Pricing of bermudan basket options

M203 Numerical finance project

Marchessaux François, Louis Faverjon, Collin Thibault

1 Introduction

This project will aim at implementing various simulation enhancement techniques to price *bermudan basket* financial derivatives. We will explore several random number generation methods with some variance reduction applications, finally ran through a *Monte-Carlo* simulation scheme. These concepts will be applied in *C++* and the results will be compared to findings for vanilla options.

This paper will first theoretically introduce the **financial** concepts tackled in the study. The **mathematical** methods used in the research will then be defined and illustrated. Finally, we will detail the coding **pipeline** that led us to our pricing results for several types of options. We will thus qualitatively illustrate our findings and propose potential improvements to our implemented pipeline. All code can be found within the GitHub: *thibaultcoo/bermudan-basket-pricing*.

Contents

2 Algorithm user guide

This section will allow the user to grasp within a few paragraphs what we aimed at producing in the code.

2.1 Overall code narrative

We aim at pricing multi-dimensional options using a general *Monte-Carlo* method first, for which the randomness is captured within a *number generators* class. The random sequence returned by a chosen generator will then allow the construction of a multi-dimensional normal probability distribution for all the underlyings of the option. A process is then built on top of this random pattern to model the joint trajectory of the basket.

As will be further explained in later sections, *Monte-Carlo* then simulates a large number of those trajectories in order to prompt out a final price for the payoff. To reduce the variability of those results and improve the stability of the method, we will successively introduce *antithetic variables*, *control variates* as well as some *quasi Monte-Carlo* using low-discrepancy random sequences.

The algorithm runs a global batch of prices comparing all methods. This allows us to appreciate the gain in stability permitted by introducing those variance reduction methods. Some numbers will be given out in the final application sections to illustrate our findings.

2.2 Classes and input parameters

The goal was to separate the key algorithm activities into readable and comprehensive classes, they are introduced below:

- **algo.cpp** which centralizes all the pricing activity, iteratively builds the *instance* object, defines the *priceables* (priced structures) and displays the tables with the pricing results.
- **instantiating.cpp** which creates a resolution object that stores the option and algorithm input parameters, as well as the generated random sequences and the diffused schemes (*Euler* and *Milstein*).
- **pricing.cpp** which encompasses in a single object all the pricing results for a single option, gathering both *European* and *Bermudan* versions. It stacks the prices, variances and confidence intervals.
- **displaying.cpp** which simply prettifies the results of our *priceables* into a readable table prompted in the terminal. It also generates a table for the input parameters used for the resolution.

The four blocks defined above are the front of our architecture to increase the ease of understanding. We briefly introduce some other classes that are key to our development, such as the scheme and option definitions:

- **BSEuler.cpp** which centralizes the simulation of underlying trajectories following the *Euler* scheme. This is where we introduce the variance reduction methods defined before, acting directly on the path diffusion.
- **BlackScholes.cpp** which is a baseline for the path creation of the underlying. Following *Black* and *Scholes* theory, the multi-dimensional diffusion is built on top of a correlated geometric *Brownian* motion.
- **BermudanBasket.cpp** which creates an object for such an option, and allows to directly solve multiple prices, considering the use of variance reduction techniques or not. A needed variation of *Monte-Carlo* was used.
- **EuropeanBasket.cpp** which acts in a similar fashion as the other instrument. Pricing both structures side by side might prompt out interesting pricing results. The regular *Monte-Carlo* baseline was used here.
- **Options.cpp** which encompasses both options defined above, and helps in centralizing the computation of moments and confidence intervals.

The other classes regarding the generation of random sequences were studied in class [Amo24]. Finally, we decided to use those parameters for our resolution:

Table 1: Simulation Parameters

Bermudan Exercise Times (months)	1, 2, 3
Spot Prices	5, 10, 20
Rates	5%
Weights	25%, 25%, 50%
Maturity (months)	3
Strike Price	10
Number of Paths	1000
Number of Paths for Quasi-MC	500
	0.06, 0.04, 0.03
Variances Matrix	0.04, 0.07, 0.045
	0.03, 0.045, 0.05

They can be freely changed within the **instantiating.cpp** initialization. In resume, the algorithm builds an instance of a resolution containing the input parameters, random sequence generators and trajectory diffusion schemes. It then builds priceables for both *european basket* and *bermudan basket* options, which are then priced with and without variance reductions enhancements. The results are then prompted out for the user to see.

3 Theoretical foundations

In this first section, we will introduce the various financial elements that will be used further in this research. Their application will be further studied later.

