Time-efficient Decentralized Exchange of Everlasting Options with Exotic Payoff Functions

Juan P. Madrigal-Cianci*, Jesper Kristensen*
*Advanced Blockchain Research.
{juanpablo,jesper}@advancedblockchain.com.

***Abstract*—We introduce a new type of derivative for cryptocurrencies: the path-dependent everlasting option. This can be understood as a generalization of the so-called *everlasting option* to a wider class of instruments for which their payoff function depends on the trajectory followed by the price of the underlying asset. This is the case, e.g., for the Asian or barrier options. In addition, we present a trading protocol for these novel derivatives and present a state-of-the-art Monte Carlo methodology for the efficient pricing of these instruments.**

1. Introduction

An *option* is a financial *derivative* (i.e., an instrument whose value depends on the value of another asset) that gives its buyer the opportunity –but not the obligation– to buy or sell its underlying asset on (or before) a future time (the so-called expiration date) and at a fixed "strike" price, provided certain conditions on the contract are satisfied. By far, the most commonly traded type of options are "European" or "American" options [1], whose price and payoff structure depend solely on the strike price and the underlying price at expiration [1]. These two kinds of options are typically referred to as "vanilla options" and, in the context of Traditional Finance (TradFi), are typically traded over large exchanges, such as the NYSE. Options whose pricing structure and contract terms are more complicated than vanilla options are typically referred to as "exotic options". In TradFi, exotic options are created by financial engineers to come up with advanced trading strategies. Typically the majority of exotic options are traded Over The Counter (OTC). This, in turn, can lend itself to a wide array of opaque or even malicious behavior, which can be avoided in the context of Decentralised Finance (DeFi) and Decentralised Exchanges (DEX) using blockchains and smart contracts as a transparent ledger for both the pricing and exchange of these instruments. To the best of our knowledge, the market for options trading in the DeFi sphere is in its early stages; with a handful of exchanges based on the following protocols [2], [3], [4]. A type of financial derivative that will be of particular interest for this work is what we will call non-expiring contracts; i.e., contract without a fixed expiration date. This is the case of, e.g., perpetual futures [5], [6] and, more recently, everlasting options [3], [7], the latter of which can be understood as a geometrically weighted basket of infinitely-many vanilla options with increasing expiration date. Inspired by this, we introduce a new type of exotic derivative, called the "path-dependent everlasting option", which can be thought of as an extension of the ideas presented in [7]. In short, the main contributions of this work are:

1) Introduction of a novel class of *exotic* options: the path-dependent everlasting option. This can be thought of as an everlasting option whose payoff depends on the path followed by the price of the underlying asset. Typical examples of (non-everlasting) path-dependent options are the so-called Asian or Barrier options.
2) An efficient protocol for the exchange of such instruments. This protocol builds upon the ideas presented in [3] and [8].
3) We propose state-of-the-art numerical methods, namely Quasi-Monte Carlo [9] and Multi-level Monte Carlo [10], for the efficient and accurate pricing of these contracts. Furthermore, we present an error analysis and a computational complexity estimator for their pricing. Given that these instruments are quite time and accuracy-sensitive in nature, it is important to have efficient pricing mechanisms, as well as theoretical bounds on their computational cost.

The rest of this paper is organized as follows. In section 1.1 we will introduce the mathematical setting and notation that will be used throughout the rest of the manuscript. We aim at presenting this section with a great deal of generality so that the methodologies presented later in the manuscript can be easily extendable to pricing models beyond the usual constant drift, constant variance Geometric Brownian Motion. In Section 2 we recall the concept of a vanilla everlasting option and introduce their new, path-dependent counterpart. The protocol for the exchange of these derivatives is presented in Section 2.1. We present, analyze and discuss a fast pricing strategy in Section 3. There, we illustrate the computational gains associated with our pricing methodologies. Lastly, we present some conclusions and finalizing remarks in Section 4.

978-1-6654-6104-7/22/$31.00 ©2022 IEEE
DOI 10.1109/Blockchain55522.2022.00066

1.1. Setup and notation

We introduce some of the notation that will be used throughout the work. Let $\mathbb{R}_+ := [0, \infty)$, and denote by $(\mathbb{R}_+, \mathcal{B}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ a filtered probability space.[1] Given a *drift* $\mu : \mathbb{R}_+^2 \to \mathbb{R}$ and a *volatility* $\sigma : \mathbb{R}_+^2 \to \mathbb{R}_+$, we model the price of an asset $S : \mathbb{R}_+ \to \mathbb{R}_+$ as the solution to the following Stochastic Differential Equation (SDE) :

$$\frac{\mathrm{d}S(t)}{S(t)} = \mu(S,t)\mathrm{d}t + \sigma(S,t)\mathrm{d}W(t), \tag{1}$$

where $W(t)$ is a standard Wiener process. Notice that here we are considering a more general version of the Geometric Brownian Motion (GBM) model typically used in the Black-Scholes formulation that accounts for non-constant drift and non-constant volatility. Being able to have this additional level of generality in the financial model is of great importance for highly-volatile markets, such as the crypto-currency one. Let us consider an option with strike price $K \in \mathbb{R}_+$ and *payoff function* $\psi : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$. Under the assumption that there exists a risk-free asset providing returns with a (risk-free) rate $r : \mathbb{R}_+ \to \mathbb{R}_+$, it follows from the arbitrage-free pricing model [11] that the price $V(t, S, K, \psi, T)$ of an option with strike price K, payoff function ψ, and expiration date T is given by

$$V(t, S, K, \psi, T) = \mathbb{E}\left[e^{-\int_t^T r(u)\mathrm{d}u}\psi(S(t), K)|\mathcal{F}_t\right], \tag{2}$$

