Smart Alpha specs

General architecture

We propose a system where there are 2 types of participants:

- Junior token holders
- Senior token holders

The system implements an epoch-based approach. The participants can enter and exit the pools at the end of each epoch but they will be allowed to signal their entry or exit at any time during the epoch.

Both types of users will receive an ERC20 token representing the proof of liquidity.

Junior token holders provide liquidity and buy risk from Senior investors. The risk in the case of Smart Alpha product is the risk of price fluctuations in the underlying token. We assume the underlying token is ETH throughout this document. Any ERC20 token could be tranched by Smart Alpha.

Investors that buy the Senior tokens will have a guaranteed USD value up to a certain price crash during an epoch.

Part of the liquidity provided by the juniors covers the losses of seniors in case of an underlying token's price crash. Similarly some of the USD denominated value generated by price appreciation of the underlying token is transferred from seniors to juniors. The rates of the "upside exposure" and "downside protection" that the Senior investors will benefit of will be calculated at the start of each epoch and is dependent of the pool's composition.

Junior and Senior entry

The mechanics of the deposit are similar for Juniors and Seniors, so they will be treated together in this document.

The deposit will happen in 2 steps:

1. signal the intention to enter the pool

2. redeem your ERC20 proof of liquidity after the next epoch started

Signal
When signalling the intention to enter the pool, the participant will send underyling tokens to the pool. This underyling will not be counted towards the current epoch's balance, thus not interfering with the current state of the pool.

Once the user has entered the pool, they can only increase the size of the deposit but can not exit the queue.

Redeem
At the epoch end, all the underlying from the queue will be transformed into junior or senior ERC20 tokens at the same price. The queued Juniors and Seniors will not participate in the current epoch's profits or losses.

After the epoch was finalized as described in the **"Advancing an epoch"** section, the users who participated in the queues for the current epoch will be able to redeem their junior or senior tokens at the price they were minted (which is the same for everyone). The redeem can be done at any time in the future, not necessarily in the epoch they entered the queue for.

Junior and Senior exit

The mechanics of the exit are similar for Juniors and Seniors, so they will be treated together in this document.

The exit will happen in 2 steps:
1. signal the intention to exit the pool
2. redeem your underlying after the epoch ended

Signal
When a participant no longer wishes to participate in the pool, they can signal an exit at any time. Since everyone participates in a risk sharing mechanism, the actual exit will happen at the end of the current epoch, when the junior or senior tokens the user wants to exit with will be converted back into underlying. The price at which the tokens will be converted will reflect the current epoch's outcome.

In order to join the exit queue, the participant will send their junior or senior tokens back to the pool and wait for the epoch to finalize. Once in the queue, the user can only increase the amount of tokens they want to exit with but can not change their minds.

Redeem
After the epoch was finalized as described in the **"Advancing an epoch"** section, all the junior and senior tokens that were queued for exit will be converted back into underyling and set aside for the users to come and redeem.

All the participants (of the same type {junior/senior}) in an epoch will claim their underlying at the same price. The redeem can be done at any time in the future, not necessarily in the epoch they entered the queue for.

Advancing an epoch

During an epoch, all the seniors have the same `upsideExposureRate` and `downsideProtectionRate`.

Step 1: finalize the current epoch
At the end of the epoch, we calculate the profits and losses of the pool based on the move of the `entryPrice` compared to the `currentPrice`.

If the `entryPrice < currentPrice` , it means the price increased and the juniors made some profits in the underlying token which we calculate using the following formula:

$$juniorprofits = \frac{(currentprice - entryprice) * (1 - upsiderate) * principal}{currentprice}$$

On the other hand, when the `entryPrice > currentPrice`, meaning the price went down since the start of the epoch, then the juniors must cover some of the losses incurred by the seniors.

The seniors are covered up until a minimum price, named `minPrice`, which is calculated using the following formula:

$$minprice = entryprice * (1 - downsiderate)$$

Next, in order to calculate the amount that the juniors must cover, named `juniorLoss`, we enter 2 scenarios: when the current price is above the minimum protected price and when it is below.

For that, we introduce the `calcPrice` in the loss formula:

> $$calcprice = MAX(currentprice, minprice)$$

Finally, we calculate the `seniorProfits` using the formula:

> $$seniorprofits = \frac{principal * entryprice}{calcprice} - principal$$

Step 2: process the entry and exit queues
Only after the epoch was finalized and the current profits and losses were accounted for and reflected into the pool state, we can determine the entry price for the queued participants.

The price of the junior tokens will be calculated using the following formula:

> $price_{junior} = \frac{liquidity_{junior}}{supply_{junior}}$ *, where the junior liquidity is calculated as*
> $liquidity_{junior} = balance_{total} - balance_{seniors}$

> *Note:* $balance_{total}$ *does not include the queued amounts*

Similarly, the price of the senior tokens will be calculated as:

> $price_{senior} = \frac{liquidity_{senior}}{supply_{senior}}$, *where* $liquidity_{senior} = balance_{senior}$

Both junior and senior token prices must be stored in the contract storage for future claims.

