nest

Whitepaper

NEST PVM with Infinite Liquidity and Complete Probability Space

nestprotocol.org

July 17, 2022

Abstract

Following the development trend of on-chain applications, liquidity has become the biggest problem for DeFi applications. The previous DeFi projects have tried to apply the traditional order book method and automated market maker models to solve this problem. However, these methods are not ideal solutions since they cannot integrate all financial services into a single protocol and share the same liquidity, resulting in resource wasting and low performance. This paper proposes a new protocol: NEST Probability Virtual Machine, which introduces the concept of discount computers and on-chain decentralized currency units, that can systematically solve the liquidity problem and integrate all financial contracts and derivatives into one protocol. It can be used for transferring all financial instruments, including tools for hedging and other economic relationship lock-ins, to on-chain applications.

Contents

1 Introduction

In the traditional financial market, i.e. CeFi (or "centralized finance"), financial intermediaries are responsible for maintaining the fairness of the trading; they are the deal makers who help the traders to find the matched counterparties. In other situations, they are also responsible for pricing financial products. The financial intermediaries include diverse exchanges and dealmakers of the loan and asset auction market. Together, they boost the efficiency of financial trading to the extreme. For instance, driven by high-frequency trading, the exchanges allow the frequencies of their order flow to arrive in nanoseconds. Although the traditional financial market has improved its efficiency, the credit risk within also increases. The inevitable hosting problem and the intervention of the financial intermediary will raise the implicit costs of financial transactions. These problems account for the main reason for developing DeFi (or "decentralized finance"). If Alice, a trader, wants to buy, sell some asset, or open a position in a centralized Financial market; she needs to go to a financial intermediary, centralized exchange in this case. After filling her required

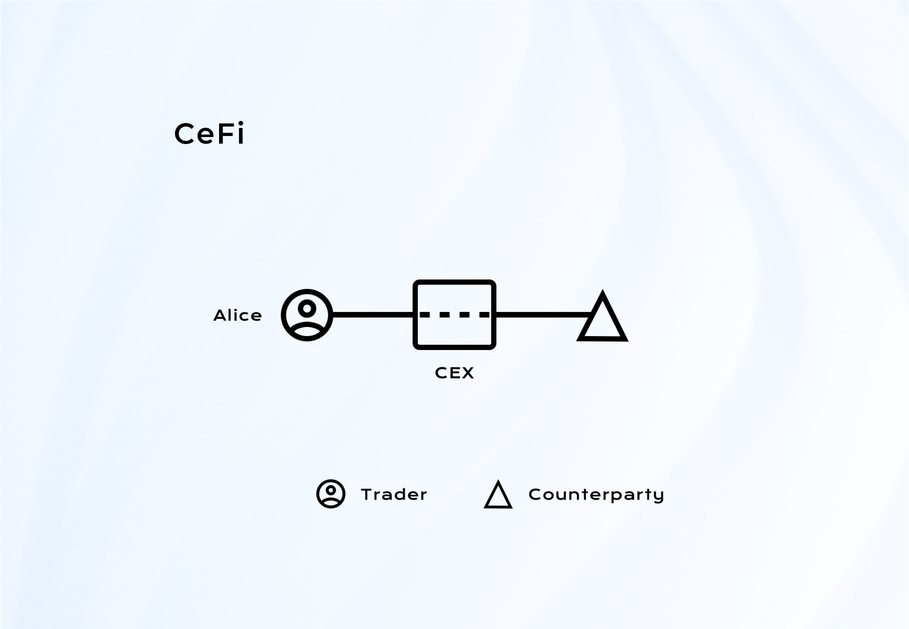

Figure 1: Diagram of CeFi

trading in an order table, the exchange will public her order and find counterparties for Alice. The diagram of this process is exhibited in Figure 1.

DeFi is an umbrella term for financial services on public blockchains. With DeFi, participants can do most things that banks support, earn interest, borrow, lend, buy insurance, trade derivatives, trade assets, and more, but it's faster and doesn't require paperwork or a third party. As with crypto generally, DeFi is global, peer-to-peer (meaning directly between two people, not routed through a centralized system), pseudonymous, and open to all.

The first generation of DeFi (DeFi 1.0) projects is based on the traditional approach of order books. For example, EtherDelta, the EtherDelta exchange, is a cryptocurrency trading platform based on the Ethereum blockchain and ERC20 standard tokens. It is worth noting that the exchange was established in Chicago in 2017. The platform is based on smart contracts, with the help of which all contracts are carried out inside the website (purchase, sale, withdrawal of funds from the wallet, etc.). Since every order has been written into a contract, waiting for counterparties, this approach is very inefficient and naturally creates a barrier to improving the market's liquidity. The problem of DeFi 1.0 includes

- the liquidity on the chain cannot be effectively ensured;
- finding the final matched counterparties still needed for centralized institutions;
- once a price deviates from the actual value, the gas fee cost of order adjustment is high.

This traditional financial approach is not appropriate for on-chain applications. We extend our story of Alice trying to trade with a DeFi 1.0 platform. She can participate in a smart contract that specifies her request and wait for counterparties to notice her order contract. The diagram of this process is exhibited in Figure 2.