where the expectation is taken with respect to the risk-neutral measure. With a slight abuse of notation we will sometimes write $V(t)$ to denote $V(t, S, K, \psi, T)$ whenever there is no source of confusion regarding the other parameters in V. We remark that in the particular case where both σ and μ are constant, and the payoff function is sufficiently simple (such as that of a European type option, which has payoff $\psi_{\text{call}}(S, K) = \max\{S - K, 0\}$ for a call or $\psi_{\text{put}}(S, K) = \max\{K - S, 0\}$ for a put), the contract at hand can be priced by solving the well-known Black-Scholes-Merton [12] partial differential equation, which yields a computationally simple or even analytical expression for the price of V. However, it is well-known (see, e.g., [11]) that once one starts considering more complicated options contracts, such as those involving potentially large number of underlying assets, or options contracts whose payoff function is path-dependent, the numerical solution to the BSM partial differential equation can quickly become computationally inefficient. Such is the case, e.g., of the arithmetic Asian option (c.f. Equation (3)), or the barrier option (c.f. Equation (4))

$$\psi_1(S(t), K) = \max\left\{\frac{1}{t}\int_0^t S(\tau) - K\mathrm{d}\tau, 0\right\}, \tag{3}$$

$$\psi_2(S(t), K) = \max\{S(t) - K, 0\}\,\mathbf{1}_{\left\{\max\limits_{0\leq t^*\leq t} S(t^*)\leq B\right\}}, \tag{4}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function and $B \in \mathbb{R}_+$ is some barrier term. We remark that many other path-dependent options exist, see, e.g., [11]. In this work, we will focus on the construction and pricing of a class of so-called perpetual options (see, e.g., [3], [5], a particular class of options contracts that do not expire and whose payoff cannot be computed using the BSM model. Thus, one typically needs to estimate the solution to (2) using Monte Carlo (MC) methods as outlined below:

1) Discretize the underlying dynamic $S(t) \approx S_h(t)$ using some appropriate numerical method (e.g., Euler-Maruyama, Milstein, etc).
2) Generate N realizations of the underlying (discretized) price dynamic $S_h(t)$ and compute a Monte Carlo estimator of V, as

$$V(t, S) \approx \frac{1}{N}\sum_{n=1}^{N} V(t, S_h^n) =: \hat{V}_h,$$

with the understanding that $\hat{V}_h \to V$ as $N \to \infty$ and $h \to 0$.

2. Path-dependent everlasting options

We now aim at extending the ideas of everlasting options, developed in [3] for vanilla payoffs under a GBM. We begin by recalling the idea of an everlasting option for an arbitrary payoff function and then proceed to construct a novel trading protocol for these instruments.

As the name suggests, everlasting options are a type of options contract without an expiration date. Since these instruments do not have an expiration date, they can allow traders to have long-term exposure or hedging without the efforts, expenses, and potential risks associated with repeatedly re-opening their position every so often [3]. Similar to perpetual futures, everlasting options rely upon the idea of a recurrent funding fee that the buyers (longs) must pay to sellers (shorts) to keep their positions open. Denote by $V_e(S, K, \psi)$ the price of the everlasting option with underlying value $S(t)$, strike price of K, and payoff function ψ, and denote a funding period of T. The mechanism of an everlasting option with a potentially exotic payoff works as follows:

1) At time $t = 0$, agent A opens a position with payoff ψ for a price of V_e. This could be a short or long position and we will assume that there is a counter-party for this contract (c.f. Section 2.1). This payoff can be as simple as a vanilla call or as complex as a basket of several exotic options with path-dependent payoffs.
2) At the close of every funding period (i.e., at time $t = kT$, $k \in \mathbb{N}$), agent A pays (if A is a buyer) or receives (if A is a seller) a funding fee $F : \mathbb{R}_+^2 \to \mathbb{R}_+$ to maintain their position. In particular, F corresponds to the *time-value* of the option, i.e.,

$$F(S, K) := V_e(S, K, \psi) - \psi(S(t); K).$$

1. Loosely speaking, $\mathcal{B}$ represents the set of all possible values the that a pricing process can take in $\mathbb{R}_+$, and $\mathcal{F}_t$ represents the history of the price process up-until some time t

Notice then that by simply considering a more general formulation of the product in [7], one can obtain a more interesting derivative. This could be useful in a wide variety of cases, ranging from pure speculation, to carefully crafted risk-hedging involving involving path-dependent strategies which involve a constant re-roll of a position.

Naturally, a crucial part of these products is the computation of V_e. Before diving into the mathematical and computational aspects of V_e, we first describe how a trading protocol for these instruments can be created.

2.1. Trading protocol

Inspired by the so-called Proactive Market-Making ideas (PMM) of the DODO framework [8] and Deri Protocol [3], as well as the Virtual Automated Market Making (VMM) of Perp, we now present a simple type of PMM for derivative trading, particularly well-suited for the type of instruments considered in this work. We remark that, given that this idea can be utilized as the backbone of other derivative-trading protocols, we aim at discussing it with a high degree of generality.

We begin with the construction of a pricing curve. We say that the market is at parity whenever the number of long positions b equals the number of short positions s. Let $w_1, w_2 \in [0,1]$ such that $w_1 + w_2 = 1$. Given a contract theoretically priced at V_e, we define the *pricing curve* $P : \mathbb{R}^3_+ \to \mathbb{R}_+$:

$$P(V_e, b, s) = V_\mathrm{e}\left[w_1\left(\frac{b}{s}\right)^m + w_2(1 + a(b-s))^+\right] + \epsilon \tag{5}$$

with $m, a, \epsilon \in \mathbb{R}_+$. Here, $\epsilon \ll 1$ is included so that P does not go to zero. Notice that P can be thought of as an extension to the trading function on the Deri protocol [3], by considering a weighted average between their *pricing curve* [3] and the term $\left(\frac{b}{s}\right)^m$, which, intuitively, penalises buyer-seller disparity according to m; indeed noting that by taking $w_1 = 0$ and $w_2 = 1$ one recovers the same pricing curve as in [3]. This is illustrated in Figures 1 and 2. We remark that a similar idea of utilising weighted averages between pricing curves has been proposed in the case of automated market makers by Curve finance (former StableSwap) [13].

