_price = black_scholes_call(S1, K, T, r, sigma1) + black_scholes_call(S2, K, T, r, sigma2) - worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho)
    return best_call_price

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

```

```
best_call_option_price = best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho)
print(f"The price of the best of call option is: {best_call_option_price:.2f}")
The price of the best of call option is: 9.53
```

Listing I.3: Best of Call Option Function and an Example

```
def margrabe(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    margrabe_price = S2 - S2 * norm.cdf(d1) + S1 * norm.cdf(d2)
    return margrabe_price
```

Listing I.4: Auxiliary Function

```
def worst_of_put(S1, S2, K, T, r, sigma1, sigma2, rho):
    worst_put_price = exp(-r * T) * K - margrabe(S1, S2, T, sigma1, sigma2, rho) + worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho)
    return worst_put_price

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

worst_put_option_price = worst_of_put(S1, S2, K, T, r, sigma1, sigma2, rho)
print(f"The price of the worst of put option is: {worst_put_option_price:.2f}")
The price of the worst of put option is: 18.60
```

Listing I.5: Worst of Put Option Function and an Example

```
def best_of_put(S1, S2, K, T, r, sigma1, sigma2, rho):
    best_put_price = exp(-r * T) * K - S1 - S2 + margrabe(S1, S2, T, sigma1, sigma2, rho) + best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho)
    return best_put_price

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
```

```
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

best_put_option_price = best_of_put(S1, S2, K, T, r, sigma1, sigma2, rho)
print(f"The price of the best of put option is: {best_put_option_price:.2f}")
The price of the best of put option is: 7.55
```

Listing I.6: Best of Put Option Function and an Example

```
#Greeks of Worst Call Option

#In order to the interest rate

def greek_rho(S1, S2, K, T, r, sigma1, sigma2, rho):
  gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))

   greek_rho_value = K * T * exp(-r * T) * multivariate_normal.cdf([gamma1,
    gamma2], mean=[0,0], cov=[[1, rho], [rho, 1]])
    return greek_rho_value

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1     # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate interest rates
interest_rates = np.arange(0, 0.11, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Interest Rate': interest_rates,
    'Option Price': [worst_of_call(S1, S2, K, T, rate, sigma1, sigma2, rho)
    for rate in interest_rates],
    'Rho': [greek_rho(S1, S2, K, T, rate, sigma1, sigma2, rho) for rate in
    interest_rates]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Interest Rate', y='Option Price', ax=axes[0], label='Option Price
    ', color='darkcyan')
axes[0].set_title('Option Price vs. Interest Rate')
axes[0].set_xlabel('Interest Rate')
axes[0].set_ylabel('Option Price')
```

```
38
39 data.plot(x='Interest Rate', y='Rho', ax=axes[1], label='Rho', color='coral')
40 axes[1].set_title('Rho vs. Interest Rate')
41 axes[1].set_xlabel('Interest Rate')
42 axes[1].set_ylabel('Rho')
43
44 plt.tight_layout()
45 plt.show()
```

Listing I.7: Interest Rate Greek Function of Worst of Call Option and an Example

```
1  #In order to the strike price
2
3  def greek(S1, S2, K, T, r, sigma1, sigma2, rho):
4    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
5    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
6
7      greek_value = - exp(-r * T) * multivariate_normal.cdf([gamma1, gamma2],
       mean=[0,0], cov=[[1, rho], [rho, 1]])
8    return greek_value
9
10 #Example usage:
11 S1 = 100  # Current S1 stock price
12 S2 = 95   # Current S2 stock price
13 T = 1 # Time to expiration in years
14 r = 0.05  # Risk-free interest rate
15 sigma1 = 0.2  # Volatility S1 stock
16 sigma2 = 0.25  # Volatility S2 stock
17 rho = 0.5 # Correlation between the 2 stocks
18
19 # Generate strike prices
20 strike_prices = np.arange(90, 111, 1)
21
22 # Create a DataFrame to organize the data
23 data = pd.DataFrame({
24     'Strike Price': strike_prices,
25     'Option Price': [worst_of_call(S1, S2, strike, T, r, sigma1, sigma2, rho)
       for strike in strike_prices],
26     'Greek': [greek(S1, S2, strike, T, r, sigma1, sigma2, rho) for strike in
       strike_prices]
27 })
28
29 # Plotting the graphs using pandas integration with matplotlib
30 fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
31
32 data.plot(x='Strike Price', y='Option Price', ax=axes[0], label='Option Price'
       , color='darkcyan')
33 axes[0].set_title('Option Price vs. Strike Price')
34 axes[0].set_xlabel('Strike Price')
35 axes[0].set_ylabel('Option Price')
```

```
36
37 data.plot(x='Strike Price', y='Greek', ax=axes[1], label='Greek', color='coral
       ')
38 axes[1].set_title('Greek vs. Strike Price')
39 axes[1].set_xlabel('Strike Price')
40 axes[1].set_ylabel('Greek')
41
42 plt.tight_layout()
43 plt.show()
```

Listing I.8: Strike Price Greek Function of Worst of Call Option and an Example

```
1  #In order to the correlation
2
3  def greek_cega(S1, S2, K, T, r, sigma1, sigma2, rho):
4    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
5    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
6    beta1 = gamma2 + sigma2 * sqrt(T)
7    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
8    rho_C = (rho * sigma1 - sigma2)/sqrt(sigma_square)
9    d1 = (beta1 - rho_C * beta2) / (sqrt(1 - rho_C ** 2))
10
11     greek_cega_value = S2 * norm.cdf(d1) * (exp(-0.5 * (beta2 ** 2)) / sqrt(2
       * np.pi)) * (sigma1 * sigma2 * sqrt(T) / sqrt(sigma_square))
12   return greek_cega_value
13
14 #Example usage:
15 S1 = 100  # Current S1 stock price
16 S2 = 95   # Current S2 stock price
17 K = 110   # Option strike price
18 T = 1 # Time to expiration in years
19 r = 0.05  # Risk-free interest rate
20 sigma1 = 0.2  # Volatility S1 stock
21 sigma2 = 0.25  # Volatility S2 stock
22
23 # Generate correlation values
24 correlations = np.arange(-0.99, 1, 0.1)
25
26 # Create a DataFrame to organize the data
27 data = pd.DataFrame({
28     'Correlation': correlations,
29     'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for
       rho in correlations],
30     'Cega': [greek_cega(S1, S2, K, T, r, sigma1, sigma2, rho) for rho in
       correlations]
31 })
32
33 # Plotting the graphs using pandas integration with matplotlib
34 fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
35
```

```
data.plot(x='Correlation', y='Option Price', ax=axes[0], label='Option Price',
     color='darkcyan')
axes[0].set_title('Option Price vs. Correlation')
axes[0].set_xlabel('Correlation')
axes[0].set_ylabel('Option Price')

data.plot(x='Correlation', y='Cega', ax=axes[1], label='Cega', color='coral')
axes[1].set_title('Cega vs. Correlation')
axes[1].set_xlabel('Correlation')
axes[1].set_ylabel('Cega')

plt.tight_layout()
plt.show()
```

Listing I.9: Correlation Greek Function of Worst of Call Option and an Example

```
#In order to the time

def greek_theta(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha2 - rho1 * alpha1) / (sqrt(1 - rho1 ** 2))
    d2 =  (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d3 = (beta2 - rho2 * beta1) / (sqrt(1 - rho2 ** 2))
    d4 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d5 = (gamma2 - rho * gamma1) / (sqrt(1 - rho ** 2))
    d6 = (gamma1 - rho * gamma2) / (sqrt(1 - rho ** 2))
    deriv_alpha1 = (log(K/S1) + (r + (sigma1**2) / 2) * T) / (2 * T * sigma1 *
     sqrt(T))
    deriv_alpha2 = (log(S1/S2) - (sigma_square / 2) * T) / (2 * T * sqrt(
    sigma_square * T))
    deriv_gamma1 = (log(K/S1) + (r - (sigma1**2) / 2) * T) / (2 * T * sigma1 *
     sqrt(T))
    deriv_gamma2 = (log(K/S2) + (r - (sigma2**2) / 2) * T) / (2 * T * sigma2 *
     sqrt(T))
    deriv_beta1 = (log(K/S2) + (r + (sigma2**2) / 2) * T) / (2 * T * sigma2 *
    sqrt(T))
    deriv_beta2 = (log(S2/S1) - (sigma_square / 2) * T) / (2 * T * sqrt(
    sigma_square * T))

     greek_theta_value = S1 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha1 **
    2)) * norm.cdf(d1) * deriv_alpha1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (
    alpha2 ** 2)) * norm.cdf(d2) * deriv_alpha2) + \
```

```
    S2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta1 ** 2)) * norm.cdf(d3) *
    deriv_beta1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf(
    d4) * deriv_beta2) + \
    K * r * exp(-r * T) * multivariate_normal.cdf([gamma1, gamma2], mean
    =[0,0], cov=[[1, rho], [rho, 1]]) - \
    K * exp(-r * T) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma1 ** 2)) *
    norm.cdf(d5) * deriv_gamma1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma2 **
     2)) * norm.cdf(d6) * deriv_gamma2)
    return greek_theta_value

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate maturity dates
maturities = np.arange(1, 11, 0.1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Maturity': maturities,
    'Option Price': [worst_of_call(S1, S2, K, time, r, sigma1, sigma2, rho)
    for time in maturities],
    'Theta': [greek_theta(S1, S2, K, time, r, sigma1, sigma2, rho) for time in
     maturities]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Maturity', y='Option Price', ax=axes[0], label='Option Price',
    color='darkcyan')
axes[0].set_title('Option Price vs. Maturity')
axes[0].set_xlabel('Maturity')
axes[0].set_ylabel('Option Price')

data.plot(x='Maturity', y='Theta', ax=axes[1], label='Theta', color='coral')
axes[1].set_title('Theta vs. Maturity')
axes[1].set_xlabel('Maturity')
axes[1].set_ylabel('Theta')

plt.tight_layout()
plt.show()
```

Listing I.10: Maturity Time Greek Function of Worst of Call Option and an Example

