An Automated Market Maker Algorithm for Fixed-Rate Trading with Flexible Maturities

Tuan Tran[1] and Duc A. Tran[2,*]

[1]Research Team, IDGB Lab Inc., Dover, DE (USA).
[2]Department of Computer Science, University of Massachusetts, Boston, MA (USA).
[*]Corresponding author: duc.tran@umb.edu.

Abstract

This paper aims to build an automated market maker (AMM) protocol to realize fixed-income products in decentralized finance. Compared to typical AMMs for cryptocurrency exchange, fixed-rate AMMs are much more difficult to design due to the inter-dependency of interest rate, loan price, and, especially, time factor. The literature lacks scientific developments. Existing efforts are either too simplistic, focusing more on computational automation and less on economic effect, or too engineering-crafted, using ad hoc formulations without economic justification. Indeed, today's primal fixed-rate AMMs, Yield AMM and Notional AMM, are too financially-risky for liquidity providers (Yield AMM) and too expensive for borrowers and lenders (Notional AMM). In this paper, we propose BondMM, a novel fixed-rate AMM protocol with two major advances. First, BondMM is superior regarding properties that reflect the financial strength of a lending institution: 1) better price impact to attract borrowers and lenders, 2) better capital efficiency to attract liquidity providers (LPs), and 3) better financial stability for both users and LPs. Second, BondMM allows for 4) issuance and trading of loans having flexibly different maturity dates using only one liquidity pool; the maturity can be arbitrarily requested, long or short term. This is an innovative feature unseen in both the literature and the practice of DeFi. Existing fixed-rate protocols would dedicate a separate liquidity pool to each maturity date, hence causing fragmented liquidity. By having a single pool shared by all maturities and LPs, BondMM improves capital efficiency, increases liquidity, and decreases computational costs. Traders can easily swap cross-maturity and increase leverage by requesting long maturities. In terms of blockchain feasibility, BondMM is constructed, whose correctness provable, using elegant closed-form mathematical expressions, hence efficient for smart contract implementation. The properties and performance of BondMM are validated with our theoretical analysis and evaluation study simulating real-world and synthetic datasets.

Keywords: Decentralized Finance, Blockchain Computing, Automated Market Makers.

1 Introduction

Fixed-income markets are the lifeblood of the global economy. They are more than twice the size of the global stock market. As blockchain technology [1] is disrupting the financial sector, we expect more and more decentralized finance (DeFi) products in the near future, and if the pattern in traditional finance applies, fixed-income DeFi should be one of the leading players in the

DeFi space, if not the leader. How far are we getting there? Today[1], the DeFi lending sector is about $40B in total value locked (TVL), which is 1.3%+ of the overall $3T+ crypto market capitalization. In particular, few fixed-income products exist [2–10], with total TVL about $5B, or 12.5%- of the DeFi lending market.

In other words, the market remains huge and wide open for fixed-income DeFi. We approach this opportunity from two different angles: 1) is it because existing lending protocols are not well designed for fixed-rate offerings, thus struggling to attract adoption? and 2) is it because there are just too few products available, suggesting room for more to enter the market? The answer is yes to both. In this paper, we propose a new fixed-rate lending and trading protocol based on automated market making (AMM) [11–14] that brings major advances to the state of the art.

1.1 State-of-the-Art Limitations

DeFi lending leaders such as AAVE [4], Compound [5], and MakerDAO [15] are designed for variable rates. They do offer fixed rates but only as an add-on product for borrowers, and the rates are so high that they account for only 1% of the total borrowing volume. This is because the lending pool needs a guaranteed spread to profit lenders at the cost of fixed-rate borrowers.

On the other hand, few protocols exist exclusively for fixed-rate lending and trading. The most notable are Yield Protocol [6] and Notional Finance [7], which inspired a following of various fixed-income DeFi applications, for example, principal-and-yield strip products with Pendle [8], Element [10], or Swivel [3], and structured products with BarnBridge [2]. Unfortunately, all these protocols and products remain unattractive to both liquidity providers (LPs) and users (borrowers, lenders, traders) for the following reasons:

Low trading volume. In theory, fixed-rate protocols operate as an AMM using the same rate for borrowing as that for lending. As the LPs lose income from bid-ask spreads, they make money only from transaction fees. Fee income is significant only if there is a lot of secondary trading, not just primary lending and borrowing transactions. However, today's products offer very short maturity durations (3-month, 6-month, and 1-year). This is not that inviting to traders due to lack of leverage, hence low secondary trading volume. Trading with speculation is the main reason as to why in traditional finance the bond market is many times larger than the lending market. Speculating traders like long-duration bonds because of the increased leverage for their bets. For example, a bond of 10-year maturity offers a leverage more than 10 times that of the 1-year due to compound effect.

Low capital efficiency. Existing fixed-rate AMM designs are either too simplistic, focusing more on automation and less on economic effect, or too engineering-driven, using ad hoc formulations without economic justification. In the Yield AMM, the mathematical formulas to ensure continuous liquidity for every possible trade are not well designed, leading to excessive liquidity provisioned for cases of extreme interest rates. These cases rarely happen, causing much LP capital to be underutilized. Better, we should concentrate liquidity distribution on realistic price/rate ranges. The Notional AMM aims to achieve this, but its pricing profits the LPs too greedily, which incurs bad rates for the service users (borrowers/lenders/traders), especially when the maturity duration is longer. This is an extreme approach because, economically speaking, no rational user would come to Notional due to high costs. Therefore, it is explainable why all products developed using these protocols or copying their AMM design keep maturity duration short. This way, borrowing and lending positions can be settled quickly without long-term risks for the LPs (in the case of Yield) and the trading cost does not look substantial to the traders (in the case of Notional).

Fragmented liquidity. Existing fixed-rate AMMs can only work with one maturity date in a liquidity pool. To offer multiple maturity dates, they must create separate pools to dedicate to each. As such, the LP liquidity is fragmented into smaller pools. This causes rates to be volatile and expensive due to limited liquidity per pool, which is undesirable to the user. Also, the liquidity in unpopular pools is underutilized, which is undesirable to the LPs. Worse, these LPs would leave for those pools more popular. We could think about optimizing a rebalancing strategy for the LPs to automatically allocate their liquidity over multiple

[1] Source: DeFiLlama.com (data analytics platform), as of Feb 26, 2025.

pools, but this is a non-trivial task, not to mention the extra computational costs and transaction/-gas fees to monitor and adjust the pools. Also due to gas fee, having separate pools makes cross-maturity trading expensive because the trader would have to exit one pool to join another pool to swap maturities.

1.2 Our contributions

Fixed-rate AMMs need deep liquidity (we need LPs) and large trading volume (we need borrowers/lenders/traders). We should not make the LPs happy simply by charging the users more. It is ideal to have a Nash equilibrium pleasing all sides maximally. Our ambition is not to find such an equilibrium; this can be a non-tractable problem computationally. Instead, we aim to do better than today's solutions in the most important aspects. As pointed out, existing fixed-rate AMM protocols are not attractive at all. We propose BondMM, a novel fixed-rate AMM protocol with major advances.

First, BondMM offers desirable properties that reflect the financial strength of a lending institution: 1) better price impact to attract borrowers and lenders, 2) better capital efficiency to attract LPs, and 3) better equity stability toward short-term liquidity coverage and long-term solvency. About the third property, a high equity means better ability to fulfill immediate transactions and resolve obligated debts in the long term. However, too high an equity means too much profit for the LPs at the cost of the users, which we should avoid. Because the business model of zero-spread fixed-rate AMMs is driven by transaction fees, we should keep LP equity as stable and close to the initial contributed capital as possible.

Second, BondMM offers desirable features unseen in the literature and practice: issuance and trading of loans having flexibly different maturity dates. The user can request any maturity duration, arbitrarily long or short term, on-demand. Importantly, this is made possible using only one liquidity pool shared by all maturities and all LPs, hence avoiding the fragmented liquidity problem. Thanks to the shared pool, traders can easily swap loans from one maturity date to another. This is executed as one single transaction, thus saving gas and transaction fees. Longer loan duration, meaning higher investment leverage, and the convenience of cross-maturity convertibility should attract the traders. Instead of splitting liquidity into separate pools for different maturities, putting all liquidity in one shared pool should improve LP capital efficiency.

In terms of construction, BondMM is a constant-function AMM with path-independence property, which is desirable for avoiding money-pumping attacks where one can strategize a series of transactions to pump and dump to exhaust pool liquidity. It is formulated, whose correctness provable, using elegant closed-form mathematical expressions, hence efficient for blockchain implementation. As to contribution to the literature, compared to typical AMMs for cryptocurrency exchange, fixed-rate AMMs are much more difficult to design due to the inter-dependency of interest rate, loan price, and, especially, time factor. The literature lacks scientific developments. Our work enhances the currently-thin scientific literature on fixed-rate AMMs and can serve as a verifiable scientific benchmark.

The properties and performance of BondMM are validated with our theoretical and experimental analyses using both real-world and synthetic datasets simulating different real-world scenarios. The results show that BondMM substantially outperforms Notional AMM and Yield AMM, especially when the maturity duration is longer. For example, when the duration is 10 years, in terms of capital efficiency, BondMM can be 2 times better than Notional and 50 times better than Yield. When there are multiple maturity dates, BondMM by using the single shared pool generates more revenue for the LPs than the individual pools each dedicated to a maturity date. For example, with 2 maturity dates, year 1 and year 2, BondMM can increase revenue by 10%+ after 2 years.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Background on fixed-rate AMM is presented in Section 3. BondMM is introduced and described in detail in Section 4. The evaluation results are analyzed in Section 5. We point out potential limitations and related issues about the realization of BondMM in Section 6. The paper concludes in Section 7.

2 Related Work

The concept of AMM started two decades ago in prediction markets, inspired by Hanson's logarithmic scoring rules (LMSR) [16, 17] as the first AMM algorithm, followed by numerous developments [16, 18–23]. Their motivation is to address the lack of liquidity and trading volume in thin order books. Unfortunately, AMM solutions for prediction markets, which would run on servers, are too complex to deploy on the blockchain. For on-chain trading, one must keep AMM simple to meet blockchain constraints regarding resources and smart contract programming.

Blockchain-based Trading AMM. Bancor [24] was the first DEX to implement an AMM, called Bonding Curve [25], for buying and selling a single token. Bonding Curve is a mathematical function of the token supply to compute the token price in such a way to satisfy the supply-demand principle and ensure infinite liquidity. AMM for two-token swaps was first discussed by Vitalik Butarin in [13] and subsequently implemented in Uniswap v2 (today's most dominant DEX). This AMM is constant-product: $x_1 x_2 = C$, where x_1, x_2 are the token amounts in the liquidity pool and C is a constant. A family of constant-function AMM (CFMM) variants have since followed; for example, constant-sum AMM used in Curve [26], weighted constant-product AMM used in Balancer [27], mixing of constant-sum and constant-product used in Curve v2 [28], and other sophisticated constant functions such as constant-power-root AMM [29] and constant-circle/ellipse-function AMM [30].