The second generation of DeFi (DeFi 2.0) introduced automated market maker (AMM) model makes decentralized exchange (DEX, e.g., Uniswap, Sushiswap, etc.) possible and lending projects. DEXs have improved the liquidity of the DeFi con-

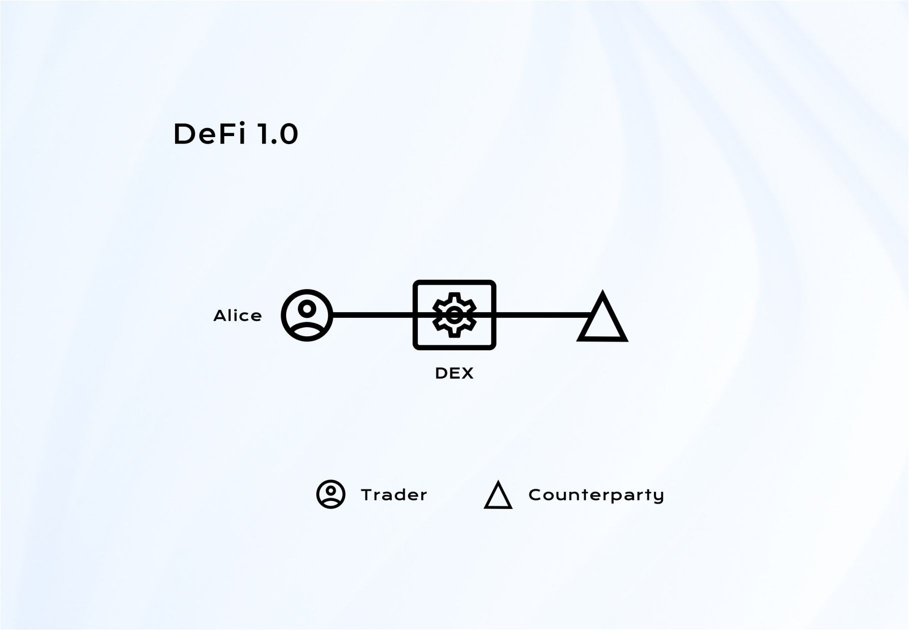

Figure 2: Diagram of DeFi 1.0

tracts and allow users to deposit tokens to earn trading fees. lending projects (e.g., Aave, Compound, etc.) provide the users a platform to earn from providing liquidity and to borrow. Projects like MakerDao vitalize the DeFi market by allowing users to generate a kind of decentralized stablecoin. This kind of stablecoin is available to anyone, anywhere; like Dai (generated by MakerDAO) is backed by a surplus of collateral that has been individually escrowed into audited and publicly viewable Ethereum smart contracts. DeFi 2.0 led to a rising wave in the crypto market in 2020 since DeFi 2.0 projects provided a better solution to the fundamental problem of the DeFi industry: lack of liquidity. With a DeFi 2.0 platform. Alice can directly trade assets with the asset pools deposited by liquidity providers by participating in a smart contract. The diagram of this process is exhibited in Figure 3.

However, whether it is AMM or liquidity pools, the solution to the liquidity problems is at the expense of the liquidity provider's flexibility: a liquidity provider needs to fix his trading strategy and bear the external market fluctuations. Once the price is favorable to the liquidity provider, the counterparty trader can opt to back

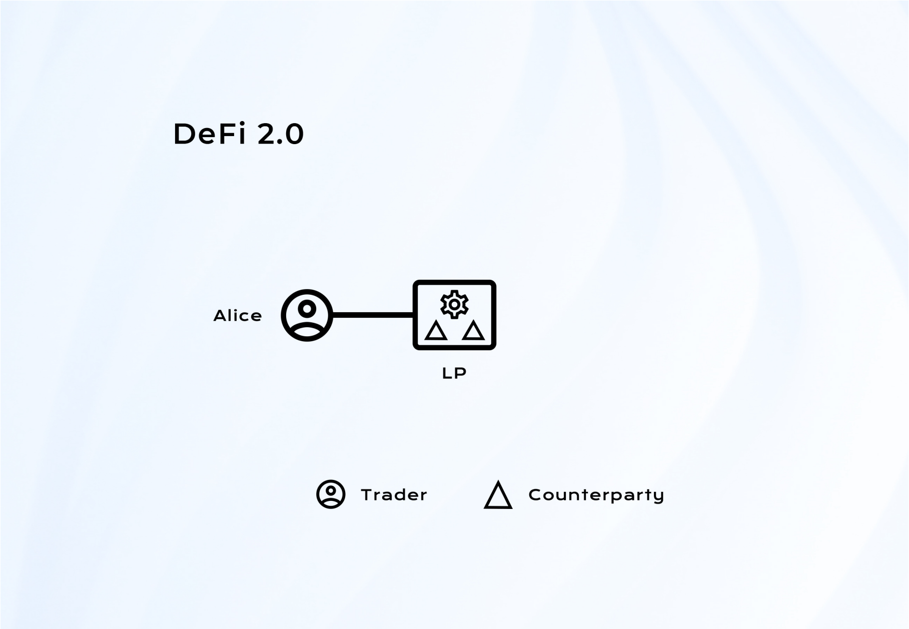

Figure 3: Diagram of DeFi 2.0

out of the deal. Once an arbitrage opportunity appears, the arbitragers flock in and cause impermanent losses to the liquidity providers. The liquidity providers has no choice but to hop for subsidies from mining, and commission or interest earned can compensate their losses. Although this asymmetric design temporarily alleviates the liquidity shortage, it generates the following problems in the long run: Firstly, many assets are occupied and deposited in DeFi protocols. At the same time, only a small volume of transactions is supported, which results in resource wasting. Most of the total value locked (TVL) is still rushing to liquidity mining for profits. In the long term, the lower the capital utilization rate, the greater the loss of the intrinsic value of the DeFi products. Secondly, in the AMM, the core variables, such as asset price and interest rate, are related to the size of the pool. When the size is small, it is easy to create arbitrage opportunities by high volume trading on one hand, and it is difficult to attract many traders and lenders when the pool is not large enough on the other hand. Moreover, DeFi products' composability and liquidity sharing are only at the technical level since the TVL of various DeFi platforms cannot be shared. This kind of liquidity created at the expense of sacrificing the liquidity providers' flexibility is not a perfect idea under the decentralized architecture.

