Napier Yield Stripping Mechanism

rev. 0.2.1

0xBakuchi

bakuchi@napier.finance

March 7, 2024

1 Abstract

Yield Stripping involves splitting a deposited yield-bearing token into principal and yield tokens. This allows users to lock in a fixed rate by selling the yield tokens upfront, enabling leveraged speculation on interest rates. This paper defines the changes in states as smart contracts based on a sophisticated Yield Stripping mechanism designed to support volatile and high-risk assets.

2 Previous Work

In this paper, we build upon the foundations of Sense Math [1]. Sense Math provides a unique framework for Yield Stripping. The concepts and methodologies introduced in Sense Math have greatly influenced this paper.
Sense Math introduces a configurable parameter for handling Yield bearing tokens, allowing for scenarios where the exchange rates of yield-bearing tokens may decrease. This flexibility allows Sense Math to effectively handle

various yield-bearing tokens, including AMM LP tokens and other risky assets. By leveraging this configurable parameter, Sense Math would ensure fair pricing and optimal system behavior, providing a robust framework for managing diverse assets.
For further details, the original Sense Math publication [1] provides the underlying principles.

3 Definitions

The following definitions are used throughout this paper.
Maturity: The timestamp when yield can be traded until. Short form: t_m.
Tranche: A smart contract that allows users to interact with the Yield Stripping Mechanism.
Target Token: A yield-bearing token that is used as a yield source of Tranche.
The equation below represents the total amount of Target token in the Tranche $T(t)$, which is worth $A(t)$ Underlying token, where $s(t)$ represents the scale or exchange rate of Target token in terms of Underlying token at time t.

$$A(t) = T(t) \times s(t) \tag{1}$$

Scale $s(t)$ is expected to increase over time based on the yield source.
Yield Token (YT): Token that represents the entitlement to claim the yield from the Tranche, and it allows collection at any time.
$ytBalance(t)$: The balance of Yield Tokens for a user at time t.
$ytSupply(t)$: The overall supply of Yield Tokens at time t.

$$\sum_u ytBalance_u(t) = ytSupply(t) \tag{2}$$

Principal Token (PT): Token that represents the right to claim the principal from the Tranche after maturity.
$ptBalance(t)$: The balance of Principal Tokens for a user at time t.
$ptSupply(t)$: The overall supply of Principal Tokens at time t.

$$\sum_u ptBalance_u(t) = ptSupply(t) \tag{3}$$

4 State Changes

This section focuses on how essential variables like Yield Tokens (YT), Principal Tokens (PT), scale, and yield change due to actions such as issuing, combining, redeeming, earning yield, transferring, and claiming. These actions mirror actual user-system interactions.
At any point in time, we define t^* as the timestamp of the latest state changes. The following variables are introduced to track user states:
$s_u(t)$: The stored maximum scale at time t when a user u claimed yield for the last time.
$y_u(t)$: The unclaimed yield of a user u at time t.

4.1 Earning Yield

At time $t < t_m$, a user u earns yield by holding YT. Let t_l be the time when a user claimed the yield for the last time, and the user's scale is equal to the maximum scale value $S(t_l)$ observed throughout the time period t_l.
The stored maximum scale for user u at time t^* is denoted as $s_u(t^*)$ and is set to the maximum scale value $S(t_l)$.
The user holds $ytBalance_u(t^*)$ from last time t_l until time t and accrues yield in Target token during the period. Let $\omega_u(t)$ be the accrued yield of the user from the last accrued time t_l to time t:

$$\omega_u = ytBalance_u(t^*) \times \left(\frac{1}{s_u(t_l)} - \frac{1}{S(t)} \right) \tag{4}$$

4.2 User Transferring YT

At any time t, a user u can transfer d_p YT to another user v. Both users' accrued yield from last time t_l until time t is added to the users' unclaimed yield ω_u and ω_v according to Equation (4),
The state changes are as follows:

$$\begin{aligned}
y_u(t) &= y_u(t^*) + \omega_u(t^*) \\
y_v(t) &= y_v(t^*) + \omega_v(t^*) \\
ytBalance_u(t) &= ytBalance_u(t^*) - d_p \\
ytBalance_v(t) &= ytBalance_v(t^*) + d_p \\
s_u(t) &= S(t) \\
s_v(t) &= S(t)
\end{aligned} \tag{5}$$