```
#In order to the volatility of the assets

def greek_vega1(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha2 - rho1 * alpha1) / (sqrt(1 - rho1 ** 2))
    d2 = (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d3 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d4 = (gamma2 - rho * gamma1) / (sqrt(1 - rho ** 2))
    deriv_alpha1 = (log(K/S1) - (r - (sigma1**2) / 2) * T) / ((sigma1 ** 2) *
    sqrt(T))
    deriv_alpha2 = (sigma1 - rho * sigma2) * (((log(S1/S2) - ((sigma_square *
    T) / 2))  / ((sqrt(sigma_square)**3) * T)))
    deriv_gamma1 = (log(K/S1) - (r + (sigma1**2) / 2) * T) / ((sigma1 ** 2) *
    sqrt(T))
    deriv_beta2 = (sigma1 - rho * sigma2) * (((log(S2/S1) - ((sigma_square * T
    ) / 2))  / ((sqrt(sigma_square)**3) * T)))

    greek_vega1_value = S1 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha1 ** 2)
    ) * norm.cdf(d1) * deriv_alpha1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (
    alpha2 ** 2)) * norm.cdf(d2) * deriv_alpha2) + \
    S2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf(d3) *
    deriv_beta2) - \
    K * exp(-r * T) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma1 ** 2)) *
    norm.cdf(d4) * deriv_gamma1)
    return greek_vega1_value

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate volatilities for stock 1
volatility1 = np.arange(0.1, 0.61, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Volatility Asset 1': volatility1,
```

```
    'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for sigma1 in volatility1],
    'Vega': [greek_vega1(S1, S2, K, T, r, sigma1, sigma2, rho) for sigma1 in volatility1]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Volatility Asset 1', y='Option Price', ax=axes[0], label='Option Price', color='darkcyan')
axes[0].set_title('Option Price vs. Volatility Asset 1')
axes[0].set_xlabel('Volatility Asset 1')
axes[0].set_ylabel('Option Price')

data.plot(x='Volatility Asset 1', y='Vega', ax=axes[1], label='Vega', color='coral')
axes[1].set_title('Vega vs. Volatility Asset 1')
axes[1].set_xlabel('Volatility Asset 1')
axes[1].set_ylabel('Vega')

plt.tight_layout()
plt.show()
```

Listing I.11: Volatility of Asset 1 Greek Function of Worst of Call Option and an Example

```
def greek_vega2(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d2 = (beta2 - rho2 * beta1) / (sqrt(1 - rho2 ** 2))
    d3 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d4 = (gamma1 - rho * gamma2) / (sqrt(1 - rho ** 2))
    deriv_alpha2 = (sigma2 - rho * sigma1) * (((log(S1/S2) - ((sigma_square * T) / 2))  / ((sqrt(sigma_square)**3) * T)))
    deriv_gamma2 = (log(K/S2) - (r + (sigma2**2) / 2) * T) / ((sigma2 ** 2) * sqrt(T))
    deriv_beta1 = (log(K/S2) - (r - (sigma2**2) / 2) * T) / ((sigma2 ** 2) * sqrt(T))
    deriv_beta2 = (sigma2 - rho * sigma1) * (((log(S2/S1) - ((sigma_square * T) / 2))  / ((sqrt(sigma_square)**3) * T)))

```

```
    greek_vega2_value = S1 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha2 ** 2)) * norm.cdf(d1) * deriv_alpha2) + \
    S2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta1 ** 2)) * norm.cdf(d2) * deriv_beta1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf(d3) * deriv_beta2) - \
    K * exp(-r * T) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma2 ** 2)) * norm.cdf(d4) * deriv_gamma2)
    return greek_vega2_value

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate volatilities for stock 2
volatility2 = np.arange(0.1, 0.61, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Volatility Asset 2': volatility2,
    'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for sigma2 in volatility2],
    'Vega': [greek_vega2(S1, S2, K, T, r, sigma1, sigma2, rho) for sigma2 in volatility2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Volatility Asset 2', y='Option Price', ax=axes[0], label='Option Price', color='darkcyan')
axes[0].set_title('Option Price vs. Volatility Asset 2')
axes[0].set_xlabel('Volatility Asset 2')
axes[0].set_ylabel('Option Price')

data.plot(x='Volatility Asset 2', y='Vega', ax=axes[1], label='Vega', color='coral')
axes[1].set_title('Vega vs. Volatility Asset 2')
axes[1].set_xlabel('Volatility Asset 2')
axes[1].set_ylabel('Vega')

plt.tight_layout()
plt.show()
```

Listing I.12: Volatility of Asset 2 Greek Function of Worst of Call Option and an Example

```
#In order to the underlying assets prices

def greek_delta1(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha2 - rho1 * alpha1) / (sqrt(1 - rho1 ** 2))
    d2 = (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d3 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d4 = (gamma2 - rho * gamma1) / (sqrt(1 - rho ** 2))
    deriv_alpha1 = 1 / (sigma1 * sqrt(T) * S1)
    deriv_alpha2 = -1 / (sqrt(T * sigma_square) * S1)
    deriv_gamma1 = 1 / (sigma1 * sqrt(T) * S1)
    deriv_beta2 = 1 / (sqrt(T * sigma_square) * S1)

    greek_delta1_value = S1 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha1 **
    2)) * norm.cdf(d1) * deriv_alpha1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (
    alpha2 ** 2)) * norm.cdf(d2) * deriv_alpha2) + \
    S2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf(d3) *
    deriv_beta2) + \
    multivariate_normal.cdf([alpha1, alpha2], mean=[0,0], cov=[[1, rho1], [
    rho1, 1]]) - \
    K * exp(-r * T) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma1 ** 2)) *
    norm.cdf(d4) * deriv_gamma1)
    return greek_delta1_value

#Example usage:
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 1 prices
asset1 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 1 Price': asset1,
    'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for
    S1 in asset1],
```

```
    'Delta': [greek_delta1(S1, S2, K, T, r, sigma1, sigma2, rho) for S1 in asset1]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 1 Price', y='Option Price', ax=axes[0], label='Option Price', color='darkcyan')
axes[0].set_title('Option Price vs. Asset 1 Price')
axes[0].set_xlabel('Asset 1 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 1 Price', y='Delta', ax=axes[1], label='Delta', color='coral')
axes[1].set_title('Delta vs. Asset 1 Price')
axes[1].set_xlabel('Asset 1 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.13: Price of Asset 1 Greek Function of Worst of Call Option and an Example

```
def greek_delta2(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d2 = (beta2 - rho2 * beta1) / (sqrt(1 - rho2 ** 2))
    d3 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d4 = (gamma1 - rho * gamma2) / (sqrt(1 - rho ** 2))
    deriv_alpha2 = 1 / (sqrt(sigma_square * T) * S2)
    deriv_gamma2 = 1 / (sigma2 * sqrt(T) * S2)
    deriv_beta1 = 1 / (sigma2 * sqrt(T) * S2)
    deriv_beta2 = -1 / (sqrt(sigma_square * T) * S2)

    greek_delta2_value = S1 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha2 ** 2)) * norm.cdf(d1) * deriv_alpha2) + \
    S2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta1 ** 2)) * norm.cdf(d2) * deriv_beta1 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf(d3) * deriv_beta2) + \
    multivariate_normal.cdf([beta1, beta2], mean=[0,0], cov=[[1, rho2], [rho2, 1]]) - \
```

```
    K * exp(-r * T) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (gamma2 ** 2)) *
    norm.cdf(d4) * deriv_gamma2)
    return greek_delta2_value

#Example usage:
S1 = 100   # Current S1 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
    'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for
    S2 in asset2],
    'Delta': [greek_delta2(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in
    asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 2 Price', y='Option Price', ax=axes[0], label='Option Price
    ', color='darkcyan')
axes[0].set_title('Option Price vs. Asset 2 Price')
axes[0].set_xlabel('Asset 2 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 2 Price', y='Delta', ax=axes[1], label='Delta', color='
    coral')
axes[1].set_title('Delta vs. Asset 2 Price')
axes[1].set_xlabel('Asset 2 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.14: Price of Asset 2 Greek Function of Worst of Call Option and an Example

```
#Crossgamma

def greek_crossgamma(S1, S2, K, T, r, sigma1, sigma2, rho):
    gamma1 = (log(S1/K) + (r - (sigma1**2) / 2) * T) / (sigma1 * sqrt(T))
    gamma2 = (log(S2/K) + (r - (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))
```

```
    sigma_square = (sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2
    alpha1 = gamma1 + sigma1 * sqrt(T)
    alpha2 = (log(S2/S1) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho1 = (rho * sigma2 - sigma1)/sqrt(sigma_square)
    beta1 = gamma2 + sigma2 * sqrt(T)
    beta2 = (log(S1/S2) - (sigma_square * T) / 2) / (sqrt(T * sigma_square))
    rho2 = (rho * sigma1 - sigma2)/sqrt(sigma_square)
    d1 = (alpha2 - rho1 * alpha1) / (sqrt(1 - rho1 ** 2))
    d2 = (alpha1 - rho1 * alpha2) / (sqrt(1 - rho1 ** 2))
    d3 = (beta1 - rho2 * beta2) / (sqrt(1 - rho2 ** 2))
    d4 = (gamma2 - rho * gamma1) / (sqrt(1 - rho ** 2))
    deriv_alpha2 = 1 / (sqrt(sigma_square * T) * S2)
    deriv_beta2 = 1 / (sqrt(sigma_square * T) * S1)
    deriv_d1 = 1 / (sqrt(sigma_square * T) * S2 * sqrt(1 - rho1 ** 2))
    deriv_d2 = (-rho1) / (sqrt(sigma_square * T) * S2 * sqrt(1 - rho1 ** 2))
    deriv_d3 = (1 / (sqrt(1 - rho2 ** 2))) * ((1 / (sigma2 * S2 * sqrt(T))) +
    (rho2 / (sqrt(sigma_square * T) * S2)))
    deriv_d4 = 1 / (sigma2 * sqrt(T) * S2 * sqrt(1 - rho ** 2))

    greek_crossgamma_value = ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha2 ** 2)
    ) * norm.cdf(d2) * deriv_alpha2) + \
    (1/(sigma1 * sqrt(T))) * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (alpha1 ** 2)
    ) * norm.pdf(d1) * deriv_d1) - \
    (1/(sqrt(sigma_square * T))) * (((1/(sqrt(2 * np.pi))) * exp(-0.5 * (
    alpha2 ** 2)) * norm.cdf(d2) * (-alpha2) * deriv_alpha2) + ((1/(sqrt(2 * np
    .pi))) * exp(-0.5 * (alpha2 ** 2)) * norm.pdf(d2) * deriv_d2)) + \
    S2 * deriv_beta2 * (((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) *
    norm.pdf(d3) * deriv_d3 + (1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2))
    * norm.cdf(d3) * beta2 * (deriv_alpha2))) + \
    deriv_beta2 * ((1/(sqrt(2 * np.pi))) * exp(-0.5 * (beta2 ** 2)) * norm.cdf
    (d3)) - \
    (K/S1) * exp(-r * T) * (1/(sigma1 * sqrt(T))) * ((1/(sqrt(2 * np.pi))) *
    exp(-0.5 * (gamma1 ** 2)) * norm.pdf(d4) * deriv_d4)
    return greek_crossgamma_value