It is better to erase this asymmetry between liquidity providers and other traders. Creating a brand new form of Defi 3.0: the system plays the role of the counterparty to all traders. This idea complies with the decentralized spirit. Everyone is in the same position to gamble with the smart contract, whether the underlying asset is Bitcoin, Ethereum, or other crypto-assets. The system burns the tokens immediately after any participant pays the system tokens for some financial products, and minting system tokens will settle the income of the financial products. We call this model ILM (Infinite Liquidity Model). It frees us from the traditional financial trading models. It forms a new financial paradigm that not only ensures composability but also uniforms programmability for all DeFi products within the same framework. With Defi 3.0, Alice can now directly trade with the system by participating in a smart contract. The system will be her counterparty in this situation. The diagram of this process is exhibited in Figure 4.

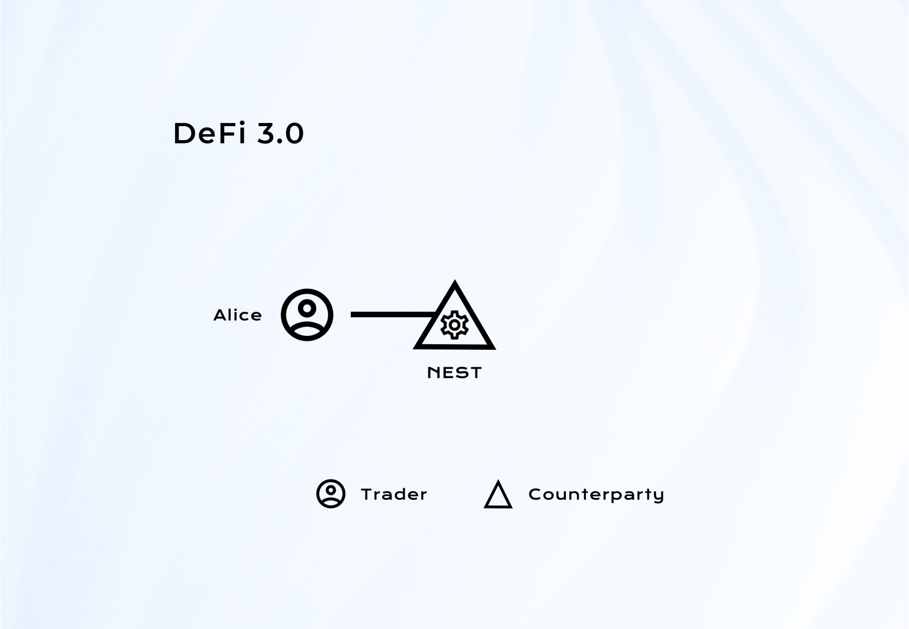

Figure 4: Diagram of DeFi 3.0

2 NEST Probability Virtual Machine

NEST Probability Virtual Machine (PVM) is a virtual machine-like structure based on the basic function library. Developers can develop various exciting applications based on the function library, similar to Ethereum virtual machine (EVM) programming.

NEST inherited the genes of Ethereum. The essence of EVM is that each program is a combination of some basic codes, which can be called as long as a certain gas fee is paid. The income of each product on the PVM can be regarded as a linear combination of some basic income functions, and this income can be obtained as long as a certain cost (that is, the discounted value of this income) is paid. The underlying logic of PVM is the same as that of EVM, which means that all EVM development can be implemented on PVM.

NEST is not just a simple copy of Ethereum. PVM can provide the calculation of random variables, which is a complete probability space relative to Ethereum. Any risk-return pair can be precisely expressed within the PVM. This means that PVM has a larger development space than EVM.

NEST will be used as the gas fee payment method for PVM, and participants can obtain customized products that cannot be produced on other platforms after paying the gas fee. Further, all developers on NEST follow the unified standard of PVM, which provides great convenience for the interoperability and integration between projects, and greatly reduces the cost of forming the NEST ecosystem.

2.1 Principles of NEST PVM

Any financial trading or product is a swap of other returns with different risks, whether the financial activity is an instant transfer, lending, an option, or any complicated derivatives.

Consider stochastic processes $\{S_t\}_{t\geq 0}$ and $\{r_t\}_{t\geq 0}$ defined on $(\Omega, \mathbb{P}, \mathcal{F})$. We use S_t to describe the asset price and r_t to describe the interest rate at time t. Let $\{\mathcal{F}_t\}_{t\geq 0}$

be a filtration generated by S_t and r_t. It is the information stream of the market at the time t. Consider a financial product need to be evaluated at time $t = 0$. Denote R_t the cash flow in at time t and C_t the cash flow out at time t. Let τ_0 and τ_1 be the time sets of cash flow out, and in, respectively, then the diagram of financial products or derivatives can be illustrated as follows.

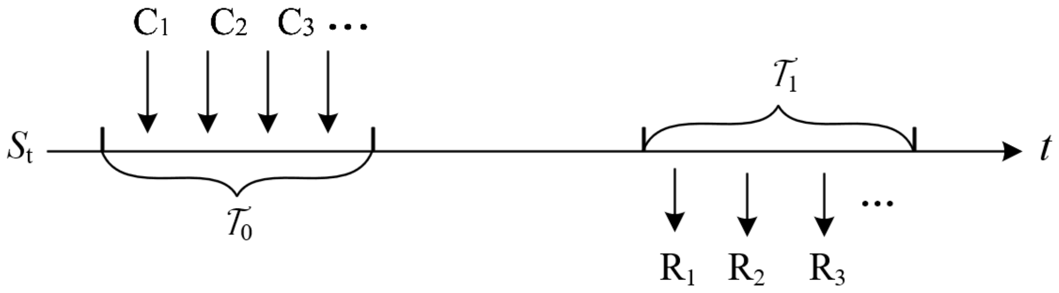

Figure 5: Diagram of financial products or derivatives

The principle of NEST PVM is that, according to a given discount algorithm, any financial products or derivatives can be decomposed into a stream of cash flow in and a stream of cash flow out. The cash flow in is the return of the financial product, and the cash flow out is the cost of the product. Moreover, the sum of the expectation of discounted future cash flow in streams is equal to or slightly smaller (in the case of deflation) than the sum of the expectation of discounted current cash flow out streams are equivalent; both cash flow in and out are uniformly paid by NEST token. With respect to the sigma-algebra $\mathcal{F}_0$ at time $t = 0$, the time of the evaluation, this process can be described as follow:

