Snek.fun Specifications

Product requirements

- **Functional Requirements**:
  - The product must allow users to connect, trade and create tokens with the following wallets:
    - Eternl
    - Nami
    - Vespr (including mobile)
  - The product must allow to create tokens with the following parameters:
    - Name
    - Ticker
    - Description
    - Token logo
  - The product must allow to buy and sell within the bonding curve of each token
- **Non-Functional Requirements**:
  - Page loading must be less than 3 seconds
  - The interface of the product must support both desktop and mobile versions
  - Order execution must not take longer than **one Cardano blockchain block**.
  - The migration from the bonding curve pool to the AMM DEX Splash pool must not take more that 5 minutes
  - The product should display the list of tokens available for trading on the home page
  - The product must display price of each token on a token page
  - The product must display order history of each token on a token page (including user orders if a wallet is connected)

    - The product must display all token holders on the token page
- **Data Requirements**:
    - Trading data and orders must be securely stored in the Postgres database

Technology

- **Smart Contracts:**
    - Language: Aiken
    - **Required Contracts:**
        - Bonding Curve Pool Contract
        - Order Contract
- **Off-chain system (batcher):**
    - Support Bonding Curve Pool Contract
    - Support Order Contract
- **Backend:**
    - Language: Scala
    - Analytics service:
        - List of tokens including filters by:
            - Market Cap
            - Last trade
            - Date
            - Trading volume
            - Bonding curve progress
        - Price charts by token
        - Order history by token
        - Token holder by token
        - Search tokens by:
            - Name
            - Ticker
            - Policy ID

- **Interface:**
  - Tech: React
  - **Pages:**
    - List of tokens page (home page)
    - Create token form page
    - Token page
    - User orders page

Bonding curve specification

We aim to construct a bonding curve suitable for memecoin launching, a functional analog of the pump.fun curve. The main functional requirements are:

1. The maximum profit of the first buyers of the token should be limited;
2. Adjust the curve parameters based on the specified token capitalization, which is achieved when the curve is saturated;
3. After curve saturation, all deposited funds and remaining tokens should be transferred into the AMM pool.

To satisfy these requirements, the curve we choose must have corresponding properties:

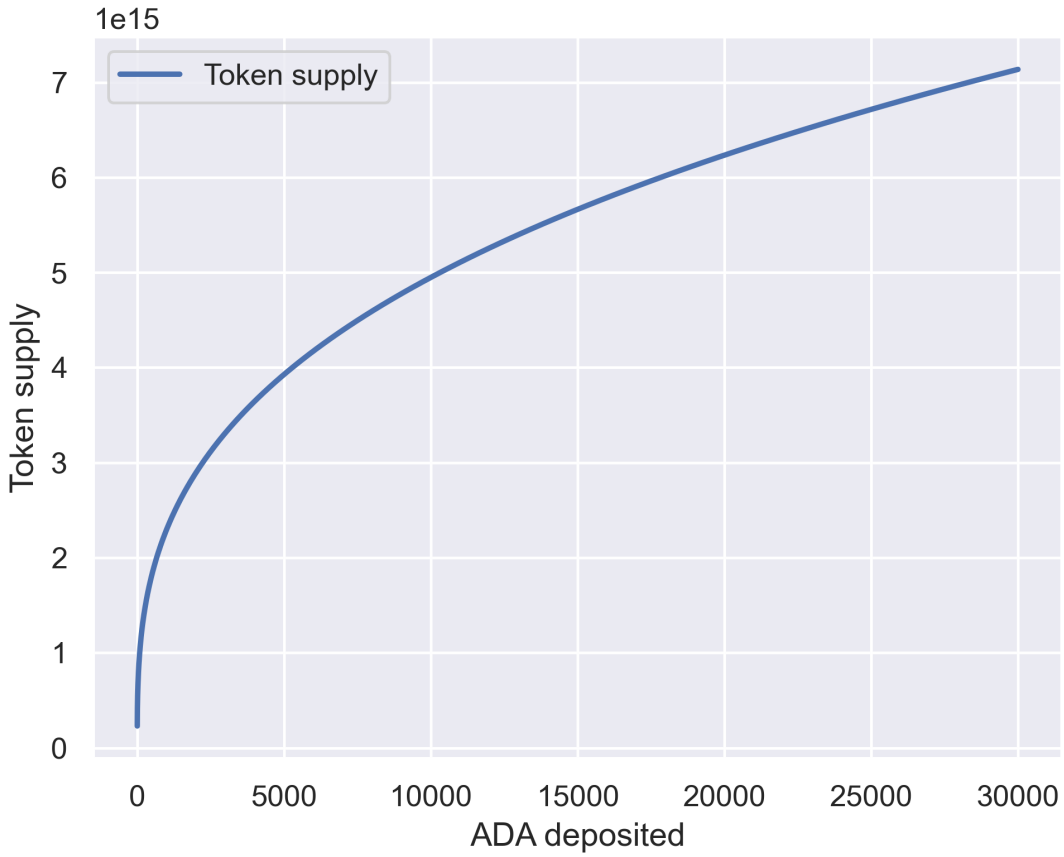
1. The Curve should allow us to adjust the initial price of the token being launched;
2. The curve should be adjustable to any final token capitalization at the saturation point;
3. The curve should have a saturation point, the price of the last token sale which corresponds to the initial price of the token in the created AMM pool

Using reverse engineering, it was found that the pump.fun curve is an exponential function, that meets the properties above, but is computationally difficult to validate on-chain. In a series of experiments, we selected a suitable curve that is easy to verify on-chain.

The original pump.fun project, according to reverse engineering, uses an exponential bonding curve, however, this requires calculating logarithms and exponents inside the smart contract. To facilitate calculations, the curve `p = a * y ^ 2 + b` was chosen as a bonding curve, since it is easier to calculate and has a sufficient growth rate to launch memcoins.

Example of target bonding curve for selling `92233720368547758O7` tokens:

Token supply changes according to:



Life cycle of the bonding curve is as follows:

1. The bonding-pool is initiated with the full emission `E` of a new token `Y` locked in the pool;
2. ADA is deposited into the pool / token `Y` is redeemed according to the `p = a * y ^ 2 + b` price curve;
3. When the bonding-pool reaches `thr_ADA_cup` through deposits, deposit/redeem actions are blocked;
4. All reserves `(bonding_pool.ADA, bonding_pool.Y)` from the bonding-pool are transferred to the newly created AMM-pool (the value of the curve parameter `a` is pre-selected so that at the saturation point `p == bonding_pool.ADA / bonding_pool.Y && bonding_pool.Y / E = 0.25`, i.e. AMM-pool is created with an asset ratio that corresponds to the last price in the bonding-pool, and the amount of `Y` token in the AMM pool is 25% of the total emission `E`).