#Example usage:
S1 = 100   # Current S1 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
```

```
    'Option Price': [worst_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2],
    'Crossgamma': [greek_crossgamma(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, ax = plt.subplots(figsize=(8, 6))

data.plot(x='Asset 2 Price', y='Crossgamma', ax=ax, label='Crossgamma', color='coral')
ax.set_title('Crossgamma vs. Asset 2 Price')
ax.set_xlabel('Asset 2 Price')
ax.set_ylabel('Crossgamma')

plt.tight_layout()
plt.show()
```

Listing I.15: Crossgamma of Worst of Call Option (Derivative of Delta1 in order to Asset 2 Price) Function and an Example

```
#Best of Call Greeks

#Cega

def greek_cega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho):

    cega_bo_call = -greek_cega(S1, S2, K, T, r, sigma1, sigma2, rho)

    return cega_bo_call

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock

# Generate correlation values
correlations = np.arange(-0.99, 1, 0.1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Correlation': correlations,
    'Option Price': [best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for rho in correlations],
    'Cega': [greek_cega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho) for rho in correlations]
```

```
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Correlation', y='Option Price', ax=axes[0], label='Option Price',
     color='mediumblue')
axes[0].set_title('Option Price vs. Correlation')
axes[0].set_xlabel('Correlation')
axes[0].set_ylabel('Option Price')

data.plot(x='Correlation', y='Cega', ax=axes[1], label='Cega', color='hotpink'
    )
axes[1].set_title('Cega vs. Correlation')
axes[1].set_xlabel('Correlation')
axes[1].set_ylabel('Cega')

plt.tight_layout()
plt.show()
```

Listing I.16: Cega of Best of Call Option Function and an Example

```
#Vega

def greek_vega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho):

    d1 = (log(S2 / K) + (r + (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))

    vega_bo_call = S2 * sqrt(T) * norm.pdf(d1) - greek_vega2(S1, S2, K, T, r,
    sigma1, sigma2, rho)

    return vega_bo_call

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate volatilities for stock 2
volatility2 = np.arange(0.1, 0.61, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Volatility Asset 2': volatility2,
    'Option Price': [best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for
    sigma2 in volatility2],
```

```
    'Vega': [greek_vega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho) for sigma2 in volatility2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Volatility Asset 2', y='Option Price', ax=axes[0], label='Option Price', color='mediumblue')
axes[0].set_title('Option Price vs. Volatility Asset 2')
axes[0].set_xlabel('Volatility Asset 2')
axes[0].set_ylabel('Option Price')

data.plot(x='Volatility Asset 2', y='Vega', ax=axes[1], label='Vega', color='hotpink')
axes[1].set_title('Vega vs. Volatility Asset 2')
axes[1].set_xlabel('Volatility Asset 2')
axes[1].set_ylabel('Vega')

plt.tight_layout()
plt.show()
```

Listing I.17: Vega of Best of Call Option Function and an Example

```
#Delta

def greek_delta_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho):

    d1 = (log(S2 / K) + (r + (sigma2**2) / 2) * T) / (sigma2 * sqrt(T))

    delta_bo_call = norm.cdf(d1) - greek_delta2(S1, S2, K, T, r, sigma1, sigma2, rho)

    return delta_bo_call

#Example usage:
S1 = 100   # Current S1 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
```

```
    'Option Price': [best_of_call(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2],
    'Delta': [greek_delta_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 2 Price', y='Option Price', ax=axes[0], label='Option Price', color='mediumblue')
axes[0].set_title('Option Price vs. Asset 2 Price')
axes[0].set_xlabel('Asset 2 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 2 Price', y='Delta', ax=axes[1], label='Delta', color='hotpink')
axes[1].set_title('Delta vs. Asset 2 Price')
axes[1].set_xlabel('Asset 2 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.18: Delta of Best of Call Option Function and an Example

```
#Worst of Put Greeks

#Cega

def greek_cega_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho):

    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    cega_wo_put = -((S2 * sqrt(T) * sigma1 * sigma2 * norm.pdf(d1)) / sigma) + greek_cega(S1, S2, K, T, r, sigma1, sigma2, rho)

    return cega_wo_put

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock

# Generate correlation values
```

```
correlations = np.arange(-0.99, 1, 0.1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Correlation': correlations,
    'Option Price': [worst_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for rho in correlations],
    'Cega': [greek_cega_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for rho in correlations]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Correlation', y='Option Price', ax=axes[0], label='Option Price', color='mediumturquoise')
axes[0].set_title('Option Price vs. Correlation')
axes[0].set_xlabel('Correlation')
axes[0].set_ylabel('Option Price')

data.plot(x='Correlation', y='Cega', ax=axes[1], label='Cega', color='tan')
axes[1].set_title('Cega vs. Correlation')
axes[1].set_xlabel('Correlation')
axes[1].set_ylabel('Cega')

plt.tight_layout()
plt.show()
```

Listing I.19: Cega of Worst of Put Option Function and an Example

```
#Vega

def greek_vega_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho):

    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    vega_wo_put = ((S2 * sqrt(T) * (sigma2 - rho * sigma1) * norm.pdf(d1)) / sigma) + greek_vega2(S1, S2, K, T, r, sigma1, sigma2, rho)

    return vega_wo_put

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
rho = 0.5 # Correlation between the 2 stocks
```

```
20
21 # Generate volatilities for stock 2
22 volatility2 = np.arange(0.1, 0.61, 0.01)
23
24 # Create a DataFrame to organize the data
25 data = pd.DataFrame({
26     'Volatility Asset 2': volatility2,
27     'Option Price': [worst_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for
       sigma2 in volatility2],
28     'Vega': [greek_vega_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for
       sigma2 in volatility2]
29 })
30
31 # Plotting the graphs using pandas integration with matplotlib
32 fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
33
34 data.plot(x='Volatility Asset 2', y='Option Price', ax=axes[0], label='Option
       Price', color='mediumturquoise')
35 axes[0].set_title('Option Price vs. Volatility Asset 2')
36 axes[0].set_xlabel('Volatility Asset 2')
37 axes[0].set_ylabel('Option Price')
38
39 data.plot(x='Volatility Asset 2', y='Vega', ax=axes[1], label='Vega', color='
       tan')
40 axes[1].set_title('Vega vs. Volatility Asset 2')
41 axes[1].set_xlabel('Volatility Asset 2')
42 axes[1].set_ylabel('Vega')
43
44 plt.tight_layout()
45 plt.show()
```

Listing I.20: Vega of Worst of Put Option Function and an Example

```
1 #Delta
2
3 def greek_delta_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho):
4
5     sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
6     d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))
7
8     delta_wo_put = -1 + norm.cdf(d1) + greek_delta2(S1, S2, K, T, r, sigma1,
       sigma2, rho)
9
10    return delta_wo_put
11
12 #Example usage:
13 S1 = 100   # Current S1 stock price
14 K = 110   # Option strike price
15 T = 1 # Time to expiration in years
16 r = 0.05  # Risk-free interest rate
```

```
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
    'Option Price': [worst_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2],
    'Delta': [greek_delta_wo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 2 Price', y='Option Price', ax=axes[0], label='Option Price', color='mediumturquoise')
axes[0].set_title('Option Price vs. Asset 2 Price')
axes[0].set_xlabel('Asset 2 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 2 Price', y='Delta', ax=axes[1], label='Delta', color='tan')
axes[1].set_title('Delta vs. Asset 2 Price')
axes[1].set_xlabel('Asset 2 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.21: Delta of Worst of Put Option Function and an Example

```
#Best of Put Greeks

#Cega

def greek_cega_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho):

    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    cega_bo_put = ((S2 * sqrt(T) * sigma1 * sigma2 * norm.pdf(d1)) / sigma) + greek_cega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho)

    return cega_bo_put

```

```
#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock

# Generate correlation values
correlations = np.arange(-0.99, 1, 0.1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Correlation': correlations,
    'Option Price': [best_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for rho
     in correlations],
    'Cega': [greek_cega_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for rho
    in correlations]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Correlation', y='Option Price', ax=axes[0], label='Option Price',
     color='orangered')
axes[0].set_title('Option Price vs. Correlation')
axes[0].set_xlabel('Correlation')
axes[0].set_ylabel('Option Price')

data.plot(x='Correlation', y='Cega', ax=axes[1], label='Cega', color='
    forestgreen')
axes[1].set_title('Cega vs. Correlation')
axes[1].set_xlabel('Correlation')
axes[1].set_ylabel('Cega')

plt.tight_layout()
plt.show()
```

Listing I.22: Cega of Best of Put Option Function and an Example

```
#Vega

def greek_vega_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho):

    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    vega_bo_put = -((S2 * sqrt(T) * (sigma2 - rho * sigma1) * norm.pdf(d1)) /
    sigma) + greek_vega_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho)
```

```
9
10      return vega_bo_put
11
12  #Example usage:
13  S1 = 100  # Current S1 stock price
14  S2 = 95   # Current S2 stock price
15  K = 110   # Option strike price
16  T = 1 # Time to expiration in years
17  r = 0.05  # Risk-free interest rate
18  sigma1 = 0.2  # Volatility S1 stock
19  rho = 0.5 # Correlation between the 2 stocks
20
21  # Generate volatilities for stock 2
22  volatility2 = np.arange(0.1, 0.61, 0.01)
23
24  # Create a DataFrame to organize the data
25  data = pd.DataFrame({
26      'Volatility Asset 2': volatility2,
27      'Option Price': [best_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for
        sigma2 in volatility2],
28      'Vega': [greek_vega_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for
        sigma2 in volatility2]
29  })
30
31  # Plotting the graphs using pandas integration with matplotlib
32  fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
33
34  data.plot(x='Volatility Asset 2', y='Option Price', ax=axes[0], label='Option
        Price', color='orangered')
35  axes[0].set_title('Option Price vs. Volatility Asset 2')
36  axes[0].set_xlabel('Volatility Asset 2')
37  axes[0].set_ylabel('Option Price')
38
39  data.plot(x='Volatility Asset 2', y='Vega', ax=axes[1], label='Vega', color='
        forestgreen')
40  axes[1].set_title('Vega vs. Volatility Asset 2')
41  axes[1].set_xlabel('Volatility Asset 2')
42  axes[1].set_ylabel('Vega')
43
44  plt.tight_layout()
45  plt.show()
```

Listing I.23: Vega of Best of Put Option Function and an Example

```
1  #Delta
2
3  def greek_delta_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho):
4
5      sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
6      d1 = (log(S2/S1)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))
```