Thus, at any moment in time (since, naturally, b and s are time-dependent) the price of the underlying option is given by P. Once this pricing curve has been constructed, users can interact with a given protocol by buying (resp. selling) potentially exotic and path-dependent option contracts from (resp. to) the liquidity providers with an adjusted mark price given by $P(V_e, b, s)$. To that end, consider a *classic* two-pool AMM (e.g., Uniswap) with a *numéraire* x and a quotable asset y (e.g., USDC and ETH) and trading function Φ (c.f. [14] for a precise definition). The protocol then works as follows:

1) A buyer is interested in buying a contract (on the quotable asset) currently begin traded at a price $p_1 = K_\mathrm{e} y$, $K_\mathrm{e} > 0$.
2) They buy the contract for a price given a trading function Φ. This money goes to the liquidity pool and the buyer is issued a token representing the value of the contract. Furthermore, this induces a price for the quotable asset, which then gets fed to P. The user also pays a small transaction fee which goes directly into the pool.
3) The buyer pays fees to maintain their position at every time $t = kT$, $k \in \mathbb{N}$. To do this in a fully decentralized and trust-less manner, the user needs to deposit some collateral and can get liquidated by the protocol or external liquidators. This collateral is not part of the liquidity pool, but part of it can go towards it if there is a liquidation.
4) The user can then sell their contract at a later time for a price p_2 determined by both equation 5 and the price of the quoted asset.
5) Liquidity providers receive the transaction and a portion of the trading fees.

Figure 1. Contour plot of $P(1, b, s)$ for $m = a = 1$ and $w_1 = w_2 = 0.5$ for the price formula in (5). Notice that, for the parameters chosen, having more buyers than sellers, produces a large reduction in price.

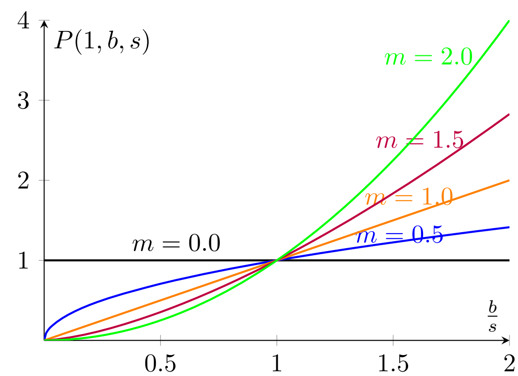

Figure 2. Plot of P vs $\frac{b}{s}$ for different values of m in (5) . Notices that larger values of m produce larger reductions in price whenever $b < s$.

Notice that Equation (5) needs to be computed every time there is a transaction. In the case where the protocol is highly active, where potentially hundreds of transactions are carried out per second, V_{e} would need to be efficiently computed. The following two sections are devoted to the construction and numerical computation of V_{e}.

2.2. Pricing V_{e}

Currently, to the best of our knowledge, everlasting options have only been thoroughly analysed (let alone implemented) in the case where the payoff function ψ of V_{e} is that of a vanilla European put or call. In a general setting, given a *payment period* $T > 0$, with a continuously accrued funding (i.e., the funding frequency $\omega \to \infty$), it is shown in [7] via the so-called no-arbitrage argument, that the price V_{e} is given by

$$V_{\text{e}}(S, K; \psi) = \frac{1}{T}\int_0^\infty V(t, S, K, \psi)e^{-t/T}\mathrm{d}t. \qquad (6)$$

As a remark, notice that (6) has a nice interpretation as the expected value of a random variable $V(\tau, \cdot, \cdot, \cdot)$ with $\tau \sim \text{Exp}\left(T^{-1}\right)$, i.e., one could alternatively write

$$V_{\text{e}}(S, K; \psi) = \mathbb{E}_{\tau\sim\text{Exp}(T^{-1})}\left[V(\tau, S, K, \psi)\right]. \qquad (7)$$

In the particular case of vanilla payoffs with $r = 0$, i.e., where $\psi_{\text{vanilla}}(S(t); K) = \psi_{\text{put}} = \max\{0, K - S(t)\}$ or $\psi_{\text{vanilla}} = \psi_{\text{call}} = \max\{0, S(t) - K\}$, there exists a closed-form formula for Equation (6) given by

$$V_{\text{e}}(S, K, \psi_{\text{vanilla}}) = \psi_{\text{vanilla}}(S(t), K) + \text{Val},$$
$$\text{with} \quad \text{Val} = \begin{cases} \frac{K}{u}\left(\frac{S}{K}\right)^{-\frac{u-1}{2}} & \text{if } S(t) \geq K, \\ \frac{K}{u}\left(\frac{S}{K}\right)^{\frac{u+1}{2}} & \text{otherwise}, \end{cases}$$

where $u = \sqrt{1 + \frac{8}{\sigma^2 T}}$.