4.3 User Claiming Yield

At time t, a user u can claim Ω Target token, which is the sum of the accrued yield ω_u from t_l to t and unclaimed yield $y_u(t^*)$:

$$\Omega = \omega_u(t^*) + y_u(t^*) \tag{6}$$

The state changes are as follows:
The user's scale is updated to the maximum scale at t, $S(t)$, and unclaimed yield is reset to zero:

$$\begin{aligned} y_u(t) &= 0 \\ s_u(t) &= S(t) \\ T(t) &= T(t^*) - \Omega \end{aligned} \tag{7}$$

4.4 Minting PT and YT before the Maturity

At time $t < t_m$, a user u deposits d_a of Underlying token to issue PT and YT.

If $ytBalance(t) > 0$, the user claims accrued yield from the last time, t_l, until the current time, t, before issuing new PT and YT. The yield is reinvested to issue more PT and YT.

Referring to Equation (4), the accrued yield for user u at time t^* is calculated as follows:

$$\omega_u(t^*) = ytBalance(t^*) \times \left(\frac{1}{s(t_l)} - \frac{1}{S(t)} \right) \tag{8}$$

Under the hood, Tranche receives d_t Target token in return for the deposit made:

$$d_t = \frac{d_a}{s(t)} + \omega_u(t^*) + y_u(t^*) \tag{9}$$

Tranche mints the amount of d_p PT and YT for each as follows:

$$d_p = d_t \times S(t) \tag{10}$$

The state changes are as follows:

$$\begin{aligned}
s_u(t) &= S(t) \\
T(t) &= T(t^*) + d_t \\
ytBalance_u(t) &= ytBalance_u(t^*) + d_p \\
ptBalance_u(t) &= ptBalance_u(t^*) + d_p \\
y_u(t) &= 0
\end{aligned} \tag{11}$$

4.5 Redeeming After the Maturity

At time $t > t_m$, a user u can redeem d_p PT, which is worth d_a Underlying token. The amount of Target token to be redeemed by Tranche is:

$$d_t = \frac{d_p}{S(t_m)} \tag{12}$$

Which is converted to d_a Underlying token: $d_a = d_t \times s(t)$ based on Equation (1),
The state changes are as follows:

$$\begin{aligned}
ptBalance_u(t) &= ptBalance_u(t^*) - d_p \\
T(t) &= T(t^*) - d_t
\end{aligned} \tag{13}$$

4.6 Combining PT and YT Simultaneously at Any Time

At any time t, a user u can redeem d_p PT and d_p YT to obtain d_t of Target token, which is worth d_a of Underlying asset.
If the user combines PT and YT before the maturity, the following formula is applied:

$$
\begin{aligned}
d_t &= \omega_u(t^*) + y_u(t^*) + d_p \frac{1}{S(t)} \\
&= d_p \times \left(\frac{1}{s_u(t_l)} - \frac{1}{S(t)} \right) + y_u(t^*) + d_p \frac{1}{S(t)} \\
&= d_p \frac{1}{s_u(t_l)} + y_u(t^*)
\end{aligned}
\tag{14}
$$

The state changes are as follows:

$$
\begin{aligned}
T(t) &= T(t^*) - d_t \\
ptBalance_u(t) &= ptBalance_u(t^*) - d_p \\
ytBalance_u(t) &= ytBalance_u(t^*) - d_p \\
y_u(t) &= 0
\end{aligned}
\tag{15}
$$

5 Proof

In this section, we provide a mathematical proof of two key claims regarding the functionality of our model. First, we demonstrate that the total yield claimable by a user u is unaffected by the timing of their claim. Second, we establish two lemmas concerning the total amount of Target token in the Tranche prior to maturity.

Let t_0 denote the UNIX timestamp when a user u issued Principal Tokens and Yield Tokens.