```
    delta_bo_put = -norm.cdf(d1) + greek_delta_bo_call(S1, S2, K, T, r, sigma1, sigma2, rho)

    return delta_bo_put

#Example usage:
S1 = 100   # Current S1 stock price
K = 110   # Option strike price
T = 1 # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate stock 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
    'Option Price': [best_of_put(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2],
    'Delta': [greek_delta_bo_put(S1, S2, K, T, r, sigma1, sigma2, rho) for S2 in asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 2 Price', y='Option Price', ax=axes[0], label='Option Price', color='orangered')
axes[0].set_title('Option Price vs. Asset 2 Price')
axes[0].set_xlabel('Asset 2 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 2 Price', y='Delta', ax=axes[1], label='Delta', color='forestgreen')
axes[1].set_title('Delta vs. Asset 2 Price')
axes[1].set_xlabel('Asset 2 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.24: Delta of Best of Put Option Function and an Example

```
from math import log, sqrt, exp
from scipy.stats import norm, multivariate_normal
import numpy as np
```

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from itertools import product

#Outperformance Option Pricing Function with Black Scholes

def margrabe(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    margrabe_price = S1 * norm.cdf(d1) - S2 * norm.cdf(d2)
    return margrabe_price

#Example usage:
S1 = 100  # Current S1 stock price
S2 = 95   # Current S2 stock price
T = 1     # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

margrabe_option_price = margrabe(S1, S2, T, sigma1, sigma2, rho)
print(f"The price of the outperformance option is: {margrabe_option_price:.2f}
    ")
The price of the outperformance option is: 11.61
```

Listing I.25: Outperformance Option Function and an Example

```
#In order to the stock prices

def greek_delta1(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    greek_delta1_value = norm.cdf(d1)
    return greek_delta1_value

#Example usage:
S2 = 95   # Current S2 stock price
T = 1 # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate asset 1 prices
asset1 = np.arange(20, 200, 1)

```

```
# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 1 Price': asset1,
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for S1 in asset1
    ],
    'Delta': [greek_delta1(S1, S2, T, sigma1, sigma2, rho) for S1 in asset1]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 1 Price', y='Option Price', ax=axes[0], label='Option Price
    ', color='gold')
axes[0].set_title('Option Price vs. Asset 1 Price')
axes[0].set_xlabel('Asset 1 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 1 Price', y='Delta', ax=axes[1], label='Delta', color='
    limegreen')
axes[1].set_title('Delta vs. Asset 1 Price')
axes[1].set_xlabel('Asset 1 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.26: Price of Asset 1 Greek Function of Outperformance Option and an Example

```
def greek_delta2(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    greek_delta2_value = -norm.cdf(d2)
    return greek_delta2_value

#Example usage:
S1 = 100   # Current S1 stock price
T = 1 # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate asset 2 prices
asset2 = np.arange(20, 200, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Asset 2 Price': asset2,
```

```
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for S2 in asset2],
    'Delta': [greek_delta2(S1, S2, T, sigma1, sigma2, rho) for S2 in asset2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Asset 2 Price', y='Option Price', ax=axes[0], label='Option Price', color='gold')
axes[0].set_title('Option Price vs. Asset 2 Price')
axes[0].set_xlabel('Asset 2 Price')
axes[0].set_ylabel('Option Price')

data.plot(x='Asset 2 Price', y='Delta', ax=axes[1], label='Delta', color='limegreen')
axes[1].set_title('Delta vs. Asset 2 Price')
axes[1].set_xlabel('Asset 2 Price')
axes[1].set_ylabel('Delta')

plt.tight_layout()
plt.show()
```

Listing I.27: Price of Asset 2 Greek Function of Outperformance Option and an Example

```
#In order to the maturity date

def greek_theta(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    greek_theta_value = S1 * norm.pdf(d1) *  (sigma / (2 * sqrt(T)))
    return greek_theta_value

#Example usage:
S1 = 100   # Current S1 stock price
S2 = 95   # Current S2 stock price
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate maturity dates
maturities = np.arange(1, 20, 1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Maturity': maturities,
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for T in maturities],
    'Theta': [greek_theta(S1, S2, T, sigma1, sigma2, rho) for T in maturities]
```

```
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Maturity', y='Option Price', ax=axes[0], label='Option Price',
    color='gold')
axes[0].set_title('Option Price vs. Maturity')
axes[0].set_xlabel('Maturity')
axes[0].set_ylabel('Option Price')

data.plot(x='Maturity', y='Theta', ax=axes[1], label='Theta', color='limegreen
    ')
axes[1].set_title('Theta vs. Maturity')
axes[1].set_xlabel('Maturity')
axes[1].set_ylabel('Theta')

plt.tight_layout()
plt.show()
```

Listing I.28: Maturity Time Greek Function of Outperformance Option and an Example

```
#In order to the correlation

def greek_cega(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    greek_cega_value = -S1 * norm.pdf(d1) *  ((sigma1 * sigma2 * sqrt(T)) / (
    sigma))
    return greek_cega_value

#Example usage:
S1 = 100   # Current S1 stock price
S2 = 95   # Current S2 stock price
T = 1 # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
sigma2 = 0.25  # Volatility S2 stock

# Generate correlation values
correlations = np.arange(-0.99, 1, 0.1)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Correlation': correlations,
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for rho in
    correlations],
    'Cega': [greek_cega(S1, S2, T, sigma1, sigma2, rho) for rho in
    correlations]
})
```

```
26
27 # Plotting the graphs using pandas integration with matplotlib
28 fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))
29
30 data.plot(x='Correlation', y='Option Price', ax=axes[0], label='Option Price',
       color='gold')
31 axes[0].set_title('Option Price vs. Correlation')
32 axes[0].set_xlabel('Correlation')
33 axes[0].set_ylabel('Option Price')
34
35 data.plot(x='Correlation', y='Cega', ax=axes[1], label='Cega', color='
      limegreen')
36 axes[1].set_title('Cega vs. Correlation')
37 axes[1].set_xlabel('Correlation')
38 axes[1].set_ylabel('Cega')
39
40 plt.tight_layout()
41 plt.show()
```

Listing I.29: Correlation Greek Function of Outperformance Option and an Example

```
#In order to the stock volatilities

def greek_vega1(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    greek_vega1_value = S1 * norm.pdf(d1) *  (((sigma1 - sigma2 * rho) * sqrt(
    T)) / (sigma))
    return greek_vega1_value

#Example usage:
S1 = 100   # Current S1 stock price
S2 = 95   # Current S2 stock price
T = 1 # Time to expiration in years
sigma2 = 0.25  # Volatility S2 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate volatilities for stock 1
volatility1 = np.arange(0.1, 0.61, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Volatility Asset 1': volatility1,
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for sigma1 in
    volatility1],
    'Vega': [greek_vega1(S1, S2, T, sigma1, sigma2, rho) for sigma1 in
    volatility1]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Volatility Asset 1', y='Option Price', ax=axes[0], label='Option
    Price', color='gold')
axes[0].set_title('Option Price vs. Volatility Asset 1')
axes[0].set_xlabel('Volatility Asset 1')
axes[0].set_ylabel('Option Price')

data.plot(x='Volatility Asset 1', y='Vega', ax=axes[1], label='Vega', color='
    limegreen')
axes[1].set_title('Vega vs. Volatility Asset 1')
axes[1].set_xlabel('Volatility Asset 1')
axes[1].set_ylabel('Vega')

plt.tight_layout()
plt.show()
```

Listing I.30: Volatility of Asset 1 Greek Function of Outperformance Option and an Example

```
def greek_vega2(S1, S2, T, sigma1, sigma2, rho):
    sigma = sqrt((sigma1**2) + (sigma2**2) - 2 * rho * sigma1 * sigma2)
    d1 = (log(S1/S2)+ ((sigma**2) / 2) * T) / (sigma * sqrt(T))

    greek_vega2_value = S1 * norm.pdf(d1) *  (((sigma2 - sigma1 * rho) * sqrt(T)) / (sigma))
    return greek_vega2_value

#Example usage:
S1 = 100   # Current S1 stock price
S2 = 95   # Current S2 stock price
T = 1 # Time to expiration in years
sigma1 = 0.2  # Volatility S1 stock
rho = 0.5 # Correlation between the 2 stocks

# Generate volatilities for stock 2
volatility2 = np.arange(0.1, 0.61, 0.01)

# Create a DataFrame to organize the data
data = pd.DataFrame({
    'Volatility Asset 2': volatility2,
    'Option Price': [margrabe(S1, S2, T, sigma1, sigma2, rho) for sigma2 in volatility2],
    'Vega': [greek_vega2(S1, S2, T, sigma1, sigma2, rho) for sigma2 in volatility2]
})

# Plotting the graphs using pandas integration with matplotlib
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

data.plot(x='Volatility Asset 2', y='Option Price', ax=axes[0], label='Option Price', color='gold')
axes[0].set_title('Option Price vs. Volatility Asset 2')
axes[0].set_xlabel('Volatility Asset 2')
axes[0].set_ylabel('Option Price')

data.plot(x='Volatility Asset 2', y='Vega', ax=axes[1], label='Vega', color='limegreen')
axes[1].set_title('Vega vs. Volatility Asset 2')
axes[1].set_xlabel('Volatility Asset 2')
axes[1].set_ylabel('Vega')

plt.tight_layout()
plt.show()
```

Listing I.31: Volatility of Asset 2 Greek Function of Outperformance Option and an Example

```
import numpy as np
from scipy.stats import norm, gamma, multivariate_normal
```

```python
from scipy.special import gamma as gamma_function, psi
import math
import matplotlib.pyplot as plt
import pandas as pd
import time
from scipy.integrate import simps

#Monte Carlo Simulation

def basket_option_price(S0, K, r, T, sigma, N, n_assets, weights, theta, rho):
    total_payoff = 0
    total_payoff_vector = []
    mean = np.zeros(n_assets)
    cov = rho

    for _ in range(N):
        # Generate random price paths for each asset
        asset_prices = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * multivariate_normal.rvs(mean, cov))

        # Calculate the weighted average price of the assets in the basket
        weighted_average_price = np.dot(asset_prices, weights)

        # Calculate the payoff of the basket option
        payoff = max(theta * (weighted_average_price - K), 0)

        # Accumulate the payoffs
        total_payoff += payoff
        total_payoff_vector.append(payoff)

    # Discount the total payoff back to present value
    discounted_payoff = total_payoff * np.exp(-r * T)

    # Compute option price
    option_price = discounted_payoff / N
    std_payoff = np.exp(-r * T) * np.std(total_payoff_vector)
    return option_price, std_payoff

# Example usage
np.random.seed(57724)
S0 = np.array([100, 100, 100, 100])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 10000  # Number of Monte Carlo repetitions
```

```
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option

option_price = basket_option_price(S0, K, r, T, sigma, N, n_assets, weights,
    theta, rho)
print("Basket option price and standard deviation of the values:",
    option_price)
Basket option price and standard deviation of the values: (27.60752366820115,
    67.96972952908918)
```