3.1 Derivatives pricing baseline

In this study, we are interested in pricing both *vanilla* and *bermudan* options, with one or several underlyings. Options are financial instruments that give their buyers the right to buy or sell an underlying component at a fixed price and a pre-determined exercise date. Those contracts are based on one underlying and called *vanilla* options. Their pricing is straightforward and was popularized in closed-form as the *Black-Scholes* formula [BS73].

3.1.1 Black-Scholes theory

The pricing depends on several market parameters and relies on multiple heavy assumptions. Interest and dividend rates are considered as constants to account for financing cost and equity behavior respectively. Then in all logic, the spot and strike values as well as the time to maturity are also important variables in the pricing of an option. Volatility is the final and key parameter in the pricing of an option. Within the *Black-Scholes* framework, it is also constant.

Among the deep set of assumptions required to implement this model, non-arbitrage argument and log-normal distribution of underlying returns are probably the most vital, but they are also the farthest from the actual market reality. Traders profit on arbitrage imbalances at every instant to equilibrate the market. Moreover, the probability of strongly negative returns is higher than what the normal distribution would suggest.

This framework provides an analytical formula for options with baseline features, namely a unique underlying asset and a single exercise date. We will introduce options exhibiting enhanced features, alongside with the pricing methodology usually used to deal with those.

3.1.2 Introducing exotic option features

When there is more than a unique underlying, we talk about *basket* options, and when there are multiple discrete pre-determined exercise dates, the option is called *bermudan*. Options exhibiting those two features are usually placed within the *exotic* range of options, outside the scope of vanillas. Those options cannot be priced with the usual analytical solutions introduced above, thus we have to rely on numerical approximations to evaluate them.

Usually, those methods imply simulating a large number of underlying trajectories, computing the discounted final option payoff on each individual path, and averaging the result to retrieve an approximated price for the instrument. This method is called *Monte-Carlo* [MU49], and relies on the (weak) *Law of Large Numbers* as well as the *Central Limit* theorem. We will thus successively introduce mathematically how the multiple underlyings are jointly diffused, and then theoretically present the *Monte-Carlo* methodology.

3.1.3 Multi-dimensional underlying diffusion

In the case where the underlyings are within the equity universe, we often choose to model their random trajectories using stochastic differential equations. So, assuming S^i our spot values and B^i their *Brownian* components, with $i = [\![1, n]\!]$:

$$\mathrm{d}S_t^i = \mu_i S_t^i \mathrm{d}t + \sigma_i S_t^i \mathrm{d}B_t^i$$

We do have μ_i and σ_i the mean and volatility of an asset i. Considering the *correlation* between those two stochastic processes, we define by $\mathcal{M}$ the *covariance matrix* associated with the two Brownian motions.

$$\mathcal{M}_{i\,j} = \rho_{ij}\sigma_i\sigma_j$$

By recalling that a *Brownian motion* B with zero mean and said *covariance* can be written as AB, with A the *Cholesky* factor of $\mathcal{M}$, such that $AA^T = \mathcal{M}$, we can rewrite the system as the following.

$$\mathrm{d}S_t^i = \mu_i S_t^i \mathrm{d}t + \sum_{j=1}^{n} A_{ij} S_t^i \mathrm{d}B_t^i$$

Finally, and to allow for practical implementation, we decide to discretize such system. Let us define the following algorithm for simulating multidimensional geometric *Brownian* motions with the log-transformed price $s_t = \log(S_t)$, with T the maturity of the option and N the discretized number of steps.

$$s_t^i = s_{t-1}^i - \frac{\sigma^2}{2}\frac{T}{N} + \sqrt{\frac{T}{N}}\left(\sum_{j=1}^{n} A_{ij} Z_k^i\right)$$

Those are the discretized price dynamics for the two underlying assets on which our exotic option is to be studied. We notice that in such context, the correlation ρ is constant, which remains unrealistic but makes computations easier.

3.1.4 Monte-Carlo methodology

Monte-Carlo is a generic approach to the computation of expectations. The (weak) *Law of Large Numbers* tells that the sample mean converges to the expected value, and the *Central Limit* theorem tells how this convergence happens.