$$\sum_{t_i \in \tau_1} \mathbb{E}(e^{-\int_0^{t_i} r_s ds} R_{t_i} | \mathcal{F}_0) \leq \sum_{t_j \in \tau_0} \mathbb{E}(e^{-\int_0^{t_j} r_s ds} C_{t_j} | \mathcal{F}_0) \tag{1}$$

Since a financial product can consist of a linear combination of basic revenue functions (BRF), each of which corresponds to a basic discount function (BDF), the cost of the product is a linear combination of these BDFs. Thus, we can design the BRF and its discount function as a developable model: a discount computer -

any financial product can be developed by this computer, and the BRF is like the instructions of the computer. The BDF is like the cost of these instructions or EVM-like gas, except that the gas is paid in NEST, and the instructions produce NEST. The calculation of the financial product P and its cost $C(P)$ can be described as:

$$P = x_1 \cdot BRF_1 + x_2 \cdot BRF_2 + ... = \mathbf{BRF} \cdot \mathbf{X}^T \tag{2}$$

$$C(P) = x_1 \cdot BDF_1 + x_2 \cdot BDF_2 + ... = \mathbf{BDF} \cdot \mathbf{X}^T \tag{3}$$

$\mathbf{X}$ is the linear combination coefficient of the **BRF** representation and **BDF** is the discount function of the **BRF**, where

$$\mathbb{E}(BRF_i|\mathcal{F}_0) \leq \mathbb{E}(BDF_i|\mathcal{F}_0) \tag{4}$$

The discount computer can develop various financial products, including options, perpetuals, leveraged trading, swaps, and lending products, to name just a few, and almost any financial product can be produced.

2.2 NEST Issuance, Settlement, and Pricing

NEST protocol issues NEST token as the only currency for cash settlement. NEST has unlimited supply. For example, when a participant gets into a contract, which assures the participant to receive a certain amount of NEST after a certain number of blocks if the underlining asset price meets certain conditions within the blocks. For example, to get into this contract, the participant needs to pay NEST, the contract's price, to the system, and the system will burn the tokens immediately after receiving them. If the condition has been met, the system will mint new NEST and pay the participant.

As readers can see, all NEST holders share the risks and benefits of NEST issuance or destruction and participate in the equilibrium of supply and demand in the secondary market for NEST. The demand for NEST is from those who buy financial products on the chain and invest in NEST, while the supply is determined by the initial offering and the NEST settled by the NEST protocol. The advantage of sharing the same settlement unit is that we can build all financial services, such as trading, lending, and derivatives, without issuing too many tokens by continuously improving the liquidity of NEST.

According to

$$\mathbb{E}(BRF_i|\mathcal{F}_0) \leq \mathbb{E}(BDF_i|\mathcal{F}_0)$$

Total discounted supply G_y satisfies:

$$\mathbb{E}(G_{t_2}) \leq \mathbb{E}(G_{t_1}),\, t_2 \geq t_1 \tag{5}$$

Total demand (D_t) is determined by trading demand, and the price of NEST (P_t) is determined by the equilibrium of $(D_t,\, G_t)$. Considering the growth of demand and the long-term deflationary nature of supply, there is a logic for a sustained rise in P_t.

Combined with NEST PVM contracts, NEST is an on-chain universal currency with scenarios, which BTC and ETH cannot achieve. BTC has no on-chain scenarios with fixed issuance, ETH follows all scenarios as gas, but its allocation is according to a fixed algorithm, not incremental. Therefore, NEST is guaranteed to clear in every scenario, which matches the completely decentralized currency envisioned by many economists, and is a further development after BTC-ETH.

2.3 The Oracle

The decentralized oracle adopted follows this idea: given an on-chain price stream, how to design a decentralized game such that the game equilibrium can output a price stream with the slightest possible deviation from the on-chain price stream. The oracle solves this problem with quotation mining, two-way options, validation cycles, price chains, and β factors. The oracle provides a price sequence that does not change the distribution of asset prices but approaches a discrete sampling model, which is determined by the structure of the decentralized game, where the quote deviation and quotation density depend on the depth of the arbitrage market and the price of the oracle token. Overall, the oracle provides an efficient decentralized way that maintains the fundamental traits of asset prices. In NEST, we tend to use highly efficient market prices and choose the most liquid underlying assets such as BTC, ETH, etc. The basic price model follows the Geometric Brownian Motion (GBM) model. Considering the characteristics of prices deviation and discrete-time, we correct the prices using the k-factor as follows,

$$k = \max(\frac{|p_2 - p_1|}{p_1}, 0.002) + \sqrt{t} \cdot \max(\sigma, \sigma_0) \tag{6}$$

where p_2 and p_1 represent the current and previous prices respectively, t, measured by second, represents the difference between the time transaction happens and the time p_2 becomes effective. Furthermore, σ the instantaneous volatility follows

$$\sigma = \frac{|p_2 - p_1|}{p_1\sqrt{T}}$$

where T represents the time-lapse between effective p_1 and p_2. σ_0 denotes the regular volatility set by the protocol (generally different values for a different financial product).

