dYdX Trading Reward Mechanism Review

Thomas Cintra
tncintra@gmail.com

Max Holloway *
max@xenophonlabs.com

March 2022

Abstract

With the advent of the trader rewards mechanism in August of 2021, the dYdX Protocol has seen significantly increased exchange volumes and revenues. While the initial rewards mechanism leads to much higher exchange revenues, it has a number of undesirable effects, such as incentivizing inorganic late-epoch trading over early epoch trading and making exchange revenues difficult to estimate. In this paper, we provide reasons for why trading surges in the end of dYdX epochs, and we derive an optimal strategy for maximizing trader rewards profits. We also provide a survey of alternative trading rewards mechanisms, and we recommend ways to improve the current mechanism to increase dYdX revenue and make the mechanism more fair for all participants.

*Disclosure: Reference to the DYDX price is necessary for this research. The authors do not own DYDX token, nor are they affiliated with dYdX Trading Inc. or any of its subsidiaries. This research was funded by the dYdX Grants DAO. Any opinions and results stated here are those of the authors, not of dYdX, its subsidiaries, nor the dYdX Grants DAO. This is not financial advice.

Contents

1 Introduction

1.1 Research Motivation

Since the first epoch of dYdX rewards, there has been a meaningful increase in fees paid near the end of the epoch. The most clear evidence of this is the prevalence of wash trading as early as epoch 0; one trader traded over $1.7B of volume through the `COMP-USD` perpetual market near the end of the epoch, mostly between two accounts in order to avoid paying a spread. This trader was not alone, as there were 80 other addresses that were also flagged for wash trading. Due to the dYdX Foundation's swift action to prohibit wash trading addresses from receiving trading rewards, wash trading decreased in future epochs. Still, the lesson is clear: incentivizing trading via the dYdX rewards mechanism has a tangible impact on how traders participate in dYdX's markets.

Given the extreme ways that some traders behave near the end of the epochs, we find it necessary to give a closer examination of dYdX's trading rewards mechanism. The primary goal of this research is to determine the extent to which the current trader rewards mechanism meets its objective of rewarding organic traders. To do this, we must first understand how the mechanism works (section 1), how rewards-profit-maximizing traders would maximize their profits (section 2), and how trading has unfolded historically (section 3). With that understanding of the current trading rewards mechanism, we can then propose ways to augment it to discourage inorganic order flow (section 4).

1.2 Mechanism Background

Recall the old formula for computing individual trader score:

$$w = f^a \times d^b \tag{1}$$

$$r = R \times \frac{w}{\sum_i w_i}, i = 1, 2, ...n \tag{2}$$

where w, f, d, and r are an individual trader's trader score, total fees paid, average open interest, and reward for a given epoch with n total traders, respectively. The constant $R = 3,835,616$ is the total amount of `DYDX` tokens disbursed, and the exponents $a = 0.7$ and $b = 0.3$ are set through governance.

Let p denote the price of `DYDX` at the end of the epoch, and $\vec{F} = (f_1, f_2, \ldots, f_n)$ denote a vector of fees paid by traders, where f_k is the amount of fees paid by trader k. Trader k's profit in `USDC` at the end of the epoch is thus:

$$P_k(\vec{F}) = R \times p \times \frac{f_k^a d_k^b}{\sum_{i=1}^n f_i^{0.7} d_i^{0.3}} - f_k \tag{3}$$

Suppose traders are engaged in market-neutral strategies to maximize open interest. For example, traders could hold equally-sized long and short positions in the `BTC-USD` market in two separate trading accounts and periodically moving

money between accounts to avoid liquidation. This entails some fees spent throughout the epoch, but we assume those to be negligible. Clearly, any capital spent in fees early in the epoch would be better used as collateral to maximize open interest, as fees can always be paid towards the end of the epoch. Since total trader score and `DYDX` price at the end of the epoch are also variable, and better understood by traders near the end of the epoch, there is further incentive for traders not to pay fees too early.

It follows that when traders start paying fees to maximize profits towards the end of the epoch, their average open interest will be approximately constant; this can also be observed empirically in previous epochs.

Thus, this rewards optimization problem can be reduced to an optimization over a fee vector $\vec{F} = [f_1, ..., f_n]$, for a vector of profit functions $\vec{P} = [P_k]$. That is, each trader k maximizes their profit $P_k(\vec{F})$, with respect to their fees f_k, concurrently alongside all of the other traders.

2 Maximizing Profits on the Current Trader Rewards Mechanism

The goal of this section is to explore the impact of trading rewards on the behavior of profit-maximizing traders. We begin with a game-theoretic approach for reward-maximization before the introduction of `stkDYDX` to the rewards formula. Using Newton's method we show that there exists a Nash equilibrium for how much reward-maximizing traders should pay in fees, and then we approximate a closed form for this equilibrium that holds under certain conditions. We examine how this equilibrium is affected by the number of traders n and the distribution of their open interest $\vec{D}$. We then show this equilibrium is unique, and examine how the introduction of `stkDYDX` impacts our solution. We conclude this section by exploring how some key assumptions we make may lead to deviations from our proposed equilibrium.

2.1 Finding a Nash Equilibrium

Consider the following thought experiment. There are n traders who unfortunately find themselves stuck in a room at the end of a dYdX trading epoch. Their task is to determine how much they will pay in fees to maximize their profits. They are told what the current total trader score is, and they know their own average open interest. Traders are told to write down how much they intend to pay in fees, and they repeat this for a number of rounds. During each round, traders decide how much they should pay in fees and write that number down. At the end of the round, they are told what the updated total trader score is, given what everyone else decided to pay in fees. Then, during the next round, they write down the updated fee amount that they want to pay, given their updated expectation on total trader score. This goes on until no trader changes the amount they write down. This resembles the problem faced by

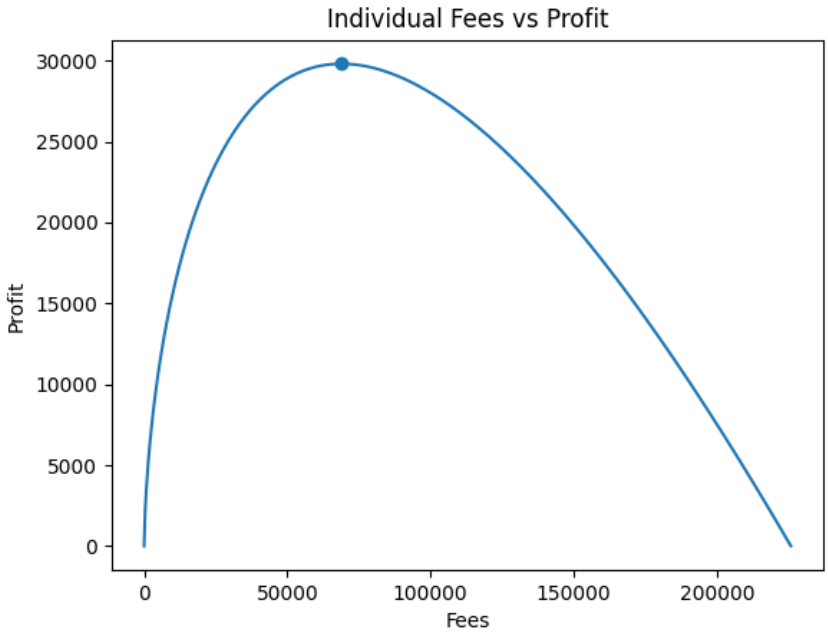

Figure 1: Profit function of trader k for constant $\vec{F}$

rewards-maximizing traders, and we will now show that for n rational agents, this process necessarily converges.