For more general payoff functions, however, the term (6) cannot be computed explicitly, but rather needs to be numerically approximated. This incurs three levels of approximation, namely:

- Approximation of the outer integral. This should be done using a numerical quadrature rule, such as a Gauss-Lebesgue quadrature, where one uses a predetermined set of quadrature points $\{w_i, t_i\}_{i=1}^{N_{\text{quad}}}$ to approximate integrals of the form $\int_0^\infty g(x)e^{-x}\mathrm{d}x$ as $\int_0^\infty g(x)e^{-x}\mathrm{d}x \approx \sum_{n=1}^{N_{\text{quad}}} w_i g(x_i)$ (provided the integral exists), i,e.,
$$T^{-1}\int_0^\infty V(t)e^{-tT^{-1}}\mathrm{d}t \approx \sum_{i=1}^{N_{\text{quad}}} w_i V(t_i T^{-1}) =: V_{\text{e,Q}},$$
where for notational simplicity we have omitted the irrelevant arguments of V
- For each t_i, there is an approximation for the simulation of the underlying dynamic by discretizing the time interval $[0, t_i]$ on t_i/h intervals of size $h \geq 0$. As an example, in the case of constant volatility and constant r, this induces an approximation of the form
$$\begin{aligned} V(t_i, S, K, \psi) &\approx \int_{\mathbb{R}^{D_h}} \frac{\psi_h(S_h(\boldsymbol{t}_i), K)e^{-\boldsymbol{t}_i C^{-1}\boldsymbol{t}_i}}{(2\pi)^{d_h}\det(C)}\mathrm{d}\boldsymbol{t}_i \\ &=: V_h(t_i, S, K, \psi) \end{aligned}$$
with $\boldsymbol{t}_i := (0, t_h, \ldots, t_i)$, and $C_{i,j} = \min\{i, j\}$ the covariance matrix of the Wiener process (c.f. [15]).
- Monte Carlo approximation of the inner (finite-dimensional) integral V_h, which is done by averaging over N_{mc} independently and identically distributed realizations of ψ_h.

Thus, given a discretisation parameter h, a set of N_{quad} quadrature points $\{w_i, t_i\}_{i=1}^{N_{\text{quad}}}$ and a sufficiently large number of Monte Carlo samples N_{MC}, on has the approximation

$$\begin{aligned} \widehat{V}_{e,h,Q}(S, K, \psi) := \sum_{i=1}^{N_{\text{quad}}} w_i \left(\sum_{n=1}^{N_{\text{MC}}} \frac{V_h^{(n)}(t_i, S, K, \psi)}{N}\right) \qquad (8) \\ \approx V_{\text{e}}(S, K, \psi). \end{aligned}$$

Notice that as a consequence of (7), one could alternatively approximate both integrals using Monte Carlo methods, however, we will chose to focus on this approach in future work. Notice, furthermore, that a similar procedure can be used to price the *Greeks* associated with such an option (c.f. [11]). We present an error bound and complexity result for the approximation in (8).

Theorem 1 (Error and Complexity MC)**.** *Suppose that, for any fixed S, K, ψ, the mapping $t \mapsto V(t, \cdot, \cdot, \cdot)$ is sufficiently smooth so that the integral converges with rate $\mathcal{O}(N_{quad}^{-\rho})$, for some $r > 1$. Suppose, furthermore, that $\hat{V}_h$ is being numerically approximated using N_{quad} points for the quadrature, a discretisation parameter h for the SDE in Equation (1) with accuracy $\mathcal{O}(h^\alpha)$, and N_{mc} Monte Carlo samples. Then, there exists a positive constant $c_s > 0$ such that*

$$\begin{aligned} \mathit{MSE}(\hat{V}_{e,h,Q}) &:= \mathbb{E}\left[\left(\hat{V}_{e,h,Q} - V_e\right)^2\right] \\ &\leq c_s\left(\sum_{i=1}^{N_{quad}} w_i\left(\frac{\mathbb{V}[V(t_i)]}{N} + h^{2\alpha}\right) + N_{quad}^{-2\rho}\right). \end{aligned}$$

Furthermore, given a tolerance $\mathsf{tol} > 0$, the computational cost required so that $\mathit{MSE}(\hat{V}_h) \leq \mathsf{tol}^2$ is upper bounded by

$$\mathit{Cost}\left(\mathit{MSE}(\hat{V}_{e,h,Q}) < \mathsf{tol}^2\right) \leq c_k\mathcal{O}(\mathsf{tol}^{-2-1/\alpha-1/\rho})$$

for some positive constant c_k.

Proof. We begin with the bound on the approximation error. With a slight abuse of notation, we will drop the dependence on S, K, ψ of V. Let's introduce the notation:

$$V_{\text{e},h,\text{Q}}(t) := \sum_{i=1}^{N_{\text{quad}}} w_i V_{\text{e},h}(t).$$

From the definition of MSE, we have that, by adding $\pm V_{e,Q}$, we obtain

$$\begin{aligned}\mathrm{MSE}(\hat{V}_h) &= \mathbb{E}\left[\left(\hat{V}_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e}}\right)^2\right] \\ &\leq 2\mathbb{E}\left[\left(\hat{V}_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e,Q}}\right)^2\right] + 2\mathbb{E}\left[\left(V_{\mathrm{e,Q}} - V_{\mathrm{e}}\right)^2\right]\end{aligned}$$

Further adding $\pm V_{\mathrm{e},h,\mathrm{Q}}$ gives

$$\begin{aligned}&\leq 4\underbrace{\mathbb{E}\left[\left(\hat{V}_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e},h,\mathrm{Q}}\right)^2\right]}_{\text{(i)}} + 4\underbrace{\mathbb{E}\left[\left(V_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e,Q}}\right)^2\right]}_{\text{(ii)}} \\ &+ 4\underbrace{\mathbb{E}\left[\left(V_{\mathrm{e,Q}} - V_{\mathrm{e}}\right)^2\right]}_{\text{(iii)}} \qquad (9)\end{aligned}$$

