The Pilgrim Arbitrary Rounds Trading Mechanism

Computing the Continuous Discretely

Changwan Park, Tei Im, Daniel Hong, Taem Park and Dongjin Jung

Test in Prod DAO LLC

1 Introduction

Previous iterations of the Pilgrim trading protocol have allowed for an arbitrary number of round units to be traded. This resulted in frontrunning issues, where consecutively executing two different buy transactions of size n and m and a single sell transaction of size $n+m$ would result in draining liquidity deposited by other users within the same pool.

To mitigate this issue, we will only allow for a fixed number of round units to be traded at a time. Let this number be k rounds. On both buy and sell transactions, only trades that are multiples of k may be made, of which a trade of size $n \cdot k$ is equivalent to executing n times a single trade of size k.

However, it is trivial that executing n transactions of size k is computationally inefficient. The paper introduces a novel method of optimally executing Pilgrim trades, both buys and sells, of size $n \cdot k$ without being required to execute n Pilgrim transactions internally.

In addition, we propose an algorithm for NFT buyouts when there are minted rounds in existence.

We then show that the market trading price of rounds – i.e. the number of base tokens required for minting a single round unit – exponentially increase under continuous buy pressure.

Finally, we prove that the novel method we proposed to generalize Pilgrim trades of size $n \cdot k$ is able to be computed without rounding errors on fixed-point arithmetic.

2 Dodging Leakage: Limiting trade size to k rounds

Under the standard Pilgrim trading mechanism, let x be the base number of rounds, which remains the same after trading. Let y be the number of base tokens locked in a Pilgrim pool. Let $P = (x, y)$. When trade transactions are executed with an arbitrarily given sequence, we may consider P as Cartesian coordinates on a 2-dimentional plane. After a single Pilgrim transaction is executed, this point P would move around this plane, even though its exact path is completely unpredictable. However, by limiting individual transaction sizes to a constant, fixed amount – let this be k rounds – we can be confident that point P will move along a predictable path. This added constraint blocks undesired user profits and pool liquidity leaks.

2.1 Minting k rounds

0. Initial condition

$$P = (x, y)$$

1. Mint k rounds and add *virtual* base token liquidity with the same pool liquidity ratio. Let this *virtual* number of base tokens added be $h_M(y)$.

$$P = \left(x + k, \frac{x+k}{x} y\right)$$

$$h_M(y) = \frac{x+k}{x} y - y = \frac{k}{x} y$$

2. Swap *virtual* base tokens to newly minted rounds, under the standard $x \cdot y = c$ Uniswap AMM curve.

$$P = \left(x, \frac{(x+k)^2}{x^2} y\right)$$

We may now calculate the number of *actual* base tokens to be deposited by the trader. Let this number be $f_M(y)$.

$$f_M(y) = \frac{(x+k)^2}{x^2} y - \frac{x+k}{x} y = \frac{kx + k^2}{x^2} y$$

3. The protocol finally sends k rounds back to the trader. Let the updated number of base tokens locked with this Pilgrim pool be $g_M(y)$:

$$g_M(y) = \frac{(x+k)^2}{x^2} y$$

4. The relationship between $h_M(y)$, $f_M(y)$, and $g_M(y)$ may be expressed as follows:

$$g_M(y) = y + h_M(y) + f_M(y)$$

2.2 Burning k `rounds`

0. Initial condition

$$P = (x, y)$$

1. Swap rounds provided with *virtual* base tokens, under the standard $x \cdot y = c$ Uniswap AMM curve.

$$P = \left(x + k, \frac{x}{x+k} y\right)$$

We may now calculate the number of *actual* base tokens to be withdrawn to the trader. Let this number be $f_B(y)$.

$$f_B(y) = y - \frac{x}{x+k} y = \frac{k}{x+k} y$$

2. Burn k rounds and remove *virtual* base token liquidity with the same pool liquidity ratio. Let this *virtual* number of base tokens removed be $h_B(y)$.

$$P = \left(x, \frac{x^2}{(x+k)^2} y\right)$$

$$h_B(y) = \frac{x}{x+k}y - \frac{x^2}{(x+k)^2}y = \frac{xk}{(x+k)^2}y$$

3. The protocol finally burns k rounds received from the trader, and returns $f_B(y)$ units of base tokens. Let the updated number of base tokens locked with this Pilgrim pool be $g_B(y)$:

$$g_B(y) = y\frac{x^2}{(x+k)^2}$$

4. The relationship between $h_B(y)$, $f_B(y)$, and $g_B(y)$ may be expressed as follows:

$$g_B(y) = y - f_B(y) - h_B(y)$$

3 Optimal Method for trading $n \cdot k$ rounds

The base number of rounds locked with a Pilgrim pool always equal x. Therefore, we only need to calculate the number of base tokens being either deposited or withdrawn. Let n be a positive integer.