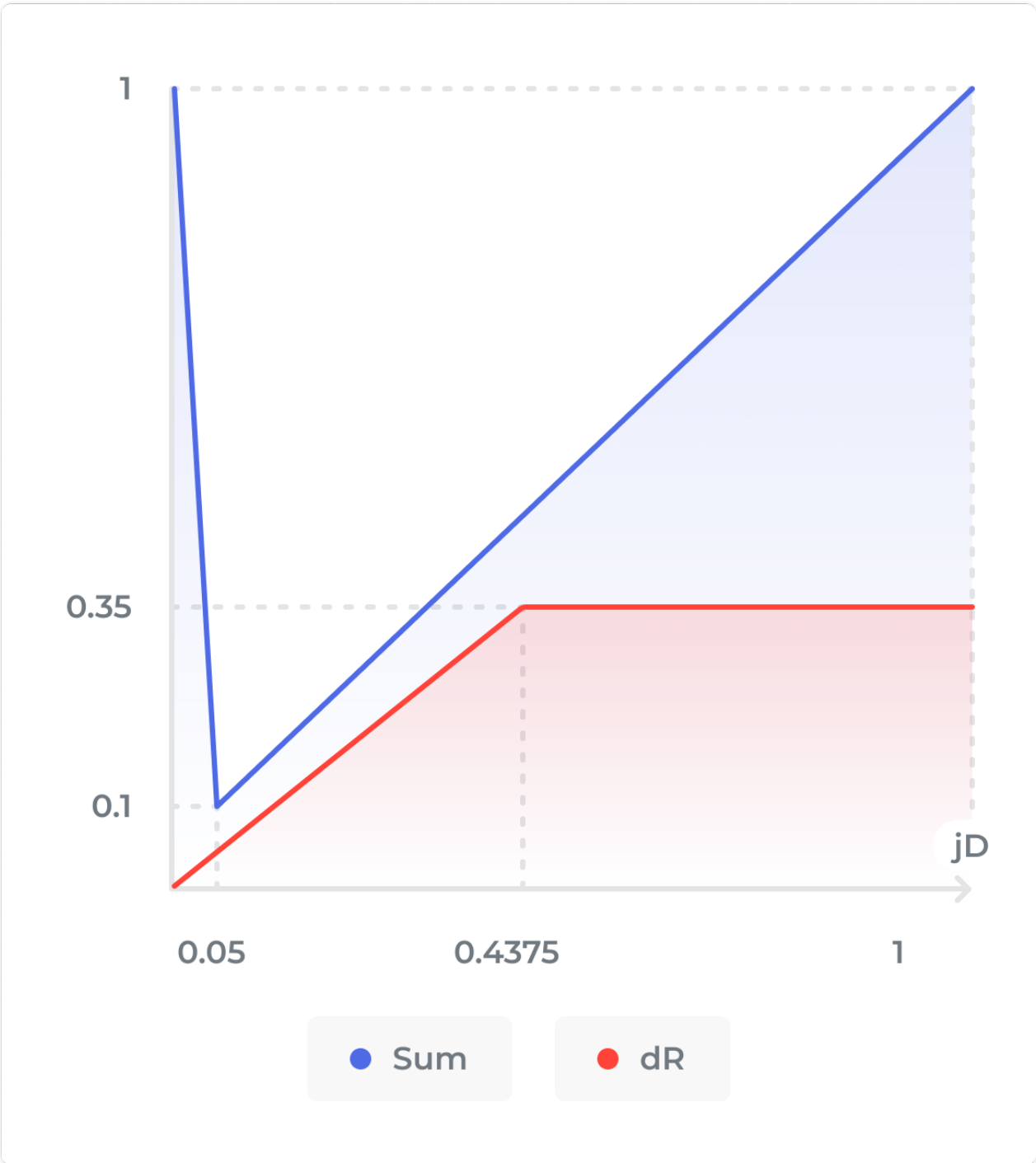
Knowing the new price for the junior and senior tokens, we can now convert the entry queues from underlying to tokens and the exit queues from tokens to underlying. In order to optimize the gas costs, we will re-use the tokens that are queued for burning which effectively leads to only one **mint** or **burn** action to be executed instead of having to do both.

Step 3: recalculate the senior rates
The senior rate model is implemented as a separate contract that can be replaced by the DAO if the need arises.

The current implementation will calculate the rates based on the pool composition represented in the following chart as `junior dominance` .

The `sum` dispalyed on the Y-axis represents the sum of `upside exposure` and `downside protection` rates.

The formulas for the 2 lines represented in the chart are:
$y_1 = -18x + 1$ when x ∈ [0, 0.05)
$y_2 = \frac{18}{19}x + \frac{1}{19}$ when x ∈ [0.05, 1]



Limits:

- **downside protection rate is limited to 80% of the junior dominance** (e.g.: when there are 30% juniors in the pool, the downside protection rate would be 24%)
- **downside protection rate has an absolute limit of 35%** (e.g.: even if there are 80% juniors in the pool, the seniors would only get 35% absolute protection)

Governance & Guardian functions

The Guardian can:

- pause & resume deposits

The DAO can:

- all Guardian functions
- change the fees structure
- change the price oracle
    - change the accounting model
- change the Guardian or itself
- change the senior rates model

Fee structure

Fees will be taken at the end of the epoch from the side that makes a profit.
The percentage of fees is set by the DAO.

Incentivization

TBD

Secondary Markets

Both Junior and Senior tokens are ERC20 compatible. Secondary markets can easily be created for these (e.g. Uniswap).