Each trader is trying to maximize their profit function $P_k(\vec{F})$ illustrated in figure 1[1]. We can solve this optimization problem using a numerical root-finding algorithm: Newton's method. We want to solve for $\frac{\partial}{\partial f_k} P_k(\vec{F}) = 0$ for all k simultaneously. Newton's method iteratively solves for f_k that maximizes P_k as follows. For a set of continuous, twice-differentiable functions P_k and an initial starting point $\vec{F}_0$, construct a sequence $\{\vec{F}_i\}$ such that:

$$\vec{F}_{i+1} = \vec{F}_i - \frac{\vec{P}'(\vec{F}_i)}{\vec{P}''(\vec{F}_i)} \tag{4}$$

where $\vec{P}(\vec{F}) = [P_k(\vec{F})]$ is the vector of profit functions computed at $\vec{F}$. To show that the vector $\vec{F}$ converges, the Newton updates must be computed in parallel for all traders at each round as denoted in the vector calculus notation above. Denote $\vec{D} = [d_k]$ as the open interest vector which we argued is constant at the end of the epoch, and D as the total open interest ($D = \sum_k d_k$). The pseudocode for our algorithm is in algorithm 1. We implement Newton's method

[1]The source code for all of the plots is in the `analysis.py` file in this repository.

in Python3 using Sympy to solve for the derivatives of P_k. For further details on our implementation please refer to this repository.

Algorithm 1 Profit Maximization Algorithm

Require: $n \geq 0$
Require: $p \geq 0$
Require: $\alpha \in [0, 1]$ ▷ Learning rate to ensure convergence
Require: $R = 3,835,616$
Ensure: $P'(f_k) = 0, \forall k \in [0, n]$
$\vec{D} \overset{iid}{\sim} \{Dirichlet[[1] \times n, 1]\}$ ▷ Randomly distribute open interest
$\vec{F} \overset{iid}{\sim} \{U[0, 1] \times \frac{D}{n}\}$ ▷ Randomly set initial guesses for fees
$r \leftarrow Inf$ ▷ Distance between successive fees vectors
while $r \geq 10^{-2}$ **do**
    $T \leftarrow \vec{D} \cdot \vec{F}$ ▷ Compute current market score
    $\vec{F}_{new} \leftarrow \vec{F}$
    for $k \in [0, n]$ **do**
        $\vec{F}_{new}[k] \leftarrow \vec{F}[k] - \alpha \times \frac{P'_k(\vec{F}[k])}{P''_k(\vec{F}[k])}$
    end for
    $r \leftarrow dist(\vec{F}_{new}, F)$ ▷ Compute root-mean-squared error
    $\vec{F}_{new} \leftarrow \vec{F}$
end while

We run our algorithm multiple times for different values of n and note that our fees vector always converges to an optimal fees vector $\vec{F}^*$. We now argue that this is indeed a Nash equilibrium.

One could prove this is a Cournot-Nash equilibrium by solving for f_k in $P'_k(f_k)$ and plugging that solution into all f_i in $P'_k(f_k)$ to show that the optimal f_k does not change. That is, if trader k chooses to pay fees according to the output of Newton's method, and then finds out that all other traders are using the same strategy, trader k will have no incentive to change the amount of fees paid. However, solving $P'_k(f_k)$ analytically would prove troublesome:

$$\frac{d}{df_k} P_k(f_k) = \frac{55309582.72 d^{0.3}}{f^{0.3}(T + d^{0.3} f^{0.7})} - \frac{55309582.72 d^{0.6} f^{0.4}}{(T + d^{0.3} f^{0.7})^2} - 1. \tag{5}$$

We can instead take a numerical and graphical approach to show why this is a Nash equilibrium. Using Sympy we can verify that $P'_k(f_k) = 0, \forall k$ for any run of our algorithm. That is, trader k is maximizing her profits and has no incentive to change her strategy. More formally: the algorithm converging implies all traders are maximizing their profit functions, and therefore are at a local equilibrium. For all runs of our algorithm, we verify the derivative of any trader's profit is 0 at our steady state solution $\vec{F}^*$ to ensure trader's are maximizing their profit functions. Again, for implementation details, please refer to our GitHub repository.

We can build some intuition on the shape of $P_k(f_k)$ by referring to figure 1, where the fees vector $\vec{F}$ is held constant except for f_k. By verifying that $P'_k(f_k) = 0, \forall k$ we ensure that every trader finds themselves at the peak of their profit curve, and would not benefit from any change in the amount of fees paid. Furthermore, all profit functions are concave down, which will later help us argue that this is a unique Nash equilibrium.

Notice that according to Newton's method with or a sufficiently small learning rate α, our algorithm converges if and only if there is no incentive for any trader to diverge from their current strategy. This is a consequence of each profit curve being concave and twice differentiable. The algorithm does converge. So there is no incentive for any trader to diverge from this strategy, even if all other traders are also using this strategy. Therefore this is a Nash equilibrium.

2.2 Computing a Closed Form Solution

We ran the profit-maximizing algorithm many times for different numbers of traders n. For large values, $n > 1000$, we noticed a very clear pattern: the ratio $d_k : f_k$ was the same for all traders. In fact, upon closer inspection we verified that the ratio of fees to open interest was:

$$\frac{f_k}{d_k} = \frac{0.7Rp}{D} \tag{6}$$

Notice that it seems the amount that a trader should pay in fees is independent of the distribution of $\vec{D}$ and the total trader score. Trader k can compute the optimal amount to pay in fees without worrying about predicting where total trader score will be at the end of the epoch or how open interest is distributed between traders. Furthermore, it follows that the sum of fees paid in a given epoch for large n can be approximated as:

$$\begin{aligned} \sum_{k=1}^{n} f_k &= \sum_{k=1}^{n} \frac{0.7Rpd_k}{D} \\ &= 0.7Rp\frac{\sum_n d_k}{D} \\ &= 0.7Rp. \end{aligned} \tag{7}$$

meaning dYdX's revenue from trading rewards is linear in the exponent a as well as the price of DYDX, p.

However, when we run our simulations for smaller values of n, this linear relationship between d_k and f_k vanishes. We observe these differences in figure 2. Notice that when there are only 2 traders they converge to paying the same amount in fees regardless of their share of open interest.

While dYdX rewards have historically attracted thousands of eligible traders, it is important to notice this peculiar behaviour at smaller n. As n increases the relationship between d_k and f_k becomes more linear, but we argue that this

is not directly due to n increasing but rather due to the distribution of open interest in the market.