3.1 Minting $n \cdot k$ rounds

A transaction for buying $n \cdot k$ rounds should be divided to n transactions, each minting k `rounds`. Let $g_{nM}(y)$ be the resulting number of base tokens locked with this Pilgrim pool – i.e. base liquidity.

$$g_{nM} = \overbrace{g_M \circ \cdots \circ g_M}^{n} = \left(\frac{(x+k)^2}{x^2}\right)^n y$$

Let $f_{nM}(y)$ be the resulting number of base tokens to be deposited by the user after this transaction.

$$f_{nM}(y) = f_M(y) + f_M(g_M(y))) + f_M(g_{2M}(y)) + \cdots + f_M(g_{(n-1)M}(y))$$

$$= \sum_{i=0}^{n-1} f_M(g_{iM}(y)) = \sum_{i=0}^{n-1} \frac{kx+k^2}{x^2} g_{iM}(y) = \frac{kx+k^2}{x^2} y \sum_{i=0}^{n-1}\left(\frac{(x+k)^2}{x^2}\right)^i$$

$$f_{nM}(y) = \frac{kx+k^2}{x^2} y \frac{1-\left(\frac{(x+k)^2}{x^2}\right)^n}{1-\frac{(x+k)^2}{x^2}} = \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} y$$

Let $h_{nM}(y)$ be the number of base tokens to be *virtually* minted.

$$h_{nM}(y) = g_{nM}(y) - f_{nM}(y) - y = \left(\frac{(x+k)^2}{x^2}\right)^n y - \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} y - y$$

$$= \frac{(2x+k)(x+k)^{2n} - (x+k)^{2n+1} + x^{2n}(x+k)}{(2x+k)x^{2n}} y - y = \frac{x(x+k)^{2n} + x^{2n}(x+k)}{(2x+k)x^{2n}} y - y$$

$$= \frac{-(2x+k)x^{2n} + x(x+k)^{2n} + x^{2n}(x+k)}{(2x+k)x^{2n}} y = \frac{x((x+k)^{2n} - x^{2n})}{(2x+k)x^{2n}} y$$

$$h_{nM}(y) = \frac{x}{2x+k}\left(\left(\frac{x+k}{x}\right)^{2n} - 1\right) y$$

3.2 Burning $n \cdot k$ rounds

A transaction for selling $n \cdot k$ rounds should be divided to n transactions, each burning k `rounds`. Let $g_{nB}(y)$ be the resulting number of base tokens locked with this Pilgrim pool – i.e. base liquidity.

$$g_{nB} = \overbrace{g_B \circ \cdots \circ g_B}^{n} = \left(\frac{x^2}{(x+k)^2}\right)^n y$$

Let $f_{nB}(y)$ be the resulting number of base tokens to be withdrawn by the user after this transaction.

$$f_{nB}(y) = f_B(y) + f_B(g_B(y))) + f_B(g_{2B}(y)) + \cdots + f_B(g_{(n-1)B}(y))$$

$$= \sum_{i=0}^{n-1} f_B(g_{iB}(y)) = \sum_{i=0}^{n-1} \frac{k}{x+k} g_{iB}(y) = \frac{k}{x+k} y \sum_{i=0}^{n-1} \left(\frac{x^2}{(x+k)^2}\right)^i$$

$$f_{nB}(y) = \frac{k}{x+k} y \frac{1 - \left(\frac{x^2}{(x+k)^2}\right)^n}{1 - \frac{x^2}{(x+k)^2}} = \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} y$$

Let $h_{nB}(y)$ be the number of base tokens to be *virtually* burned.

$$h_{nB}(y) = y - f_{nB}(y) - g_{nB}(y) = y - \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} y - \left(\frac{x^2}{(x+k)^2}\right)^n y$$

$$= y - \frac{(x+k)^{2n+1} - x^{2n}(x+k) + (2x+k)x^{2n}}{(2x+k)(x+k)^{2n}} y = y - \frac{(x+k)^{2n+1} + x^{2n+1}}{(2x+k)(x+k)^{2n}} y$$

$$= \frac{(2x+k)(x+k)^{2n} - (x+k)^{2n+1} - x^{2n+1}}{(2x+k)(x+k)^{2n}} y = \frac{x((x+k)^{2n} - x^{2n})}{(2x+k)(x+k)^{2n}} y$$

$$h_{nB}(y) = \frac{x}{2x+k} \left(1 - \left(\frac{x}{x+k}\right)^{2n}\right) y$$

4 Trading with N base tokens

This section describes the logic of which rounds are newly minted or burnt when an arbitrary number of base tokens $N \in \mathbb{N}$ is given, instead of in round units.