Listing I.32: Basket Option Price using Monte Carlo Simulation and an Example

```
#Levy's log-normal moment matching

def levy_method(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
    ] * sigma[j] * rho[i][j] * T)
    m = 2 * np.log(M) - 0.5 * np.log(V_square)
    v_square = np.log(V_square) - 2 * np.log(M)
    d1 = (m - np.log(K) + v_square) / (np.sqrt(v_square))
    d2 = d1 - np.sqrt(v_square)

    value_basket = Df * (M * norm.cdf(d1) - K * norm.cdf(d2))
    return value_basket

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
     0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

option_price_levy = levy_method(S0, sigma, r, T, weights, rho, K)
print("Basket option price:", option_price_levy)
Basket option price: 28.051966214249894
```

Listing I.33: Basket Option Price using Levy's log-normal moment matching and an Example

```python
#The reciprocal gamma approximation by Milevsky and Posner

def gamma_method(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
    ] * sigma[j] * rho[i][j] * T)
    alpha = (2 * V_square - M ** 2) / (V_square - M ** 2)
    beta = (V_square - M ** 2) / (V_square * M)

    value_basket = Df * (M * gamma.cdf(1/K, a = alpha - 1, scale = beta) - K *
     gamma.cdf(1/K, a = alpha, scale = beta))
    return value_basket

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
     0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

option_price_gamma = gamma_method(S0, sigma, r, T, weights, rho, K)
print("Basket option price:", option_price_gamma)
Basket option price: 24.495775089623557
```

Listing I.34: Basket Option Price using the reciprocal gamma approximation by Milevsky and Posner and an Example

```python
#Milevsky and Posner's approximation via higher moment

def sign_function(number):
    if number == 0:
        sign = 0
    else:
        sign = math.copysign(1, number)
    return sign

def higher_moments_method(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
```

```
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights, F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i] * sigma[j] * rho[i][j] * T)
    std = np.sqrt(V_square - (M ** 2))
    third_moment = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            for k in range(len(weights)):
                third_moment += weights[i] * weights[j] * weights[k] * F[i] * F[j] * F[k] * np.exp(sigma[i] * sigma[j] * rho[i][j] * T) *  np.exp(sigma[i] * sigma[k] * rho[i][k] * T) * np.exp(sigma[j] * sigma[k] * rho[j][k] * T)
    skewness = ((third_moment - 3 * M * (std ** 2) - (M ** 3)) / (std ** 3))
    w = 0.5 * ((8 + 4 * (skewness ** 2) + 4 * np.sqrt((4 * skewness ** 2) + (skewness ** 4))) ** (1/3)) + (2 / ((8 + 4 * (skewness ** 2) + 4 * np.sqrt((4 * skewness ** 2) + (skewness ** 4))) ** (1/3))) - 1
    b = 1 / (np.sqrt(np.log(w)))
    a = 0.5 * b * np.log((w * (w - 1))/(std ** 2))
    d = sign_function(skewness)
    c = d * M - np.exp(((1 / (2 * b)) - a) / b)
    Q = a + b * np.log((K - c) / d)

    value_basket = Df * (M - K + (K - c) * norm.cdf(Q) - d * np.exp((1 - 2 * a * b) / (2 * (b ** 2))) * norm.cdf(Q - (1 / b)))
    return value_basket

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

option_price_higher_moments = higher_moments_method(S0, sigma, r, T, weights, rho, K)
print("Basket option price:", option_price_higher_moments)
Basket option price: 27.995069983977984
```

Listing I.35: Basket Option Price using Milevsky and Posner's approximation via higher moment and an Example

```
# Example usage
```

```
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho_base = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1,
    0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 300000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option

# Values to iterate for off-diagonal elements of the correlation matrix
rho_off_diagonal_values = np.arange(-0.3, 1.0, 0.1)

# Create lists to store the rho values and deltas for each method
rho_values = []
levy = []
mp_gr = []
mp_4m = []
montecarlo = []

# Iterate over the off-diagonal correlation values
for rho_value in rho_off_diagonal_values:
    # Update the correlation matrix
    rho = rho_base.copy()
    for i in range(rho.shape[0]):
        for j in range(rho.shape[1]):
            if i != j:
                rho[i, j] = rho_value

    # Calculate the delta for the updated correlation matrix
    value_levy = levy_method(S0, sigma, r, T, weights, rho, K)
    value_mp_gr = gamma_method(S0, sigma, r, T, weights, rho, K)
    value_mp_4m = higher_moments_method(S0, sigma, r, T, weights, rho, K)
    value_montecarlo = basket_option_price(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho)[0]

    # Store the rho value and deltas
    rho_values.append(rho_value)
    levy.append(value_levy)
    mp_gr.append(value_mp_gr)
    mp_4m.append(value_mp_4m)
    montecarlo.append(value_montecarlo)

# Convert lists to DataFrame
df = pd.DataFrame({
    'Levy': levy,
```

```
    'MP-GR': mp_gr,
    'MP-4M': mp_4m,
    'Monte Carlo Simulation': montecarlo
}, index=rho_values)

# Plotting Deltas using pandas
plt.figure(figsize=(8, 6))
plt.plot(df.index, df['Levy'], marker='o', linestyle='-', color='orange',
    label='Levy')
plt.plot(df.index, df['MP-GR'], marker='o', linestyle='-', color='violet',
    label='MP-GR')
plt.plot(df.index, df['MP-4M'], marker='o', linestyle='-', color='red', label=
    'MP-4M')
plt.plot(df.index, df['Monte Carlo Simulation'], marker='o', linestyle='-',
    color='lightseagreen', label='Monte Carlo')

plt.xlabel("Correlation")
plt.ylabel("Option Price")
plt.title('Basket Call Option Price Varying Correlation using Different
    Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
```

Listing I.36: Basket Call Option Price Varying Correlation using Different Methods

```
# Example usage
np.random.seed(57724)
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho_base = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1,
    0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 300000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option

# Initial vector of asset prices
S_initial = np.array([20.0, 20.0, 20.0, 20.0])
# Step by which each element in the vector will be incremented
step = 10
# Number of steps
num_steps = 19

# Create lists to store the rho values and deltas for each method
asset_price_values = []
levy = []
```

```
mp_gr = []
mp_4m = []
montecarlo = []
lower_bounds = []
upper_bounds = []

# Iterate over the number of steps
for i in range(num_steps):
    # Update the vector of asset prices
    S0 = S_initial + i * step
    # Calculate the delta for the updated correlation matrix
    value_levy = levy_method(S0, sigma, r, T, weights, rho, K)
    value_mp_gr = gamma_method(S0, sigma, r, T, weights, rho, K)
    value_mp_4m = higher_moments_method(S0, sigma, r, T, weights, rho, K)
    value_montecarlo = basket_option_price(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho)[0]
    confidence_interval_lower_bounds = basket_option_price(S0, K, r, T, sigma,
     N, n_assets, weights, theta, rho)[0] - norm.ppf(0.95) * (
    basket_option_price(S0, K, r, T, sigma, N, n_assets, weights, theta, rho)
    [1] / np.sqrt(N))
    confidence_interval_upper_bounds = basket_option_price(S0, K, r, T, sigma,
     N, n_assets, weights, theta, rho)[0] + norm.ppf(0.95) * (
    basket_option_price(S0, K, r, T, sigma, N, n_assets, weights, theta, rho)
    [1] / np.sqrt(N))
    # Store the asset prices and deltas
    asset_price_values.append(S0[0])
    levy.append(value_levy)
    mp_gr.append(value_mp_gr)
    mp_4m.append(value_mp_4m)
    montecarlo.append(value_montecarlo)
    lower_bounds.append(confidence_interval_lower_bounds)
    upper_bounds.append(confidence_interval_upper_bounds)

# Convert lists to DataFrame
df = pd.DataFrame({
    'Levy': levy,
    'MP-GR': mp_gr,
    'MP-4M': mp_4m,
    'Monte Carlo Simulation': montecarlo,
    'Lower Bounds CI': lower_bounds,
    'Upper Bounds CI': upper_bounds
}, index=asset_price_values)

# Plotting Prices using pandas
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['Levy'], marker='o', linestyle='-', color='orange',
    label='Levy')
plt.plot(df.index, df['MP-GR'], marker='o', linestyle='-', color='violet',
    label='MP-GR')
```

```
plt.plot(df.index, df['MP-4M'], marker='o', linestyle='-', color='red', label='MP-4M')
plt.plot(df.index, df['Monte Carlo Simulation'], marker='o', linestyle='-', color='lightseagreen', label='Monte Carlo')
plt.fill_between(df.index, df['Lower Bounds CI'], df['Upper Bounds CI'], color='black', alpha=1, label='95% CI')

plt.xlabel("Asset Prices")
plt.ylabel("Option Price")
plt.title('Basket Call Option Price Varying Asset Prices using Different Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
```

Listing I.37: Basket Call Option Price Varying Asset Prices using Different Methods

```
# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the basket
rho_base = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 400000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option

# Values to iterate for sigma
sigma_values = np.arange(0.1, 0.7, 0.1)

# Create lists to store the rho values and deltas for each method
volatilities_values = []
levy = []
mp_gr = []
mp_4m = []
montecarlo = []

# Iterate over the sigma values
for sigma_value in sigma_values:
    # Update the sigma vector
    sigma = np.array([sigma_value] * n_assets)
    # Calculate the delta for the updated correlation matrix
    value_levy = levy_method(S0, sigma, r, T, weights, rho, K)
    value_mp_gr = gamma_method(S0, sigma, r, T, weights, rho, K)
    value_mp_4m = higher_moments_method(S0, sigma, r, T, weights, rho, K)
```

```
    value_montecarlo = basket_option_price(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho)[0]
    # Store the volatilities and deltas
    volatilities_values.append(sigma_value)
    levy.append(value_levy)
    mp_gr.append(value_mp_gr)
    mp_4m.append(value_mp_4m)
    montecarlo.append(value_montecarlo)

# Convert lists to DataFrame
df = pd.DataFrame({
    'Levy': levy,
    'MP-GR': mp_gr,
    'MP-4M': mp_4m,
    'Monte Carlo Simulation': montecarlo
}, index=volatilities_values)

# Plotting Prices using pandas
plt.figure(figsize=(8, 6))
plt.plot(df.index, df['Levy'], marker='o', linestyle='-', color='orange',
    label='Levy')
plt.plot(df.index, df['MP-GR'], marker='o', linestyle='-', color='violet',
    label='MP-GR')
plt.plot(df.index, df['MP-4M'], marker='o', linestyle='-', color='red', label=
    'MP-4M')
plt.plot(df.index, df['Monte Carlo Simulation'], marker='o', linestyle='-',
    color='lightseagreen', label='Monte Carlo')

plt.xlabel("Volatility")
plt.ylabel("Option Price")
plt.title('Basket Call Option Price Varying Volatility using Different Methods
    ')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
```

