Dynamic Liquidity Token Protocol: A Decentralized Approach to Tokenomics and Automated Market-Making

Abstract

This new standard operates on the principles of bonding curves and leverages the underlying logic from the Power and Bonding Curve contracts to facilitate seamless token transactions. By employing a continuous minting and burning mechanism, the contract allows users to buy and sell tokens directly with ETH, thus eliminating the need for traditional liquidity pools and intermediaries. The contract ensures that token prices are dynamically adjusted based on the reserve balance and total supply, promoting a fair valuation of tokens. Ultimately, this design aims to create a decentralized and gas efficient trading environment, fostering a sustainable ecosystem for token holders and contributors alike. This paper outlines the motivation and technical frameworks.

1 Introduction

One of the primary goals of this contract design is to eliminate the reliance on separate liquidity pool contracts, such as those used by decentralized exchanges like Uniswap. In conventional systems, liquidity providers must interact with both a token contract and an external liquidity pool, which not only increases the complexity of the interaction but also leads to higher transaction costs. Every action, such as providing liquidity, swapping tokens, or withdrawing liquidity, involves multiple steps, each requiring its own transaction, resulting in elevated gas fees.

This token contract integrates the liquidity mechanism directly into the contract itself. By doing so, it enables users to buy and sell tokens without interacting with an external pool. This drastically reduces the overhead involved in managing liquidity, as users no longer need to approve third-party contracts or perform multiple transactions to achieve a single outcome. Instead, all liquidity-related actions are handled within the token contract, making the process more streamlined and cost-efficient.

2 Technical Overview

2.1 Power Function Smart Contract for Token Bonding Curves

The Power smart contract leverages bonding curves to provide a dynamic and decentralized token pricing mechanism. Bonding curves are mathematical models that relate token price to supply, typically increasing the price as more tokens are minted. This mechanism is particularly relevant for projects seeking an automated market-making model, ensuring liquidity without needing traditional market makers.

2.2 Motivation

Current bonding curve implementations require precise mathematical functions to ensure stability, efficiency, and accuracy during token issuance or redemption. The objective of the Power contract is to:

- Ensure token price and supply are dynamically balanced based on demand.
- Enable predictable and deterministic pricing mechanisms through mathematical rigor.

2.3 Key Features

2.3.1 Fixed-Point Arithmetic

Smart contracts on Ethereum are limited in their ability to handle floating-point arithmetic. To overcome this limitation, the Power contract uses **fixed-point arithmetic**, specifically 128.128-bit representation (128 bits for integer and 128 bits for the fractional part).

- **`FIXED_1`** represents the base unit (1 in fixed-point terms) and is set to 2^127.
- **`FIXED_2`** is twice `FIXED_1`.
- **`MAX_NUM`** defines the maximum number that can be handled in the system, preventing overflow.

2.3.2 Logarithmic and Exponential Functions

Efficient calculation of logarithmic and exponential functions is critical to the bonding curve's pricing mechanism.

- **Natural Logarithm (ln)**: The natural logarithm is crucial for calculating the percentage change in token supply relative to price. To ensure efficient computation, constants **`LN2_NUMERATOR`** and **`LN2_DENOMINATOR`** approximate natural logarithmic base changes.
- **Exponential Function (exp)**: The contract also computes exponentials efficiently. This is necessary for adjusting prices based on supply. A significant portion of the contract is dedicated to ensuring the exponential functions are precise and efficient.
- **Optimal Log/Exp Approximations**: Due to the complexity of accurately calculating logarithmic and exponential values on-chain, **`optimal log`** and **`optimal exp`** approximations are precomputed for specific ranges. These precomputed values are stored in **`maxExpArray`**, allowing the contract to rapidly calculate powers and logarithms without expensive on-chain computations.