To improve LP capital efficiency, several AMM models have come with the feature of liquidity concentration. This feature was first introduced in Uniswap v3 [31] where LPs can specify a price range to apply their liquidity. Other liquidity-concentrated AMMs followed, including Trader Joe [32], Curve v2 [28], and OBMM [14]. Among key academic contributions recently, Goyal et al. [33] and Millionis et al. [34] each proposed a theoretical framework for finding the capital-optimal pricing function for CFMM if a belief about future prices is known. More discussions on LP liquidity strategy are presented in the work of He et al. [35], who introduced the notion of opportunity cost and proposed an optimal AMM design to maximize the LP's utility considering these costs.

AMM designs based on CFMM emphasize mathematical correctness and simplicity. Their pricing rule is ad hoc, mostly technological, and without precedent in traditional finance. LP price demands cannot be well approximated by any function. Recently, Park [36] points out some conceptual flaws of today's AMMs from the perspective of economics, which makes them more vulnerable to front-running exploitation risks than traditional finance. Milionis et al. [37] present a general two-asset exchange framework unifying CFMMs and limit-order books (LOB). Unlike CFMM, LOB can represent any LP price demand using limit orders. The sacrifice in LOB, however, is the large state size of the order book. The contribution of their work is an analysis and quantification of this tradeoff between the two exchange models.

Blockchain-based Fixed-Rate AMM. Almost all AMM developments, including both commercial and academic designs, are for swapping of crypto assets. AMMs for lending purposes, in particular fixed-rate financing, are rare. To the best of our knowledge, the only fixed-rate AMM protocols worth mentioning are Yield Protocol [6] and Notional Finance [7], which inspired a following of various fixed-income DeFi commercial products such as Pendle [8], Element [10], or Swivel [3], and BarnBridge [2]. We will compare to them directly in the next section. In the academic literature, we are aware of no research contributions to fixed-rate AMM design. Our work in this paper is the first academic research effort in this direction. Note that, despite great traction on asset-trading AMMs, it is not trivial to apply them directly on fixed-rate AMMs.

3 Preliminaries

We are interested in loans of fixed rate having a finite maturity. Someone borrowing $b = \$100$ at interest rate $r = 5\%$ yearly for $T = 10$ years has to pay back a total of $a = b \cdot (1 + r)^T = \$162$, which consists of the principal amount b and interest amount $a - b = \$62$. Depending on products, these amounts can be collected by the lender in smaller payments periodically scheduled or as a lump sum at maturity time. The latter applies to zero-coupon bonds where the interest total is prepaid in advance at the beginning and the principal total is one single future payment. Each bond

has a “face value" equal to the principal amount, which is materialized only at bond expiration. Until then, bonds can be traded at floating price driven by supply and demand. The interest prepay has the same effect as pricing the bond at a discount from the face value. Whoever holds a bond at maturity will be paid its face-value amount.

Applying to the above example, the loan can be implemented as a 10-year bond with face value of \$100 and, without interest, its price would be \$100. If the \$62 interest is prepaid to the lender (bond buyer) at purchase time, it is equivalent to a \$62 discount, resulting in an effective price of \$100 − \$62 = \$38. When this bond matures 10 years later, its holder will receive \$100 paid by the borrower. During the bond's life, it can be resold at will. Most bond volume comes from trading activities as traders love to speculate on bond interests due to macroeconomic dynamics. Intuitively, the resale price should be below the \$100 face value to attract buyers and increase if less time remains to maturity.

In what follows, for the sake of theoretical formulation, we assume the zero-coupon bond model to represent fix-rate loans. For implementation in practice, this model can easily be adapted to enable other repayment methods, for example, distributing yields over the time.

3.1 Fixed-Rate AMM

Consider bonds that mature at some time T in the future. Bonds are implemented in the form of a fungible token. Let us call it “bond token", or simply “bond". Money value is in the form of some numeraire “cash", say DAI. Without loss of generality, let the bond face value be 1 DAI.

At any time instant, a fixed-rate AMM is associated with the tuple $(x,\ y,\ r,\ p)$ where x, y are the respective amounts of bond and cash in the liquidity pool, and r is the reference interest rate (annualized). The reference rate is what the participants sees instantaneously as representative of the AMM market. In practice, a spread between the lending rate and borrowing rate can be introduced, but for the the sake of theoretical presentation, assume no rate spread and so r is the marginal interest rate for both lending and borrowing. This corresponds to a marginal bond price p. In theory, bond rate and price have the following relationship,

$$p = \mathrm{e}^{-tr} \Leftrightarrow r = \frac{1}{t} \ln \frac{1}{p} \tag{1}$$

where t is time to maturity. The actual bond price in a trade depends on the order size, i.e., the number of bonds traded in the order. Typically, r, p are computed from the pool reserves (x, y). Different AMM designs have different formulas for them.

Transactions. There are four types of actions from the user: bond minting (borrowing), bond burning (borrower exit), bond buying (lending), and bond selling (lender exit). Each action results in a change of state:

$$\begin{pmatrix} x \\ y \\ r \\ p \end{pmatrix} \rightarrow \begin{pmatrix} x_{new} = x + \Delta x \\ y_{new} = y + \Delta y \\ r_{new} \\ p_{new} \end{pmatrix}$$

Borrowing ($\Delta x > 0,\ \Delta y < 0$) is the action of bond minting. To borrow, one must deposit a collateral according to some collateral-to-loan ratio, for example 150% for volatile collateral. The loan amount, $|\Delta y|$ DAI, is instantly transferred from the pool to the borrower. At the same time, a corresponding number of Δx bonds are minted into the liquidity pool, readily available for buyers.

Lending ($\Delta x < 0,\ \Delta y > 0$) is the action of bond buying. One who buys $|\Delta x|$ bonds effectively becomes a lender and has to pay a corresponding price Δy DAI to the pool. This buyer, by holding the bonds, will receive from the pool a lump sum of $|\Delta x|$ DAI at maturity time.

Exiting a borrower position ($\Delta x < 0,\ \Delta y > 0$) is the action of bond burning. A borrower can payoff a loan, fully or partially, early. To do so, the borrower sends Δy DAI (the amount wanted to payoff) to the pool which in turn burns a corresponding $|\Delta x|$ bonds. The pro-rata collateral is returned to the borrower.

Exiting a lender position ($\Delta x > 0,\ \Delta y < 0$) is the action of bond selling. A lender may not wait until maturity time to redeem the bond. At any time, he or she can sell bonds to the pool to get early money. Selling Δx bonds will get back a corresponding $|\Delta y|$ DAI.

Calculation. Given an order ($\Delta x,\ \Delta y$) input as an amount in either Δx or Δy, the formula

to calculate the other amount is according to the specific rate-price model of the AMM.

3.2 Existing Models

There are wo main models: the Constant-Function model [6] and the Notional Finance model [7].

3.2.1 Constant-Function Model

Intuitively, one can borrow the idea from constant-function AMM (CFMM) [12, 13, 38] which is the de facto standard for decentralized cryptocurrency exchange. Using the constant-function model, the bond price is formulated to satisfy a constant-function invariant,

$$F(x + \Delta x, y + \Delta y) = F(x, y) = C \text{ (constant)}.$$

The interest rate is then derived from this price, $r = \frac{1}{t}\ln(1/p)$. Function F should respect the supply-demand law such that the rate should increase with more borrowing demand and decrease with more lending demand. Let $\phi = \frac{x}{y}$ denote the bond-to-cash ratio in the liquidity pool.

Constant-Product: $F(x, y) = x^w y^{1-w} = C$, constant $w \in (0, 1)$. For cryptocurrency-trading DEXes, this model is used in Uniswap AMM [39] by setting $w = \frac{1}{2}$. In general case, $w \in (0, 1)$, it is the Balancer AMM model [40]. Existing fixed-rate protocols adopting this model include Pendle V1 [8] and Apwine (now Spectra) [41]. The reference bond price is

$$p = \frac{-dy}{dx} = \frac{w}{1-w}\frac{y}{x} = \frac{w}{1-w}\phi^{-1}. \tag{2}$$

The reference interest rate is thus

$$r = \frac{1}{t}\ln\frac{1}{p} = \frac{1}{t}\ln\frac{1-w}{w} + \frac{1}{t}\ln\phi. \tag{3}$$

A drawback of this model is that the price is not affected by time to maturity. It remains constant if the pool does not change (no transaction occurs). This does not make sense because the bond price should increase and approach the face value as we get closer to maturity. Else, buyers would wait till near maturity time to place a buy order to get an immediate profit. This results in terrible capital efficiency for the LPs because their capital sits idle for a long time.

Constant-Power-Sum: $F(x, y) = x^{1-t/T} + y^{1-t/T} = C$, where t is time to maturity. This is the AMM model used in Yield Protocol [6]. It has the following reference bond price

$$p = \frac{-dy}{dx} = \left(\frac{y}{x}\right)^{t/T} = \phi^{-t/T} \tag{4}$$

and interest rate

$$r = \frac{1}{t}\ln(1/p) = \frac{1}{T}\ln\frac{x}{y} = \frac{1}{T}\ln(\phi). \tag{5}$$

This model resolves the drawback of the constant-product model in that the bond price does depend on time to maturity. Also, the price curve gradually flattens toward maturity ($t \to 0$). However, when there is less bond than cash in the pool, i.e., $\phi < 1$, the AMM formula results in a bond price exceeding the face value and, equivalently, a negative interest rate. No lender would then join. It is thus a waste to reserve much LP capital to cover this unrealistic case.

3.2.2 Notional Finance Model

Similar to Uniswap V3 [31] concentrating liquidity on a realistic price interval, Notional Finance [7], followed by Pendle V2 [42], use a fixed-rate AMM that concentrates liquidity around an anchor rate. Its rate function is an extension of the constant-power-sum rate,

$$r = \kappa \cdot \ln(\phi) + r^*, \tag{6}$$

where r^* is the anchor rate and constant κ is a coefficient to tune the liquidity concentration. A lower κ means higher concentration, thus more flattening price curve. In the non-scaling case where $\kappa = 1/T$ and $r^* = 0$, Eq. (6) is precisely the rate formula of Yield Protocol.

Notional AMM expresses rate r as a simple interest: $p(1 + rt) = 1$. So, the reference price is

$$p = (1 + rt)^{-1} = \left(1 + t\kappa \cdot \ln(\phi) + tr^*\right)^{-1}. \tag{7}$$

However, instead of relying on the mathematics of a constant-function invariant, Notional has a crafted formula to price a trade order. To compute Δy given Δx, let $\overline{\phi} = \frac{x+\Delta x}{y-\Delta x}$. The average price for

the order is set to

$$\overline{p} = \left(1 + t\kappa \cdot \ln(\overline{\phi}) + tr^*\right)^{-1}, \tag{8}$$

resulting in $\Delta y = -\overline{p}\Delta x$.

A side effect of this pricing design is that the trade price $\overline{p}$ is always more expensive then post-trade AMM reference price (the new reference price p updated after the trade). Especially, this results in an expensive trade price for bonds of long maturity, thus unattractive to traders. Also, Notional's pricing curve is no longer path-independent. Ideally, every AMM should be path-independent [13]. Another drawback of Notional's trade price formula is the difficulty of deriving Δx if the order is input in terms of Δy. There requires solving a non-trivial equation having no closed-form solution. This could explain why, as of now, Notional does not allow orders to input Δy. As later shown in our evaluation, by being greedy on the LP side, Notional would benefit the LPs in the long term but, due to expensive price impact, it suffers from liquidity and solvency risks in the early phase of a bond's life.