4.1 Minting with N base tokens

Let the round unit multiplier $m \in \mathbb{N}$ be the number of rounds minted with N base tokens, i.e. $m \cdot k$. Deriving m:

$$f_{nM}(y) = \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} y = \frac{kx + k^2}{x^2} y \frac{1 - \left(\frac{(x+k)^2}{x^2}\right)^n}{1 - \frac{(x+k)^2}{x^2}}$$

$$f_{nM}(y) = \frac{kx + k^2}{x^2} y \frac{1 - \left(\frac{(x+k)^2}{x^2}\right)^n}{1 - \frac{(x+k)^2}{x^2}} = \frac{kx + k^2}{x^2} y \frac{x^2}{(x+k)^2 - x^2} \left(\left(\frac{x+k}{x}\right)^{2n} - 1\right)$$

$$f_{nM}(y) = \frac{(x+k)y}{2x+k}\left(\left(\frac{x+k}{x}\right)^{2n} - 1\right)$$

$$\left(\frac{x+k}{x}\right)^{2n} = 1 + \frac{f_{nM}(y)(2x+k)}{(x+k)y}$$

Substituting $f_{nM}(y)$ as N,

$$n = \frac{1}{2}\frac{\log_2\left(1 + \frac{N(2x+k)}{(x+k)y}\right)}{\log_2\left(\frac{x+k}{x}\right)}$$

Therefore the protocol can mint $m \cdot k$ rounds at the minimum, where $m = \lfloor n \rfloor$.

4.2 Burning with N base tokens

Let the round unit multiplier $m \in \mathbb{N}$ be the number of rounds burnt with N base tokens being returned, i.e. $m \cdot k$. Deriving m:

$$f_{nB}(y) = \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}}y = \frac{k}{x+k}y\frac{1-\left(\frac{x^2}{(x+k)^2}\right)^n}{1-\frac{x^2}{(x+k)^2}}$$

$$f_{nB}(y) = \frac{k}{x+k}y\frac{1-\left(\frac{x^2}{(x+k)^2}\right)^n}{1-\frac{x^2}{(x+k)^2}} = \frac{k}{x+k}y\frac{(x+k)^2}{(x+k)^2-x^2}\left(1-\left(\frac{x}{x+k}\right)^{2n}\right)$$

$$f_{nB}(y) = \frac{(x+k)y}{2x+k}\left(1-\left(\frac{x}{x+k}\right)^{2n}\right)$$

$$\left(\frac{x}{x+k}\right)^{2n} = 1 - \frac{f_{nB}(y)(2x+k)}{(x+k)y}$$

Substituting $f_{nB}(y)$ as N,

$$n = \frac{1}{2}\frac{\log_2\left(1 - \frac{N(2x+k)}{(x+k)y}\right)}{\log_2\left(\frac{x}{x+k}\right)}$$

Therefore the protocol can burn $m \cdot k$ rounds at the minimum, where $m = \lfloor n \rfloor$.

5 Contract Loss Analysis

We prove that the total number of base tokens deposited from users must be always greater or equal to the total amount of base tokens withdrawn from users within the same Pilgrim pool. As blockchain-based systems always process transactions linearly, we ignore concurrency, i.e. the total number of rounds locked within a Pilgrim pool must stay the same during a transaction.

Definition 1. *User Action A*: Any arbitrary round trader may perform either one of the following two actions: mints, or burns.

$$A := \begin{cases} \mathrm{M}_{n\cdot k} := \text{Mint } n \cdot k \texttt{ rounds}. \ \forall n \in \mathbb{N} \\ \mathrm{B}_{m\cdot k} := \text{Burn } m \cdot k \texttt{ rounds}. \ \forall m \in \mathbb{N} \end{cases} \tag{1}$$

Denote $M_k = M$, $B_k = B$ for brevity.