The correct procedure follows

- when it comes to a call option, the long price is $(1+k)p$ while the short price is $\frac{p}{1+k}$
- when it comes to a put option, the long price is $\frac{p}{1+k}$ while the short price is $(1+k)p$

where p represents the base price.

2.4 Time Domain

τ_0 and τ_1 denote the time domain can be instants or intervals. A moment can be a definite moment or a random moment (e.g., a stoppage), and in finance, an interval is often used to determine some mean or stoppage. Although the time on the blockchain is discrete, these discrete differences can be ignored for a longer period and compensated for in a shorter period based on the k-factor, and thus can be interpreted approximately in terms of continuous-time intervals.

2.5 Discount Computers

We abstract all financial products (services) as an exchange of revenue streams and expenditure streams, and the revenue streams are represented by linear combinations of the BRFs. Then any financial product development only needs to determine the linear combination of BRFs to obtain its cost (present value) by the linear combination of BDFs. Such linear combinations are the same as we use computer programming, therefore we call this figuratively as the discount computer. Any financial product corresponds to a piece of computer programming. The composable DeFi becomes program designing and calling in the same framework, reducing the difficulty of understanding and risk management.

2.6 BRF and BDF

BRF can be a deterministic value (e.g., block 13678933 to get 1000 NEST) or a random variable after introducing some oracle price information. Here, we consider the basic types of deterministic values, random variables of NEST pricing oracle and probability random variables consist of polynomial functions, exponential functions, logarithmic functions, absolute value functions, maximum-minimum functions, and definite integral operators. The base discount function (BDF) contains normal distribution functions, polynomial functions, exponential functions, logarithmic functions, etc. Considering that the reality does not need so many revenue functions and the complexity of the calculation, we choose a relatively simple list of functions, which can be gradually improved later. As mentioned earlier, the BRF is the basic instructions of the discount computer, while a financial product is a program that combines these instructions.

2.7 Discount Rate and Interest Rate Oracle

In principle, the discount rate reflects the risk-free return of the on-chain world. We can choose a risk-free interest rate statistic on the chain such as the PoS yield of

ETH or the decentralized interest rate oracle (a design as follows: given the number of NEST issues per year, anyone who locks NEST can share these tokens in the proportional ratio) as the discount rate. However, this paradigm is more suitable for a traditional centralized world. In a decentralized world, we can take the discount rate to a relatively small value, even zero, to give the NEST deflationary properties and thus ensure a steady rise in the NEST price.

2.8 Pricing Unit Transformation

If a flat or ETH is required as the numeraire in NEST, it is sufficient to introduce the NEST/USDT or NEST/ETH price, which can be obtained through the NEST oracle. Suppose the liquidity of NEST is large enough that the settlement of a single financial product has little impact on the price. In that case, the financial product with the introduced price is no different from the traditional financial product. The pricing based on the risk-neutral measure ($\mathbb{E}^Q$) can perfectly solve the calculation of the discount function, which can be used for hedging or asset portfolio management.

$$\mathbb{E}^Q(BRF_i|\mathcal{F}_0) \leq \mathbb{E}^Q(BDF_i|\mathcal{F}_0) \tag{7}$$

2.9 Financial Product Development

The development of a financial product in NEST PVM is the same as writing smart contracts, i.e., finding a vector with BRF as the base for the target return, and that vector represents this financial product. The product of that vector and the corresponding BDF base is the cost of the financial product, i.e.; it is only necessary to pay this cost in the time domain τ_0 to obtain the financial product. That financial product will get the NEST minted by the NEST contract in the time domain τ_1, and its quantity is the product of that vector and the BRF. This process is just like writing smart contracts by which all financial products can be implemented with the discount computer programming of NEST PVM. Developers do not have to operate the liquidity of tokens and just need the NEST to be liquid enough.

3 Applications

NEST PVM has a wide range of applications, covering almost all financial services and different trading structures (including peer-to-peer, many-to-many, etc.). It is a revolutionary design in the history of blockchains and can lock in various on-chain economic relationships.

The accessible financial products include European options, perpetual contracts and probability coins.

3.1 European Options

At present, the European option smart contract on NEST PVM supports ETH, and the option price and strike price are calculated based on the price of ETH/USDT. Compared with the traditional European option, the buyer of the European option in NEST PVM, namely the option holder, is a participant in the smart contract, while the seller of the option is the smart contract. Participants can buy European options from the system on NEST PVM and then either sell the option to the system or hold the option until exercise time. The main difference between the European option smart contract and the traditional European option is that the option purchase, option sell, and option exercise in the smart contract are settled by NEST and allow participants to buy only one European option share.

Participants are required to select the type of option (call or put), expiration date (at least 30 days from the purchase date), the strike price (K), and the amount of NEST (X_0) to be used to purchase the option. T is the exact time of a day as the day that the option is purchased ($t_0 = 0$). The system calculates the price of the option based on participant demand and the historical (Used to calculate the historical rate of return of the underlying option[1]: μ). The current price (S_0) of the underlying asset on the predictor (the price of call option c_0, put option p_0) and options share that can be purchased by the participant with X_0 NEST's (call option

[1] The backtracking cycle of historical returns will have a significant impact on the whole model, so a more extended period should be considered.

share $n = \frac{X_0}{c_0}$, put option shares $n = \frac{X_0}{p_0}$). Note that the system considers the NEST to have the same value as the USDT when calculating shares.