5.1 Total Claimable Yield is Unaffected by Claim Timing

Let $\Omega_u(t_a, t_b)$ represent the total yield claimable by user u during the period from t_a to t_b, where $t_a \leq t_b$. Assume that the user holds $ytBalance_u$ Yield Tokens from t_0 to t_n. The claimable yield from t_0 to t_n is given by:

$$\Omega_u(t_0, t_n) = ytBalance_u \times \left(\frac{1}{s_u(t_0)} - \frac{1}{S(t_n)} \right) \quad (t_0 \leq t \leq t_n) \tag{16}$$

Assuming the user claims an yield at arbitrary times t_i ($t_0 < t_1 \leq t_2 < \cdots \leq t_n$), we can show that the yield accrued by the user is always equivalent to the total claimable yield from t_i to t_n.

$$\begin{aligned} \sum_{i=0}^{n-1} \Omega_u(t_i, t_{i+1}) &= ytBalance_u \times \sum_{i=0}^{n} \left(\frac{1}{s_u(t_i)} - \frac{1}{S(t_n)} \right) \\ &= \Omega_u(t_0, t_n) \end{aligned} \tag{17}$$

As a reminder $s_u(t_i) := S(t_i)$, implying that every time a user claims yield, the user's scale is updated to the current maximum scale at the time. Thus, the total claimable yield by the maturity doesn't depend on claim timing and how the scale changes over the course of the maturity.

5.2 Lemma 1

This lemma shows that before the maturity ($t < t_m$), the user's yield token balance equals their principal token balance, i.e., $ytBalance_u(t) = ptBalance_u(t)$.

5.3 Lemma 2

This lemma states that the total amount of Target token in the tranche at any time before the maturity ($t < t_m$) equals the total amount of Target token that can be redeemed by Principal, plus the sum of all unclaimed interests by Yield tokens.
The detailed proof is conducted in the following three steps:

5.3.1 User Claiming Yield Before the Maturity Satisfies Lemma 2

We start with the assumption that a user u accrued $\omega_u(t^*)$ from t_l to t and has unclaimed yield $y_u(t^*)$. We show that both $T(t)$ and $y_u(t)$ decrease by the same amount $\omega_u(t^*) + y_u(t^*)$, proving that the equation holds true for t.

5.3.2 Minting PT and YT Before the Maturity Satisfies Lemma 2

$T(t)$ increases by $d_t := \frac{d_p}{s(t)}$.
And $ytSupply(t)$ increases by $\left(\frac{d_p}{s(t)} + \omega_u(t^*) + y_u(t^*)\right) \times S(t)$, and $\omega_u(t)+y_u(t)$ is 0. Hence, the equation holds true for t.

5.3.3 Combining PT and YT Satisfies Lemma 2

We assume that a user u issued PT at t_0, and at t, the user is about to combine PT and YT.

$$T(t^*) = \frac{ytSupply(t^*)}{S(t_0)} + \omega_u(t^*) + y_u(t^*) \tag{18}$$

For simplicity, a user u has not have any unclaimed yield before t. Thus $y_u(t)$ is equal to zero. And then the user has already claimed yield which is worth $\omega_u(t^*)$ Target token right before combining PT and YT:

$$\omega_u(t^*) = ytBalance_u(t^*) \times \left(\frac{1}{s_u(t_0)} - \frac{1}{S(t^*)}\right) \tag{19}$$

Where $s_u(t_0)$ is the maximum scale at the time of issuance $S(t_0)$. Let ΔT be the total withdrawn amount of Target token. Based on Equation (4.6), the Target token in Tranche after the operation becomes:

$$\begin{aligned} T(t^*) - \Delta T &= T(t^*) - \left(\frac{d_p}{s_u(t^*)} + d_p \times \left(\frac{1}{s_u(t_0)} - \frac{1}{S(t^*)}\right)\right) \\ &= T(t^*) - \left(\frac{d_p}{S(t^*)} + d_p \times \left(\frac{1}{s_u(t_0)} - \frac{1}{S(t^*)}\right)\right) \\ &= T(t^*) - \frac{d_p}{s_u(t_0)} \\ &= T(t^*) - \frac{d_p}{S(t_0)} = 0 \end{aligned} \tag{20}$$

Where $\frac{d_p}{S(t_0)}$ is equal to the amount of Target token that the user deposited at the issuance. Even if the current scale is smaller than the one at the time of issuance, the equation holds true for t.
Hence, a user can redeem their same amount of PT and YT at any time and withdraw Target token.

References

[1] ABDK Consulting. (n.d.). *Sense Math.* Retrieved from `https://hackmd.io/f6QDr5jyRd278h6RMF37ew`