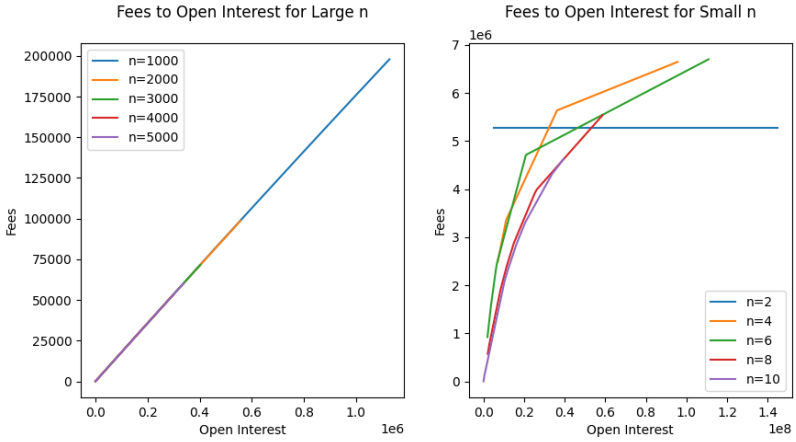

Figure 2: Distribution of fees to open interest for varying n

2.2.1 Distribution of Open Interest

So far we have presented our algorithm using a flat Dirichlet distribution to allocate open interest in our simulations. A Dirichlet distribution allows us to distribute an exact quantity, D, amongst n traders while controlling the weight, α_k, of each trader d_k, which we set to $\alpha_k = 1$ for all traders.

However, the irregular behavior observed for small n prompted us to modify our algorithm to account for whales in the market, i.e. those with disproportionately large average open interest. Recall the probability density function of the Dirichlet distribution: for n points sampled from $Dir(x_1, ..., x_n; \alpha_1, ..., \alpha_n)$ the pdf is:

$$\frac{1}{B(\vec{\alpha})} \prod_{i=1}^{n} x_i^{\alpha_i - 1} \tag{8}$$

where $B(\vec{\alpha})$ is the multivariate beta function, a normalizing constant[2]. A flat Dirichlet distribution where $\alpha_0 = \alpha_1 = ... = 1$ gives equal weight to all traders, such that for large n, no traders are expected to account for a large share of open interest. When we have small n, such as $n = 2$, each trader accounts for a very large share of open interest, and therefore can significantly affect the total trader score by paying more or less in fees.

To simulate this effect we can instead provide very large values of α_i for a small number of traders, such that their share of open interest is much larger.

[2]https://en.wikipedia.org/wiki/Beta_function

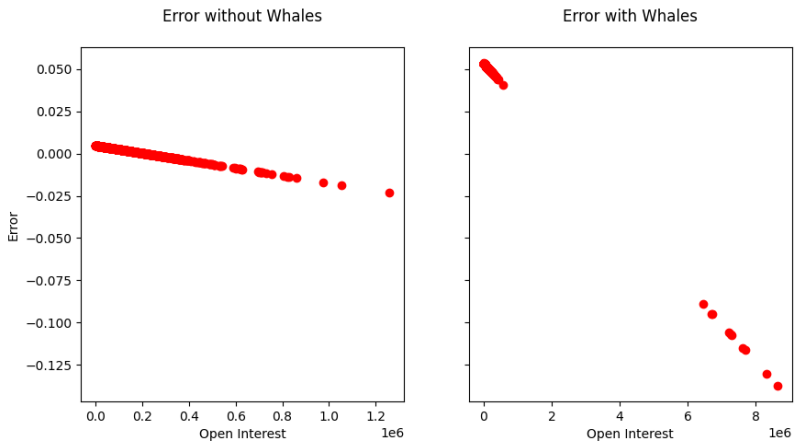

Figure 3: Error of closed form solution $f_k = \frac{0.7Rpd_k}{D}$ with and without whales. Whales correspond to the ten points on the lower right corner of the right-hand plot.

This better reflects the actual distribution of open interest, where some addresses account for upwards of 3% of open interest, and some of these accounts are likely held by the same trader/firm. We will refer to the select traders with disproportionately large shares of open interest as "whales". For the remainder of this section we will be exploring results for simulations with 10 whales of varying sizes. We choose to simulate 10 whales based on historical data presented in dYdX's trader rewards dashboard.

When we account for this uneven distribution in our algorithm, we notice that as the open interest of a particular trader becomes much larger than the mean, and therefore changes in their fees paid significantly impact the overall trader score, their optimal fees paid *decreases* relative to equation (6). We can observe this pattern in figure 3, which displays the difference between our hypothesized closed form $f_k = \frac{0.7Rpd_k}{D}$ and the actual equilibrium found by Newton's method. Notice that without whales most traders can safely pay according to our closed form solution. However when we inflate the open interest of 10 traders this is no longer the case; if they followed the closed-form, the majority of traders will be underpaying in fees, whereas the whales will be overpaying relative to the optimal fees vector found by a run of our algorithm. That is to say, Newton's method converges to a Nash equilibrium, and the proposed closed-form is merely an approximation that does not hold true when we take market whales into account.

Interestingly, the existence and size of whales affect the expected sum of fees paid in any particular epoch. In figure 4 we notice that as the weight of the 10 whales increases, the total sum paid in fees decreases. The red line indicates

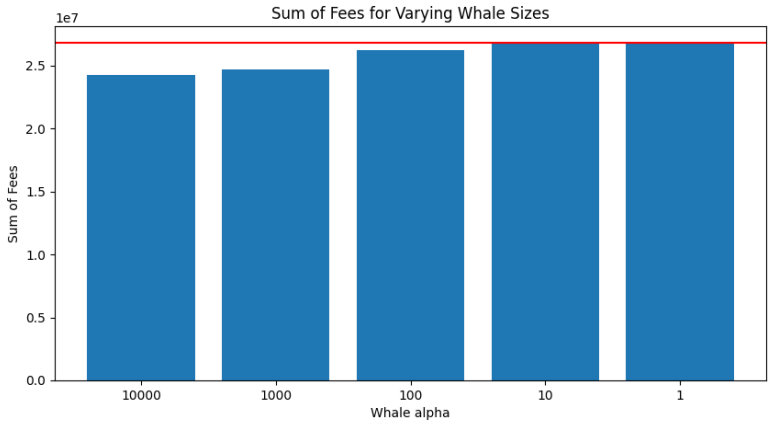

Figure 4: Sum of fees for varying whale sizes. The red line is $0.7Rp$. Recall that α is the weight parameter for traders in the Dirichlet distribution, meaning a larger α creates larger whales. Simulations use $p = \$10$ and $D = 1,500,000,000$.

the expected sum $0.7Rp$ where p was arbitrarily set to \$10.

Furthermore, we can show the advantage whales have over smaller traders in maximizing profits per amount spent in fees. Traders holding a greater share of open interest will generate more profit per dollar spent in fees, as displayed in figure 5. Notice that non-whales will profit around 43% off the amount spent in fees, whereas whales profit close to 50%.