The variables involved in this subsection are listed below:

- t_0 or 0: The time spot of buying the European option is denoted as time 0
- T: The time spot when the option is exercised (> 0)
- $t \in [0, T]$: The time between the buying and the exercising of the option ($0 \leq t \leq T$)
- S_t: The price of the underlying asset at the time t (> 0)
- μ: The rate of return on the price of the underlying asset
- σ: The volatility of the price of the underlying asset (> 0)
- K: The strike price of the option (> 0)
- n: The shares owned by this participant of the option (> 0)
- c_t: The value of a European call option at t (≥ 0)
- p_t: The value of a European put option at t (≥ 0)

At the time the option is exercised, the contract automatically extracts the current price of the last block S_T to settle the proceeds of the option exercise. According to the above definition of options, the payoff from the execution of European call options is:

$$\text{Call option: } [S_T - K]^+,$$

$$\text{Put option: } [K - S_T]^+,$$

where $[x]^+ := \max[x, 0]$.

Call contracts mint $n \times [S_T - K]^+$ NEST immediately, and put contracts mint $n \times [K - S_T]^+$ NEST. After execution time, participants can withdraw the NEST from the contract to their wallet.

We assume the price of the underlying asset S_t follows the following Geometric

Brownian Motion (GBM):

$$dS_t = \mu S_t dt + \sigma S_t dW_t,\ t \in [0, T],$$

where W_t, $t \in [0, T]$ is a standard Wiener process (Brownian Motion), μ is the rate of return on the underlying asset's price, σ is the volatility of the underlying asset's price.

According to the initial price of the asset S_0 we can obtain:

$$S_t = S_0 \exp((\mu - \frac{1}{2}\sigma^2)t + \sigma W_t) \tag{8}$$

where W_t can be considered as a normally distributed random variable($W_t \sim N(0, t)$).

Taking $t = T$ and submit (t) into $[S_T - K]^+$ and $[K - S_T]^+$, taking expectation yields to:

$$\mathbb{E}([S_T - K]^+ | S_0) = S_0 e^{\mu T} \Phi(d_1) - K\Phi(d_2), \tag{9}$$

$$\mathbb{E}([K - S_T]^+ | S_0) = K\Phi(-d_2) - S_0 e^{\mu T} \Phi(-d_1), \tag{10}$$

where:

$$d_1 = \frac{1}{\sigma\sqrt{T}}\left[\ln\frac{S_0}{K} + (\mu + \frac{\sigma^2}{2})T\right],$$

$$d_2 = \frac{1}{\sigma\sqrt{T}}\left[\ln\frac{S_0}{K} + (\mu - \frac{\sigma^2}{2})T\right] = d_1 - \sigma\sqrt{T},$$

While Φ is the cumulative distribution function of the standard normal distribution. The price of the European call option at $t = 0$ is:

$$c_0 = \mathbb{E}([S_T - K]^+ | S_0),$$

The price of the European put option at $t = 0$ is:

$$p_0 = \mathbb{E}([K - S_T]^+ | S_0).$$

3.2 Selling European Options

Participants can sell their European options to the system at some time point t $t \in (0, T)$ before the European option expires. The system will calculate the price of the corresponding option c_t (call option) or p_t (put option) at this time according to the price S_t of the underlying asset:

$$\begin{aligned} c_t &= \mathbb{E}([S_T - K]^+ | S_t), \\ p_t &= \mathbb{E}([K - S_T]^+ | S_t). \end{aligned}$$

With the same calculates:

$$\mathbb{E}([S_T - K]^+ | S_t) = S_t e^{\mu(T-t)} \Phi(\hat{d}_1) - K\Phi(\hat{d}_2), \tag{11}$$

$$\mathbb{E}([K - S_T]^+ | S_t) = K\Phi(-\hat{d}_2) - S_t e^{\mu(T-t)} \Phi(-\hat{d}_1), \tag{12}$$

where:

$$\hat{d}_1 = \frac{1}{\sigma\sqrt{(T-t)}} \left[\ln \frac{S_t}{K} + (\mu + \frac{\sigma^2}{2})(T-t) \right],$$

$$\hat{d}_2 = \frac{1}{\sigma\sqrt{(T-t)}} \left[\ln \frac{S_t}{K} + (\mu - \frac{\sigma^2}{2})(T-t) \right] = \hat{d}_1 - \sigma\sqrt{(T-t)}.$$

Depending on the option price at that time, the system mints and pays either $n \times c_t$ (call option) or $n \times p_t$ (put option) NEST to the participant.

3.3 Perpetual Contracts

The perpetual contract in NEST PVM is a smart contract acting as a futures exchange, the central counterparty for all participants to trade. However, because NEST is directly burned and minted by the system, and the system directly calls the price information of the underlying asset on the oracle to conduct settlement calculation, the algorithm is different from that of centralized futures. Hence, the perpetual contract in NEST PVM only needs to consider a price time series without considering the convergence of price differences. At present, the smart contract of perpetual contract option on NEST PVM also supports ETH: futures opening, closing, and closing positions are calculated based on the spot price of ETH/USDT.