Notice that the last three terms in inequality (9) are, up to a constant 4, the error contributions of (i) the variance of the Monte Carlo estimator, (ii) the discretization bias squared and (iii) the integration error. From this and Jensen's inequality we have that

$$\begin{aligned}&\text{(i)} = 4\mathbb{E}\left[\left(V_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e,Q}}\right)^2\right] \leq c_s \sum_{i=1}^{N_{\text{quad}}} w_i \frac{\mathbb{V}[V(t_i)]}{N_{\text{mc}}}, \\ &\text{(ii)} = 4\mathbb{E}\left[\left(V_{\mathrm{e},h,\mathrm{Q}} - V_{\mathrm{e,Q}}\right)^2\right] \leq c_s \sum_{i=1}^{N_{\text{quad}}} w_i h^{2\alpha}, \\ &\text{(iii)} \leq c_s N_{\text{quad}}^{-2\rho}.\end{aligned}$$

for some constant c_s. Adding the previous upper bounds proves the first result. To obtain the second result we make each error contribution smaller than tol^2. This in turn implies that $h \lesssim \mathsf{tol}^{1/\alpha}$, $h^{-2} \lesssim N_{\text{mc}}$ and $\mathsf{tol}^{-1/\rho} \lesssim N_{\text{quad}}$. Thus, the computational cost needed to obtain an MSE bounded by tol^2 grows as $\mathcal{O}(\mathsf{tol}^{-2-1/\alpha-1/\rho})$, as desired. □

Notice that, even for moderate values of α and r, the cost-tolerance complexity of our estimator grows rather fast. In a practical setting, this would severely limit the efficiency of our proposed AMM. We now present a way of alleviating this issue.

3. Efficient pricing

We present a Multi-level Quasi-Monte Carlo (MLQMC) approach for the fast and accurate computation of V_{e}. We remark that MLQMC methods have been used several times in the literature for the pricing of (non-everlasting) exotic options, as discussed in, e.g., [10]. Given the novelty of everlasting options, however, to the best of our knowledge, these methods have not yet been used to price these instruments. In the remainder of this section, we present a brief overview of these methodologies and propose an MLQMC algorithm for the fast computation of V_{e}. As the name suggests, MLQMC methods have two main components; Quasi-Monte Carlo methods, which improve the convergence of crude Monte Carlo methods by considering sequences of quasi-random numbers, and Multi-level Monte Carlo methods, which improve upon crude Monte Carlo methods relying upon a discretized model by considering a hierarchy of discretizations.

3.1. Quasi-Monte Carlo (QMC)

We recall the main ideas behind Quasi Monte Carlo (QMC) methods in an abstract setting. Consider the integral I of a function $f : \mathbb{R}^d \to \mathbb{R}$ over the d-dimensional unit sduare given by

$$I = \int_{[0,1]^d} f(x_1, \dots, x_d)\, dx_1 \dots dx_d.$$

A Crude Monte Carlo estimator $\hat{I}_{CMC}$ that uses N independent and identically distributed replicas of $X := (x_1, \dots, x_d)$, achieves an error

$$|I - I_{\text{CMC}}| \leq c_{1-\alpha/2} \frac{\sqrt{\mathbb{V}[f(X)]}}{\sqrt{N}}$$

with asymptotic confidence $1-\alpha$. The idea of Quasi Monte Carlo (QMC) sampling is to consider, instead, a *purely deterministic* sample $\{X^{(1)}, \dots, X^{(N)}\}$ to improve the rate $1/\sqrt{N}$, while keeping the simple structure of the sample average estimator $\hat{I}_{\text{QMC}} = \frac{1}{N}\sum_{i=1}^{N} f(X^{(i)})$ with equal weights $1/N$. It relies on the observation that a random uniform sample does not seem to cover "uniformly" the hypercube and hopefully there exist better designs that achieve this goal. To that end, QMC uses a *Low Discrepacy Sequence* (LDS), which, intuitively, cover the unit square on a more uniform way (c.f. Figure 3). Several multi-dimensional LDS are known (e.g., Sobol, Halton, Faure sequences, c.f. [16]) and their study is an active field of research for both number theorists and financial mathematicians. It is known (see e.g., [9] for a thorough discussion on this) that using (deterministic) "samples" from an LDS, QMC estimators are of the form:

$$|I - I_{\text{QMC}}| < \mathcal{O}\left(\frac{(\log N)^{(d-1)}}{N}\right),$$

i.e., the error decays as N^{-1} rather than the slower $N^{-1/2}$ of standard Monte Carlo. There is a caveat, however, and it is that this constant on this error is unknown and typically difficult to estimate. Furthermore, since the points from an LDS are not random points, one cannot use the Central Limit Theorem to compute the error of the estimator. An easy idea to overcome this is to randomize the QMC formula. Let $U \sim \mathcal{U}([0,1]^d)$. If $\mathcal{P} = \{X^{(1)}, \dots, X^{(N)}\}$ is a low discrepancy point set, so is

$$P_U = \{\{X^{(1)} + U\}, \{X^{(2)} + U\}, \dots, \{X^{(N)} + U\}\}$$

where the same shift is applied to all points and again $\{\cdot\}$ denotes the fractional part. P_U is called a randomly shifted point set. We could then compute $\hat{\mu}_{\text{QMC}}^{(j)}$, $j = 1, \dots, k$, for few randomly shifted point sets and average the obtained results. The resulting randomly shifted QMC estimator is

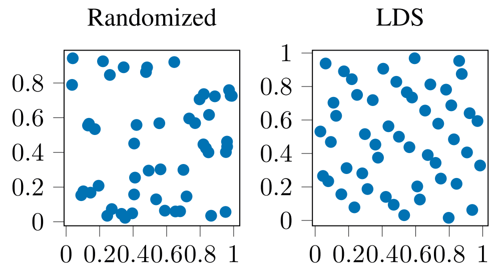

Figure 3. Left: Sequence of 50 uniform random numbers on the unit square. Right: LDS on the unit square.

then $\hat{\mu}_{QMC} = \frac{1}{k}\sum_{j=1}^{k}\hat{\mu}_{\text{QMC}}^{(j)}$. Since $U^{(j)} \sim \mathcal{U}([0,1]^d)$, so is $\{X^{(i)} + U^{(j)}\}$ for any $i = 1,\ldots,N$. It follows that $\hat{\mu}_{\text{QMC}}$ is an *unbiased* estimator of $\mu = \mathbb{E}\psi$. Moreover, since $\hat{\mu}_{\text{QMC}}^{(j)}$ are independent, the variance of the estimator is $\mathbb{V}\hat{\mu}_{\text{QMC}} = \frac{\sigma_{\text{QMC}}^2}{k}$ with $\sigma_{\text{QMC}}^2 = \mathbb{E}(\hat{\mu}_{\text{QMC}}^{(j)} - \mu)^2 = O\left(\frac{(\log N)^{2(d-1)}}{N^2}\right)$ hence, very small, in general, and can be estimated by the standard sample variance estimator $\hat{\sigma}_{\text{QMC}}^2 = \frac{1}{k-1}\sum_{j=1}^{k}(\hat{\mu}_{\text{QMC}}^{(j)} - \hat{\mu}_{\text{QMC}})^2$. Lastly, notice that

Algorithm 1 Randomly shifted QMC.