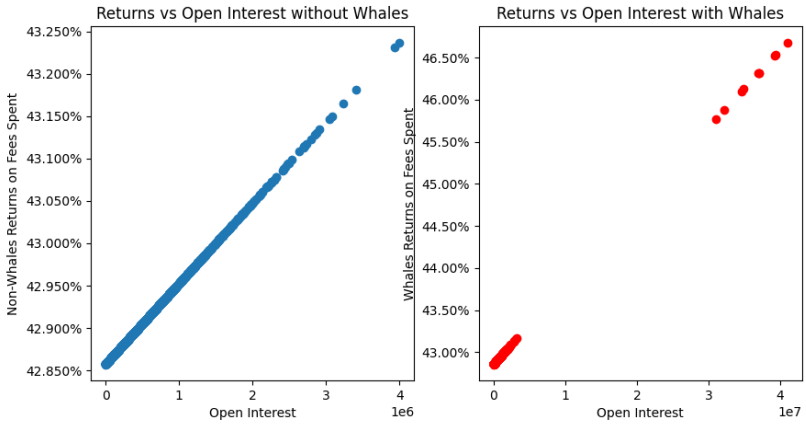

Figure 5: Profit over fees for whales vs non-whales. Whales correspond to the ten points on the upper right corner of the right-hand plot.

2.2.2 Aside on Price Sensitivity

We can briefly examine the price sensitivity of our algorithm in terms of total fees paid. We run Newton's method with $D = 150,000,000$ and 10 whales at $\alpha = 100$ at varying prices. Results are shown in figure 6 and indicate the expected revenue from the trading rewards program.

2.2.3 Aside on Trader Profits

Figure 5 shows the profits that traders currently receive by employing this strategy, before considering slippage. We see that whales make more than smaller traders, however these results yield quite an interesting result: even smaller traders stand to profit upwards of 43% on fees spent (before slippage).

Although accounting for slippage is difficult for whales, we can show that it is negligible for smaller traders. It is common to see spreads on dYdX's `BTC-USD` market as low as \$1 (0.0023%), with a notional size on the best offer of over \$40,000. For small traders who space out their trades, they may be able to enjoy slippage that is negligible compared to the trading fee paid to dYdX, thus making 40+% returns per fee spend viable in reality.

We also do not take into account exchange deposit and withdrawal fees, which may cost a non-negligible amount for small traders. If the trader's strategy requires frequent deposits or withdrawals, then this can quickly eat into their profits. Nevertheless, since this component of the trader's profits depend on their strategy, we cannot speak to precisely how much a trader will need to pay in ethereum gas fees.

It is important to note that returns per fee spend is not the same as return on investment. For a trader with average leverage ratio l, their return on investment

is approximated by the following:

$$r_{oi} = \frac{.7Rpr_f l}{D}, \tag{9}$$

where r_{oi} is the return on equity allocated to the dYdX platform, l is their average leverage ratio, and r_f is the return per fees ($r_f \approx 0.4$); see appendix for the derivation. When $p = \$5.5$ and $D = 1,500,000,000$, $l = 10$, and $r_f = 0.43$, we see that the trader's return on investment is 4.2% per epoch, or 64% per year.

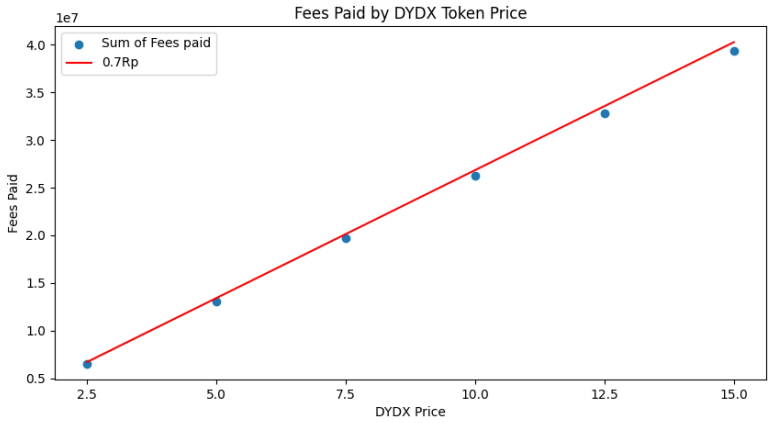

Figure 6: Sum of fees paid according to Nash Equilibrium for varying DYDX token prices. The red line indicates the expected sum of fees paid according to our closed form $0.7Rp$.

2.3 Uniqueness

We can intuit that our pure-strategy Nash equilibrium is unique. Given that all traders have similar concave profit function, it would be unexpected for there to exist multiple Nash equilibria. In fact, we have already developed some reasoning for why our solution is unique when constructing our Newton's method algorithm; we begin at an arbitrary, randomly sampled fee vector and it always converges to the same ratio of fees to open interest. If there existed multiple equilibria, we would expect a different convergence pattern on at least one run of our algorithm.

However, we can be a little more robust with our uniqueness proof using fixed point theorems from the mathematical field of real analysis. We reproduce our uniqueness proof in the appendix. The key takeaway is that our proposed Nash equilibrium is unique, and therefore we expect traders trying to maximize their profit from the rewards program to pay fees according to our optimal fees vector $\vec{F}^*$.

2.4 Takeaways

Let's summarize our conclusions on trader behaviour before the introduction of `stkDYDX`:

1. Traders will try to pay most of their fees at the end of the epoch.
2. For n profit-maximizing traders with total open interest D, there exists a Nash equilibrium for the fees paid, $\vec{F^*}$, where no trader has an incentive to increase or decrease their amount of fees paid.
3. $\vec{F^*}$ is unique.
4. For a flat Dirichlet distribution of open interest, $\vec{F^*}$ can be approximated by a closed-form equation $\frac{f_k}{d_k} = \frac{aRp}{D}$ for all traders k, and dYdX revenue from fees scales linearly with a and p. Recall that $a = 0.7$
5. However, this closed form overestimates fees paid by big traders, and underestimates for small traders depending on the distribution of open interest. Large whales holding significant shares of the total open interest introduce an error in our closed form that can be explored by running Newton's method with varying whale sizes.
6. Markets with large whales will pay a smaller sum in fees compared to markets where open interest is more evenly distributed. For example, at a `DYDX` token price $p = \$10$ and total open interest $D = 1,500,000,000$, an even distribution of open interest will yield approximately \$1M more in fees than if there was the same total interest, but with whales of weight $\alpha_k = 10,000$.
7. Traders with a larger share of open interest (i.e. whales) have a higher ROI compared to smaller traders.

2.5 Introducing the Safety Module

We have shown that there exists a unique pure-strategy Nash equilibrium for the rewards profit maximization problem. Furthermore, we have shown how rational agents can find this optimal strategy, and how they should compute the optimal fees they must pay. We will now analyze how introducing `stkDYDX` to the Cobb-Douglas reward function will impact how traders choose to pay their fees. The new trader score function is:

$$w = f^a \times d^b \times [\max(10, g)]^c \tag{10}$$

where g is a trader's average `stkDYDX` held, and $c = 0.05$. Notice that we have previously split a trader's strategy into two phases. Phase (1) occurs at the beginning of the epoch when traders decide how they will maximize their open interest via, say, highly leveraged market neutral strategies. In phase (2) traders decide how much to pay in fees. We made this distinction because there

is no incentive for traders to pay fees before the very end of the epoch, and it is clear that introducing `stkDYDX` does not change this fact. The introduction of `stkDYDX` only changes how traders will compute the amount to pay in fees at the end of the epoch.