Listing I.38: Basket Call Option Price Varying Volatilities using Different Methods

```
# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho_base = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1,
    0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
N = 300000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
```

```
theta = 1 #To determine if it is a call or a put option

# Values to iterate for strike price
strike_values = np.arange(20, 200, 10)

# Create lists to store the rho values and deltas for each method
strike_price_values = []
levy = []
mp_gr = []
mp_4m = []
montecarlo = []

# Iterate over the sigma values
for strike_price_val in strike_values:
    # Update the sigma vector
    K = strike_price_val
    # Calculate the delta for the updated correlation matrix
    value_levy = levy_method(S0, sigma, r, T, weights, rho, K)
    value_mp_gr = gamma_method(S0, sigma, r, T, weights, rho, K)
    value_mp_4m = higher_moments_method(S0, sigma, r, T, weights, rho, K)
    value_montecarlo = basket_option_price(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho)[0]
    # Store the volatilities and deltas
    strike_price_values.append(strike_price_val)
    levy.append(value_levy)
    mp_gr.append(value_mp_gr)
    mp_4m.append(value_mp_4m)
    montecarlo.append(value_montecarlo)

# Convert lists to DataFrame
df = pd.DataFrame({
    'Levy': levy,
    'MP-GR': mp_gr,
    'MP-4M': mp_4m,
    'Monte Carlo Simulation': montecarlo
}, index=strike_price_values)

# Plotting Prices using pandas
plt.figure(figsize=(8, 6))
plt.plot(df.index, df['Levy'], marker='o', linestyle='-', color='orange',
    label='Levy')
plt.plot(df.index, df['MP-GR'], marker='o', linestyle='-', color='violet',
    label='MP-GR')
plt.plot(df.index, df['MP-4M'], marker='o', linestyle='-', color='red', label=
    'MP-4M')
plt.plot(df.index, df['Monte Carlo Simulation'], marker='o', linestyle='-',
    color='lightseagreen', label='Monte Carlo')

plt.xlabel("Strike Price")
plt.ylabel("Option Price")
```

```
plt.title('Basket Call Option Price Varying Strike Price using Different
    Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
```

Listing I.39: Basket Call Option Price Varying Strike Price using Different Methods

```
#Greeks with Levy's log-normal moment matching

#In order to the stock prices

def access_vector_entry(vector):
    # Ask the user for the index of the entry they want to access
    index = int(input("Enter the index of the entry you want to access: "))

    # Check if the index is valid
    if 0 <= index < len(vector):
        # Access the specified entry of the vector
        specific_entry = vector[index]
        return index, specific_entry
    else:
        return None

def delta(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
    ] * sigma[j] * rho[i][j] * T)
    m = 2 * np.log(M) - 0.5 * np.log(V_square)
    v_square = np.log(V_square) - 2 * np.log(M)
    part1_deriv_V_square = 0
    for j in range(len(weights)):
        if j != access_vector_entry(weights)[0]:
            part1_deriv_V_square += 2 * access_vector_entry(weights)[1] *
    weights[j] * np.exp(r * T) * F[j] * np.exp(access_vector_entry(sigma)[1] *
    sigma[j] * rho[access_vector_entry(rho)[0]][j] * T)
        else:
            part1_deriv_V_square += 2 * (access_vector_entry(weights)[1] ** 2)
     * np.exp(r * T) * access_vector_entry(F)[1] * np.exp((access_vector_entry(
    sigma)[1] ** 2) * T)
    deriv_V_square = ((part1_deriv_V_square) / V_square) - 2 * ((
    access_vector_entry(weights)[1] * np.exp(r * T)) / M)
```

```
    d1 = (m - np.log(K) + v_square) / (np.sqrt(v_square))
    deriv_root_v_square = 0.5 * (1 / (np.sqrt(v_square))) * deriv_V_square

    greek_delta = Df * ((access_vector_entry(weights)[1] * np.exp(r * T) *
    norm.cdf(d1) + M * norm.pdf(d1) * deriv_root_v_square))
    return greek_delta

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
     0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

greek_delta_value = delta(S0, sigma, r, T, weights, rho, K)
print("Delta value:", greek_delta_value)
Enter the index of the entry you want to access: 0
Delta value: 0.16006495776781238
```

Listing I.40: Asset Price Greek of Levy's log-normal moment matching and an Example

```
#In order to the correlation

def cega(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
    ] * sigma[j] * rho[i][j] * T)
    m = 2 * np.log(M) - 0.5 * np.log(V_square)
    v_square = np.log(V_square) - 2 * np.log(M)
    deriv_V_square = access_vector_entry(weights)[1] * access_vector_entry(
    weights)[1] * access_vector_entry(F)[1] * access_vector_entry(F)[1] *
    access_vector_entry(sigma)[1] * access_vector_entry(sigma)[1] * T * np.exp(
    sigma[access_vector_entry(sigma)[0]] * sigma[access_vector_entry(sigma)[0]]
     * rho[access_vector_entry(rho)[0]][access_vector_entry(rho)[0]] * T)
    d1 = (m - np.log(K) + v_square) / (np.sqrt(v_square))
    deriv_root_v_square = (1 / (np.sqrt(v_square) * V_square)) *
    deriv_V_square

    greek_cega = Df * M * norm.pdf(d1) * deriv_root_v_square
```

```
    return greek_cega

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

greek_cega_value = cega(S0, sigma, r, T, weights, rho, K)
print("Cega value:", greek_cega_value)
Enter the index of the entry you want to access:  0
Enter the index of the entry you want to access:  1
Cega value: 2.318447222035616
```

Listing I.41: Correlation Greek of Levy's log-normal moment matching and an Example

```
#In order to the volatility

def vega(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i] * sigma[j] * rho[i][j] * T)
    m = 2 * np.log(M) - 0.5 * np.log(V_square)
    v_square = np.log(V_square) - 2 * np.log(M)
    deriv_V_square = 0
    for j in range(len(weights)):
        if j != access_vector_entry(weights)[0]:
            deriv_V_square += 2 * access_vector_entry(weights)[1] * weights[j] * access_vector_entry(F)[1] * F[j] * sigma[j] * rho[access_vector_entry(rho)[0]][j] * T * np.exp(access_vector_entry(sigma)[1] * sigma[j] * rho[access_vector_entry(rho)[0]][j] * T)
        else:
            deriv_V_square += 2 * (access_vector_entry(weights)[1] ** 2) * (access_vector_entry(F)[1] ** 2) * access_vector_entry(sigma)[1] * T * np.exp((access_vector_entry(sigma)[1] ** 2) * T)
    d1 = (m - np.log(K) + v_square) / (np.sqrt(v_square))
    deriv_root_v_square = 0.5 * (1 / (np.sqrt(v_square) * V_square)) * deriv_V_square
```

```
24
25      greek_vega = Df * M * norm.pdf(d1) * deriv_root_v_square
26      return greek_vega
27
28 #Example usage
29 S0 = [100, 100, 100, 100]   # Initial price of the assets
30 sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
31 r = 0    # Risk-free interest rate
32 T = 5   # Time to maturity
33 weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
34 rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
       0.5, 0.5, 1]])   # Correlation matrix
35 K = 100   # Strike price
36
37 greek_vega_value = vega(S0, sigma, r, T, weights, rho, K)
38 print("Vega value:", greek_vega_value)
39 Enter the index of the entry you want to access:  0
40 Vega value: 17.34096914765987
```

Listing I.42: Volatility Greek of Levy's log-normal moment matching and an Example

```
1  #Greeks with The reciprocal gamma approximation by Milevsky and Posner
2
3  #In order to the stock prices
4
5  def greek_delta(S0, sigma, r, T, weights, rho, K):
6      Df = np.exp(-r * T)
7      F = []
8      for s0 in S0:
9          Fi = s0 * np.exp(r * T)
10         F.append(Fi)
11     M = np.dot(weights,F)
12     V_square = 0
13     for i in range(len(weights)):
14         for j in range(len(weights)):
15             V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
       ] * sigma[j] * rho[i][j] * T)
16     alpha = (2 * V_square - M ** 2) / (V_square - M ** 2)
17     beta = (V_square - M ** 2) / (V_square * M)
18     deriv_V_square = 0
19     for j in range(len(weights)):
20         if j != access_vector_entry(weights)[0]:
21             deriv_V_square += 2 * access_vector_entry(weights)[1] * weights[j]
        * np.exp(r * T) * F[j] * np.exp(access_vector_entry(sigma)[1] * sigma[j] *
        rho[access_vector_entry(rho)[0]][j] * T)
22         else:
23             deriv_V_square += 2 * (access_vector_entry(weights)[1] ** 2) * np.
       exp(r * T) * access_vector_entry(F)[1] * np.exp((access_vector_entry(sigma)
       [1] ** 2) * T)
24     x = np.linspace(0.00001, 1/K, 10000)
```

```
    function_to_integrate = np.log(x) * gamma.pdf(x, a =(alpha-1), scale = beta)
    function_to_integrate_1 = np.log(x) * gamma.pdf(x, a = alpha, scale = beta)
    deriv_g_order_alpha = simps(function_to_integrate, x) + gamma.cdf(1/K, a = (alpha-1), scale = beta) * (np.log(1/beta) - ((gamma_function(alpha - 1) * psi(alpha - 1)) / gamma_function(alpha-1)))
    deriv_g_order_beta = -gamma.cdf(1/K, a =(alpha-1), scale = beta) * ((alpha - 1)/ beta) + gamma.cdf(1/K, a = alpha, scale =beta) * (gamma_function(alpha) / (gamma_function(alpha-1) * beta))
    deriv_g_alpha = simps(function_to_integrate_1, x) + gamma.cdf(1/K, a = alpha, scale = beta) * (np.log(1/beta) - ((gamma_function(alpha) * psi(alpha)) / gamma_function(alpha)))
    deriv_g_beta = -gamma.cdf(1/K, a = alpha, scale = beta) * (alpha / beta) + gamma.cdf(1/K, a = (alpha+1), scale = beta) * (gamma_function(alpha + 1) / (gamma_function(alpha) * beta))
    deriv_alpha = ((2 * V_square * M * access_vector_entry(weights)[1] * np.exp(r * T) - (M ** 2) * deriv_V_square) / ((V_square - M ** 2) ** 2))
    deriv_beta = ((-access_vector_entry(weights)[1] * np.exp(r * T)) / (M ** 2)) - ((access_vector_entry(weights)[1] * np.exp(r * T) * V_square - M * deriv_V_square)/ (V_square ** 2))

    delta_greek = Df * (access_vector_entry(weights)[1] * np.exp(r * T) * gamma.cdf(1/K, a =(alpha-1), scale=beta) + M * (deriv_g_order_alpha * deriv_alpha + deriv_g_order_beta * deriv_beta) - K * (deriv_g_alpha * deriv_alpha + deriv_g_beta * deriv_beta))
    return delta_greek

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

value_greek_delta = greek_delta(S0, sigma, r, T, weights, rho, K)
print("Delta value:", value_greek_delta)
Enter the index of the entry you want to access:  0
Delta value: 0.14611034390053368
```