2.3.3 Precomputed Constants and Scaling Factors

To maximize performance and accuracy, the contract includes a set of precomputed constants. These constants help to quickly approximate logarithmic and exponential functions, which are otherwise computationally expensive:

- **`OPT_LOG_MAX_VAL`** and **`OPT_EXP_MAX_VAL`** are the maximum values for which logarithmic and exponential approximations remain efficient and accurate.
- **`maxExpArray[]`** stores values to facilitate fast computation of powers for different exponents, ensuring the contract remains efficient and low on gas fees.

2.4 Mathematical Implementation

The bonding curve relies heavily on **logarithmic growth** to model the relationship between the token supply and price. As more tokens are minted (i.e., more capital is added to the curve), the price grows logarithmically, ensuring that early token buyers receive a discount while late buyers pay progressively more.

The Power contract implements this relationship through an optimized **log-exp approximation**:

- **Logarithmic Growth**:

$$log(x) = \sum coefficients\ from\ precomputed\ arrays$$

  The contract uses precomputed scaling factors to make this computation feasible in Solidity's constrained environment.

- **Exponential Growth**:

$$exp(x) = \sum coefficients\ from\ precomputed\ arrays$$

  As with the logarithmic growth, precomputed arrays allow the contract to efficiently compute exponential values based on the token supply.

This mathematical rigor ensures that the Power contract can handle large numbers with minimal precision loss and avoid common pitfalls in Ethereum smart contracts, such as overflows.

2.5 Mathematical Framework of the Bonding Curve Contract

The Bonding Curve contract governs the pricing dynamics of a token through a mathematical relationship between supply and reserve balance. This whitepaper focuses on the mathematical model behind the bonding curve, illustrating how token purchases and sales are calculated based on various inputs like token supply, reserve balance, and reserve ratio. The contract employs a power function to represent the nonlinear nature of the curve, ensuring a predictable and continuous price gradient.

2.5.1 Key Parameters and Variables

- Token Supply (`_supply`): The total supply of the token available at a given time.

- Reserve Balance (`_reserveBalance`): The amount of reserve (e.g., ETH) held by the contract.

- Reserve Ratio (`_reserveRatio`): A percentage (out of 1,000,000) that defines the ratio of reserves to the total token market cap.

- Deposit Amount (`_depositAmount`): The amount of reserve deposited by a buyer in exchange for new tokens.

- Sell Amount (`_sellAmount`): The amount of tokens a user wishes to sell back to the contract for reserves.

2.6. Purchase Return Calculation

The formula for calculating the number of tokens issued in exchange for a deposit follows a bonding curve, with the following logic:

$$\text{NewTokenSupply} = \text{CurrentSupply} \times \left(\frac{\text{ReserveBalance} + \text{DepositAmount}}{\text{ReserveBalance}} \right)^{\frac{\text{ReserveRatio}}{1{,}000{,}000}}$$

The function calculatePurchaseReturn in the contract calculates the number of new tokens to be issued based on the deposit amount, reserve balance, token supply, and reserve ratio.

2.7 Key Steps:

- Special Case (Linear Bonding Curve): When the reserve ratio equals `MAX_RESERVE_RATIO` (1,000,000), the price is linear. The number of new tokens is directly proportional to the deposit amount:

$$\Delta\text{Tokens} = \frac{\text{Supply} \times \text{DepositAmount}}{\text{ReserveBalance}}$$

- Non-linear Case: For non-maximal reserve ratios, the formula employs an exponential function that is handled using the power function (power()), a crucial component for non-linear curves. This function computes the increase in token supply by raising the adjusted reserve-to-supply ratio to a power proportional to the reserve ratio.

The formula used is derived from integrating the price formula $(P(x))$ over the range of supply and incorporates the reserve ratio as a weighting factor.

2.8 Sale Return Calculation

For token sales, the function `calculateSaleReturn` computes how much reserve the seller will receive based on how many tokens they are selling. The number of reserve tokens returned is derived from the inverse function of the purchase return.