4 Solution: BondMM Protocol

We propose a new fixe-rate AMM protocol, named BondMM. Our approach is driven by the following reasons. In existing models, the AMM state is (x, y) and the interest rate is a time-independent function of the bond-to-cash ratio in the pool expressed in the count x of bond tokens, $r = R(\phi), \phi = \frac{x}{y}$. So if the state does not change over the time, this representation means that the supply and demand, i.e., number of bonds (asset) vs. cash, does not change. Per supply-demand law, the bond price should also stay the same. This contradicts the fact that when no transaction occurs the bond price $p = e^{-rt}$ automatically increases as we get closer to maturity $t \to 0$.

Another reason against this state definition is its unsuitability for the case having in the same liquidity pool bonds of different maturity dates. Because a bond of maturity date T_i is not fungible with a bond of maturity date $T_j \neq T_i$, which have different pricing, we would need to define more than one type of bond token. As such, the count x is not a good representation of the total bond supply in the pool. From the perspective of economics, a better choice should be the total cash value of these heterogeneous bonds.

AMM State. We thus propose to represent the AMM state as (X, y) where X is a nominal quantity in cash (DAI) to represent the present cash-value of bonds (of possibly different maturity dates) in the pool. This value is initialized according to the cash deposit and initial rate at pool creation and updated per transaction according to our pricing mechanic (presented later). The reference rate is a time-independent function of the bond-to-cash ratio expressed in X, not x:

$$r = R(\psi), \psi = \frac{X}{y}. \tag{9}$$

State (X, y) is changed only if triggered upon a transaction. Therefore, rate r remains constant as long as no transaction occurs, which is a desirable feature for avoiding arbitrage.

BondMM supports multiple maturity dates in the same pool. Any arbitrary short or long value for maturity date T_i can be requested by the order. At the same time, there can simultaneously exist bonds that have different T_i's. An order with maturity date T_i is executed as if it were fed into a fixed-rate AMM having a single maturity T_i, whose pool has $x_i = X/p_i$ bonds and y DAI, where

$$p_i = e^{-r(T_i - \tau)} \tag{10}$$

is the reference price of the T_i-bond at current time τ. So when no transaction occurs, X does not change over time but the count of bonds in the supply $x_i = X/p_i$ will automatically decrease. This is consistent with the increase of price p_i over time, thus obeying the supply-demand law. Below is a summary:

At current time τ, maturity date T_i :
state (X, y)
$\downarrow$
reference rate $r = R(X/y)$
$\downarrow$
T_i**-bond:** { **rate** $r_i = r$; **price** $p_i = e^{-r(T_i-\tau)}$; **count** $x_i = Xe^{r(T_i-\tau)}$

The details for our AMM are as follows.

4.1 Reference Rate Function

At any time, the AMM has one reference rate regardless of maturity date. However, the actual price to execute an order varies depending on the requested maturity date. As a result, the effective rate applied to the transacted bonds differs per maturity date. Specifically, we formulate the reference rate as the following logarithmic function of the bond-to-cash ratio:

$$r = R(\psi) = \kappa \cdot \ln(\psi) + r^* = \kappa \cdot \ln\left(\frac{X}{y}\right) + r^*. \tag{11}$$

Here, constant $r^* \in (0, 1)$ is the anchor rate, which should be chosen to reflect the long-term market mean, and constant $\kappa \in (0, 1)$ is the coefficient to tune the concentration of liquidity on the price curve. Note that our rate function is similar to Notional Finance in the use of the log form, but the crucial difference is that we use the bond-value proportion $\psi = \frac{X}{y}$ instead of bond-count proportion $\phi = \frac{x}{y}$. Another major difference is in how we design price functions. We will present this later, where we will see that the mathematics behind BondMM enables computation with elegant closed-form expressions.

4.2 Bond Tokens

Bonds are categorized into groups, each corresponding to a maturity date. Bonds of maturity date T_i, i.e., the T_i-bonds, are implemented on the blockchain as tokens that are fungible per T_i. The simplest way is to implement them as ERC-20 tokens. However, this way would lead to creation of many separate ERC-20 tokens for different maturity dates. Alternatively, which is more efficient, we can implement all tokens with different maturity dates together by defining a single ERC-721 non-fungible token (NFT). Whereas ERC-20 can represent only one type of fungible tokens, an ERC-721 NFT can represent multiple types of fungible tokens, corresponding to different maturity dates in our case.

Over time, bond tokens are minted or burnt upon incoming transactions. In the case of bond buying, the protocol will mint bond tokens to transfer to the buyer. In the case of bond selling, the protocol will burn the bonds received from the seller.

4.3 Liquidity Pool Creation

Suppose that we want to set up the AMM with initial cash $y = y_0$ DAI. At the beginning we should attract borrowers and so the initial rate $r = r_0$ should be set low enough to compete with the public market. The anchor rate r^* should be chosen to represent a belief about the rate in an equilibrium state. For example, if the market rate is 3% now but is expected to average around 5% in the future, then we could set $r_0 = 2.5\%$ and $r^* = 5\%$. Given these choices, we calculate the initial $X = X_0$ according to Eq. (11):

$$X_0 = y_0 e^{(r_0 - r^*)/\kappa}.$$

Note that the pool consists of cash only. Bond tokens stay outside the pool, in the hand of the lenders (bond holders).

4.4 Virtual State

For ease of explanation, assume for now that all bonds have the same maturity date T. Let p be the reference price for this maturity. Our intuition is that, because the bond supply is represented by their cash value X, we can imagine having $x = X/p$ bonds in the pool. Therefore, in our design, the pricing at AMM state (X, y) is equivalent to that of a two-asset pool having x bonds and y DAI such that the following equations hold true at any time-to-maturity t,

$$\begin{cases} X = xp \\ p = e^{-rt} \\ r = \kappa \ln \frac{X}{y} + r^* \end{cases} \tag{12}$$

We refer to (x, y) as the *virtual state* of the AMM. Note that the virtual state is maturity-specific. In general, the AMM at state (X, y) will have a virtual state (x_i, y) corresponding to each different maturity date T_i. For now, since we assume only one maturity date, there is only one virtual state.

Solving the above equation system, Eqs. (12), gets us the following relationships relating the state (X, y), virtual state (x, y), reference rate r, and reference price p.

Proposition 1. *Let t be time-to-maturity. Then,*

- *The reference rate r and reference price p can be expressed in terms of virtual state (x, y) as*

follows:

$$r = \frac{1}{1+\kappa t}\left(\kappa \ln \frac{x}{y} + r^*\right) \tag{13}$$

$$p = \left[\left(\frac{x}{y}\right)^{\kappa} e^{r^*}\right]^{-t/(1+\kappa t)} \tag{14}$$

- *The bond count x in virtual state (x, y) can be expressed in terms state (X, y) as follows:*

$$x = X^{1+\kappa t}\left(\frac{e^{r^*}}{y^{\kappa}}\right)^{t}. \tag{15}$$

Proof. The formula for the reference rate is

$$\begin{aligned} r = R(\psi) = \kappa \cdot \ln(\psi) + r^* = \kappa \cdot \ln\left(\frac{X}{y}\right) + r^* \\ = \kappa \cdot \ln\left(\frac{xe^{-rt}}{y}\right) + r^* = \kappa\left(\ln \frac{x}{y} - rt\right) + r^* \\ \Rightarrow r = \frac{1}{1+\kappa t}\left(\kappa \ln \frac{x}{y} + r^*\right). \end{aligned}$$

Then, it is easy to see the reference price as

$$\begin{aligned} p = \mathrm{e}^{-tr} = \exp\left[\frac{-t}{1+\kappa t}\left(\kappa \ln \frac{x}{y} + r^*\right)\right] \\ = \left[\left(\frac{x}{y}\right)^{\kappa} e^{r^*}\right]^{\frac{-t}{1+\kappa t}}. \end{aligned}$$

Next, using Eq. (14), we have

$$\begin{aligned} X = xp \\ = x\left[\left(\frac{x}{y}\right)^{\kappa} e^{r^*}\right]^{-t/(1+\kappa t)} \\ = x^{1/(1+\kappa t)}\left(\frac{e^{r^*}}{y^{\kappa}}\right)^{-t/(1+\kappa t)} \\ \Rightarrow x = \left[X\left(\frac{e^{r^*}}{y^{\kappa}}\right)^{t/(1+\kappa t)}\right]^{1+\kappa t} = X^{1+\kappa t}\left(\frac{e^{r^*}}{y^{\kappa}}\right)^{t}. \end{aligned}$$

□

We obtain the following important result showing that given the AMM configuration at any earlier time, we can compute the current virtual state (x, y) from the current reference rate r. This result applies to any generic rate function.

Proposition 2. *Assume a generic rate function $r = R(X/y)$. Suppose that at some earlier point in time, for example at initialization, the AMM was at state (X_1, y_1) with reference rate r_1. Let t be the current time-to-maturity, when the reference rate is r. Then the virtual state (x, y) is related to r as follows:*

$$x = x_1 \frac{1+R^{-1}(r_1)}{R^{-1}(r_1)} \frac{R^{-1}(r)}{1+R^{-1}(r)} \exp \int_{r_1}^{r} \frac{tdr}{1+R^{-1}(r)} \tag{16}$$

$$y = x_1 \frac{1+R^{-1}(r_1)}{R^{-1}(r_1)} \frac{1}{1+R^{-1}(r)} \exp \int_{r_1}^{r} \frac{tdr}{1+R^{-1}(r)} \tag{17}$$

where $x_1 = X_1/p_1 = X_1 e^{r_1 T}$.

Proof. We have

$$r = R\left(\frac{X}{y}\right) \Rightarrow y = \frac{X}{R^{-1}(r)} = \frac{xe^{-rt}}{R^{-1}(r)}. \tag{18}$$

The reference price is the instantaneous price at time-to-maturity t, hence

$$\frac{-dy}{dx} = p = e^{-rt}.$$

Thus we solve the following ODE system to find x and y,

$$\begin{cases} y = \frac{xe^{-rt}}{R^{-1}(r)} \\ \frac{dy}{dx} = -e^{-rt} \end{cases}$$

with initial condition (x_1, y_1, r_1).