Let an infinite sequence of independent and identically distributed random variables X_i with expected value $\mu < \infty$, then the (weak) *Law of Large Numbers* states that the sample mean converges in probability to the expected value:

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{\text{p.as.}} \mu$$

Assuming the existence of the above-introduced theorem, and assuming that the variance of all the X_i are defined, then the *Central Limit* theorem poses:

$$\sqrt{n}(\bar{X}_n - \mu) \xrightarrow{\text{d}} N(0, \sigma^2)$$

The last theorem implies that, for n large, $P(|\bar{X}_n - E(X)| \leq \frac{1.96\sigma}{\sqrt{n}}) \approx 0.95$, providing a 95% confidence interval for $E(X)$:

$$\left[\bar{X}_n - \frac{1.96\sigma}{\sqrt{n}}, \bar{X}_n + \frac{1.96\sigma}{\sqrt{n}}\right]$$

We are thus able to compute the final payoff of an option for a number of diffused underlying basket trajectories, for which the expectation mathematically converges to the mean. Discounting this result under the *risk-neural* probability framework thus provides a robust approximation of the price of an option. But this method is not perfect, and the subsequent section will introduce key developments that will exhibit an even more consistent overall pricing methodology.

3.2 Monte-Carlo enhancements

We previously defined all the basic elements needed in our research. The goal is now to introduce advancements to those baseline tools, for which we will then appreciate the qualitative gains that are made for pricing derivatives.

3.2.1 Longstaff-Schwartz algorithm

The main setback of *Monte-Carlo* is its inherent inconvenience in pricing options with path dependency. For *bermudan* options for example, pricing requires comparing at each exercise date both a future and current option value. The individual having rational expectations, chooses the direction that carries the highest value. Measuring this future option value can be theoretically imagined using another *Monte-Carlo* on top of the first one.

This is extremely inefficient, and we often rather look into performing a linear regression to evaluate this future value. This method is called the *Longstaff-Schwartz* algorithm [LS01].

It involves simulating possible future price paths of the underlying asset using wisely-chosen regression coefficients, instead of simulating thousands of new dynamics. The unique predicted trajectory is then used to build a future anticipated option value, which thus helps the user make an exercise decision. Here is a step-by-step explanation of the algorithm. First, we simulate multiple paths of the underlying asset price using the multi-dimensional geometric *Brownian* motion dynamics introduced in 3.1.3.

In this context, r is the risk-free rate, σ is the volatility, 1 is the time increment, and Z_i^k is a standard normal random variable. For each path, we calculate the cash flows at each potential exercise date. For an American option, the cash flow CF_t at time t if exercised is the known *European* call option payoff at t.

At each exercise date, starting from the second-to-last and moving backwards to the first, the algorithm performs a least-squares regression of the cash flows discounted to that time against a set of basis functions $f(S_t)$, such as polynomials of the stock price. The regression estimates the expected payoff of holding the option, conditional on S_t:

$$\mathbb{E}[e^{-r\Delta t}CF_{t+\Delta t}|S_t] \approx \sum_{j=0}^{n} \beta_j f_j(S_t),$$

where β_j are the regression coefficients, n is the number of basis functions used and $\Delta t = 1$. At each exercise date t, we compare the immediate exercise value with the continuation value estimated by the regression and act on it:

$$\underline{\text{Exercise if}}\ \max(S_t - K, 0)\ \text{(for call)}\ > \sum_{j=0}^{n} \beta_j f_j(S_t).$$

The option value is thus the expected discounted payoff over all paths, accounting for the possibility of early exercise based on the regression estimates at each decision point.

3.2.2 Pseudo control variates

Some enhancements to *Monte-Carlo* techniques also involve reducing the variance of the expected value of a random variable. This improves the reliability and stability of the technique. Their wide use in financial modelling [Car+99] demonstrate the effectiveness of such techniques.

Pseudo control variates combine the primary random variable of interest with a correlated auxiliary random variable, with an unknown analytical relationship linking both variables. The auxiliary random variable, also called control variate, is chosen so as to mimick the behavior of the primary variable.

Consider a random variable X representing the quantity of interest, and a control variable Y whose expected value $E(Y)$ is known. The principle of control variates involves forming a new estimator:

$$Z = X + \beta(Y - E(Y)),$$

where β is a parameter that can be optimized to minimize the variance of Z. The optimal value of β, denoted as β^*, is given by:

$$\beta^* = -\frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(Y)}.$$

Substituting β^* back into the equation for Z yields the control variate estimator with reduced variance. The variance of the new estimator Z is:

$$\mathrm{Var}(Z) = \mathrm{Var}(X) - \frac{\mathrm{Cov}^2(X,Y)}{\mathrm{Var}(Y)}.$$

In the context of *basket bermudan* options, suppose X represents the simulated payoff of the option, and we choose a simpler but related option Y as the control variate. Assume Y is a *European basket* option whose expected payoff $E(Y)$ can be calculated analytically or through a simpler model. The control variate estimator for the *bermudan* option payoff then becomes:

$$Z = X + \beta^*(Y - E(Y)),$$

where β^* is adjusted to minimize the variance based on simulated data.