Listing I.43: Asset Price Greek of the reciprocal gamma approximation by Milevsky and Posner and an Example

```
#In order to the correlation

def greek_cega(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
```

```
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i
    ] * sigma[j] * rho[i][j] * T)
    alpha = (2 * V_square - M ** 2) / (V_square - M ** 2)
    beta = (V_square - M ** 2) / (V_square * M)
    deriv_V_square = 2 * weights[access_vector_entry(weights)[0]] * weights[
    access_vector_entry(weights)[0]] * F[access_vector_entry(F)[0]] * F[
    access_vector_entry(F)[0]] * sigma[access_vector_entry(sigma)[0]] * sigma[
    access_vector_entry(sigma)[0]] * T * np.exp(sigma[access_vector_entry(sigma
    )[0]] * sigma[access_vector_entry(sigma)[0]] * rho[access_vector_entry(rho)
    [0]][access_vector_entry(rho)[0]] * T)
    x = np.linspace(0.00001, 1/K, 10000)
    function_to_integrate = np.log(x) * gamma.pdf(x, a = (alpha-1), scale =
    beta)
    function_to_integrate_1 = np.log(x) * gamma.pdf(x, a = alpha, scale = beta
    )
    deriv_g_order_alpha = simps(function_to_integrate, x) + gamma.cdf(1/K, a =
     (alpha-1), scale = beta) * (np.log(1/beta) - ((gamma_function(alpha - 1) *
     psi(alpha - 1)) / gamma_function(alpha-1)))
    deriv_g_order_beta = -gamma.cdf(1/K, a =(alpha-1), scale = beta) * ((alpha
     - 1)/ beta) + gamma.cdf(1/K, a = alpha, scale = beta) * (gamma_function(
    alpha) / (gamma_function(alpha-1) * beta))
    deriv_g_alpha = simps(function_to_integrate_1, x) + gamma.cdf(1/K, a =
    alpha, scale = beta) * (np.log(1/beta) - ((gamma_function(alpha) * psi(
    alpha)) / gamma_function(alpha)))
    deriv_g_beta = -gamma.cdf(1/K, a = alpha, scale = beta) * ((alpha)/ beta)
    + gamma.cdf(1/K, a = (alpha+1), scale = beta) * (gamma_function(alpha + 1)
    / (gamma_function(alpha) * beta))
    deriv_alpha = -(((M ** 2) * deriv_V_square) / ((V_square - M ** 2) ** 2))
    deriv_beta = ((M * deriv_V_square)/ (V_square ** 2))

    cega_greek = Df * (M * (deriv_g_order_alpha * deriv_alpha +
    deriv_g_order_beta * deriv_beta) - K * (deriv_g_alpha * deriv_alpha +
    deriv_g_beta * deriv_beta))
    return cega_greek

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
```

```
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

value_greek_cega = greek_cega(S0, sigma, r, T, weights, rho, K)
print("Cega value:", value_greek_cega)
Enter the index of the entry you want to access:  0
Enter the index of the entry you want to access:  1
Cega value: 1.5083135100970173
```

Listing I.44: Correlation Greek of the reciprocal gamma approximation by Milevsky and Posner and an Example

```
#In order to the volatility

def greek_vega(S0, sigma, r, T, weights, rho, K):
    Df = np.exp(-r * T)
    F = []
    for s0 in S0:
        Fi = s0 * np.exp(r * T)
        F.append(Fi)
    M = np.dot(weights,F)
    V_square = 0
    for i in range(len(weights)):
        for j in range(len(weights)):
            V_square += weights[i] * weights[j] * F[i] * F[j] * np.exp(sigma[i] * sigma[j] * rho[i][j] * T)
    alpha = (2 * V_square - M ** 2) / (V_square - M ** 2)
    beta = (V_square - M ** 2) / (V_square * M)
    deriv_V_square = 0
    for j in range(len(weights)):
        if j != access_vector_entry(weights)[0]:
            deriv_V_square += 2 * access_vector_entry(weights)[1] * weights[j] * access_vector_entry(F)[1] * F[j] * sigma[j] * rho[access_vector_entry(rho)[0]][j] * T * np.exp(access_vector_entry(sigma)[1] * sigma[j] * rho[access_vector_entry(rho)[0]][j] * T)
        else:
            deriv_V_square += 2 * (access_vector_entry(weights)[1] ** 2) * (access_vector_entry(F)[1] ** 2) * access_vector_entry(sigma)[1] * T * np.exp((access_vector_entry(sigma)[1] ** 2) * T)
    x = np.linspace(0.00001, 1/K, 10000)
    function_to_integrate = np.log(x) * gamma.pdf(x, a = alpha - 1, scale = beta)
    function_to_integrate_1 = np.log(x) * gamma.pdf(x, a = alpha, scale = beta)
    deriv_g_order_alpha = simps(function_to_integrate, x) + gamma.cdf(1/K, a = alpha - 1, scale = beta) * (np.log(1/beta) - ((gamma_function(alpha - 1) * psi(alpha - 1)) / gamma_function(alpha-1)))
```

```
    deriv_g_order_beta = -gamma.cdf(1/K, a = alpha - 1, scale = beta) * ((alpha - 1)/ beta) + gamma.cdf(1/K, a = alpha, scale = beta) * (gamma_function(alpha) / (gamma_function(alpha-1) * beta))
    deriv_g_alpha = simps(function_to_integrate_1, x) + gamma.cdf(1/K, a = alpha, scale = beta) * (np.log(1/beta) - ((gamma_function(alpha) * psi(alpha)) / gamma_function(alpha)))
    deriv_g_beta = -gamma.cdf(1/K, a = alpha, scale = beta) * ((alpha)/ beta) + gamma.cdf(1/K, a = alpha + 1, scale = beta) * (gamma_function(alpha + 1) / (gamma_function(alpha) * beta))
    deriv_alpha = -(((M ** 2) * deriv_V_square) / ((V_square - M ** 2) ** 2))
    deriv_beta = ((M * deriv_V_square)/ (V_square ** 2))

    vega_greek = Df * (M * (deriv_g_order_alpha * deriv_alpha + deriv_g_order_beta * deriv_beta) - K * (deriv_g_alpha * deriv_alpha + deriv_g_beta * deriv_beta))
    return vega_greek

#Example usage
S0 = [100, 100, 100, 100]   # Initial price of the assets
sigma = [0.4, 0.4, 0.4, 0.4]   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = [0.25, 0.25, 0.25, 0.25]    # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price

value_greek_vega = greek_vega(S0, sigma, r, T, weights, rho, K)
print("Vega value:", value_greek_vega)
Enter the index of the entry you want to access:  0
Vega value: 11.281524028235635
```

Listing I.45: Volatility Greek of the reciprocal gamma approximation by Milevsky and Posner and an Example

```
#Greeks with Monte Carlo Simulation

#In order to the stock prices

def delta_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta, rho, epsilon):
    total_payoff_plus = 0
    total_payoff_minus = 0
    mean = np.zeros(n_assets)
    cov = rho
    S0_plus = S0.copy()
    S0_minus = S0.copy()
    S0_plus[access_vector_entry(S0_plus)[0]] = S0_plus[access_vector_entry(S0_plus)[0]] + epsilon
```

```
    S0_minus[access_vector_entry(S0_minus)[0]] = S0_minus[access_vector_entry(S0_minus)[0]] - epsilon

    for _ in range(N):
        # Generate random price paths for each asset
        normal_multivariate = multivariate_normal.rvs(mean, cov)
        asset_prices_plus = S0_plus * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * normal_multivariate)
        asset_prices_minus = S0_minus * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * normal_multivariate)

        # Calculate the weighted average price of the assets in the basket
        weighted_average_price_plus = np.dot(asset_prices_plus, weights)
        weighted_average_price_minus = np.dot(asset_prices_minus, weights)

        # Estimate payoff for perturbed basket prices
        payoff_plus = max(weighted_average_price_plus - K, 0)
        payoff_minus = max(weighted_average_price_minus - K, 0)

        # Accumulate the payoffs
        total_payoff_plus += payoff_plus
        total_payoff_minus += payoff_minus

    # Estimate Delta using finite differences
    delta = (total_payoff_plus - total_payoff_minus) / (2 * epsilon)

    # Compute Delta
    delta_estimated = delta / N

    return delta_estimated

# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 5000000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

delta_option = delta_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta, rho, epsilon)
print("Delta option:", delta_option)
```

```
Enter the index of the entry you want to access:  0
Delta option: 0.15985148601680993
```

Listing I.46: Asset Price Greek using Monte Carlo Simulation and an Example

```
#In order to the correlation

def cega_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta, rho, epsilon):
    total_payoff_plus = 0
    total_payoff_minus = 0
    mean = np.zeros(n_assets)
    cov = np.identity(n_assets)
    cov_plus = rho.copy()
    cov_minus = rho.copy()
    cov_plus[access_vector_entry(cov_plus)[0]][access_vector_entry(cov_plus)[0]] = cov_plus[access_vector_entry(cov_plus)[0]][access_vector_entry(cov_plus)[0]] + epsilon
    cov_plus[access_vector_entry(cov_plus)[0]][access_vector_entry(cov_plus)[0]] = cov_plus[access_vector_entry(cov_plus)[0]][access_vector_entry(cov_plus)[0]] + epsilon
    cov_minus[access_vector_entry(cov_minus)[0]][access_vector_entry(cov_minus)[0]] = cov_minus[access_vector_entry(cov_minus)[0]][access_vector_entry(cov_minus)[0]] - epsilon
    cov_minus[access_vector_entry(cov_minus)[0]][access_vector_entry(cov_minus)[0]] = cov_minus[access_vector_entry(cov_minus)[0]][access_vector_entry(cov_minus)[0]] - epsilon
    cov_plus = np.linalg.cholesky(cov_plus)
    cov_minus = np.linalg.cholesky(cov_minus)

    for _ in range(N):
        # Generate random price paths for each asset
        normal_multivariate = multivariate_normal.rvs(mean, cov)
        asset_prices_plus = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * np.dot(cov_plus,normal_multivariate))
        asset_prices_minus = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * np.dot(cov_minus,normal_multivariate))

        # Calculate the weighted average price of the assets in the basket
        weighted_average_price_plus = np.dot(asset_prices_plus, weights)
        weighted_average_price_minus = np.dot(asset_prices_minus, weights)

        # Estimate payoff for perturbed basket prices
        payoff_plus = max(weighted_average_price_plus - K, 0)
        payoff_minus = max(weighted_average_price_minus - K, 0)

        # Accumulate the payoffs
        total_payoff_plus += payoff_plus
        total_payoff_minus += payoff_minus