Specifically, we have

$$-e^{-rt}dx = dy = \frac{\partial y}{\partial x}dx + \frac{\partial y}{\partial r}dr,$$

leading to

$$\left[\frac{\partial y}{\partial x} + e^{-rt}\right]dx + \frac{\partial y}{\partial r}dr = 0. \tag{19}$$

Let $U(r) \triangleq \frac{1}{R^{-1}(r)}$. From Eq. (18), we have

$$\frac{\partial y}{\partial x} = e^{-rt}U(r) \tag{20}$$

$$\frac{\partial y}{\partial r} = xe^{-rt}\Big[U'(r) - tU(r)\Big]. \tag{21}$$

Plugging these into Eq. (19),

$$\Big[e^{-rt}U(r)+e^{-rt}\Big]dx+xe^{-rt}\Big[U'(r)-tU(r)\Big]dr=0$$
$$\Leftrightarrow \frac{dx}{x}+\frac{U'(r)-tU(r)}{1+U(r)}dr=0.$$

This ODE is separable, having the solution below

$$x=\frac{x_1(1+U(r_1))}{1+U(r)}\exp\left[t(r-r_1)-t\int_{r_1}^{r}\frac{dr}{1+U(r)}\right]$$

(replacing U, we obtain:)

$$=\frac{x_1(1+\frac{1}{R^{-1}(r_1)})}{1+\frac{1}{R^{-1}(r)}}\exp\left[t(r-r_1)-t\int_{r_1}^{r}\frac{dr}{1+\frac{1}{R^{-1}(r)}}\right]$$
$$=x_1\frac{1+R^{-1}(r_1)}{R^{-1}(r_1)}\frac{R^{-1}(r)}{1+R^{-1}(r)}\exp\int_{r_1}^{r}\frac{tdr}{1+R^{-1}(r)}.$$

Then,

$$y=xe^{-rt}U(r)=xe^{-rt}\frac{1}{R^{-1}(r)}$$
$$=x_1\frac{1+R^{-1}(r_1)}{R^{-1}(r_1)}\frac{1}{1+R^{-1}(r)}\exp\int_{r_1}^{r}\frac{tdr}{1+R^{-1}(r)}.$$

□

As a corollary, we have the result below showing nice closed-forms for computing the virtual state (x,y) from reference rate r when R is our logarithmic rate function.

Corollary 3. *Let t be the current time-to-maturity, when the reference rate is r. Then the virtual state (x,y) is related to r as follows:*

$$x=X_0e^{r_0}\cdot\left[e^{\kappa^{-1}(r-r_0)}\cdot\frac{e^{\kappa^{-1}(r_0-r^*)}+1}{e^{\kappa^{-1}(r-r^*)}+1}\right]^{t\kappa+1} \tag{22}$$

$$y=y_0\cdot\left[\frac{e^{\kappa^{-1}(r_0-r^*)}+1}{e^{\kappa^{-1}(r-r^*)}+1}\right]^{t\kappa+1}. \tag{23}$$

Proof. The proof is obvious by applying Proposition 2 setting $(X_1,y_1,r_1)=(X_0,y_0,r_0)$ and $R(\psi)=\kappa\ln\psi+r^*$ and doing some simple derivations. □

4.5 Constant-Function Invariant

We have an important result below showing that the AMM virtual state has a constant-function invariant.

Proposition 4. *At any time-to-maturity t, the virtual state (x,y) satisfies the following constant-function invariant*

$$Kx^{\alpha}+y^{\alpha}=C \tag{24}$$

where C is constant specific to t, i.e., constant over all transactions happening at this time, and

$$\alpha\triangleq\frac{1}{1+t\kappa},\ K\triangleq e^{-tr^*\alpha} \tag{25}$$

The pool inventory must satisfy this equality before and after any order that is executed at time-to-maturity t.

Proof. Because $r=\kappa\ln\psi+r^*$, we have $\psi=e^{\kappa^{-1}(r-r^*)}$ and so

$$\frac{y}{x}=\frac{y}{Xe^{rt}}=\psi^{-1}e^{-rt}=e^{-\kappa^{-1}(r-r^*)-rt}$$
$$\Rightarrow r=\frac{r^*+\kappa\ln\frac{x}{y}}{1+t\kappa}$$
$$\Rightarrow\frac{r-r^*}{\kappa}=\frac{\frac{r^*}{\kappa}+\ln\frac{x}{y}}{1+t\kappa}-\frac{r^*}{\kappa}$$
$$=\frac{r^*}{\kappa}\left(\frac{1}{1+t\kappa}-1\right)+\frac{\ln\frac{x}{y}}{1+t\kappa}$$
$$=\frac{-tr^*+\ln\frac{x}{y}}{1+t\kappa}. \tag{26}$$

Now, let

$$C\triangleq y_0^{\frac{1}{1+t\kappa}}\left[e^{\kappa^{-1}(r_0-r^*)}+1\right]$$

where y_0 and r_0 are the cash reserve and rate at the pool initialization. We have

$$C^{t\kappa+1}=y_0\cdot\left(e^{\kappa^{-1}(r_0-r^*)}+1\right)^{t\kappa+1}$$
$$=y\left(e^{\kappa^{-1}(r-r^*)}+1\right)^{t\kappa+1}\text{ (from Eq. (23))}$$
$$=y\left(e^{\frac{-tr^*}{1+t\kappa}}\left(\frac{x}{y}\right)^{\frac{1}{1+t\kappa}}+1\right)^{t\kappa+1}\text{ (from Eq. (26))}$$
$$\Rightarrow C=y^{\frac{1}{1+t\kappa}}\left[K\left(\frac{x}{y}\right)^{\frac{1}{1+t\kappa}}+1\right]=Kx^{\alpha}+y^{\alpha}.$$

□

As a corollary, the state (X, y) also has a constant-function invariant.

Corollary 5. *At any time-to-maturity t, the state (X, y) satisfies the following constant-function invariant*

$$y^\alpha\left(\frac{X}{y}+1\right)=C \tag{27}$$

where C and α are constants specific to t formulated in Proposition 4; they are constant over all transactions happening at this time. The pool inventory must satisfy this equality before and after any order that is executed at time-to-maturity t.

Proof. We have

$$r=\kappa\ln(X/y)+r^*\Rightarrow e^{rt}=\left(\frac{X}{y}\right)^{\kappa t}e^{r^*t}$$

and so

$$\begin{aligned}C&=Kx^\alpha+y^\alpha=K(Xe^{rt})^\alpha+y^\alpha\\&=K(X\left(\frac{X}{y}\right)^{\kappa t}e^{r^*t})^\alpha+y^\alpha\\&=K(X^{1/\alpha}e^{r^*t}y^{-\kappa t})^\alpha+y^\alpha\\&=Xy^{-\kappa t\alpha}+y^\alpha=y^\alpha\left(\frac{X}{y}+1\right).\end{aligned}$$

□

As a constant-function AMM whose reference price depends only on the AMM state, our AMM satisfies the path-independent property. It thus makes no difference whether to execute a single large order or to split it into a sequence of smaller orders. Path independence is considered an axiom of any AMM [20]. An AMM without this property is vulnerable to attacks where one could make free gain from the AMM.

4.6 Order Execution

Now we are ready to present how to execute an order $(\Delta x, \Delta y)$ having any maturity date T_i. This maturity can be chosen arbitrarily, anew or among the existing maturity dates. Let τ be the current time when the AMM is at state (X, y) with reference rate r. The corresponding virtual state (x, y) for maturity T_i will have

$$x=x_i=\frac{X}{p_i}=\frac{X}{e^{-r(T_i-\tau)}}=Xe^{r(T_i-\tau)}. \tag{28}$$

We show below how to calculate Δy from Δx. The other direction, computing Δx from Δy, can similarly be derived.

Order pricing. Thanks to the constant-function invariant in Eq. (24), we have

$$K(x+\Delta x)^\alpha+(y+\Delta y)^\alpha=C=Kx^\alpha+y^\alpha,$$

where

$$t=T_i-\tau,\ \alpha=\frac{1}{1+t\kappa},\ K=e^{-tr^*\alpha}.$$

We then express Δy in terms of Δx,

$$\begin{aligned}\Delta y&=\left[C-K(x+\Delta x)^\alpha\right]^{1/\alpha}-y\\&=\left(Kx^\alpha+y^\alpha-K(x+\Delta x)^\alpha\right)^{1/\alpha}-y.\end{aligned} \tag{29}$$

We can now calculate the price of an order given the state.

Proposition 6. *The average price for executing an order of size Δx at time-to-maturity t can be expressed in terms of current state (X, y) as follows:*

$$\overline{p}=\frac{y}{\Delta x}\left[1-\left(\frac{X}{y}+1-\left(\left(\frac{X}{y}\right)^{\frac{1}{\alpha}}+\frac{K^{\frac{1}{\alpha}}\Delta x}{y}\right)^\alpha\right)^{\frac{1}{\alpha}}\right] \tag{30}$$

where α, K are defined in Proposition 4.

Proof. From (15) we have,

$$K^{\frac{1}{\alpha}}x=K^{\frac{1}{\alpha}}X^{1+\kappa t}\left(\frac{e^{r^*}}{y^\kappa}\right)^t=X^{\frac{1}{\alpha}}y^{-\kappa t}. \tag{31}$$

Eq. (29) then becomes

$$\Delta y=\left(Kx^\alpha+y^\alpha-K(x+\Delta x)^\alpha\right)^{\frac{1}{\alpha}}-y$$

$$= \left(X y^{-\kappa t \alpha} + y^\alpha - (X^{1/\alpha} y^{-\kappa t} + e^{-r^* t} \Delta x)^\alpha \right)^{\frac{1}{\alpha}} - y$$
$$= y \left(\frac{X}{y} + 1 - \left(\left(\frac{X}{y} \right)^{\frac{1}{\alpha}} + e^{-r^* t} \frac{\Delta x}{y} \right)^\alpha \right)^{\frac{1}{\alpha}} - y.$$

Putting this in $\overline{p} = \frac{-\Delta y}{\Delta x}$, we get the formula needed to prove. □

In the case that $\Delta x > 0$ (bond selling), the AMM will burn these Δx T_i-bonds and transfer $|\Delta y|$ DAI to the seller. In the otherwise case, $\Delta x < 0$ (bond buying), the AMM will receive Δy DAI and mint $|\Delta x|$ T_i-bonds to send to the buyer. Note that, because of this pricing, the effective rate applied to the transacted bonds is $\overline{r} = -\frac{1}{t} \ln \overline{p}$.

State update. After the transaction, the AMM changes state from (X, y) to $(X_{new}, y_{new} = y + \Delta y)$. Applying the constant-function invariant in Corollary 5, we have

$$y^\alpha \left(\frac{X}{y} + 1 \right) = (y_{new})^\alpha \left(\frac{X_{new}}{y_{new}} + 1 \right)$$
$$= (y + \Delta y)^\alpha \left(\frac{X_{new}}{y + \Delta y} + 1 \right)$$
$$\Rightarrow X_{new} = (y + \Delta y) \left[\left(\frac{y}{y + \Delta y} \right)^\alpha \left(\frac{X}{y} + 1 \right) - 1 \right]. \quad (32)$$

The new bond-to-cash ratio is

$$\psi_{new} = \frac{X_{new}}{y_{new}} = \left(\frac{y}{y + \Delta y} \right)^\alpha \left(\frac{X}{y} + 1 \right) - 1 \quad (33)$$
$$= \left(\frac{y}{y + \Delta y} \right)^\alpha \left(\psi + 1 \right) - 1. \quad (34)$$

The new reference rate is

$$r_{new} = \kappa \ln \psi_{new} + r^*$$
$$= \kappa \ln \left[\left(\frac{y}{y + \Delta y} \right)^\alpha \left(\psi + 1 \right) - 1 \right] + r^*.$$

4.7 Cross-Maturity Swap

Because multiple maturity dates can coexist in the liquidity pool, the user can easily convert bonds from one maturity date T_1 to another maturity date $T_2 \neq T_1$. A practical reason for doing this is to swap interest rates. Our AMM processes this swap as a sequence of two instantaneously consecutive sub-transactions: 1) sell T_1-bonds to get Δy DAI and then 2) immediately spend this Δy DAI to buy T_2-bonds.

$$\begin{bmatrix} X \\ y \\ r \end{bmatrix} \xrightarrow[\overline{p}_1]{\Delta x_1, \Delta y_1, T_1} \begin{bmatrix} X_1 \\ y_1 \\ r_1 \end{bmatrix} \xrightarrow[\overline{p}_2]{\Delta x_2, \Delta y_2, T_2} \begin{bmatrix} X_2 \\ y_2 = y \\ r_2 \end{bmatrix}.$$

Here, $\Delta y_1 = -\Delta y$, $\Delta y_2 = \Delta y$. Note that $\Delta y > 0$, $\Delta x_1 > 0$ (sell bond), $\Delta x_2 < 0$ (buy bond). $|\Delta x_1|$ is the number of T_1-bonds to be converted and $|\Delta x_2|$ is the number of T_2-bonds to receive back. $\overline{p}_1$ and $\overline{p}_2$ are the corresponding trade prices.