1: Generate $U^{(1)},\ldots,U^{(k)} \overset{iid}{\sim} \mathcal{U}([0,1]^d)$
2: **for** $j = 1,\ldots,k$ **do**
3: compute $\hat{\mu}_{\text{QMC}}^{(j)} = \frac{1}{N}\sum_{i=1}^{N} f(\{X^{(i)} + U^{(j)}\})$
4: **end for**
5: Compute $\hat{\mu}_{\text{QMC}} = \frac{1}{k}\sum_{j=1}^{k}\hat{\mu}_{\text{QMC}}^{(j)}$
6: Compute $\hat{\sigma}_{QMC}^2 = \frac{1}{k-1}\sum_{j=1}^{k}(\hat{\mu}_{QMC}^{(j)} - \hat{\mu}_{QMC})^2$
7: Output $\hat{\mu}_{QMC}$ and

$$I_\alpha = \left[\hat{\mu}_{\text{QMC}} - c_{1-\alpha/2}\frac{\hat{\sigma}_{\text{QMC}}}{\sqrt{k}},\ \hat{\mu}_{\text{QMC}} + c_{1-\alpha/2}\frac{\hat{\sigma}_{\text{QMC}}}{\sqrt{k}}\right]$$

.

the QMC method is designed of rintegrals with respect to the Lebesgue measure in the unit square. In our case, since we are typically interested in integrating against multivariate Gaussian measures, one then needs to transforms the points in the unit square into the real line with the inverse normal cumulative distribution function (c.f. [9], [16] for a detailed account of how to implement these methods).

3.2. Multi-level Monte Carlo (MLMC)

Multi-level Monte Carlo methods are a set of computational techniques that exploit the relationship between the number of samples, discretization accuracy, and computational complexity of an MC estimator in such a way that the overall computational cost associated with obtaining an MC estimator with a given accuracy is much lower than its single level (i.e., "plain" MC) counterpart. These methods have been successfully implemented in the fields of option pricing [10], [17], uncertainty quantification for different applications in science and engineering [18], and Bayesian

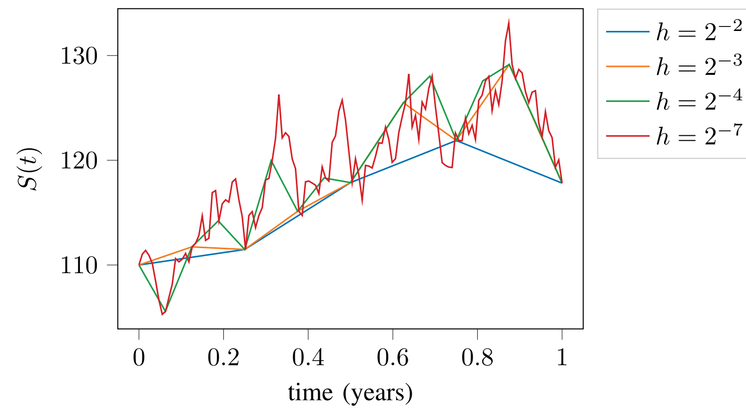

Figure 4. Discretisation of the SDE in (1) at different levels of accuracy.

inference [19], [20]. In the interest of brevity, we briefly present the main idea behind these methods and refer the interested reader to [10] and the references therein for an in-depth discussion about them. The idea behind MLMC is fairly simple; instead of considering simulations at a fixed discretisation parameter $h = h_L$, one considers a hierarchy of discretisation levels $h_\ell = 2^{-\ell}h_0$, $\ell = 0,1,\ldots,L$ approximating V by V_{h_ℓ} with increasing accuracy (and hence cost) in ℓ. One can then estimate $\mathbb{E}[V] \approx \mathbb{E}[V_{h_L}] \approx V_L^{\text{MLMC}}$ with

$$V_L^{\text{MLMC}} := \frac{1}{N_0}\sum_{i=1}^{N_0} V_{h_0}^{(i,0)} + \sum_{\ell=1}^{L}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\left[V_{h_\ell}^{(i,\ell)} - V_{h_{\ell-1}}^{(i,\ell)}\right],$$

where $V_{h_\ell}^{(i,\ell)}$ and $V_{h_{\ell-1}}^{(i,\ell)}$ denote correlated realizations of V discretised with a time-step h_ℓ and $h_{\ell-1}$ using the same random input (c.f. Figure 4). This correlation is important for reducing the cost of computing the MLMC estimator. The cost of the estimator is controlled by two parameters; namely, the level-wise sample sizes N_ℓ and the number of levels L. It is shown in [10] that by cleverly choosing N_ℓ as a function of (i) $\mathbb{V}\left[V_{h_\ell}^{(i,\ell)} - V_{h_{\ell-1}}^{(i,\ell)}\right]$ (ii) the cost of the simulation at level ℓ and the desired error tolerance tol, MLMC provides an estimator of $\mathbb{E}[V]$ with the same tolerance than its single-level counterpart, albeit at a much cheaper cost. We recall such a result next.