```

```
    # Estimate Cega using finite differences
    cega = (total_payoff_plus - total_payoff_minus) / (2 * epsilon)

    # Compute Cega
    cega_estimated = cega / N

    return cega_estimated

# Example usage
start = time.time()
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho = np.array([[1, 0.90, 0.90, 0.90],[0.90, 1, 0.90, 0.90],[0.90, 0.90, 1,
    0.90],[0.90, 0.90, 0.90, 1]])   # Correlation matrix
K = 100   # Strike price
N = 10000000  # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

cega_option = cega_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta
    , rho, epsilon)
end = time.time()
time_execution = end - start
print("Cega option:", cega_option)
print(f"The function took {time_execution:.5f} seconds to execute.")
Enter the index of the entry you want to access:  0
Enter the index of the entry you want to access:  1
Cega option:  2.376090
The function took 937.54998 seconds to execute.
```

Listing I.47: Correlation Greek using Monte Carlo Simulation and an Example

```
#In order to the volatility

def vega_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta, rho,
    epsilon):
    total_payoff_plus = 0
    total_payoff_minus = 0
    mean = np.zeros(n_assets)
    cov = rho
    sigma_plus = sigma.copy()
    sigma_minus = sigma.copy()
    sigma_plus[access_vector_entry(sigma_plus)[0]] = sigma_plus[
    access_vector_entry(sigma_plus)[0]] + epsilon
```

```
    sigma_minus[access_vector_entry(sigma_minus)[0]] = sigma_minus[access_vector_entry(sigma_minus)[0]] - epsilon

    for _ in range(N):
        # Generate random price paths for each asset
        normal_multivariate = multivariate_normal.rvs(mean, cov)
        asset_prices_plus = S0 * np.exp((r - 0.5 * sigma_plus**2) * T + sigma_plus * np.sqrt(T) * normal_multivariate)
        asset_prices_minus = S0 * np.exp((r - 0.5 * sigma_minus**2) * T + sigma_minus * np.sqrt(T) * normal_multivariate)

        # Calculate the weighted average price of the assets in the basket
        weighted_average_price_plus = np.dot(asset_prices_plus, weights)
        weighted_average_price_minus = np.dot(asset_prices_minus, weights)

        # Estimate payoff for perturbed basket prices
        payoff_plus = max(weighted_average_price_plus - K, 0)
        payoff_minus = max(weighted_average_price_minus - K, 0)

        # Accumulate the payoffs
        total_payoff_plus += payoff_plus
        total_payoff_minus += payoff_minus

    # Estimate Vega using finite differences
    vega = (total_payoff_plus - total_payoff_minus) / (2 * epsilon)

    # Compute Vega
    vega_estimated = vega / N

    return vega_estimated

# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 5000000  # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

vega_option = vega_monte_carlo(S0, K, r, T, sigma, N, n_assets, weights, theta, rho, epsilon)
print("Vega option:", vega_option)
```

```
Enter the index of the entry you want to access:  0
Vega option: 17.213066855820717
```

Listing I.48: Volatility Greek using Monte Carlo Simulation and an Example

```
# Example usage
np.random.seed(57724)
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
     0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 500000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

# Initial vector of asset prices
S_initial = np.array([20.0, 20.0, 20.0, 20.0])
# Step by which each element in the vector will be incremented
step = 10
# Number of steps
num_steps = 19

# Create lists to store the asset prices and deltas
asset_prices = []
delta_levy = []
delta_mp_gr = []
delta_montecarlo = []

# Iterate over the number of steps
for i in range(num_steps):
    # Update the vector of asset prices
    S = S_initial + i * step
    # Calculate the delta for the updated vector of asset prices
    delta_value_levy = delta(S, sigma, r, T, weights, rho, K)
    delta_value_mp_gr = greek_delta(S, sigma, r, T, weights, rho, K)
    delta_value_montecarlo = delta_monte_carlo(S, K, r, T, sigma, N, n_assets,
     weights, theta, rho, epsilon)
    # Store the asset prices and deltas
    asset_prices.append(S[0])  # Use the first asset price as the x-axis value
    delta_levy.append(delta_value_levy)
    delta_mp_gr.append(delta_value_mp_gr)
    delta_montecarlo.append(delta_value_montecarlo)

# Convert lists to DataFrame
df_deltas = pd.DataFrame({
```

```
    'Levy': delta_levy,
    'MP-GR': delta_mp_gr,
    'Monte Carlo Simulation': delta_montecarlo
}, index=asset_prices)

# Plotting Deltas using pandas
plt.figure(figsize=(8, 6))
plt.plot(df_deltas.index, df_deltas['Levy'], marker='o', linestyle='-', color=
    'orange', label='Levy')
plt.plot(df_deltas.index, df_deltas['MP-GR'], marker='o', linestyle='-', color
    ='violet', label='MP-GR')
plt.plot(df_deltas.index, df_deltas['Monte Carlo Simulation'], marker='o',
    linestyle='-', color='lightseagreen', label='Monte Carlo')

plt.xlabel("Asset Price")
plt.ylabel("Delta")
plt.title('Delta of a Basket Call Option using Different Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
Enter the index of the entry you want to access:  0
```

Listing I.49: Delta of a Basket Call Option using Different Methods

```
# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
sigma = np.array([0.4, 0.4, 0.4, 0.4])   # Volatility of the assets
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho_base = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1,
    0.5],[0.5, 0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 10000000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

# Values to iterate for off-diagonal elements of the correlation matrix
rho_off_diagonal_values = np.arange(-0.3, 1.0, 0.1)

# Create lists to store the rho values and deltas for each method
rho_values = []
cega_levy = []
cega_mp_gr = []
cega_montecarlo = []

# Iterate over the off-diagonal correlation values
```

```
for rho_value in rho_off_diagonal_values:
    # Update the correlation matrix
    rho = rho_base.copy()
    for i in range(rho.shape[0]):
        for j in range(rho.shape[1]):
            if i != j:
                rho[i, j] = rho_value

    # Calculate the delta for the updated correlation matrix
    cega_value_levy = cega(S0, sigma, r, T, weights, rho, K)
    cega_value_mp_gr = greek_cega(S0, sigma, r, T, weights, rho, K)
    cega_value_montecarlo = cega_monte_carlo(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho, epsilon)

    # Store the rho value and deltas
    rho_values.append(rho_value)
    cega_levy.append(cega_value_levy)
    cega_mp_gr.append(cega_value_mp_gr)
    cega_montecarlo.append(cega_value_montecarlo)

# Convert lists to DataFrame
df_cegas = pd.DataFrame({
    'Levy': cega_levy,
    'MP-GR': cega_mp_gr,
    'Monte Carlo Simulation': cega_montecarlo
}, index=rho_values)

# Plotting Deltas using pandas
plt.figure(figsize=(8, 6))
plt.plot(df_cegas.index, df_cegas['Levy'], marker='o', linestyle='-', color='
    orange', label='Levy')
plt.plot(df_cegas.index, df_cegas['MP-GR'], marker='o', linestyle='-', color='
    violet', label='MP-GR')
plt.plot(df_cegas.index, df_cegas['Monte Carlo Simulation'], marker='o',
    linestyle='-', color='lightseagreen', label='Monte Carlo')

plt.xlabel("Correlation")
plt.ylabel("Cega")
plt.title('Cega of a Basket Call Option using Different Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
Enter the index of the entry you want to access:  0
Enter the index of the entry you want to access:  1
```

Listing I.50: Cega of a Basket Call Option using Different Methods

```
# Example usage
np.random.seed(57724)
S0 = np.array([100.0, 100.0, 100.0, 100.0])   # Initial price of the assets
```

```
r = 0    # Risk-free interest rate
T = 5   # Time to maturity
weights = np.array([0.25, 0.25, 0.25, 0.25])   # Weights of assets in the
    basket
rho = np.array([[1, 0.5, 0.5, 0.5],[0.5, 1, 0.5, 0.5],[0.5, 0.5, 1, 0.5],[0.5,
     0.5, 0.5, 1]])   # Correlation matrix
K = 100   # Strike price
N = 5000000    # Number of Monte Carlo repetitions
n_assets = 4  # Number of assets in the basket
theta = 1 #To determine if it is a call or a put option
epsilon = 0.01

# Values to iterate for sigma
sigma_values = np.arange(0.1, 0.7, 0.1)

# Create lists to store the sigma values and cegas for each method
sigma_store = []
vega_levy = []
vega_mp_gr = []
vega_montecarlo = []

# Iterate over the sigma values
for sigma_value in sigma_values:
    # Update the sigma vector
    sigma = np.array([sigma_value] * n_assets)

    # Calculate the cega for the updated sigma vector
    vega_value_levy = vega(S0, sigma, r, T, weights, rho, K)
    vega_value_mp_gr = greek_vega(S0, sigma, r, T, weights, rho, K)
    vega_value_montecarlo = vega_monte_carlo(S0, K, r, T, sigma, N, n_assets,
    weights, theta, rho, epsilon)

    # Store the sigma value and cegas
    sigma_store.append(sigma_value)
    vega_levy.append(vega_value_levy)
    vega_mp_gr.append(vega_value_mp_gr)
    vega_montecarlo.append(vega_value_montecarlo)

# Convert lists to DataFrame
df_vegas = pd.DataFrame({
    'Levy': vega_levy,
    'MP-GR': vega_mp_gr,
    'Monte Carlo Simulation': vega_montecarlo
}, index=sigma_store)

# Plotting Deltas using pandas
plt.figure(figsize=(8, 6))
plt.plot(df_vegas.index, df_vegas['Levy'], marker='o', linestyle='-', color='
    orange', label='Levy')
```

```
plt.plot(df_vegas.index, df_vegas['MP-GR'], marker='o', linestyle='-', color='violet', label='MP-GR')
plt.plot(df_vegas.index, df_vegas['Monte Carlo Simulation'], marker='o', linestyle='-', color='lightseagreen', label='Monte Carlo')

plt.xlabel("Volatility Asset")
plt.ylabel("Vega")
plt.title('Vega of a Basket Call Option using Different Methods')
plt.grid(False)  # Enable gridlines if you want them
plt.legend()
plt.show()
Enter the index of the entry you want to access:  0
```

Listing I.51: Vega of a Basket Call Option using Different Methods