Smartlink

Farms & Fixed Token Supply

A Mathematical Model for Tezos

Jeremy Martin

V 1.0 - October 2021

Table of contents

Background

Introduction

The aim of this paper is to give the mathematical and technical basis for farms implementation on Tezos. A farm is a smart contract which allows investors to stake liquidity pool tokens on a periodic basis in exchange for a reward token.

Constraints of the project

There are unique constraints for this architecture as described below:

- All farms have a fixed number of periods to distribute tokens while they are active
- The reward token cannot be minted, therefore the reward supply must be provided at the origination
- We want to be able to increase the reward supply during a farm's life
- We want a decreasing ratio of rewards for every period calculated from a fixed percentage rate
- Users can stake or unstake any number of liquidity pool tokens at any time
- A user can take all his rewards from all the elapsed periods, except the current period

In this paper we propose to take one period = one week. Note that this solution only works for a small number of periods, since we will repeat this number of periods, typically under 100. For instance, a farm distributing rewards for 2 months would correspond to 9 periods.

Variable definitions

Variable	Description	Type	Example
R_{total}	Total reward to distribute in Tokens	$nat * 10^{-3}$	20,000.000 Tokens
w_i	The week number i	nat	Week 2
I_{max}	Number of weeks of distribution	nat	5 Weeks
$R(w_i)$	Reward to distribute for week w_i	$nat * 10^{-3}$	3,500.000 Tokens
T	Weekly degressive rate	$nat * 10^{-2}$	75 %

The mathematical model

Calculating the number of tokens to distribute the week w_i :

We want to determine $R(w_i)$. We have I_{max} and T fixed for this period. Note that we also assume that R_{total} is fixed for this part, and we start at week 1.

We typically want to distribute all the rewards over the weeks until the week $w_{I_{max}}$:

$$R_{total} = \sum_{i=1}^{I_{max}} R(w_i) \tag{1}$$

We also enforce the following weekly degressive rate in our model:

$$R(w_{i+1}) = T * R(w_i)$$

And the corresponding cumulative value:

$$R(w_{I_{max}}) = T^{I_{max}-1} * R(w_1) \tag{2}$$

We can apply classic telescoping for this kind of geometric sequence:

$$R_{total} * T = \sum_{i=2}^{I_{max}+1} R(w_i) \tag{3}$$

Now, by subtracting **(1)** and **(3)** we get:

$$R_{total} - R_{total} * T = \sum_{i=1}^{I_{max}} R(w_i) - \sum_{i=2}^{I_{max}+1} R(w_i)$$

$$R_{total} * (1 - T) = R(w_1) - R(w_{I_{max}+1})$$

Adding the degressive constraint **(2)**:

$$R_{total} * (1 - T) = R(w_1)(1 - T^{I_{max}})$$

We can conclude:

$$R(w_1) = \frac{R_{total} * (1 - T)}{(1 - T^{I_{max}})}$$

And more generally, the reward at week i.
$\forall i \in [1, I_{max}]$:

$$\boxed{R(w_i) = \frac{T^{i-1} * R_{total} * (1 - T)}{(1 - T^{I_{max}})}} \quad \textbf{(4)}$$

Example of distribution

This distribution example will have the following values:

R_{total}	I_{max}	T
20,000.000	5	0.75

By applying formula (4) we get the following farming plan **(A)** :

Week 1	Week 2	Week 3	Week 4	Week 5	Total
6,555.697	4,916.773	3,687.580	2,765.685	2,074.263	19,999.998

Dynamically changing the total number of tokens to distribute

The goal is to add an entry point to the contract that can increase the number of reward tokens to distribute. Let's say we want to increase the number of tokens to distribute by Δ at week j. For the sake of the demonstration, we assume that j <= I_{max}.

Note that if we are already at week I_{max}, then we will increase only the last week's supply to distribute since new rewards for week j will begin at the week $j + 1$. If we are in the first week, it is equivalent to changing the whole distribution.

We will distribute $R_{total} + \Delta$ tokens in total. We need to calculate R'_{total} for the new process.

We already distributed a certain number of tokens, so what is left is :

$$R'_{total} = R_{total} + \Delta - \sum_{i=1}^{j-1} R(w_i)$$

$$\boxed{R'_{total} = \Delta + \sum_{i=j}^{I_{max}} R(w_i)} \quad \textbf{(5)}$$

Furthermore, we keep T unchanged and calculate a new I'_{max} for the serie, in that case :

$$I'_{max} = I_{max} - (j-1) \ .$$

Example of a dynamic distribution

Using our example distribution **(A)**, we now propose an increase of 50,000 tokens on week 3.