2.5.1 Finding a Nash Equilibrium with `stkDYDX`

In order to simulate a Nash equilibrium with `stkDYDX` we must decide how and how much `stkDYDX` to distribute among traders. We show how traders could optimally decide on how much `stkDYDX` to hold within the rewards ecosystem in the appendix. However, the current state of the `DYDX` safety staking pool does not demonstrate the same optimization behavior as the rewards mechanism, as there are much fewer stakers than traders (2,700 stakers, according to Etherscan), and not all of those stakers are large traders on the protocol.

We instead distribute the total amount of `stkDYDX`, G, similarly to how we distributed total average open interest D, except we establish a floor for each individual g_k at $g_k = 10$ in accordance with the rewards function. We find that our algorithm still converges to an optimal fees vector $\vec{F}^*$.

Therefore, if we can arrive at an adequate estimate for the total `stkDYDX`, G, then we can approximate the optimal fees vector using our algorithm. Furthermore, we argue that the impact of `stkDYDX` on the Nash equilibrium from previous sections is minimal; the exponent on `stkDYDX` is very small and few traders hold significant amounts in `stkDYDX`. Therefore, we postulate that an adequate approximation of G will lead to surprisingly accurate approximations of the optimal fees vector $\vec{F}^*$.

To approximate G we use the dYdX API to get the average `stkDYDX` held in an epoch and multiply by the number of eligible traders in said epoch. We find that total average `stkDYDX` is in the order of tens of millions, and we explore the sensitivity of our Nash equilibrium for varying G in figure 7. Notice that our x-axis is $d_k^{0.28} g_k^{0.05}$ to capture the relationship between d_k and g_k in the rewards function. We argue that rough approximations of G are sufficient for small traders, but for larger traders, small fluctuations in G might have significant impacts on the optimal amount to pay in fees. We argue this sensitivity to G is small due to its relatively small exponent $c = 0.05$.

2.6 Assumptions

1. *All traders are trying to maximize profits from rewards.* This assumption that our model makes is obviously false. However, our model is mostly used as a means to estimate the optimal fees a trader should pay near the end of an epoch, and it accounts for the amount of fees already paid throughout the epoch. Thus, we would expect the true trader score to be lower in practice than what is derived by our mechanism, since not all traders near the end of the epoch are optimizing for their rewards.

2. *All traders agree on a `DYDX` price p.* When rewards-profit maximizing traders pay x USDC in fees in order to receive an estimated y of `DYDX`

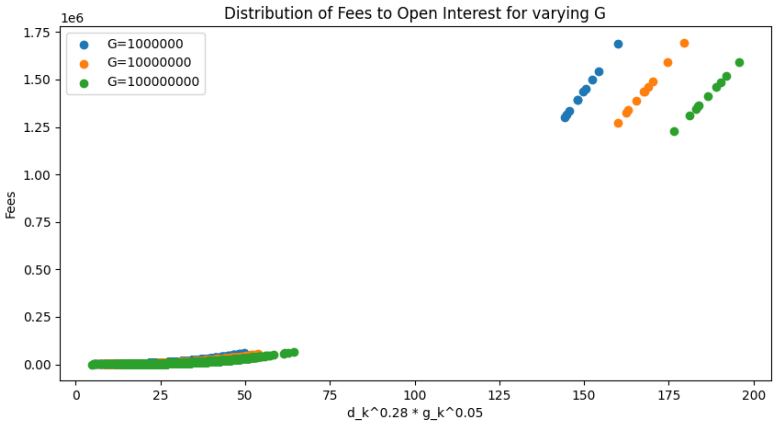

Figure 7: Fees distribution for varying G. Whales correspond to the thirty points on the upper right corner of the plot.

in end of epoch rewards, this is equivalent to locking x USDC into a collateral account and going long an estimated y of `DYDX` futures contracts, with expiry of one week after the epoch end. We assume that traders mark this virtual futures contracts to the `DYDX` price at the end of the epoch. However, since the reward quantity y and end-of-epoch `DYDX` price is difficult to estimate throughout the epoch, risk-averse traders would shade their fee payments lower. Even near the end of the epoch, when y and `DYDX` price are easier to estimate, there is still (at the time of writing) a negative perpetual funding rate for `DYDX`; when perpetuals sell at a discount, we expect that the virtual `DYDX` futures will also trade at a discount. Thus, we have reason to believe that the agreed-upon `DYDX` price used in our model is an over-estimate, and that the observed total trader score will be lower in practice.

3. *All traders are market neutral on* ***`DYDX`*** or *have means to short* ***`DYDX`****.* This assumption, is related to assumption (2), and it is an unrealistic assumption. If not all traders can short `DYDX`, then risk-averse traders will be less willing to pay fees, whereas speculative traders may be willing to pay more in fees than the current `DYDX` price would warrant. This assumption is critical to understanding historical epoch data.

3 Comparison to Historical dYdX Data

We now compare our optimal-fees algorithm with historical dYdX data. We use dYdX's HTTP API to obtain data on `stkDYDX`; we use dYdX's trader rewards

dashboards to find the total number of eligible traders and the total trader score; and we use dYdX epoch reviews to get data on fees paid. Using this data and the closing price p of `DYDX` token the day before the end of each epoch, we run our simulations to compute optimal fees for traders. The data from the dYdX API call is processed in our public code repository, and it can be found in figure 8.

We generate 10 market whales each with weight $\alpha = 100$ and run our algorithm. Notice that epoch 0 occurred before the implementation of the rewards mechanism so our analysis only applies to epochs 1-7. The historical data we gathered along with our algorithm's predictions are displayed in table 1.

epoch	fees	openInterest	stakedDYDX	numTraders	totalTraderScore	close
1	33900000.0	170293617.912	0.0	9979	64760847	20.972
2	64000000.0	196342688.532	0.0	11175	156288754	18.849
3	55000000.0	200980039.716	0.0	8502	136583326	14.181
4	39000000.0	175289739.45	0.0	6448	91997522	7.398
5	24000000.0	164673930.09	20574299.548	4876	56500160	7.415
6	17000000.0	127578968.971	25490720.0	5008	44815330	7.436
7	15000000.0	109582046.721	36520010.0	5347	40647086	4.534

Table 1: Historical trading rewards data.

Our results are displayed in figure 10. Furthermore, the expected versus actual profits can be computed using the total trader score for each trader in each simulation. To explore these, refer to the `utils.py` file in our repository.

3.1 Discussion

We begin with figure 10, which compares the 'predicted' optimal fees from our algorithm vs. the 'actual' observed fees paid by dYdX traders. This graph is by-and-large the most important to understand, as it can be used as a tool for evaluating the merit of our prior game theoretical analysis.

What we observe are two distinct periods: (1) epochs 1-4, and (2) epochs 5-7. We handle each period separately, explaining why these results appear.