3.2.3 Antithetic variables

We previously saw that combining variables help in reducing the variance of a simulation. Another method implies modifying the primary variable while maintaining the same distributional properties, and then pairing the modified variable with its original counterpart. The newly introduced variable was coined the *antithetic* variable [Gla03].

The variance of the estimator can thus be reduced because each tends to cancel out the fluctuations of the other. This modification is usually characterized by a rotation or a change is the sign.

Suppose X and $-X$ two Gaussian random variables with the same mean and variance. For variance reduction, we define a new estimator $X' = \frac{X+(-X)}{2}$, and it follows that:

$$V(X') = \frac{V(X) + \mathrm{cov}(X,-X)}{2}.$$

Given that X and $-X$ are perfectly negatively correlated as $\mathrm{cov}(X,-X) = -V(X)$ the variance of the estimator reduces to:

$$V(X') = \frac{V(X) - V(X)}{2} = 0.$$

For practical applications in financial modeling, such as simulating the payoff of a European basket call option, the payoff function can be modeled as:

$$h(X) = \left(\sum_{i=1}^{d} \alpha_i S_i^T(X) - K \right)^+ ,$$

where $S_i^T(X)$ represents the price of the i-th asset influenced by the random variable X at time T, α_i are the portfolio weights, and K is the strike price.

Using the antithetic variables method, we simulate X and $-X$ and calculate the payoff for both, thus estimating the option price by averaging these payoffs:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} \frac{h(X_i) - h(-X_i)}{2}.$$

This estimator, $\bar{X}$, is more efficient due to its lower variance while retaining the same expected value as the original simulation model. This approach is especially advantageous for options where the payoff depends on the path and multiple valuation points, such as in *bermudan* options.

3.2.4 Quasi Monte-Carlo

This method is an enhancement over the traditional *Monte-Carlo* approach that employs *low-discrepancy* sequences instead of random numbers. These sequences are designed to cover the multidimensional space more uniformly, leading to faster convergence rates in numerical integration, particularly useful in high-dimensional settings. Some commonly used *low-discrepancy* sequences are the Sobol [Sob67], Halton [Hal60], or Faure [Fau82] sequences.

These sequences ensure that the points are distributed more evenly across the integration domain compared to purely random sequences. The error in numerical integration typically reduces at a rate of:

$$\frac{(\log N)^d}{N}$$

where N is the number of points and d is the dimensionality of the integral, compared to the $\frac{1}{\sqrt{N}}$ rate observed with random sequences. The ability of quasi *Monte-Carlo* to improve convergence rates is crucial for efficiently pricing exotic options, as it allows for achieving higher accuracy with fewer simulation runs:

$$E(f) \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i),$$

where f is the payoff function and $\mathbf{x}_i$ are the points generated by a low-discrepancy sequence. The method's efficacy in high-dimensional spaces (due to the assets and exercise dates) ensures a more representative sampling of the underlying asset price paths:

$$\prod_{i=1}^{d}[a_i, b_i] \ni (x_1, x_2, \ldots, x_d),$$

which directly influences the pricing accuracy for *bermudan* options namely.

3.3 Random number generation algorithms

The stochasticity in the diffusion of our underlying trajectory is modelled with *Brownian* motions, following *Black-Scholes* theory. We thus need some random sequences. In this paper, we will start from the *uniform* distribution, which can be generated both with *pseudo-random* or *quasi-random* sequences.

Whereas pseudo-random sequences exhibits strong statistical randomness, quasi-random numbers have a *low-discrepancy* property which assures an evenly-distributed sequence, preventing clusters and increasing the spread of values.

3.3.1 Pseudo-random sequences

In this study we use *Ecuyer combined* to generate those sequences. They are an application of the *combined Linear Congruential* method, which combines several multiplicative congruential generators with the objective of attaining satisfying properties. Linear Congruential Generators (LCGs) are defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m,$$

where a, c, and m are constants. The choice of these constants significantly affects the quality of the random numbers generated.

The Ecuyer Combined method specifically utilizes two such generators with carefully selected parameters to avoid correlation between the sequences and to increase the period dramatically. The method can be described by the equations:

$$X_{n+1} = (a_1X_n + c_1) \mod m_1,$$
$$Y_{n+1} = (a_2Y_n + c_2) \mod m_2,$$

where (X_n) and (Y_n) are the sequences generated by the first and second LCGs, respectively. The combined sequence Z_n is then typically calculated using a combination formula such as, where m is usually the smaller of m_1 and m_2.

$$Z_n = (X_n - Y_n) \mod m,$$

The period of the combined generator is the least common multiple (LCM) of the periods of the individual generators, under certain conditions. If m_1 and m_2 are co-prime (i.e., their greatest common divisor (GCD) is 1), the period of Z_n can be maximized to $\text{lcm}(m_1 - 1, m_2 - 1)$.