The variables related to this subsection are listed blow:

- t_0 or 0: The time of opening a perpetual position is denoted as t_0 or 0

- T: The time spot of closing this perpetual position (> 0)

- $t \in [0, T]$: a time spot between the opening time and the closing time of this perpetual position ($0 \leq t \leq T$)

- S_t: The spot price of the underlying asset at time t (> 0)

- μ: The historical rate of return of the price of the underlying asset

- σ: The long-term volatility of the price of the underlying asset (> 0)

- σ_0: Spot volatility of the price of the underlying asset (≥ 0)

- t_1: The time spot which has an effective price information and nearest to and before time t (> 0)

- t_2: The time spot which has an effective price information and nearest to and before time t_1 (> 0)

- $P_{t,1}$: The effective price of the underlying asset at time t_1 (> 0)

- $P_{t,2}$: The effective price of the underlying asset at time t_2 (> 0)

- $C(t)$: The compensatory factor of the price of the underlying asset (> 0)

- L: The leverage ratio of this future position ($L = 1, 2, 3, 4, 5$)

- $R(t)$: The dynamic rate of return of this position at time t

- $M(t)$: The margin balance at time t (≥ 0)

- $M_m(t)$: The maintenance margin level at time t (> 0)

- a: The minimum retained balance factor ($= 10$)

As with centralized futures contracts, participants are required to choose which position to open: long or short; You need to determine the leverage ratio (L) at the time of opening, which determines the margin ratio; Finally, he needs to decide how much of the margin to put up (M_0). Unlike centralized perpetuities, the opening and closing prices of NEST PVM perpetuities are based on the Nest prognostic price sequence rather than the futures market price.

The price sequence on the oracle machine is not always updated in time compared with the price sequence in the exchanges, so the most recent available price information ($P_{t,1}$) may be accepted by the system a short time (t_1) before the participant opening ($t = 0$) or closing ($t = T$) a position. To prevent arbitrage opportunity, we introduce a compensating coefficient ($C(t)$) to calculate the opening price (S_0) and the closing price (S_T):

$$C(t) = \max(\frac{|P_{t,2} - P_{t,1}|}{P_{t,1}}, 0.002) + \sqrt{t_1} \times \max(\sigma(t), \sigma_0(t)),$$

where, $P_{t,2}$ is the effective price before and nearest to $P_{t,1}$, the time lag between the time of taking effect of $P_{t,2}$ and the time of taking effect of $P_{t,1}$ is: t_2; $\sigma(t)$ is the long-term volatility of the price of the underlying asset at time t; $\sigma_0(t)$ is the spot volatility of the price of the underlying asset at time t calculated based on $P_{t,1}$ and $P_{t,2}$:

$$\sigma_0 = |\frac{P_{t,1} - P_{t,2}}{P_{t,2}\sqrt{t_2}}| \ .$$

After the compensatory factor is introduced, the opening and closing prices of participants are defined as:

$$\text{Long opening price: } S_0 = P_{0,1} \times (1 + C(0)),$$

$$\text{Long closing price: } S_T = \frac{P_{T,1}}{1 + C(T)},$$

$$\text{Short opening price: } S_0 = \frac{P_{0,1}}{1 + C(0)},$$

$$\text{Short closing price: } S_T = P_{T,1} \times (1 + C(T)).$$

Unlike daily and real-time settlement, perpetual contracts in NEST PVM are settled on a per-blockchain basis and are calculated differently from futures market price sequences. In NEST PVM, we defined the dynamic rate of return of this position at a time t ($t \in [0, T]$):

$$R(t) = L \times \frac{S_t e^{-\mu t} - S_0}{S_0} \ .$$

Where, according to the position, S_t is the closing price at time t, S_0 is the opening price at time 0, L is the leverage ratio, and μ is the historical rate of return of the price of the underlying asset. With the dynamic rate of return of this position, we define the margin balance at t as:

$$\text{Long margin balance: } M_t = M_0 \times (1 + R(t)),$$

$$\text{Short margin balance: } M_t = M_0 \times (1 - R(t)).$$

After the participant opens a position, he can either close his position or open more positions in the same position or close the position, at any later point in time t ($t \in [0, +\infty)$). If he chooses to close, he will receive NEST equal to his margin balance. When his margin balance falls below the margin maintenance level: $M_m(t)$, another participant can forcibly close his contract and receive NEST equal to his margin balance. The margin maintenance level in the system is defined as:

$$M_m(t) = \max(M_t \times L \times 0.02, a),$$

where a is the minimum remaining balance parameter in the system, which we set as 10 at present.

If the participant chooses to open an additional position with the same leverage at time t, his positions are combined and his equivalent opening price S is calculated. If the participant closes his position at time T $(T > t)$, then the margin balance is:

$$(1 \pm L\frac{S_T e^{-\mu T} - S_0}{S_0})M_0 + (1 \pm L\frac{S_T e^{-\mu(T-t)} - S_t}{S_t})M_t,$$

where all ± takes + or − at the same time. Suppose another participant opens a position with S and posts a margin $M_0 + M_t$ at time t. At T, his margin balance is:

$$(1 \pm L\frac{S_T e^{-\mu(T-t)} - S}{S})(M_0 + M_t),$$

where all ± takes + or − simultaneously as the former formula. We want to make the margin balance of these two participants equal, so from the above two equations, one can be obtained:

$$(1 \pm L\frac{S_T e^{-\mu T} - S_0}{S_0})M_0 + (1 \pm L\frac{S_T e^{-\mu(T-t)} - S_t}{S_t})M_t = (1 \pm L\frac{S_T e^{-\mu(T-t)} - S}{S})(M_0 + M_t),$$

The equivalent opening price S can be obtained from the above equation:

$$S = \frac{(M_0 + M_t)S_0 S_t}{S_0 M_t + S_t M_0 e^{-\mu t}}.$$

The combined initial margin is equal to the sum of the two margin payments.

3.4 Probability Coins (PRC)

The probability coins (PRC) give the holder the right to request the system to mint NEST for the holder at some moment with a predetermined probability. To exercise a PRC, the holder needs to decide when to exercise and the probability parameter N. Probability coins with parameter N can request the system to mint N NEST when exercising with probability $\frac{1}{N}$, or the holder will receive nothing with probability $1 - \frac{1}{N}$.

One can see that, with parameter N, the expectation of the outcome of exercising a PRC is equal to 1, and the variation of this outcome is $N - 1$.