Theorem 2 (MLMC cost complexity [10])**.** *Suppose that there exists positive constants $\alpha,\beta,\gamma,c_1,c_2,c_3$ such that:*

$$|\mathbb{E}[V_{h_\ell} - V]| \le c_1 2^{-\alpha\ell},$$
$$|\mathbb{V}[V_{h_\ell} - V]| \le c_2 2^{-\beta\ell},$$
$$\mathit{Cost}(V_{h_\ell}) \le c_3 2^{-\gamma}.$$

Then, there exists a number of samples $N_\ell = N_\ell(\mathsf{tol})$ such that

$$\mathit{MSE}(V_L^{MLMC}) < \mathsf{tol}^2. \tag{10}$$

Furthermore, the computational cost of a MLMC estimator *satisfying* (10) *is bounded by*

$$\textit{Cost}\left(\textit{MSE}(V_L^{MLMC}) < \mathsf{tol}^2\right) \leq c_4 \begin{cases} \mathsf{tol}^{-2} & \beta > \gamma \\ \mathsf{tol}^{-2}(\log(\mathsf{tol}))^2 & \beta = \gamma \\ \mathsf{tol}^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma. \end{cases}$$

3.3. Multi-level Quasi-Monte Carlo (MLQMC)

Our proposed methodology for the efficient computation of V_e is outlined below. The "inner loop" component of such a procedure is based upon the work of [10].

Algorithm 2 Pricing everlasting option

1: **procedure** MLQMC-PRICING($\mathsf{tol}, L_{\text{min}}$)
2: Set $N_{\text{quad}} = \lceil \mathsf{tol}^{-2/r} \rceil$.
3: **for** $i = 1, \ldots, N_{\text{quad}}$ **do**
Set $L = 0$.
4: Estimate $[V_{h_L}(t_i) - V(t_i)]$ using 32 random offsets and $N_L = 1$.
5: **while** $\sum_{\ell=0}^{L}[V_{h_L}(t_i) - V(t_i)] > \frac{\mathsf{tol}^2}{3}$ **do**
6: Double N_ℓ on the level with the largest

$$\frac{[V_{h_\ell}(t_i) - V(t_i)]}{2^\ell N_\ell}$$

.

7: # if not at minimum level or bias too large, add level
8: **if** $L < L_{\text{min}}$ or $|E_{h_L - h_{L-1}}|^2 < \mathsf{tol}^2/3$ **then**
9: set $L = L + 1$ and go to line 4.
10:

One can extend Theorem 2 to a multi-level estimator of the price of V_e. We present such a result next.

Theorem 3 (MLMC cost complexity for V_e)**.** *Suppose that the mapping $t \mapsto V(t, \cdot, \cdot, \cdot)$ is sufficiently smooth. Then, under the same assumptions as Theorems 1 and 2, one has that the computational cost of estimating V_e using the Algorithm 2 so that its MSE is bounded by* tol^2 *has a cost* $\textit{Cost}\left(\textit{MSE}(V_{e,L}^{MLMC}) < \mathsf{tol}^2\right) < K$ *with*

$$K \leq c_4 \begin{cases} \mathsf{tol}^{-2-1/\rho} & \beta > \gamma \\ \mathsf{tol}^{-2-1/\rho}(\log(\mathsf{tol}))^2 & \beta = \gamma \\ \mathsf{tol}^{-2-(\gamma-\beta)/\alpha-1/\rho} & \beta < \gamma. \end{cases}$$

Proof. This theorem is a consequence of theorems 1 and 2. Indeed, proceeding as in the proof of Theorem 1, it just suffices to observe that the terms (i) and (ii) in 9 are the Multi-level Monte Carlo contributions to the MSE, which can be bound using Equation (10). □

In order to showcase the advantages of our proposed method, we present a simple "sanity check" example where we compare the time-to-solution for the pricing of an everlasting Asian option using (a) Crude Monte Carlo (CMC) and (b) our Multi-level Quasi Monte Carlo (MLQMC) approach. To that end, consider an arithmetic Asian option

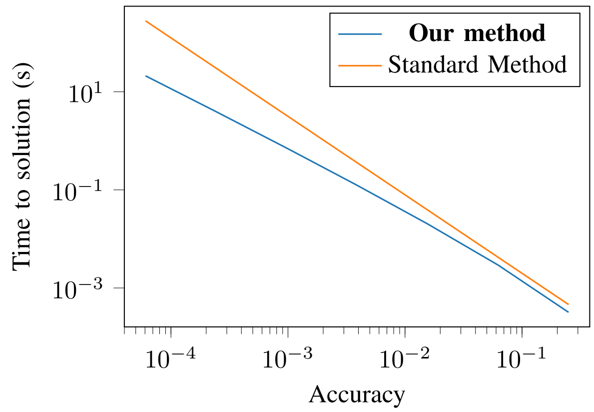

Figure 5. cost-tolerance complexity. As we can see, the proposed methodology is clearly more efficient than plain Monte Carlo.

	time (s)	Option price	Error bound
Standard Method	284.73	6.246	0.024
Our method	2.43	6.263	0.023

TABLE 1. RESULTS.

with $K = 120$, $S_0 = 100$, $r = 0.1$, $\sigma = 0.3$ and a payoff given by

$$\psi_h(S(t_i), K) = \max\left\{\frac{1}{N}\sum_{n=0}^{N} S(t_{nh}) - K, 0\right\}, \quad h = t_i/N.$$

Notice that, in this case, the accuracy of the payoff ψ_h depends on the discretisation parameter h, with the understanding that $\psi_h \to$ (3) as $h \to 0$. We construct S_h using a Milstein discretisation scheme [15], and estimate values for α, β, γ given by $\alpha = 0.99$, $\beta = 1.3$ and $\gamma = 0.95$. This in agreement with what one might theoretically expect for this type of scheme and the type of payoff (c.f. [15], [10]. We price this option using both methods to a tolerance of $\mathsf{tol} = 5 \times 10^{-3}$, i.e., we aim to price the option with either method so that the mean square error is less than tol^2. We estimated $\rho > 5$ (which suggests that V is a sufficiently smooth function of t), and as such we consider $N_{\text{quad}} = 10$. For the MLQMC algorithm, we chose $\mathsf{L} = 7$ with $h_\ell = 2^{-\ell-1}$, $\ell = 0, 1, \ldots, 7$. We present our results in Table 1. As we can see, we can gain an improvement of over 100 times for the pricing of each option. We remark that further computational gains can be obtained by computing each MLQMC estimator in an embarrassingly parallel fashion. A complexity plot based on the estimated values of α, β, γ is presented in Figure 5.

