Distribution

Distribution can be done in either TruMATIC or MATIC.

`__distributeRewardsUpdateTotal`

Reads from the allocations data structure and distributes the correct amount of TruMATIC or MATIC to recipients. It takes as its arguments `distributor` d and `recipient` r and a flag `inMatic` indicating whether the distribution is in MATIC or TruMATIC.

Variable Changes

`userShares`

- $s'_r = s_r + \alpha_d(r)\left(\frac{1}{\pi_d(r)} - \frac{1}{P}\right)(1-\psi)$ recipient gets truMATIC equivalent to the PnL from owning truMATIC minus the distribution fee ψ.
- $s'_{treasury} = s_{treasury} + \alpha_d(r)\left(\frac{1}{\pi_d(r)} - \frac{1}{P}\right)\psi$. The distribution fee gets sent to the treasury.
- $s'_d = s_d - \alpha_d(r)\left(\frac{1}{\pi_d(r)} - \frac{1}{P}\right)$distributor loses truMATIC equivalent to PnL from owning truMATIC.
- ▼ *Overflows and order of operations*

  $$\text{amount to move} = \frac{(\alpha_d(r)\underline{\underline{\pi_d(r)}})\cdot 10^{18}}{\overline{\overline{\pi_d(r)}}} - \text{ceil}(\frac{(\alpha_d(r)\underline{\underline{P_d(r)}})\cdot 10^{18}}{\overline{\overline{P_d(r)}}})$$
  $$= \text{mulDiv}(\alpha_d(r)\underline{\pi_d(r)}, 10^{18}, \overline{\pi_d(r)}) - \text{ceil}(\text{mulDiv}(\alpha_d(r)\underline{P_d(r)}, 10^{18}, \overline{P_d(r)}))$$

  Order of magnitude of the constituent terms of the muldiv is 10^30, 10^18, 10^52 respectively so this should prevent all overflows. In general this will *underestimate* the amount that needs to be transferred.

`allocations`

- $\pi'_d(r) = P$ - update the allocation share price so that next time distribute is called, recipients are only paid the incremental rewards.
- $\alpha'_d(r) = \alpha_d(r)$ - MATIC amount that's allocated does not change.

▼ *Calculations for variable change*

For a share price change from P_0 to P_1 the total rewards for a distribution of α MATIC is:

$$\begin{aligned}\alpha\,\mathrm{MATIC} &= \alpha/P_0\,\mathrm{TruMATIC}\\ &\implies \alpha P_1/P_0\,\mathrm{MATIC}\end{aligned}$$

that is, the rewards accrued in MATIC are $\alpha\left(\frac{P_1}{P_0}-1\right)$, which in TruMATIC with the share price at P_1 is $\alpha\left(\frac{1}{P_0}-\frac{1}{P_1}\right)$.

In the case of distribution, the allocated amount is $\alpha_d(r)$, the initial price $\pi_d(r)$ and the final price is the current price P. Substituting this in gives the above expressions. Note that the allocation share price needs to be updated to the current share price P now since the next time distribution is called, the pnl should be calculated from the previous distribution (which happened at the share price).

`_distributeRewards`

Distributes rewards from the caller to a single recipient but does not change the allocations.

Called by `_distributeRewardsUpdateTotal` and `_distributeAll`.

If the distribution is in MATIC, the correct amount of MATIC is transferred from the distributor's wallet to the recipient's.

If the distribution is in TruMATIC, the correct amount of TruMATIC is transferred from the distributor's wallet to the recipient's.

`distributeAll (public)`

Takes `inMatic` as an argument. Runs through all recipients for a single distributor and distributes all rewards, whether in MATIC or TruMATIC, via `_distributeRewards`.

Variable Changes

`allocations`

- Calls `_distributeRewards(recipient, distributor, inMATIC)` for all recipients in `recipients(distributor)`

`totalAllocated`

- $\alpha_d' = \alpha_d$ - total allocated does not change during distribution
- $\pi_d' = P$ - since every $\pi_d(r)$ is set to P, their weighted average is also P.