Definition 2. *Strategy S*: The finite sequence of ith *user action* A_i.

$$S = \{A_i\}_{i=0}$$

Definition 3. *Net token delta* $G(y, S)$: Total base token balance delta of a user before and after following strategy S, while assuming no other operations have taken place. Let y be the initial amount of base tokens locked with a Pilgrim pool.

Definition 4. *Net round delta* $R(y, S)$: Total round balance delta of a user before and after following strategy S, while assuming no other operations have taken place. Let y be the initial amount of base tokens locked with a Pilgrim pool.

Definition 5. *Strategy Equivalence*: If $G(y, S_1) = G(y, S_2)$, we define that S_1 and S_2 are *equivalent* and express the relation as $S_1 \iff S_2$.

Definition 6. *Contract loss* T: $T := \{S_L | G(y, S_L) > 0\}$

Lemma 1. *Minting Degeneracy:* $\forall m, n \in \mathbb{N}, \quad \{M_{m\cdot k}, M_{n\cdot k}\} \iff \{M_{(m+n)\cdot k}\}$

Proof. We show that $f_{mM}(y) + f_{nM}(g_{mM}(y)) = f_{(m+n)M}(y)$.

$$f_{nM}(g_{mM}(y)) = \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} g_{mM}(y)$$

$$= \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} \left(\frac{(x+k)^2}{x^2}\right)^m y = \frac{(x+k)^{2n+2m+1} - x^{2n}(x+k)^{2m+1}}{(2x+k)x^{2n+2m}} y$$

$$f_{mM}(y) + f_{nM}(g_{mM}(y))$$

$$= \frac{(x+k)^{2m+1} - x^{2m}(x+k)}{(2x+k)x^{2m}} y + \frac{(x+k)^{2n+2m+1} - x^{2n}(x+k)^{2m+1}}{(2x+k)x^{2n+2m}} y$$

$$= \frac{x^{2n}(x+k)^{2m+1} - x^{2n+2m}(x+k) + (x+k)^{2n+2m+1} - x^{2n}(x+k)^{2m+1}}{(2x+k)x^{2n+2m}} y$$

$$= \frac{(x+k)^{2(m+n)+1} - x^{2(m+n)}(x+k)}{(2x+k)x^{2(m+n)}} y = f_{(m+n)M}(y)$$

□

Corollary 1. $\forall n \in \mathbb{N}, \quad \{M_{n\cdot k}\} \iff \{\overbrace{M, \cdots, M}^{n}\}$

Proof. Apply mathematical induction to **Lemma 1**.

□

Lemma 2. *Burning Degeneracy:* $\forall m, n \in \mathbb{N}, \quad \{B_{m\cdot k}, B_{n\cdot k}\} \iff \{B_{(m+n)\cdot k}\}$

Proof. We show that $f_{mB}(y) + f_{nB}(g_{mB}(y)) = f_{(m+n)B}(y)$.

$$f_{nB}(g_{mB}(y)) = \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} g_{mB}(y)$$

$$= \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} \left(\frac{x^2}{(x+k)^2}\right)^m y = \frac{(x+k)^{2n}x^{2m} - x^{2n+2m}}{(2x+k)(x+k)^{2n+2m-1}} y$$

$$f_{mB}(y) + f_{nB}(g_{mB}(y))$$

$$= \frac{(x+k)^{2m} - x^{2m}}{(2x+k)(x+k)^{2m-1}} + \frac{(x+k)^{2n}x^{2m} - x^{2n+2m}}{(2x+k)(x+k)^{2n+2m-1}}$$

$$= \frac{(x+k)^{2m+2n} - x^{2m}(x+k)^{2n} + (x+k)^{2n}x^{2m} - x^{2n+2m}}{(2x+k)(x+k)^{2n+2m-1}}$$

$$= \frac{(x+k)^{2(m+n)} - x^{2(m+n)}}{(2x+k)(x+k)^{2(m+n)-1}} = f_{(m+n)B}(y)$$

□

Corollary 2. $\forall n \in \mathbb{N}, \quad \{B_{n\cdot k}\} \iff \{\overbrace{B, \cdots, B}^{n}\}$

Proof. Apply mathematical induction to **Lemma 2**.