4. Finalising remarks

In our current work, we have presented a new type of derivative applicable to the DeFi space. In addition, we have presented a new type of trading mechanism for these derivatives. We remark that our presented trading protocol can, at least in theory, be extended beyond the case of everlasting options (i.e., one could use it to trade, e.g., vanilla or other exotic derivatives). Furthermore, we presented a theoretical analysis of the computational complexity of their pricing. Lastly, we have advocated for the use of advanced Monte Carlo techniques for the pricing of these contracts.

Our simple numerical results illustrate a rather large computational advantage of our proposed methodologies over standard pricing techniques.

There are a handful of possible research directions one could look into. From a modeling perspective, one could, e.g., incorporate even more general models for the pricing process, such as a general Lévy process. One could also look into proposing and pricing other types of financial derivatives that could be of interest in the DeFi sphere, such as options with a change of numéraire. From a protocol perspective, a much-needed piece of missing literature is robust modeling and quantification of the uncertainty associated with automated trading protocols. We are currently working towards this goal, and in future work, we aim to expand and further investigate, via numerical simulations, the behavior of our proposed protocol.

References

[1] J. C. Hull, *Options futures and other derivatives*. Pearson Education India, 2003.

[2] C. Alexander, J. Choi, H. Park, and S. Sohn, "Bitmex bitcoin derivatives: Price discovery, informational efficiency, and hedging effectiveness," *Journal of Futures Markets*, vol. 40, no. 1, pp. 23–43, 2020.

[3] R. C. 0xAlpha, Daniel Fang, "The echange protocol for everlasting options," Deri Finance, Tech. Rep., 10 2021. [Online]. Available: https://github.com/deri-finance/whitepaper/blob/master/deri_everlasting_options_whitepaper.pdf

[4] A. Juliano, "dydx: A standard for decentralized margin trading and derivatives," *URl: https://whitepaper. dydx. exchange*, 2018.

[5] H. U. Gerber and E. S. Shiu, "Pricing perpetual options for jump processes," *North American Actuarial Journal*, vol. 2, no. 3, pp. 101–107, 1998.

[6] A. E. Kyprianou and M. R. Pistorius, "Perpetual options and canadization through fluctuation theory," *The Annals of Applied Probability*, vol. 13, no. 3, pp. 1077–1098, 2003.

[7] D. White and S. Bankman-Fried, *Everlasting Options*, 2021.

[8] "Proactive Market Making Algorithm," Tech. Rep., 4 2022. [Online]. Available: https://dodoex.github.io/docs/docs/pmm

[9] P. l'Ecuyer, "Randomized quasi-monte carlo: An introduction for practitioners," in *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. Springer, 2016, pp. 29–52.

[10] M. B. Giles, "Multilevel monte carlo methods," *Acta Numerica*, vol. 24, pp. 259–328, 2015.

[11] P. Glasserman, *Monte Carlo methods in financial engineering*. Springer, 2004, vol. 53.

[12] F. Black and M. Scholes, "The pricing of options and corporate liabilities," in *World Scientific Reference on Contingent Claims Analysis in Corporate Finance: Volume 1: Foundations of CCA and Equity Valuation*. World Scientific, 2019, pp. 3–21.

[13] M. Egorov, "Stableswap-efficient mechanism for stablecoin liquidity," *Retrieved Feb*, vol. 24, p. 2021, 2019.

[14] G. Angeris, A. Evans, and T. Chitra, "Replicating market makers," *arXiv preprint arXiv:2103.14769*, 2021.

[15] L. C. Evans, *An introduction to stochastic differential equations*. American Mathematical Soc., 2012, vol. 82.

[16] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of monte carlo methods*. John Wiley & Sons, 2013.

[17] D. J. Higham, "An introduction to multilevel monte carlo for option valuation," *International Journal of Computer Mathematics*, vol. 92, no. 12, pp. 2347–2360, 2015.

[18] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, "Multilevel monte carlo methods and applications to elliptic pdes with random coefficients," *Computing and Visualization in Science*, vol. 14, no. 1, pp. 3–15, 2011.

[19] T. J. Dodwell, C. Ketelsen, R. Scheichl, and A. L. Teckentrup, "A hierarchical multilevel markov chain monte carlo algorithm with applications to uncertainty quantification in subsurface flow," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 3, no. 1, pp. 1075–1108, 2015.

[20] J. P. Madrigal-Cianci, F. Nobile, and R. Tempone, "Analysis of a class of multi-level markov chain monte carlo algorithms based on independent metropolis-hastings," *arXiv preprint arXiv:2105.02035*, 2021.

[21] R. C. Merton, "Theory of rational option pricing," *The Bell Journal of economics and management science*, pp. 141–183, 1973.

[22] M. B. Giles, "Mlmc for nested expectations," in *Contemporary Computational Mathematics-A Celebration of the 80th Birthday of Ian Sloan*. Springer, 2018, pp. 425–442.

[23] A. L. Teckentrup, R. Scheichl, M. B. Giles, and E. Ullmann, "Further analysis of multilevel monte carlo methods for elliptic pdes with random coefficients," *Numerische Mathematik*, vol. 125, no. 3, pp. 569–600, 2013.

[24] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Mathematics of computation*, vol. 23, no. 106, pp. 221–230, 1969.