Applying Eq. (30) on the first sub-transaction, we have $\Delta y_1 = yA - y$ where

$$A = \frac{X}{y} + 1 - \left(\left(\frac{X}{y} \right)^{\frac{1}{\alpha_1}} + \frac{e^{-r^* t_1} \Delta x_1}{y} \right)^{\alpha_1}$$

and so

$$y_1 = y + \Delta y_1 = y(A^{1/\alpha_1} - 1). \quad (35)$$

Similarly, applying Eq. (30) on the second sub-transaction, we have $\Delta y_2 = y_1 B - y_1$ where

$$B = \frac{X_1}{y_1} + 1 - \left(\left(\frac{X_1}{y_1} \right)^{\frac{1}{\alpha_2}} + \frac{e^{-r^* t_2} \Delta x_2}{y_1} \right)^{\alpha_2}$$

and so

$$y_2 = y_1 + \Delta y_2 = y_1(B^{1/\alpha_2} - 1). \quad (36)$$

Because $y = y_2$, combining Eq. (35) and Eq. (36), we have the equality

$$A^{\frac{-\alpha_2}{\alpha_1}} = B$$
$$= \frac{X_1}{y_1} + 1 - \left[\left(\frac{X_1}{y_1} \right)^{\frac{1}{\alpha_2}} + \frac{e^{-r^* t_2} \Delta x_2}{y_1} \right]^{\alpha_2}. \quad (37)$$

To find an expression for $\frac{X_1}{y_1}$, applying the constant-function invariant to the first sub-transaction, we have

$$y^{\alpha_1} \left(1 + \frac{X}{y} \right) = y_1^{\alpha_1} \left(1 + \frac{X_1}{y_1} \right) \quad (38)$$
$$\Rightarrow \frac{X_1}{y_1} + 1 = \left(\frac{y}{y_1} \right)^{\alpha_1} \left(1 + \frac{X}{y} \right) \quad (39)$$

$$\Rightarrow \frac{X_1}{y_1} = \left(\frac{y}{y_1}\right)^{\alpha_1}\left(1+\frac{X}{y}\right) - 1 \tag{40}$$

$$= A^{-1}\Big(\underbrace{1+\frac{X}{y}}_{D}\Big) - 1 \text{ (from Eq. (35))} \tag{41}$$

$$= A^{-1}D - 1. \tag{42}$$

Plugging this expression of $\frac{X_1}{y_1}$ in Eq. (37), we have

$$A^{\frac{-\alpha_2}{\alpha_1}} = A^{-1}D - \left[(A^{-1}D-1)^{\frac{1}{\alpha_2}} + \frac{e^{-r^*t_2}\Delta x_2}{yA^{\frac{1}{\alpha_1}}}\right]^{\alpha_2}.$$

We can easily solve this equation to get the amount Δx_2 of T_2-bonds expressed in terms of the AMM state (X, y) and the amount Δx_1 of T_1-bonds. Hence, the following result regarding the pricing for cross-maturity bond conversion.

Proposition 7. *Suppose that the AMM is at state (X, y) when the user wants to trade in Δx_1 T_1-bonds to get T_2-bonds. Let τ be the current time. Then, the corresponding number of T_2-bonds to return is*

$$\Delta x_2 = e^{r^*t_2}yA^{\frac{1}{\alpha_1}-\frac{1}{\alpha_2}} \times \left[\left(D - A^{1-\frac{\alpha_2}{\alpha_1}}\right)^{\frac{1}{\alpha_2}} - \left(D-A\right)^{\frac{1}{\alpha_2}}\right] \tag{43}$$

where

$$t_1 = T_1 - \tau,\ t_2 = T_2 - \tau,\ D = \frac{X}{y} + 1,$$
$$\alpha_1 = \frac{1}{1+\kappa t_1},\ \alpha_2 = \frac{1}{1+\kappa t_2},$$
$$A = \frac{X}{y} + 1 - \left[\left(\frac{X}{y}\right)^{1/\alpha_1} + \frac{e^{-r^*t_1}\Delta x_1}{y}\right]^{\alpha_1}.$$

Rate impact. How does a cross-maturity bond swap impact the rate? Applying the constant-function invariant to the two sub-transactions of the swap, we have the following equalities

$$y^{\alpha_1}\left(1+\frac{X}{y}\right) = y_1^{\alpha_1}\left(1+\frac{X_1}{y_1}\right) \tag{44}$$

$$y_1^{\alpha_2}\left(1+\frac{X_1}{y_1}\right) = y_2^{\alpha_2}\left(1+\frac{X_2}{y_2}\right) \tag{45}$$

leading to

$$y_1^{\alpha_2-\alpha_1}y^{\alpha_1}\left(1+\frac{X}{y}\right) = y_2^{\alpha_2}\left(1+\frac{X_2}{y_2}\right) \tag{46}$$

$$\Rightarrow y_1^{\alpha_2-\alpha_1}\left(1+\frac{X}{y}\right) = y^{\alpha_2-\alpha_1}\left(1+\frac{X_2}{y}\right) \tag{47}$$

(because $y_2 = y$).

We thus obtain the following equivalences:

$$r_2 < r \Leftrightarrow X_2 < X \tag{48}$$

$$\Leftrightarrow \left(\frac{y_1}{y}\right)^{\alpha_2-\alpha_1} < 1 \tag{49}$$

$$\Leftrightarrow \alpha_2 < \alpha_1, \text{ because } y_1 > y \tag{50}$$

$$\Leftrightarrow T_1 < T_2. \tag{51}$$

Thus the impact of bond conversion is that rate r will decrease if converting long-to-short bonds and increase if converting short-to-long bonds. This makes sense according to the principle of supply and demand. If lenders keep converting for shorter loans in hopes of making quick return, the rate will be lower, which protects the solvency capacity of the AMM. On the other hand, if they keep converting for longer term, the rate will rise, thus encouraging more short-term lending, which reduces the time window of the AMM liability.

Note that before the conversion, the cash value of the T_1-bonds is Δy_1 if sold to the AMM. After the conversion, the cash value of the T_2-bonds if sold to the AMM is also $\Delta y_2 = \Delta y_1$ which is the exact cash amount to spend to buy the T_2-bond earlier in the swap. This is thanks to the path independence of our AMM. Thus, no trader can gain free cash by cross-maturity conversion.

4.8 Financial Stability

Thanks to constant-function invariance and path independence properties, the liquidity pool always has sufficient cash to fulfill orders that sell bonds to the pool or that redeem bonds at maturity. However, this may not be guaranteed if the LPs withdraw a lot of liquidity leading to cash shortage. In addition, if the AMM allows for much more lending than borrowing, the LP profit would be negative. Therefore, for practical implementation, when the LPs withdraw liquidity or a new lending position being created, i.e., bond buying, we should allow this action only if the AMM is able

to maintain its equity above a certain threshold ρ, for example $\rho = 90\%$ of the initial equity to cap the loss, if any, to 10% max.

Let b_i and l_i be the number of active T_i-bonds issued by borrowers and that currently held by lenders, respectively. If no new transaction comes, the total cash in the pool at any time τ is

$$E(\tau) = y + \sum_{T_i \le \tau} (b_i - l_i) \tag{52}$$

because the pool will have, for each expired maturity date $T_i \le \tau$, received b_i (DAI) from the borrowers and paid l_i (DAI) to the lenders. We call this quantity $E(\tau)$ the net equity at time τ.

Consider a bond-buying order $(\Delta x, \Delta y)$ that arrives now for some maturity date T. If we executed this order it would reduce the equity regarding those maturity dates that are T or later. Hence, we will execute this order only if

$$\min_{T_i \ge T} E(T_i) + \Delta y - |\Delta x| \ge \rho y_0. \tag{53}$$

4.9 LP Addition and Withdrawal

An LP can contribute liquidity anytime by adding a cash amount to the pool. Let this amount be $\Delta y > 0$. As a result, the AMM moves state from (X, y) to $(X^{new}, y^{new} = y + \Delta y)$. It is required that the reference rate be unchanged. This is equivalent to keeping the same bond-to-cash ratio. That is,

$$\psi^{new} = \frac{X^{new}}{y^{new}} = \frac{X^{new}}{y + \Delta y} = \psi = \frac{X}{y}.$$

Hence,

$$X^{new} = \frac{X(y + \Delta y)}{y}.$$

Now think of the AMM as a company whose shareholders are the LPs. When all maturity dates expire, the long-term LP equity is

$$E = E(\max\{T_i\}) = y + \sum_{T_i} (b_i - l_i).$$

After the new LP is added, the new equity is $E + \Delta y$. The new LP will own a fraction $s = \frac{\Delta y}{E + \Delta y}$ of the pool. The other LPs will have their share diluted accordingly.

When withdrawing liquidity, that LP can withdraw a fraction $s_1 \le s$ of the pool at the most. If the state is (X, y), the cash amount withdrawn will be $s_1 y$. The new state will be

$$X^{new} = X(1 - s_1),\ y^{new} = y(1 - s_1)$$

to keep the reference rate and bond-to-cash ratio unchanged. The other LPs will have their share diluted accordingly.