□

Lemma 3. *Minting Inversion:* $\forall n \in \mathbb{N}, \quad \{M_{n\cdot k}, B_{n\cdot k}\} \iff \{\}$

Proof. We show that $f_{nM}(y) = f_{nB}(g_{nM}(y))$.

$$f_{nB}(g_{nM}(y)) = \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} g_{nM}(y) = \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} \left(\frac{(x+k)^2}{x^2}\right)^n y$$

$$= \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} y = f_{nM}(y)$$

□

Lemma 4. *Burning Inversion:* $\forall n \in \mathbb{N}, \quad \{B_{n\cdot k}, M_{n\cdot k}\} \iff \{\}$

Proof. We show that $f_{nB}(y) = f_{nM}(g_{nB}(y))$.

$$f_{nM}(g_{nB}(y)) = \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} g_{nB}(y) = \frac{(x+k)^{2n+1} - x^{2n}(x+k)}{(2x+k)x^{2n}} \left(\frac{x^2}{(x+k)^2}\right)^n y$$

$$= \frac{(x+k)^{2n} - x^{2n}}{(2x+k)(x+k)^{2n-1}} y = f_{nB}(y)$$

□

Lemma 5. *Round Recovery:* $\forall S, R(y, S) \geq 0.$

Proof. The total number of rounds burned cannot exceed the total amount of rounds minted.

□

Theorem 1. *Contract Loss Nullity:* $T = \emptyset$

Proof. Assume that S_L exists which satisfies $G(y, S_L) > 0$.

$$S_L = \{A_i\}_{i=0}^{t} = \{A_0, \cdots, A_t\}, \quad t \in \mathbb{N}$$

By **Corollary 1** and **Corollary 2**, all A_is may be exploded by substituting with sequences only containing M or B. Let *equivalent scenario* be S_D. S_D only includes *user actions* M or B.

$$S_L \iff S_D = \{A_{i,d}\}_{i=0}^{t_d} = \{A_{0,d}, \cdots, A_{t_d,d}\}, \quad t_d \in \mathbb{N}, \quad A_{i,d} \in \{M, B\}$$

By **Lemma 3** and **Lemma 4**, all neighboring $A_{i,d}$s with different *user action* types may be inverted. By *inversion* lemma, *equivalent scenario* S_I is in one of the three forms $S_{I,M}$, $S_{I,B}$, and $S_{I,}$. By reapplying **Corollary 1** and **Corollary 2**, we finally get

$$S_D \iff S_I = \begin{cases} S_{I,M} = \{\overbrace{M, \cdots, M}^{n}\} = \{M_{n\cdot k}\}, \forall n \in \mathbb{N} \\ S_{I,B} = \{\overbrace{B, \cdots, B}^{m}\} = \{B_{m\cdot k}\}, \forall m \in \mathbb{N} \\ S_{I,0} = \{\} \end{cases} \tag{2}$$

Case 1: $S_{I,M} = \{M_{n\cdot k}\}$: The user minted $n \cdot k$ rounds: $R(y, S_{I,M}) > 0$. The user must deposit base tokens for minting, thus $G(y, S_{I,M}) < 0$. Therefore $S_{I,M} \notin T$.

Case 2: $S_{I,B} = \{B_{m\cdot k}\}$: The user burned $m \cdot k$ rounds: $R(y, S_{I,M}) < 0$. The user must withdraw base tokens for burning, thus $G(y, S_{I,M}) > 0$. **Proof by contradiction**: By **Lemma 5**, the user cannot burn $n \cdot k$ rounds because the user never minted. **Case 2** never exists. $\Rightarrow\Leftarrow$

Case 3: $S_{I,0} = \{\}$. The total number of rounds minted and burned are equal, thus $G(y, S_{I,M}) = 0$. Therefore $S_{I,0} \notin T$.

$$\therefore \forall S, \nexists G(y, S) > 0 \iff T = \emptyset$$

□

6 NFT Buyout Mechanism

A protocol user U_T may want to claim ownership of the NFT locked with a Pilgrim pool. After the transfer, all corresponding rounds should be burnt back to its base token at a fixed value; metaNFT owners and round holders must be able to liquidate their positions. There may be $n \cdot k$ rounds minted from this NFT by a set of users U_M, with $g_{nM}(y)$ base tokens accounted as liquidity for this Pilgrim pool contract (i.e. both *virtual* and *actual* liquidity). Let U_O be the original NFT owner. Let α be the total locked number of base tokens with this Pilgrim pool (i.e. *actual* liquidity only).