In the first period, we observe that the total fees paid massively outpaced the amount of fees expected in epochs 1-4. In the first epoch, we see that the total sum of fees paid is less than the predicted sum of fees paid; this likely arises due to the fact that calculating an optimal amount of fees paid is nontrivial (it took us a month!). In epochs 2-4, we observe a massive over-correction in the total amount of fees paid; this could be for a number of reasons, such as anticipating that most other traders are not optimizing for rewards, or just typical platform demand manifesting in exchange order flow. However, what we find most likely is that traders opened positions with large open interest in order to maximize rewards, and in the process of maintaining these leveraged positions, they overspent fees throughout the epoch due to `DYDX`'s price decline by the end of the epoch. For reference, observe figure 9, which plots dYdX exchange's volume vs. `DYDX` price. This figure demonstrates, on a log-scale, the price evolution of the

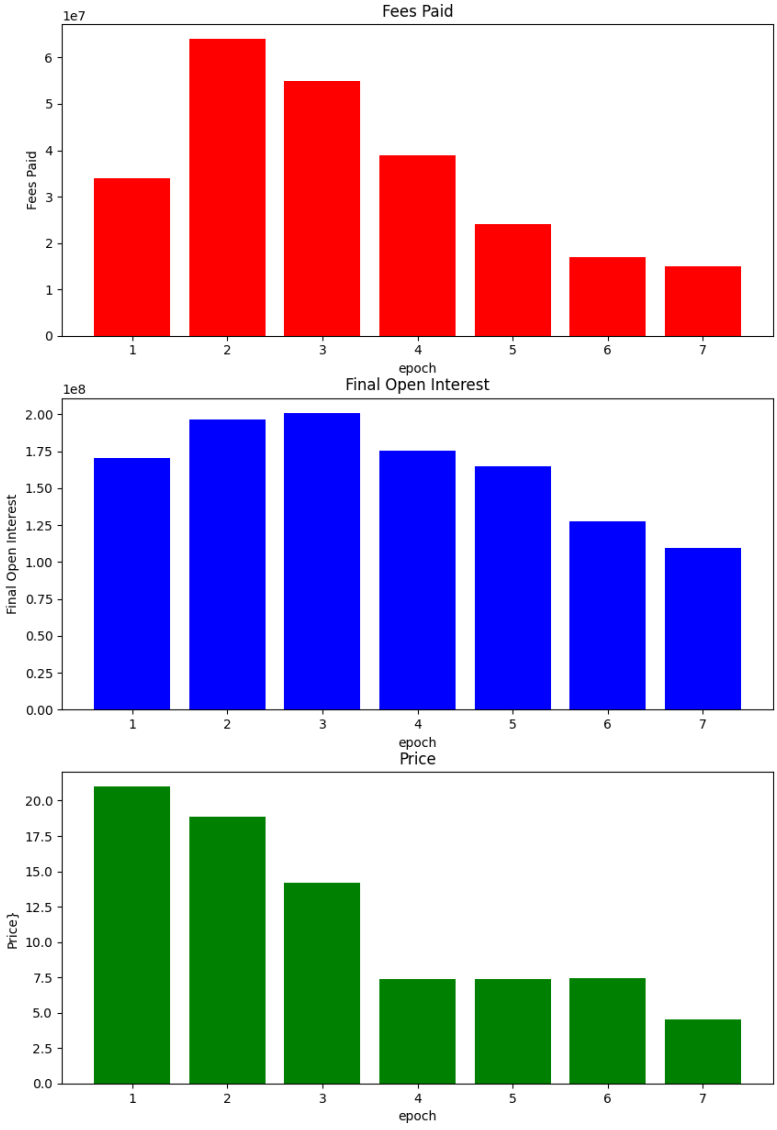

Figure 8: Historical trading rewards data.

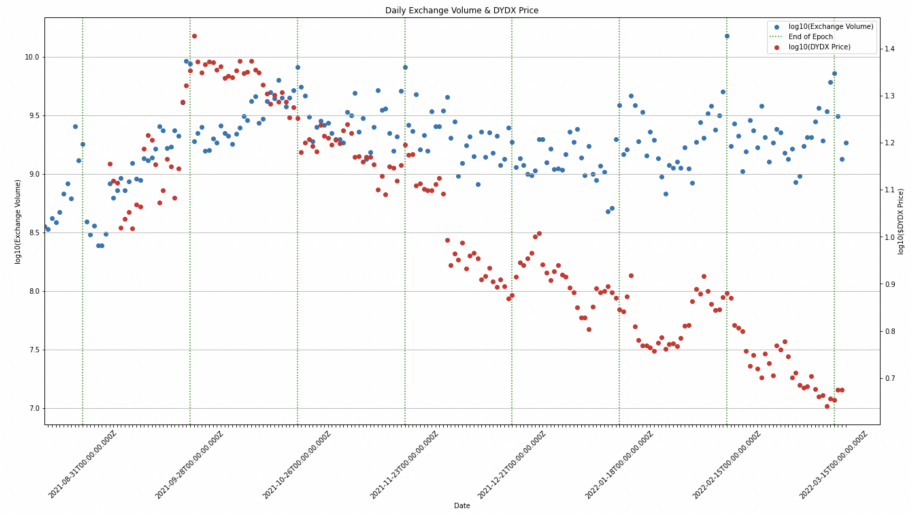

Figure 9: Historical dYdX exchange trading volume and `DYDX` price.

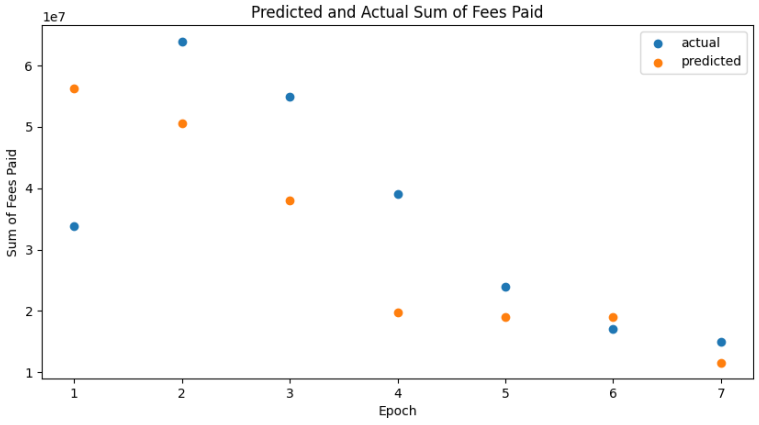

Figure 10: Comparison between actual sum of fees paid and predicted sum of fees paid.

DYDX token throughout the epochs. Recall that in order for traders to maximize their rewards, they must maintain leveraged positions in order to heighten their open interest. The higher the leverage, the more often traders must rebalance their portfolios to avoid liquidation. The more traders rebalance, the more fees they pay throughout the epoch. It is quite likely that risk-neutral traders who valued DYDX rewards at a higher price in the beginning of the epoch than the price received at the end of the epoch would surpass their optimal amount of fees paid throughout the epoch by rebalancing their portfolio too often. This behavior would explain why the observed fees paid in epochs 2, 3, and 4 were higher than that predicted by our algorithm, since our algorithm computes optimal fees paid based on the price of DYDX at the end of each epoch. Traders should thus be cautious when applying our algorithm at the end of an epoch if the DYDX price has dropped since the epoch's beginning; if the price decreased over the epoch, they should pay less in fees than our algorithm suggests.