Regarding the fraction s_1, removing $y s_1$ (DAI) from the pool now would reduce the equity regarding all the active maturity dates. Thus, we require

$$\min_{T_i} E(T_i) - y s_1 \ge \rho y_0. \tag{54}$$

This LP then can withdraw a share fraction at most

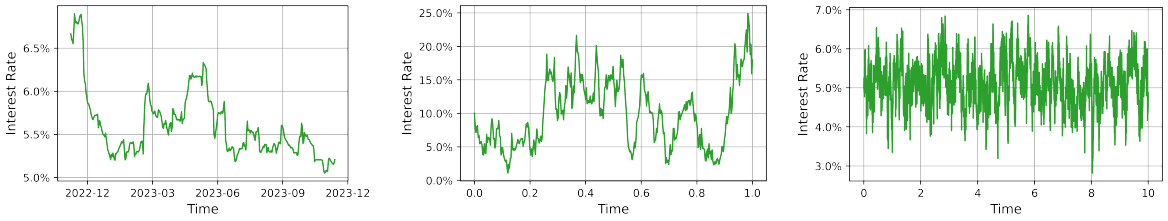
$$s_1 \le \min\left\{ s, \frac{\min_{T_i} E(T_i) - \rho y_0}{y} \right\}.$$

5 Numerical Results

We conducted an evaluation study with both real-world and synthetic datasets to analyze and validate key properties of BondMM[2]. There are two cases for evaluation:

- Same maturity date: All bonds have the same maturity date. We compare BondMM to Yield and Notional, the two main existing fixed-rate AMM models, with respect to four important aspects: interest rate, price impact, equity, and capital efficiency. Yield was the first mover, whereas Notional is the AMM model for leading fixed-rate commercial products today. Two maturity dates are evaluated: 1) all bonds mature after 1 year since the pool creation, and 2) all bonds mature after 10 years since the pool creation.
- Multiple maturity dates: Different maturity dates co-exist. Because Yield and Notional are not applicable here, we investigate BondMM only. We consider three practical products based on BondMM: 1) *Single-maturity pool*: All bonds expire after 2 years since the pool creation; 2) *Periodic-maturity pool*: All bonds expire at the current year's end. Bonds transacted in year 1 expire at the

[2] All datasets, plots, and simulation source codes are available for download at https://www.cs.umb.edu/~duc/research/bondmm/



(a) AAVE (365 tx, daily frequency) (b) CIR (500K+ tx, minute frequency) (c) VASICEK (5M+ tx, minute frequency)

Fig. 1 Market rate datasets used in the experiments: (a) Real-world ETH stable rates from AAVE protocol over one year of daily frequency; (b) Synthetic rates generated by the CIR model over 1 year of minute frequency; and (c) Synthetic rates generated by the VASICEK model over 10 years of minute frequency.

end of year 1. Bonds transacted in year 2 expire at the end of year 2; 3) *Mixed-maturity pool*: At anytime, we can have bonds that expire at the end of the current year n or bonds that expire at the end of the next year $n+1$, where $n = 1, 2, 3, ...$

There are four transaction activities: borrowing, lending, closing a borrowing position, and closing a lending position. These are modeled in our simulation as transactions of bond buying and bond selling. Moreover, we randomize the types of transaction with equal probability. Precisely, a sell order is treated as either a new borrowing position or a lending closing position with equal probability 1/2. Similarly, a buy order is treated as either a new lending position or a borrowing closing position with equal probability 1/2. We assume no LP change during the simulation run.

5.1 Single Maturity: BondMM vs. Yield vs. Notional

In this comparison, each AMM protocol would start with the same initial cash $y_0 = 1$ DAI and same initial rate r_0 and perform under the same input sequence of orders. The value of r_0 is given by the input dataset used in each simulation run. In BondMM and Notional, the anchor rate is set to $r^* = r_0$ and the concentration coefficient is set to $\kappa = 0.02$ (it is often observed in traditional exchanges that the price usually moves within 2% of the current price upon a transaction.)

Datasets. We evaluated with one real-world dataset from the AAVE and two synthetic datasets.

- Real-world (AAVE): We collected a time series of borrowing stable rates from the ETH lending pool on AAVE Protocol with daily frequency over one year from Dec 1st, 2022 to Nov 30, 2023. This data set represents a market scenario in which the market rates are trending in one direction towards more lending over the time. Figure 1(a) plots the AAVE interest rates. The initial rate r_0 is about 6.6%.
- Synthetic (CIR and VASICEK): To evaluate with higher-frequency and larger data, we generated two synthetic datasets using the Cox–Ingersoll–Ross (CIR) [43] and VASICEK model [44] with realistic statistics such as max, min, average, volatility, etc. These are the two most popular models in financial mathematics to generate interest rates over time. The CIR dataset has 525,600 transactions every minute over 1 year, representing a market with a wider rate variation. The VASICEK dataset has 5,256,000 transactions every minute over 10 years, representing a normal market scenario where interest rates are mean-reversing up and down normally. Figures 1(b, c) plot our synthetic CIR and VASICEK market rates, respectively. The initial rate r_0 is 10% in the CIR dataset and 5% in the VASICEK dataset.

For comparing the protocols under the same sequence of orders, we generate a realistic order input using the above market rate data. Specifically, given a time series of market rates (r_t), from the AAVE, CIR, or VASICEK dataset, we reconstruct a corresponding time series of orders (Δx_t) such that executing these orders on an independent interest rate protocol, which we call the "Linear Protocol", would result in these market rates (r_t). We then feed these same orders (Δx_t)

to BondMM, Yield, and Notional to compare. The Linear Protocol is described by the following equation, $r_t = r_{\min} + (r_{\max} - r_{\min}) \ln \frac{x_t}{y_t}$, where parameters $r_{\min}, r_{\max}$ are chosen symmetrically around initial rate r_0.

5.1.1 Interest Rate

We plot the interest rates over the time of the protocols as a result of processing the input orders; see Figure 2 for different settings: AAVE dataset with 1-year maturity, CIR dataset with 1-year maturity, and VASICEK dataset with 10-year maturity. In all scenarios, it is obvious that the interest rate curve of Yield is the most volatile, which oftentimes reaches negative rates. In contrast, Notional and BondMM are much more stable and have similar rate volatility. Especially, extreme volatility is observed for Yield in the case of 1-year maturity for both the AAVE dataset (Figure 2(a)), where the rate can go negative reaching -150%, and the CIR dataset (Figure 2(b)) where the rate can reach -100%. These rates are absolutely surreal. Even for the VASICEK dataset which represents a stable market, Yield rates still fluctuate widely and can go below zero; see Figure 2(c).

We can explain this though. The high rate volatility of Yield is due to the fact that the concentration coefficient κ in the rate formula of Yield is equal to $1/T$, which is much worse than $\kappa = 0.02$ set in the other protocols. Indeed, for maturity of $T = 10$ years, Yield has $\kappa = 1/10 = 0.1$, five times worse than BondMM and Notional. For maturity of $T = 1$ year, it is $\kappa = 1/1 = 1.0$, or 50 times worse. When the order size is the same to the three protocols, the one with the worst concentration coefficient κ will force the interest rate to fluctuate the most. It is not desirable to have a lending protocol with extremely volatile rates, and, in this aspect, Yield is not practical. Both BondMM and Notional offer stable rates because they can choose an arbitrarily small value for κ. In contrast, Yield does not have this flexibility.

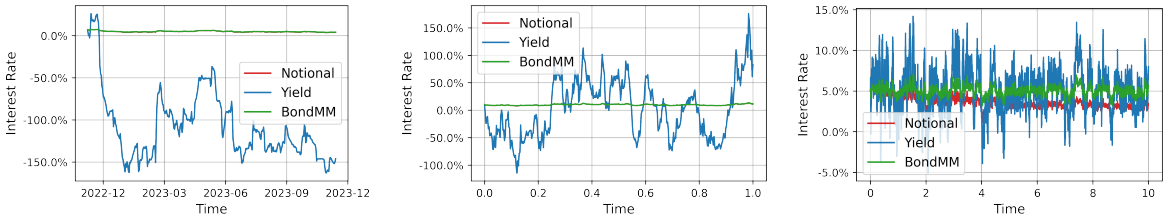
5.1.2 Price Impact

Traders do not like price slippage when executing a large order. Worse slippage means higher trading cost. To compare the price impact of the three protocols, we computed the trade price of each order as a percentage slippage from the current price. Applying to the different datasets and different maturities, Figure 3 shows the price-impact histogram, plotting the number of orders that experience a given price slippage. As seen, Yield is extremely expensive. Its histogram is almost flat over all price impact values, implying a very high degree of price slippage. Notional is not that extreme, but more expensive than BondMM, especially with longer maturity. Let us look at the 95%-quantile worst-case scenario. With 1-year maturity, Notional has a price impact about 1.5 times higher than BondMM; 0.17% vs. 0.12% for AAVE dataset (Figure 3(a)) and 0.04% vs. 0.027% for CIR (Figure 3(b)) . The gap is wider for the 10-year maturity, as seen in Figure 3(c), where Notional (0.34%) is more than twice costlier than BondMM (0.16%) and Yield (0.89%) is 5+ times costlier. This evaluation obviously shows that BondMM pricing is the most attractive for traders.

5.1.3 Equity

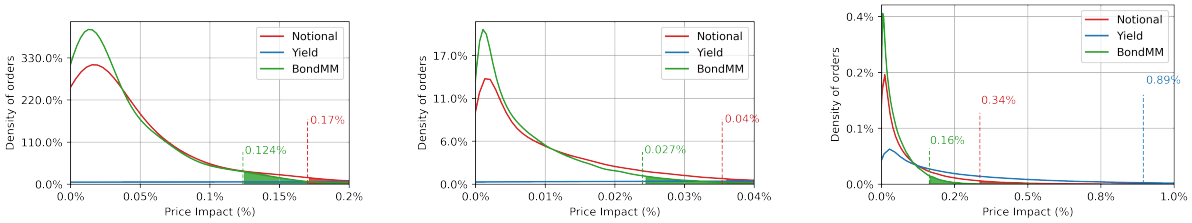
If the equity loss is too much, no rational LPs would want to provide liquidity to the protocol. On the opposite side, if the equity gain is too much, no traders would use the protocol because it causes loss for them via implicit costs such as price impact. Therefore, a fair bond protocol needs equity to be as close to the initial equity as possible. In other words, the equity PnL should approach zero as closely as possible, except when the net bond position is one-directional (much more lending volume than borrowing and the other way around). Another important criterion is the stability of the equity PnL: the more stable it is, the better LPs are protected against market uncertainty.

Figure 4(a) (for the AAVE dataset with 1-year maturity), Figure 4(b) (for the CIR dataset with 1-year maturity), and Figure 4(c) (for VASICEK dataset with 10-year maturity) show the LP equity of each protocol over the time. In all three scenarios, BondMM is the absolute winner because its equity is not only the closest to initial equity but also is the most stable. For the short maturity (1-year), BondMM and Notional have comparable equity. This is understandable because the bond maturity is too short for these two protocols to substantially differentiate from each other.



(a) AAVE: 1-year maturity (b) CIR: 1-year maturity (c) VASICEK: 10-year maturity

Fig. 2 Interest rate of each protocol as a result of processing input orders over the time. In plots (a) and (b), the Notional and BondMM curves are highly similar, hence seen as the same.



(a) AAVE: 1-year maturity (b) CIR: 1-year maturity (c) VASICEK: 10-year maturity

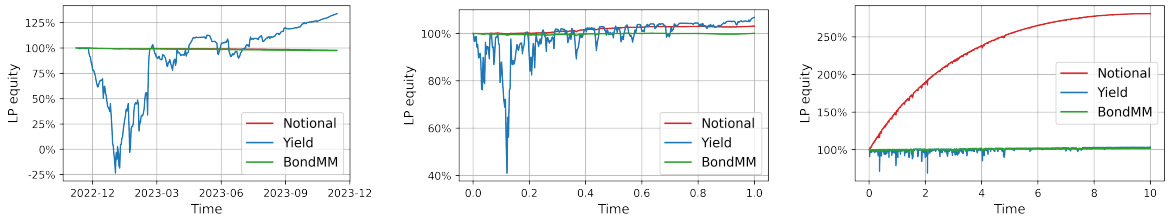
Fig. 3 Price impact measured as a percentage slippage from current price when pricing an order. The y-axis is the proportion of orders having a given slippage. In plots (a) and (b), the Yield impact is very small, hence seen close to the y = 0% line.