6.1 NFT Buyout

Let X be the total number of rounds, which equals the sum of initial round liquidity invariant x and the total number of minted rounds $n \cdot k$.

Definition 7. *price* C_n: The number of base tokens required for minting a single round unit when $n \cdot k$ total rounds are minted for this pool, or the ratio between $g_{nM}(y)$ and initial round liquidity invariant x.

The number of base tokens E_T that U_T is required to pay back to this Pilgrim pool for a full buyout may be calculated as follows:

$$E_T = X \times C_n = (x + n \cdot k) \times \frac{g_{nM}(y)}{x} = \frac{x + n \cdot k}{x} g_{nM}(y)$$

6.2 Round Holder Liquidations

The total set of round holders U_M owns $n \cdot k$ rounds. On a full pool buyout, all existing round holders must be able to liquidate their positions to E_M base tokens, which may be calculated as follows:

$$E_M = n \cdot k \times C_n = n \cdot k \times \frac{g_{nM}(y)}{x} = \frac{n \cdot k}{x} g_{nM}(y)$$

6.3 NFT Holder Liquidations

The original NFT holder U_O must also be able to liquidate its position back to E_O units of base tokens.

$$E_O = E_T - E_M + \alpha = g_{nM}(y) + \alpha$$

6.4 metaNFT Buyout

The metaNFT owner or an approved user may pay E_B to U_O to reclaim NFT ownership locked with the PilgrimPair contract. As the liquidator must pay the number of base tokens the original NFT holder should receive in case of a full buyout, $E_B = E_O$.

$$E_B = E_O$$

7 Exponentiality of Pilgrim Round Valuation

Let X be the total number of rounds, which equals the sum of initial round liquidity invariant x and the total number of minted rounds $n \cdot k$. Let the **Pilgrim value function** $P(X) = Y = C_n$. We now prove that $P(X)$ is an exponential function.

Theorem 2. *$P(X) = Y$ is an exponential function.*

Proof.

$$P(X) = \frac{g_{nM}(y)}{x} = \left(\frac{(x+k)^2}{x^2}\right)^n \frac{y}{x}$$

$$X = x + n \cdot k \iff n = \frac{X - x}{k}$$

Substitute n,

$$P(X) = \left(\frac{(x+k)^2}{x^2}\right)^{\frac{X-x}{k}} \frac{y}{x} = \left(\frac{(x+k)^2}{x^2}\right)^{-\frac{x}{k}} \frac{y}{x} \left(\frac{(x+k)^2}{x^2}\right)^{\frac{X}{k}}$$

Define functions A, B:

$$A(x,y,k) = \left(\frac{(x+k)^2}{x^2}\right)^{-\frac{x}{k}} \frac{y}{x}, \quad B(x,k) = \frac{(x+k)^2}{x^2}$$

Therefore by assuming that x, y, k are constants, we get $P(X)$:

$$P(X) = A \times B^{\frac{X}{k}}$$

which has the form of exponential function.

□

Let us observe the property of the derivative of $P(X)$:

$$\frac{dP(X)}{dX} = \frac{A(x,y,k) \ln B(x,k)}{k} B(x,k)^{\frac{x}{k}}$$

The derivative of $P(X)$ is heavily related to k. Therefore, one must initialize the value of k in a meticulous manner to improve valuation accuracy and capital efficiency of the Pilgrim valuation function $P(X)$.

8 Computation on Fixed Point Arithmetic

Blockchain runtimes only provide fixed-point arithmetic, which often results in rounding errors. In this section, we propose calculation methods that do not void **Theorem 1** under fixed-point arithmetic. Let $m \cdot k, m \in \mathbb{Z}^+$ be the total number of rounds minted, and let y be the number of base tokens locked with this Pilgrim pool. The contract must internally store the value of m. Let y_{init} be the initial number of base tokens accounted with this Pilgrim pool on initialization; note that y_{init} is considered as *virtual* liquidity here.

The mechanism should follow the following definitions in the exact order as given:

Definition 8. *Error function* $e(x)$: Let $e(x)$ be the erroneous result of equation x computed on fixed point arithmetic.

Definition 9. $\tilde{y} = \tilde{g}_{mM}(y_{init}) = e(g_{mM}(y_{init})) = e\left(\left(\frac{(x+k)^2}{x^2}\right)^m\right) y_{init}$

Definition 10. $\tilde{h}_{mM}(y_{init}) = e\left(\frac{x}{2x+k}(\tilde{g}_{mM}(y_{init}) - y_{init})\right)$

Definition 11. $\tilde{f}_{mM}(y_{init}) = \tilde{g}_{mM}(y_{init}) - y_{init} - \tilde{h}_{mM}(y_{init})$

The contract stores the value of $\tilde{g}_{mM}(y_{init})$, $\tilde{h}_{mM}(y_{init})$, $\tilde{f}_{mM}(y_{init})$.