The formula is:

$$\text{NewReserveBalance} = \text{CurrentReserveBalance} \times \left(\frac{\text{CurrentSupply} - \text{SellAmount}}{\text{CurrentSupply}} \right)^{\frac{\text{ReserveRatio}}{1{,}000{,}000}}$$

The difference between the original and the new reserve balance gives the number of reserve tokens returned to the seller.

Key Steps:

- Special Case (Linear Bonding Curve): When the reserve ratio is maximal, the function becomes a simple linear proportion:

$$\Delta\text{Reserves} = \frac{\text{ReserveBalance} \times \text{SellAmount}}{\text{Supply}}$$

- Non-linear Case: For non-maximal reserve ratios, the function employs the inverse power function, adjusting the reserve balance by scaling down according to the proportion of tokens being sold.

2.9 Power Function

The power function used in both purchase and sale calculations approximates an exponential relationship between token supply and reserve balance. It takes into account the reserve ratio, which adjusts the curve's steepness, allowing for a flexible range of bonding curve shapes.

In both calculatePurchaseReturn and calculateSaleReturn, the power function is implemented as:

$$\text{Result} = \left(\frac{\text{BaseN}}{\text{BaseD}}\right)^{\frac{\text{Exponent}}{MAX\_RESERVE\_RATIO}}$$

where BaseN and BaseD are the reserve balances after a deposit or token sale, respectively. This expression is calculated with high precision and scaled appropriately to ensure stability across a wide range of values.

2.10 Reserve Ratio and Bonding Curve Shape

The reserve ratio plays a crucial role in determining the shape of the bonding curve:

- High Reserve Ratio (approaching 1,000,000): The bonding curve becomes nearly linear. Price per token remains relatively constant, increasing proportionally with token purchases or sales.

- Low Reserve Ratio: The bonding curve becomes highly non-linear, with token prices accelerating as more tokens are purchased. This creates a steep price increase when the token supply grows, discouraging large purchases and incentivizing early buyers.

2.11 Metadata

The token contract incorporates ERC-7572, which introduces the concept of contract-level metadata through the `contractURI()` function. This feature allows the developer to specify and update metadata related to the token contract easily. The metadata can include essential information such as the token's website, social networks, description, and other relevant details that can enhance the user's understanding and interaction with the token.

The metadata is structured as a stringified JSON, allowing for a flexible and extensible format where any relevant information can be included. This design enables the incorporation of various attributes, such as social media links, documentation references, and governance details, providing users with comprehensive insights into the token's functionality and ecosystem.

Authors of the ERC-7572 standard—Devin Finzer, Alex Atallah, and Ryan Ghods—recognized the importance of having a standardized approach to contract metadata in the Ethereum ecosystem. By utilizing this standard, the token contract ensures that essential information is readily available,

promoting transparency and fostering user trust. The ability to update the metadata dynamically further allows for ongoing communication of relevant information to users, ensuring that they have access to the most current and accurate data regarding the token and its ecosystem. This aspect of the contract enhances user experience and engagement, aligning with the goals of decentralization and transparency that underpin blockchain technology.

3 Conclusion

In conclusion, the innovative token contract harnesses the power of a bonding curve to create a streamlined and efficient liquidity model that eliminates the need for separate liquidity pool contracts, such as those used by Uniswap. By allowing the developer to maintain centralized control over liquidity, the contract not only reduces transaction costs but also simplifies the user experience in trading and interacting with the token. The dynamic pricing mechanism provides a fair and transparent method for token valuation, benefiting both buyers and sellers while fostering continuous trading activity.

4 Acknowledgements

The bonding curve and power contracts utilized in this idea are derived from the pioneering work of the Bancor Protocol. Their innovative approach to liquidity and decentralized trading has significantly influenced the development of this contract, providing a robust framework for managing token economics and liquidity dynamics.

“If I have seen further than others, it is by standing upon the shoulders of giants.”