In the second period, namely epochs 5-7, we see that traders' observed total fees paid closely reflects the amount of fees predicted by our algorithm. For the same reason that traders *overpaid* in fees in previous epochs, we see that traders paid nearly the right amount in epochs 5-7. Take from table 1 the price of DYDX at the end of epochs 4, 5, and 6: \$7.398, \$7.415, \$7.436. These prices are all very near each other, which means that rewards-optimizing traders' optimal open interest calculation in the beginning of the epoch was roughly correct, and thus they were able to pay nearly the optimal amount in fees over the entire epoch. This helps to explain why epochs 5 and 6 had a sum of fees paid nearly identical to our algorithm's computed optimal fee spend. In epoch 7, the DYDX price dropped 40%, and this manifested in a fairly large over-payment in fees for that epoch, where traders paid roughly 22% more than the optimal amount in fees.

Overall, we see that for the small number of epochs where trader rewards were offered, our optimal-fee calculating algorithm gives results resoundingly close to the amount of fees observed by traders in epochs where the DYDX price is close near the beginning and end. In epochs where the total observed fees are higher than what our algorithm predicts, this is consistently accompanied by a decaying DYDX price.

It must be noted that due to the limited number of data points present, we do not claim that our results are statistically significant. Given the fact that we only observed seven data points, it possible that our results are the product of coincidence. However, given this small amount of data on previous epochs, we believe that a majority of trading volume can be eloquently explained by a model composed entirely of rewards-profit-maximizing traders.

4 Survey of Alternative Trader Reward Mechanisms

In this section, we analyze a number of alternative trading rewards mechanisms, giving the benefits and drawbacks of each. Mechanism 0 is an exploration of improving parameters within the existing mechanism, whereas Mechanisms 1-3 are examples of mechanisms that would replace the current mechanism entirely if they were implemented. Furthermore, we analyze a pre-epoch sale into stable coins, which can be used to augment and improve any of the rewards mechanisms listed here. We conclude with suggestions for changes to be made to the dYdX rewards parameters.

4.1 Mechanism 0: Existing Mechanism, with Different Parameters

4.1.1 Overview

Although this paper was motivated by observations of atypical trading behavior related to the rewards mechanism, it is still only fair that we share the merits of the original mechanism as well. As such, in this section we enumerate the current parameters of the dYdX rewards mechanism, we give suggestions for how they could be changed to further dYdX's objectives, and we give the benefits and drawbacks of the current mechanism with these updated parameters.

4.1.2 Analysis of Current Trader Rewards Parameters

The dYdX trader rewards mechanism currently has the following rewards parameters.

1. $R = 3,835,616$: The total reward, denominated in dYdX token, to be distributed at the end of each epoch.
2. $a = 0.67$: The weight in the rewards formula associated with a trader's fees paid.
3. $b = 0.28$: The weight in the rewards formula associated with a trader's average open interest.
4. $c = 0.05$: The weight in the rewards formula associated with a trader's quantity of `stkDYDX`.
5. $T = 28$: The number of days in a trading rewards epoch.

We proceed sequentially, giving suggestions for if and how each of these parameters should be updated.

Parameter R. The current rewards amount is substantial. Our paper finds evidence that the amount of value of rewards, Rp, is roughly linearly correlated with the amount of expected fee revenue for the exchange. Therefore, if the

exchange wants to bootstrap short-term revenue, it can achieve that end by increasing token rewards. However, changing the token distribution would affect the tokenomics for the protocol, which warrants more research than what was provided here. We do not give any recommendations for a change in the R parameter.

Parameter a. The current a parameter composes the greater part of the current rewards formula weighting. As we found in section 2, the total amount of fees paid to the exchange is roughly linear in the a parameter (i.e. $\sum_k f_k \approx \alpha a$, for some value α). Therefore, increasing the a parameter by 10%, for example, would increase the expected fees by approximately 10%. Considering the substantial amount of fees generated by the dYdX protocol already, a 10% increase in fees paid in the past month would have resulted in $1.5M of marginal revenue.

The only problem with raising the a parameter is that we must then lower either the b or c parameters. We now argue that it would be acceptable to lower the b parameter without causing meaningful change to the dYdX protocol.

Parameter b. The b parameter in the dYdX trader rewards formula is meant to incentivize long term growth of the protocol. Upon the implementation of the trader rewards mechanism, we see that open interest jumped nearly a full order of magnitude higher than its point before the rewards (see figure 11). Here, we argue that the b parameter can be decreased, for two reasons: (1) we demonstrate evidence in section 2 that the dYdX trader rewards mechanism has not inspired substantive growth in long-term users, and (2) open interest is correlated, but remains quite high even when the dollar value of trader rewards decreases. Despite the value of rewards dropping to a fifth of its all-time high, open interest remains high. This gives some, albeit not perfect, evidence that open interest would remain high even if it had less weight in the rewards formula. However we do not have a model to quantify how much we expect open interest to drop as a result of decreasing the b parameter.

Even if open interest were to decrease after decreasing the b parameter, we still must ask: why is this a bad thing? Open interest does not contribute to protocol revenues; favoring open interest means the exchange incentivizes high-leverage, long-duration trades, which have a higher likelihood of liquidation than lower average leverage trades; favoring open interest makes it more difficult for rewards-profit maximizing traders to maximize rewards, and thus decreases protocol revenues; and there is no clear relationship between long-term user growth and the incentivization of open interest. Of course, it would be unfair to say that open interest is a useless metric to incentivize. High open interest on the exchange indicates that traders are increasing their position sizes on the platform, which is an important sign of protocol growth. High open interest demonstrates that dYdX is a place where legitimate traders transact.

All things considered, we find that the protocol would see increased revenue by increasing its a parameter, and in order to preserve the sum-to-1 Cobb-Douglas function, we propose that the increase in a come at the expense of the parameter b. In particular, we propose increasing the a parameter to $a = 0.8$ and decreasing the b parameter to $b = 0.15$. We expect that this will result in

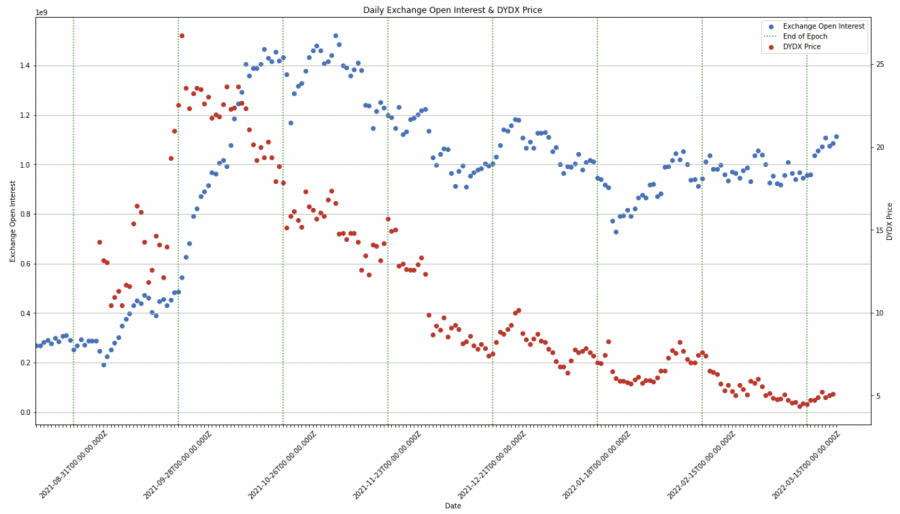

Figure 11: Historical dYdX exchange open interest and `DYDX` price. Open interest is correlated, however its magnitude of change is much more resilient to changes in `DYDX` price.

two changes: first, it will increase the dYdX exchange revenue by roughly 20% (i.e. 0.8/0.67) due to the approximately linear relationship between a and total fees paid, and second, it will increase the dYdX exchange revenue by making it less difficult for traders to optimize their rewards. For epoch 7, we believe this change would have the effect of a \$3.2M+ increase in epoch revenue, and for even higher-fees-paid epochs, such as epoch 2, this change would have had the effect of a \$13M+ increase in epoch revenue.