8.1 Minting $n \cdot k$ rounds

The resulting amount of rounds minted is equivalent to $(m+n) \cdot k$.

Definition 12. $\tilde{g}_{nM}(\tilde{y}) = e(g_{nM}(y)) = \tilde{g}_{(n+m)M}(y_{init}) - \tilde{g}_{mM}(y_{init})$

Definition 13. $\tilde{h}_{nM}(\tilde{y}) = e(h_{nM}(y)) = \tilde{h}_{(n+m)M}(y_{init}) - \tilde{h}_{mM}(y_{init})$

By applying **Definition 10**, $\tilde{h}_{nM}(\tilde{y})$ may be rewritten as

$$\tilde{h}_{nM}(\tilde{y}) = e\left(\frac{x}{2x+k}(\tilde{g}_{(n+m)M}(y_{init}) - y_{init})\right) - \tilde{h}_{mM}(y_{init}).$$

Definition 14. $\tilde{f}_{nM}(\tilde{y}) = e(f_{nM}(y)) = \tilde{f}_{(n+m)M}(y_{init}) - \tilde{f}_{mM}(y_{init})$

By applying **Definition 11**, $\tilde{f}_{nM}(\tilde{y})$ may be rewritten as

$$\tilde{f}_{nM}(\tilde{y}) = \tilde{g}_{(n+m)M}(y_{init}) - y_{init} - \tilde{h}_{(n+m)M}(y_{init}) - \tilde{f}_{mM}(y_{init}).$$

All the intermediate values required to calculate $\tilde{g}_{nM}(\tilde{y})$, $\tilde{f}_{nM}(\tilde{y})$, and $\tilde{h}_{nM}(\tilde{y})$ have the form of $\tilde{g}_{kM}(y_{init})$, or stored with the contract itself; $\tilde{g}_{mM}(y_{init})$, $\tilde{h}_{mM}(y_{init})$, and $\tilde{f}_{mM}(y_{init})$.

Here, we always get consistent values already affected by fixed point rounding. This is because we always initiate calculation based on y_{init}, not y. Therefore, despite being on fixed-point arithmetic, we get consistent results for $\tilde{g}_{nM}(\tilde{y})$, $\tilde{f}_{nM}(\tilde{y})$, and $\tilde{h}_{nM}(\tilde{y})$.

When values $\tilde{g}_{nM}(\tilde{y})$, $\tilde{f}_{nM}(\tilde{y})$, and $\tilde{h}_{nM}(\tilde{y})$ are calculated, intermediate values $\tilde{g}_{(n+m)M}(y_{init})$, $\tilde{h}_{(n+m)M}(y_{init})$, and $\tilde{f}_{(n+m)M}(y_{init})$ are also calculated.

Pilgrim smart contract implementations should store these values for future calculation and remove values $\tilde{g}_{mM}(y_{init})$, $\tilde{h}_{mM}(y_{init})$, and $\tilde{f}_{mM}(y_{init})$.

8.2 Burning $n \cdot k$ rounds

The resulting amount of rounds minted is equivalent to $(m-n) \cdot k$.

Definition 15. $\tilde{g}_{nB}(\tilde{y}) = e(g_{nB}(y)) = \tilde{g}_{mM}(y_{init}) - \tilde{g}_{(m-n)M}(y_{init})$

Definition 16. $\tilde{h}_{nB}(\tilde{y}) = e(h_{nB}(y)) = \tilde{h}_{mM}(y_{init}) - \tilde{h}_{(m-n)M}(y_{init})$

Definition 17. $\tilde{f}_{nB}(\tilde{y}) = e(f_{nB}(y)) = \tilde{f}_{mM}(y_{init}) - \tilde{f}_{(m-n)M}(y_{init})$

Following the same logic as **8.1**, we always get consistent results for $\tilde{g}_{nB}(\tilde{y})$, $\tilde{h}_{nB}(\tilde{y})$, and $\tilde{f}_{nB}(\tilde{y})$, even on fixed-point arithmetic.