The choice of parameters a_1, c_1, m_1 and a_2, c_2, m_2 must be made carefully to ensure the statistical independence of the sequences (X_n) and (Y_n) and to maximize the period and uniformity of the distribution of Z_n.

The Ecuyer Combined method provides several advantages. It increases period, making it suitable for simulations that require a large number of random numbers. It also improves uniformity and randomness properties, while reducing correlation between outputs compared to individual LCGs.

3.3.2 Quasi-random sequences

Reducing the variance further can be done by introducing quasi-random sequences to strengthen the uniform repartition of our randomly generated values. As seen in class [Amo24], we implement a *Van der Corput* sequence [Van35].

The sequence is generated by reversing the digits of a number in a given base. Consider a sequence in base b. The nth term of the Van der Corput sequence, denoted $\phi_b(n)$, is computed by reversing the digits of the integer n in its base b representation. If n has the following base b representation:

$$n = d_k b^k + d_{k-1} b^{k-1} + \cdots + d_1 b + d_0,$$

where $0 \leq d_i < b$ for all i, then $\phi_b(n)$ is defined as:

$$\phi_b(n) = \frac{d_0}{b} + \frac{d_1}{b^2} + \cdots + \frac{d_k}{b^{k+1}}.$$

This sequence is particularly noted for its low-discrepancy, which means the sequence fills the unit interval more uniformly than uncorrelated random points. This property is quantitatively expressed through the "discrepancy" measure, which is small for sequences like Van der Corput compared to purely random sequences. The star discrepancy D^* of the first N terms of the sequence is typically of the order $O\left(\frac{\log N}{N}\right)$.

4 European basket pricing application

In the next two sections, we are going to implement iteratively the predefined methods within a *C++* pricing algorithm.

4.1 Raw pricing

To do..

4.2 Enhanced pricing

To do..

5 Bermudan basket pricing application

To do..

5.1 Raw pricing

To do..

5.2 Enhanced pricing

To do..

6 Conclusion

Throughout this project, we dived into the pricing of multi-dimensional underlying options and studied the positive influence of *Monte-Carlo* variance enhancing features. Within a *C++* framework, we applied notions of *control variates* and *antithetic variables* to further reduce the variance of the results of our simulations. Furthermore, we implemented some quasi-random sequences and succesfully observed a significant increase in the robustness of our results.

Potential improvements to our method would be the addition of other exotic features, such as *barriers* to consider discontinuities in the payoff. This choice would be motivated by the massive popularity of barrier options in the structured products universe, especially in continental Europe.

Advancements regarding the further variance reduction for the simulations could be attained with the use of more advanced low-discrepancy sequences, such as [Fau82]. This method is particularly effective when the dimension is a prime number related to the base of the sequence. They key idea is to permute the digits of the Van der Corput [Van35] sequence in a specific way that depends on the factorial number system. This sequence is known for its improved distribution properties and its increased performance for high dimension problems.

References

[Van35] J.G. Van der Corput. *Verteilungsfunktionen I.* Vol. 38. Netherlands Academy of Arts and Sciences, 1935, pp. 813–821.

[MU49] Nicholas Metropolis and Stanislaw Ulam. *The Monte Carlo method.* Vol. 44. 247. Taylor & Francis, 1949.

[Hal60] John H. Halton. *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals.* Vol. 2. 1. Springer, 1960.

[Sob67] Ilya M. Sobol. *Distribution of points in a cube and approximate evaluation of integrals.* Vol. 7. 4. Pergamon, 1967.

[BS73] Fischer Black and Myron Scholes. *The Pricing of Options and Corporate Liabilities.* Vol. 81. 3. University of Chicago Press, 1973.

[Fau82] Henri Faure. *Discrepance de suites associees a un systeme de numeration (en dimension s).* Vol. 41. 4. Institute of Mathematics Polish Academy of Sciences, 1982.

[Car+99] Murray Carlson et al. *Effective Monte Carlo variance reduction techniques for valuing American options.* Vol. 45. 7. INFORMS, 1999.

[LS01] Francis A Longstaff and Eduardo S Schwartz. *Valuing American options by simulation: A simple least-squares approach.* Vol. 14. 1. Oxford University Press, 2001.

[Gla03] Paul Glasserman. *Monte Carlo methods in financial engineering.* Vol. 53. 2003.

[Amo24] Kaiza Amouh. *Numerical Finance.* Lecture notes in Master 203 program. Paris, France, 2024.