A holder who has M PRCs can exercise all the PRC at one time. The holder can receive MN NEST with probability $\frac{1}{N}$, or nothing with probability $1 - \frac{1}{N}$. The expectation of the outcome of exercising a PRC is equal to M, and the variation of this outcome is $M^2(N - 1)$.

Or the holder can exercise them one by one, the outcome of every PRC is independent with each other. With this circumstance, the expectation of the total outcome is equal to M, but the variation of this outcome is $M(N - 1)$.

Furthermore, the holder can decide to existence any PRC at a time as long as he still holds some PRC. This will give the holder more complicated probability space of outcome.

3.5 Warrant and Impermanent Loss (IL)

Liquidity provider (LP) in AMM protocol inevitably suffers from IL. In this subsection, we propose, as one application based on NEST PVM protocol, a possible solution for it.

The AMM protocol of **Uniswap** (V2) adopts one common constant product function, i.e.

$$x_t \cdot y_t = k \tag{13}$$

where x_t and y_t represent the quantities of numeraire and traded token in the liquidity pool respectively. It is also required that the ratio of the two assets inside the pool always represents the price of the traded token, i.e.

$$p_t = \frac{x_t}{y_t} \tag{14}$$

Assuming individual LP holds (x_0, y_0), with in hedge period T (measured by year), he/she can pay following quantity

$$(e^{\mu T} - 2e^{\frac{\mu T}{2} - \frac{\sigma^2 T}{8}} + 1)x_0 \tag{15}$$

of NEST to get a derivative where μ and σ^2 represent the mean and variance of price in the liquidity pool. He/She can get following quantity

$$\sqrt{k}(\frac{p_t}{\sqrt{p_0}} + \sqrt{p_0} - 2\sqrt{p_t})$$

of NEST if exercising the derivative at time $t \in [0, T]$ or following quantity

$$\sqrt{k}(\frac{p_T}{\sqrt{p_0}} + \sqrt{p_0} - 2\sqrt{p_T})$$

of NEST if exercising it after T.

As we can see, the above functions are written as a relatively complicated function with non-linear structure regarding p_0, p_t, etc. This makes it practically difficult, although in principle not impossible, for the LP to find the counterparty who is willing to sell the specific derivatives. However, notice that the flexibility of BRF and BDF in NEST PVM allows any function forms for the cash flow in and out; NEST PVM can simply issue one warrant according to above functions. The omnipotent power of ILM is not limited to providing infinite liquidity but all possible types of liquidities based on any function.

4 Further Applications

There are many possible applications for NEST PVM to develop in the near future.

4.1 Trading, Price Coins, and Stablecoins

A native asset is actually equal to a price coin (NEST), which is equivalent to splitting an asset into a dynamic price and a fixed settlement unit. This model can only be effectively implemented in a fully decentralized world: the traditional centralized world has the credit risk of cashing out. Thus, trading is equivalent to exchanging NEST for various price coins or settling out of NEST with various price coins. Or using the native asset to exchange the corresponding price coin at a ratio of 1 to 1 (which is slightly off due to the price deviation of oracles). By analogy, a stablecoin pegged to a fiat currency such as the US dollar is a USDT price coin.

4.2 Exponential and Logarithmic Coins

A new paradigm is the exponential coin, where the percentage of price fluctuations feeds into the growth of returns in an exponential manner. Compared to leveraged coins, exponential coins have many advantages: no need to close positions, faster growth, interchangeable transfers, free stacking at the same address, etc. For example, when the price doubles, the exponential coin with the e base can grow 7.4 times, and when the price triples, the exponential coin grows 20 times.

4.3 Revenue Swaps

Revenue swaps are cost swaps, as they are all equivalent to discounted of revenue streams.

4.4 Lending

Lending becomes much easier by pledging assets accepted by NEST PVM to receive the corresponding NEST, repaying them to receive the collateralized asset, and then being liquidated when the liquidation ratio is exceeded, where the core parameters are the liquidation ratio, collateral rate, interest rate, etc.

4.5 Insurance

Based on the characteristics at the end of the event, price insurance can be made to swap the loss with the premium.

4.6 Interest Rate Derivatives

It is possible to design various interest rate derivatives through price information of the base interest rate oracle.

4.7 NFT Applications

It is possible to lock in all economic relationships in on-chain games or NFTs based on NEST, i.e., all game assets can be designed to some probability coins or derivatives above. In this way, its game assets corresponding to NFT can be cashed in NEST PVM, regardless of the existence of that game, thus building a consistent variable in the game world.

4.8 Multi-Party Trading

To design a transaction between two or more participants: A and B can create a contract based on NEST PVM, each paying a certain amount of NEST at the current moment and receiving the corresponding NEST in the future so that NEST PVM can participate in the allocation between them, which realizes a multi-player competition and game structure.

5 Summary

NEST PVM offers a new paradigm: financial products are considered as programming over BDFs. The cost is the expense of calling those functions, just like EVM.

The difference is that the economic relations of the discount computer are inherent. The new paradigm can cover almost all financial products (services) which can be bought at any time and settled with unlimited liquidity, where market makers, margin (call), and fear of being unable to settle are not required. As long as the NEST liquidity is sufficient, it would be easy to recreate the traditional financial market. Moreover, as difficulties of issuance and settlement are solved, traditional derivatives exchanges can focus on the secondary market, thus significantly reducing their costs. In addition, NEST PVM can bring the fundamental consistent variables for the metaverse, with the ability to traverse different games to lock in economic relationships. Last but not the least, the omnipotent NEST PVM includes two unique features. First, ILM guarantees infinite liquidity to the system which allows any user can get any size of position at any time. Secondly, NEST PVM allows a complete probability space which extends all risk-return combinations for the product development.