Since we distributed **6,555.697 + 4,916.773 = 11,472.470** tokens already, we only have **8,527.530** tokens left in the pool before the increase, so we now get the following values :

R'_{total}	I'_{max}	T
58,527.530	3	0.75

All in all, we calculate the following distribution with **(4)** and **(5)** :

Week 1	Week 2	Week 3	Week 4	Week 5	Total
6,555.697	4,916.773	25,309.202	18,981.902	14,236.426	69,999.999

The Smart-Contract

Structures

- ☐ **farm_config[record]** : The global configuration of the pool.

Field	Type
creation_time	nat
total_reward	$nat * 10^{-3}$
total_weeks	nat
degressivity_rate	$nat * 10^{-2}$

☐ **farm_points[map]**: The total cumulative points of all users for every week.

Key	Value
week	nat

☐ **user_points[map]**: The total share points of any user.

Key	Value
week + address	nat

☐ **user_stake[map]:** The total number of LP tokens staked by any user.

Key	Value
address	nat

☐ **reward_at_week[map]**: The total number of LP tokens to distribute by week.

Key	Value
week	nat

Constants

☐ **week_in_seconds** : 604800 (corresponding to the number of seconds in a week)

Constructor

☐ **make_farm** : Creates the farm.

Argument	Type
total_reward	$nat * 10^{-3}$
total_weeks	nat
degressivity_rate	$nat * 10^{-2}$

- Feeds **farm_config** and the three other arguments
- $\forall$ week **i**, creates **farm_points[i] = 0**
- $\forall$ week **i**, computes **reward_at_week[i]** with the formula **(4)**

Internals

☐ **get_current_week** : Computes the current week

Argument	Type
()	-

- Gets the number of seconds with built-in **Tezos.now**
- Subtracts this value from **farm_config.creation_time**
- Divides by **week_in_seconds** and takes the whole upper part

☐ **send_reward** : Sends reward tokens to **user_address.**

Argument	Type
token_amount	nat
user_address	$address$

- Calls **transfer()** from the token reward contract to send reward tokens to the user

Entry points

☐ **stake** : The function to stake LP tokens.

Argument	Type
token_amount	$nat * 10^{-3}$

- Computes the current week with **j = get_current_week**
- Computes the number of seconds **S** left before the end of the current week **j**
- Creates or increases **user_stake[user_address]** by **token_amount**
- $\forall$ week **i >= j**, creates **user_points[user_address][i]** if it does not exist already
- Adds **S** $*$ **token_amount** to **user_points[user_address][j]** and **farm_points[j]**
- $\forall$ week **i > j**, adds **week_in_seconds** $*$ **token_amount** to **user_points[user_address][i]** and **farm_points[i]**

☐ **unstake** : The function to unstake LP tokens.

Argument	Type
token_amount	$nat * 10^{-3}$

- Checks that **user_stake[user_address] > token_amount**
- Computes the current week with **j = get_current_week**
- Computes the number of seconds **S** left before the end of the current_week **j**
- Subtracts **S** $*$ **token_amount** from **user_points[user_address][j]** and **farm_points[j]**
- $\forall$ week **i > j**, add **week_in_seconds** $*$ **token_amount** to **user_points[user_address][i]** and **farm_points[i]**

☐ **claim_all** : The function to claim the token rewards for the user

Argument	Type
()	-

- Computes the current week with **j = get_current_week**

- $\forall$ week **i < j**, if **user_points[user_address][i] > 0**, compute the sum C_i of **user_points[user_address][i]** divided by **farm_points[i]**
- $\forall$ week **i < j**, send the C_i percentage of **reward_at_week[i]** to **user_address**

☐ **increase_reward** : The function to increase the token rewards

Argument	**Type**
token_amount	$nat * 10^{-3}$

- Computes the current week with **j = get_current_week**
- Computes the new values R'_{total} and I'_{max} with the formula **(5)**
- Updates **farm_config** with the new values R'_{total}
- $\forall$ week **i >= j**, computes the new **reward_at_week[i]** with the formula **(4)** applied to the new values R'_{total} and I'_{max}