8.3 Contract Loss Analysis under Fixed Point Arithmetic

We show that **Theorem 1**: *Contract Loss Nullity* also holds on fixed point(*FP*) arithmetic. Let $l \cdot k$ be the amount of total rounds minted before calculation. Assume that the number of burned rounds is not greater than the total number of rounds minted.

Lemma 6. *Minting Degeneracy under FP:* $\forall m, n \in \mathbb{N}, \quad \{M_{m\cdot k}, M_{n\cdot k}\} \iff \{M_{(m+n)\cdot k}\}$

Proof. We show that $e(f_{mM}(y)) + e(f_{nM}(g_{mM}(y))) = e(f_{(m+n)M}(y))$.

By **Definition 13**,

$$e(f_{mM}(y)) = \tilde{f}_{(m+l)M}(y_{init}) - \tilde{f}_{lM}(y_{init})$$

$$e(f_{nM}(g_{mM}(y))) = \tilde{f}_{(m+n+l)M}(y_{init}) - \tilde{f}_{(m+l)M}(y_{init})$$

Therefore,

$$e(f_{mM}(y)) + e(f_{nM}(g_{mM}(y))) = \tilde{f}_{(m+n+l)M}(y_{init}) - \tilde{f}_{lM}(y_{init})$$

Following **Definition 13**,

$$\tilde{f}_{(m+n+l)M}(y_{init}) - \tilde{f}_{lM}(y_{init}) = e(f_{(m+n)M}(y))$$

□

Lemma 7. *Burning Degeneracy under FP:* $\forall m, n \in \mathbb{N}, \quad \{B_{m\cdot k}, B_{n\cdot k}\} \iff \{B_{(m+n)\cdot k}\}$

Proof. We show that $e(f_{mB}(y)) + e(f_{nB}(g_{mB}(y))) = e(f_{(m+n)B}(y))$.

By **Definition 17**,

$$e(f_{mB}(y)) = \tilde{f}_{lM}(y_{init}) - \tilde{f}_{(l-m)M}(y_{init})$$

$$e(f_{nB}(g_{mB}(y))) = \tilde{f}_{(l-m)B}(y_{init}) - \tilde{f}_{(l-m-n)B}(y_{init})$$

Therefore,

$$e(f_{mB}(y)) + e(f_{nB}(g_{mB}(y))) = \tilde{f}_{lB}(y_{init}) - \tilde{f}_{(l-m-n)B}(y_{init})$$

Following **Definition 17**,

$$\tilde{f}_{lB}(y_{init}) - \tilde{f}_{(l-m-n)B}(y_{init}) = e(f_{(m+n)B}(y))$$

□

Lemma 8. *Minting Inversion under FP:* $\forall n \in \mathbb{N}, \quad \{M_{n\cdot k}, B_{n\cdot k}\} \iff \{\}$

Proof. We show that $e(f_{nM}(y)) = e(f_{nB}(g_{nM}(y)))$.

By **Definition 13**,

$$e(f_{nM}(y)) = \tilde{f}_{(n+l)M}(y_{init}) - \tilde{f}_{lM}(y_{init})$$

By **Definition 17**,

$$e(f_{nB}(g_{nM}(y))) = \tilde{f}_{(n+l)M}(y_{init}) - \tilde{f}_{(n+l-n)M}(y_{init})$$

□

Lemma 9. *Burning Inversion under FP:* $\forall n \in \mathbb{N}, \quad \{B_{n\cdot k}, M_{n\cdot k}\} \iff \{\}$

Proof. We show that $e(f_{nB}(y)) = e(f_{nM}(g_{nB}(y)))$.

By **Definition 17**,

$$e(f_{nB}(y)) = \tilde{f}_{lM}(y_{init}) - \tilde{f}_{(l-n)M}(y_{init})$$

By **Definition 13**,

$$e(f_{nM}(g_{nB}(y))) = \tilde{f}_{(l-n+n)M}(y_{init}) - \tilde{f}_{(l-n)M}(y_{init})$$

□

Theorem 3. *Contract Loss Nullity under FP:* $T = \emptyset$

Proof. Equivalent to proof of **Theorem 1**.

Corollary 1 and **Corollary 2** also holds on fixed point arithmetic, simply using mathematical induction based on **Lemma 6** and **Lemma 7**. By applying **Lemma 8** and **Lemma 9** similar to **Theorem 1** with **Lemma 5**, we have proved **Contract Loss Nullity** on fixed point arithmetic.

□