However, Yield's equity curve is extremely bad, particularly during the early phase of the protocol. In the AAVE scenario, Yield's equity can decrease below zero, which is unacceptable. In the CIR scenario, the LPs can lose as much as 60% of the initial equity. We see that this equity drop happens when the rate also goes negative, seeing a lot of bond buying (lending). As such, the pool is obligated to owe the lender large amounts. In contrast, BondMM and Notional still maintains stable equity close to the initial. The results here point out a critical weakness of Yield: it would potentially collapse in a lending-heavy market.

In the more stable market with the VASICEK dataset (Figure 4(c)), where the rates are mean-reversed, Yield is able to stabilize its equity. In this case, the more interesting observation is about Notional. Due to its greedy trade pricing, as the market is stable with balanced net bond positions, Notional accumulates a lot more cash from trading compared to the other protocols, explaining why its equity grows higher noticeably. This is not good though. Notional quickly doubles equity after two years, too greedy and unrealistic; no traders would come to Notional. As aforementioned, we want an equity that is stable and as close to the initial equity as possible. We see that BondMM satisfies both conditions. Yield is still unstable during early time, when it can lose 30% of equity; see Figure 5 for a zoom-in plot comparing BondMM and Yield.

5.1.4 Capital Efficiency

While equity is a factor to represent fairness between the LP and the traders, capital efficiency affects LP exclusively. It is always desirable to maximize it. To compare capital efficiency of the three protocols, we assume that they reach an equilibrium state having the same cash reserve in the pool and the the same rate as market rate, and then compute how much bond volume can be fulfilled by each protocol to change the rate by the same amount. The larger this volume, the more efficient the LP capital is utilized. In other words, we have a better liquidity depth. Specifically, we set the market rate at 5% and the results are shown in Figure 6 for two cases of maturity, $T = 1$ year and $T = 10$ years. Yield is obviously the least capital efficient. BondMM and Notional have similar capital efficiency with 1-year maturity. With



(a) AAVE: 1-year maturity (b) CIR: 1-year maturity (c) VASICEK: 10-year maturity

Fig. 4 LP equity over the time. In plot (a), the Notional and BondMM curves are highly similar, hence seen as the same.

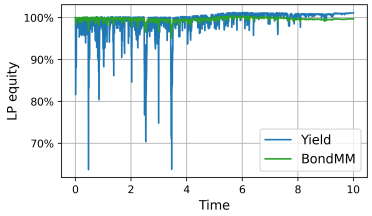


Fig. 5 LP equity over the time: BondMM versus Yield for the VASICEK dataset with 10-year maturity.

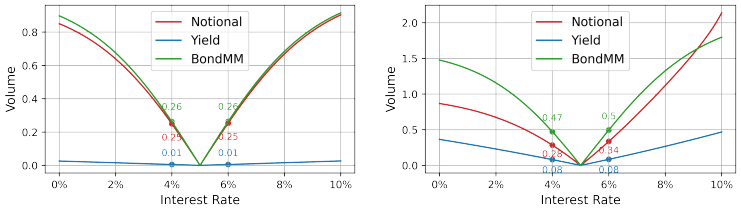


Fig. 6 Capital efficiency: The y-axis is the bond volume, relative with the bond quantity in the pool, that can be fulfilled to reach a given rate. Here, (left) 1-year maturity; and (right) 10-year maturity.

the longer 10-year maturity, BondMM is noticeably more efficient. For example, to move the current rate from 5% to 4%, BondMM can accommodate a trade volume 0.47 (in bond unit), where Notional and Yield can only do 0.28 and 0.08. Put another way, BondMM is almost twice more capital efficient than Notional and 50 times more than Yield. This study confirms our hypothesis that our rate formulation results in a better liquidity concentration around realistic rate ranges (in this evaluation, the 5% market rate), especially when the maturity duration increases.

5.2 Multiple Maturities: Shared Pool vs. Separate Pools

Now we evaluate the case allowing different maturity dates to co-exist. Because Yield and Notional are not designed for this case, we consider BondMM only and compare three product settings (described at the beginning of Section 5): the single-maturity pool, the periodic-maturity pool, and the mixed-maturity pool. Our hypothesis is that the mixed-maturity pool would profit the LPs the most.

These three pools start at the same time and compete with each other. Rationally, if their rates differ, the user will tend to the pool with the best rate, thus moving the rate of this pool closer to the other pools. Therefore, to simulate this equilibrium-approaching behavior, we assume that bond orders will go into each pool such that the reference rate is the same across all three pools. Then we will find out which pool brings the best revenue for the LPs. LP revenue comes from transaction fee. Our evaluation assumes a realistic fee structure such that the fee for 0.5, 1, 2-years-to-maturity bonds is 0.1%, 0.2%, 0.4% extra charge to the transaction rate, respectively, and this fee decreases toward zero as time approaches maturity (details of the fee are expressed in our simulation source code).

Datasets. We apply the Vasicek model to simulate a time series of rates over time. To see the difference between the pools more clearly, we consider two imbalanced market scenarios leading to creation of two datasets:

- The VASICEK/DOWN dataset: The trend is more lending demand than borrowing. The rate starts at $r_0 = 5\%$ and can go up and down, but it tends to decrease to an average of 4%. Thus we set $r^* = 4\%$. As a result, we obtain a series of 1,051,000 rate values at every minute over 2 years. Figure 7(a)(leftmost) plots these rates.

- The VASICEK/UP: The trend is more borrowing demand than lending. The rate starts at $r_0 = 5\%$ and can go up and down, but it tends to increase to an average of 6%. Thus we set $r^* = 6\%$. As a result, we obtain a series of 1,051,000 rate values at every minute over 2 years. Figure 7(b)(leftmost) plots these rates.

5.2.1 LP Revenue

In the rate-down market scenario, which is simulated by using the VASICEK/DOWN dataset, the revenue of each pool accumulated over time is plotted in Figure 7(a)(middle). It is observed that in the first year, all the pools perform almost identically. However, starting in the second year, the mixed-maturity pool starts to earn more fee compared to the other pools. At the end of year 2, it has a 28% revenue margin, compared to 25% for the periodic-maturity pool and 23% for the single-maturity pool. In the rate-up market scenario (VASICEK/UP dataset), the mixed-maturity pool also makes the largest revenue, 35% after 2 years, compared to 30% for both periodic-maturity and single-maturity pools. Between the latter two pools, the single-maturity pool is slightly better. Combining both scenarios, it is convincing to conclude that the mixed-maturity pool brings the most money to the LPs.

5.2.2 Cash in the Pool

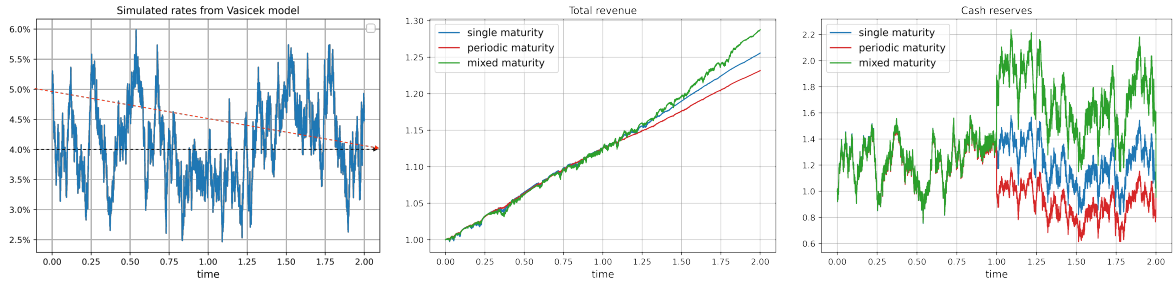
Figure 7(a)(rightmost) and Figure 7(b)(rightmost) plot the pool's cash amount over time for the two market scenarios, respectively. It is consistently observed that the mixed-maturity pool has the most cash reserve. Note that the cash reserve should increase if there is more lending (cash into the pool) and decrease if there is more borrowing (cash out of the pool). Interestingly, the mixed-maturity pool tends to have increasing cash reserve over time in both scenarios, even when the rate is uptrend (i.e., the general market is lender-favorable). Perhaps, it is because the pool allows transactions with bonds of different maturity dates in the same pool. This flexibility, plus the capability to fulfill the largest transaction volume (resulting in the highest revenue), makes the mixed-maturity pool a better choice for the LPs than the single-maturity pool and periodic-maturity pool.

6 Discussions

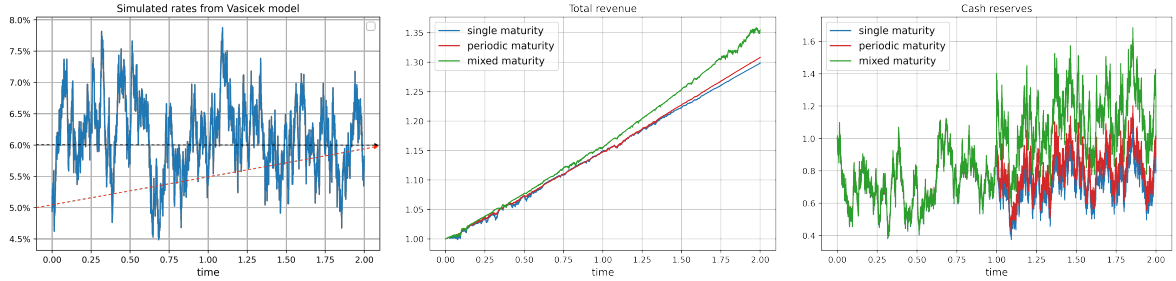
At the very core, BondMM is like a typical trading AMM in that it is based on constant-function bonding curves. These constant functions on one hand ensure unbounded liquidity for every possible trade but on the other hand may lead to unnecessary liquidity provisioned for cases of extreme prices. This is inefficient in capital use for those LPs that place liquidity in those extreme ranges (rarely happening). Although in the previous sections we have shown BondMM more capital-efficient than existing fixed-rate AMMs (Yield and Notional), this efficiency could be further improved. Currently, a limitation of BondMM is that it does not allow LPs to specify preferences on what bond price or interest-rate range to concentrate their liquidity. In our future work, we will pursue this direction. We will learn from existing developments on optimal AMM liquidity customization strategy to maximize LP payoff, which we already discussed in Section 2, and explore ways to make similar LP capital efficiency improvements on BondMM.

Another issue we have not addressed in the paper is how to structure the transaction fees. In a fixed-rate AMM, the LPs have to pay interests to the lenders when lending activities dominate borrowing activities. In the current version of BondMM, we set a stop-loss threshold at 10% to cap the LP net loss, if any, at 10%. The limitation is that this threshold is manual set, which does not take into account the fee structure. Ideally, we should increase this threshold to allow for more lenders to join but and at the same time, to be financially solvent, increase the fee charging lenders (which is reasonable due to increasing demands from the lenders). Fee mechanism design is an important problem for our future work.

The third issue of setting up the parameters for BondMM such that they represent a specific market. There are three key parameters: liquidity concentration coefficient κ, initial interest rate r_0, and anchor interest rate r^*. The current version of BondMM is limited in that these parameters must be set manually depending on the market at the time of AMM pool creation. Ideally, these parameters should follow the progressing real-time state of the market. As we discussed above, we will investigate ways to adjust κ automatically to satisfy LP concentration preferences over the time.



(a) VASICEK/DOWN dataset, representing a market scenario where rate is trending down, i.e., more lending than borrowing



(b) VASICEK/UP dataset, representing a market scenario where rate is trending up, i.e., more borrowing than lending

Fig. 7 Separate pools (single-maturity, periodic-maturity) vs. Shared pool (mixed-maturity): In both market scenarios, the multi-maturity pool has the largest revenue and cash reserve.

Regarding initial rate r_0, we only need to set it at the beginning and so choosing a rate that is competitive with the current market rate is sufficient. However, the choice for the anchor rate r^* needs further thinking into the market in the future. One solution is to allow changing this rate via the smart contract when necessary and approved by the owners and user community of the protocol.

Regarding scalability, the performance of BondMM if having high-frequency transactions depends on the processing capability of the underlying blockchain network. Fortunately, this capability is increasingly improved with faster blockchain networks being developed at both layer 1 and layer 2. The mathematics of BondMM is simple enough not to raise any scalability bottleneck by itself.

Last but not least, although our paper is focused on the computing aspects of building a fixed-rate AMM, we would like to emphasize the importance of regulatory and governance matters. As BondMM is fully automated and decentralized without any entry point for centralized management, if future regulations require the user to KYC or submit certain documents, we would need a service layer on top of BondMM to process these documents to satisfy the regulation. Also, the smart contracts that implement BondMM must be rigorously audited to ensure zero programming risks for real-world implementation. All these are common for every DeFi protocol, not just BondMM. We leave these issues, as well as other related operational risks and governance matters, beyond the scope of our paper and invite the reader to [45] for experts' opinions and recommendations, some specific to lending service.

7 Conclusions

We have proposed BondMM, a new AMM protocol for fix-rate lending and trading. It is better than today's fixed-rate AMMs in terms of price impact (more attractive to users), capital efficiency (more attractive to LPs), and equity stability (better balance between trading costs to users and equity for LPs to avoid liquidity and solvency risks). In addition, an innovative feature unseen in the literature and practice of DeFi is that BondMM allows co-existence of different maturity dates in the same liquidity pool. This avoids the

LP liquidity fragmentation which is incurred if we use separate pools for different maturity dates. The user can go into BondMM and choose any arbitrary maturity date to transact with bonds. This is desirable in terms of user experience as well as investment leverage power. In the future work, we will investigate the directions to overcome the potential limitations and improve the performance of BondMM as just discussed in the previous section.

Declarations

Funding

The authors did not receive support from any organization for the submitted work.

Conflict of interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] D. A. Tran, M. T. Thai, and B. Krishnamachari, Eds., *Handbook on Blockchain*. Springer, 2022.

[2] BarnBridge, "Barnbridge white paper," 2021. [Online]. Available: https://github.com/BarnBridge/BarnBridge-Whitepaper

[3] Swivel Finance. [Online]. Available: https://swivel.finance/

[4] AAVE, "AAVE protocol (white paper)," 2020. [Online]. Available: https://github.com/aave/aave-protocol/blob/master/docs/Aave_Protocol_Whitepaper_v1_0.pdf

[5] R. Leshner and G. Hayes, "Compound: The money market protocol (white paper)," 2019. [Online]. Available: https://compound.finance/documents/Compound.Whitepaper.pdf

[6] A. Niemerg, D. Robinson, and L. Livnev, "Yieldspace: An automated liquidity provider for fixed yield tokens (white paper)," 2020. [Online]. Available: https://yield.is/YieldSpace.pdf

[7] Notional Finance, "Notional protocol (white paper)," 2020. [Online]. Available: https://docs.notional.finance/developers/whitepaper/whitepaper

[8] Pendle Finance. [Online]. Available: https://www.pendle.finance/

[9] Term Finance. [Online]. Available: https://term.finance/

[10] Element Finance, "The element protocol construction paper." [Online]. Available: https://paper.element.fi/

[11] J. C. Schlegel, M. Kwaśnicki, and A. Mamageishvili, "Axioms for constant function market makers," in *Proceedings of the 24th ACM Conference on Economics and Computation*, ser. EC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1079. [Online]. Available: https://doi.org/10.1145/3580507.3597720

[12] G. Angeris, A. Evans, T. Chitra, and S. P. Boyd, "Optimal routing for constant function market makers," in *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, D. M. Pennock, I. Segal, and S. Seuken, Eds. ACM, 2022, pp. 115–128. [Online]. Available: https://doi.org/10.1145/3490486.3538336

[13] V. Buterin, "On path independence," 2017. [Online]. Available: https://vitalik.ca/general/2017/06/22/marketmakers.html

[14] T. Tran, D. A. Tran, and T. Nguyen, "Order book inspired automated market making," *IEEE Access*, vol. 12, pp. 36 743–36 763, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3372402

[15] MakerDAO, "The maker protocol: Makerdao's multi-collateral dai (mcd) system (white paper)," 2019. [Online]. Available: https://makerdao.com/en/whitepaper

[16] R. Hanson, "Combinatorial information market design," *Inf. Syst. Frontiers*, vol. 5, no. 1, pp. 107–119, 2003. [Online]. Available: https://doi.org/10.1023/A:1022058209073

[17] ——, "Logarithmic market scoring rules for modular combinatorial information aggregation," *Journal of Prediction Markets*, vol. 1, no. 1, pp. 3–15, 2007. [Online]. Available: https://EconPapers.repec.org/RePEc:buc:jpredm:v:1:y:2007:i:1:p:3-15

[18] A. Othman and T. Sandholm, "Automated market-making in the large: The gates hillman prediction market," in *Proceedings of the 11th ACM Conference on Electronic Commerce*, ser. EC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 367–376. [Online]. Available: https://doi.org/10.1145/1807342.1807401

[19] Y. Chen and D. M. Pennock, "A utility framework for bounded-loss market makers," in *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, ser. UAI'07. Arlington, Virginia, USA: AUAI Press, 2007, p. 49–56.

[20] A. Othman, D. M. Pennock, D. M. Reeves, and T. Sandholm, "A practical liquidity-sensitive automated market maker," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, sep 2013. [Online]. Available: https://doi.org/10.1145/2509413.2509414

[21] Y. Chen, L. Fortnow, N. Lambert, D. M. Pennock, and J. Wortman, "Complexity of combinatorial market makers," in *Proceedings of the 9th ACM Conference on Electronic Commerce*, ser. EC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 190–199. [Online]. Available: https://doi.org/10.1145/1386790.1386822

[22] J. Abernethy, Y. Chen, and J. Wortman Vaughan, "An optimization-based framework for automated market-making," in *Proceedings of the 12th ACM Conference on Electronic Commerce*, ser. EC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 297–306. [Online]. Available: https://doi.org/10.1145/1993574.1993621

[23] J. Peterson and J. Krug, "Augur: a decentralized, open-source platform for prediction markets," *CoRR*, vol. abs/1501.01042, 2015. [Online]. Available: http://arxiv.org/abs/1501.01042

[24] B. Network, 2017. [Online]. Available: https://bancor.network/

[25] E. Hertzog, G. Benartzi, and G. Benartzi, "Bancor protocol: Continuous liquidity and asynchronous price discovery for tokens through their smart contracts; aka "smart tokens"," 2017. [Online]. Available: https://whitepaper.io/document/52/bancor-whitepaper/

[26] M. Egorov, "Stableswap: A non-custodial portfolio manager, liquidity provider, and price sensor," 2019. [Online]. Available: https://berkeley-defi.github.io/assets/material/StableSwap.pdf

[27] B. Finance, 2019. [Online]. Available: https://balancer.fi/

[28] M. Egorov, "Automatic market-making with dynamic peg," 2021. [Online]. Available: https://resources.curve.fi/pdf/crypto-pools-paper.pdf

[29] M. Wu and W. McTighe, "Constant Power Root Market Makers," *arXiv e-prints*, p. arXiv:2205.07452, May 2022.

[30] Y. Wang, "Automated market makers for decentralized finance (defi)," 2020. [Online]. Available: https://arxiv.org/pdf/2009.01676.pdf

[31] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," 2021. [Online]. Available: https://whitepaper.io/document/52/bancor-whitepaper/

[32] "Trader joe v2.1 - liquidity book docs," 2022. [Online]. Available: https://docs.traderjoexyz.com/

[33] M. Goyal, G. Ramseyer, A. Goel, and D. Mazieres, "Finding the right curve: Optimal design of constant function market makers," in *Proceedings of the 24th*

ACM Conference on Economics and Computation, ser. EC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 783–812. [Online]. Available: https://doi.org/10.1145/3580507.3597688

[34] J. Milionis, C. C. Moallemi, and T. Roughgarden, "A myersonian framework for optimal liquidity provision in automated market makers," in *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, ser. LIPIcs, V. Guruswami, Ed., vol. 287. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 81:1–81:19. [Online]. Available: https://doi.org/10.4230/LIPIcs.ITCS.2024.81

[35] X. D. He, C. Yang, and Y. Zhou, "Optimal design of automated market makers on decentralized exchanges," 2024. [Online]. Available: https://arxiv.org/abs/2404.13291

[36] A. Park, "Conceptual flaws of decentralized automated market making," 2023. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3805750

[37] J. Milionis, C. C. Moallemi, and T. Roughgarden, "Complexity-approximation trade-offs in exchange mechanisms: Amms vs. lobs," in *Financial Cryptography and Data Security: 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1–5, 2023, Revised Selected Papers, Part I.* Berlin, Heidelberg: Springer-Verlag, 2023, p. 326–343. [Online]. Available: https://doi.org/10.1007/978-3-031-47754-6_19

[38] G. Angeris, A. Agrawal, A. Evans, T. Chitra, and S. Boyd, *Handbook on Blockchain*. Springer, 2022, ch. Constant Function Market Makers: Multi-asset Trades via Convex Optimization, pp. 415–444.

[39] H. Adams, N. Zinsmeister, and D. Robinson, "Uniswap v2 core," 2020. [Online]. Available: https://uniswap.org/whitepaper.pdf

[40] F. Martinelli and N. Mushegian, "Stableswap - efficient mechanism for stablecoin liquidity," 2019. [Online]. Available: https://balancer.fi/whitepaper.pdf

[41] Spectra Finance. [Online]. Available: https://www.spectra.finance/

[42] V. Nguyen, "A study on amm for trading fixed yield and pendle's v2 principal token amm," October 2022. [Online]. Available: https://github.com/pendle-finance/pendle-v2-resources/blob/main/whitepapers/V2_AMM.pdf

[43] J. C. Cox, J. E. Ingersoll, and S. A. Ross, "A theory of the term structure of interest rates," *Econometrica*, vol. 53, no. 2, pp. 385–407, 1985. [Online]. Available: http://www.jstor.org/stable/1911242

[44] O. Vasicek, "An equilibrium characterization of the term structure," *Journal of Financial Economics*, vol. 5, no. 2, pp. 177–188, 1977. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304405X77900162

[45] A. Capponi, G. Iyengar, and J. Sethuraman, "Decentralized finance: Protocols, risks, and governance," *Found. Trends Priv. Secur.*, vol. 5, no. 3, pp. 144–188, 2023. [Online]. Available: https://doi.org/10.1561/3300000036