Parameter c. The safety staking module weight, c, has the effect of incentivizing deposits into the safety staking module. It was added just before epoch 5 (see the commonwealth post) as a way to convert short-term traders into long-term dYdX holders. Currently the `stkDYDX` term plays a relatively insignificant role in trader rewards, and we see no reason to increase it, due to the fact that it already holds approximately 30M `DYDX`, according to Etherscan. Further commentary on the staking module is beyond the scope of this paper, however we currently assume that these reserves with a current USD value north of $150M are sufficiently large and do not require further incentivization. However, we leave it to future research to further optimize the c parameter.

Parameter T. The epoch length parameter has significance not only to the trader rewards mechanism, but also to other modules in the dYdX ecosystem, such as time locks on safety and liquidity staking modules, as well as liquidity provider rewards. Since the scope of this paper is limited to the trader rewards mechanism, we do not consider the downstream effect that changing the epoch length parameter would have on the rest of the dYdX ecosystem. For this reason, we do not give a recommendation for changing the current epoch length.

4.1.3 Mechanism Properties

1. *Drawback:* As shown in section 2, this mechanism incentivizes inorganic trading near the end of the epoch.

2. *Benefit:* This mechanism is reasonably straightforward if traders know the optimal amount to pay in fees. They need only pay excess fees at one point in the epoch, making it quite accessible for *anyone* to attempt profit-maximizing behavior for trading rewards. This makes the trading rewards mechanism relatively fair and accessible. Furthermore, since the rewards mechanism is more accessible, this also contributes to increases in dYdX trading revenues.

3. *Benefit:* This mechanism has been shown to work quite well historically, as it has clearly attracted significantly more exchange volume and open interest. By keeping the same rewards mechanism, there is also less need for existing rewards-profit-maximizing traders to modify their optimization algorithms.

4.2 Mechanism 1: Median of Smaller Trading Intervals

4.2.1 Overview

This mechanism is almost identical to the current trading rewards mechanism, except an individual trader's fee term f is defined as the median fees paid over smaller trading intervals throughout the epoch, rather than the sum of all fees paid throughout the epoch.

Formally, let d be an interval length (e.g. d =1 hour), and let f_i be the amount of fees paid by a trader on the time interval $(id, (i+1)d)$. Then the

new rewards formula would let the fee term for that trader be

$$f = \text{median}_{i \in \{1,2,\ldots,T/d\}}(f_i).$$

This fee term would be used to compute a trader's score according to the same current trader score formula, namely

$$w = f^a \times d^b \times [\text{Max}(10, g)]^c,$$

where w is the trader's trader score, f is fees, a is fee weight (currently 0.67), d is open interest, b is open interest weight (currently 0.28), g is `stkDYDX`, and c is staked `DYDX` weight.

4.2.2 Mechanism Properties

1. *Benefit:* This mechanism makes it easier for the dYdX Protocol to estimate trading revenues, since traders will ramp up fee payments more frequently instead of just once per 28 days.

2. *Benefit:* The engineering work required to implement this solution is minimal, as it only requires computing traders' scores in each of the pre-defined intervals.

3. *Drawback:* This mechanism requires significant trader up time in order for traders to get any rewards at all. If a trader pays fees in fewer than $\frac{T}{2d}$ intervals, they will thus have 0 trader rewards. If d =1 hour, then under this formula, traders must pay fees in at least $672/2 = 336$ of the hour-long intervals throughout the epoch in order to get any rewards. Alternatively, if d =1 day, then under this formula, traders must pay fees in at least 14 of the day-long intervals throughout the epoch in order to get any rewards. A small value for d makes it impossible for infrequent traders to get rewards. A large value for d leads to the same mechanism failure as the current mechanism. For instance, if d =7 days, then traders must still pay fees in at least two intervals to get rewards, yet they can still perform the same end-of-epoch behavior observed today on the smaller intervals, where each week we would expect to see a large ramp up in volume.

4. *Drawback:* Since this mechanism is so taxing on rewards-profit-maximizing traders, it will likely lead to fewer traders optimizing their rewards, and thus lead to lower exchange revenues.

Due to its drawbacks with respect to inaccessible rewards maximization, we find that this mechanism is not a suitable replacement for the current rewards mechanism.

4.3 Mechanism 2: Shorter Epochs Mechanism

4.3.1 Overview

This mechanism is also quite similar to the current mechanism, except with a changed epoch length parameter T. If taken to the extreme, epochs could

be very short (e.g. 100 milliseconds), giving traders an incentive to spend fees throughout the epoch, rather than all at the end.

4.3.2 Mechanism Properties

1. *Benefit:* This mechanism makes it easier for the dYdX Protocol to estimate trading revenues, since traders will ramp up fee payments more frequently instead of just once per 28 days.

2. *Drawback:* There are multiple modules in the dYdX governance stack that use the epoch variable (e.g. liquid staking module, safety staking module, governance time locks). Changing the epoch length is not a trivial change engineering-wise, and it may also have downstream effects on other mechanisms that use the epoch parameter.

3. *Drawback:* This shifts the inorganic volume from occurring in a large way at the end of each 28 day epoch to occurring in a small way at the end of each smaller epoch. This mechanism still does not deter inorganic volume, and in fact as the epoch length gets smaller, the rewards received by infrequent traders also gets smaller. This also means that market makers, who currently need only modify their algorithms near the end of the 28 day dYdX trading epoch, would need to modify their algorithms at the end of each smaller epoch.

Due to the engineering lift necessary to implement this mechanism, as well as its dubious benefits to the inorganic late epoch trading phenomenon, we find that this mechanism is not a suitable replacement for the current rewards mechanism.

4.4 Mechanism 3: Mini-Intervals Mechanism

4.4.1 Overview

This mechanism is similar to the current mechanism, with two additions: (1) it distinguishes between epochs and what we call "trading intervals" where traders earn rewards, and (2) it does not disclose the start/end time of trading intervals until after the epoch has completed. We show here that by adding these components to the current trader rewards mechanism, the exchange can incentivize nearly constant-rate trading volume across the epoch with no impact to overall epoch open interest. This makes it harder for rewards-maximizing trader to optimize their rewards, which has benefits and drawbacks discussed in the "Mechanism Properties" section.

First, we describe how this mechanism distinguishes an "epoch" from a "trading interval". The epoch length is a parameter in dYdX governance smart contracts on which a large amount of behavior depends. Safety staking and liquidity provider rewards also depend on this parameter for their own mechanism. Thus, changing the epoch length has effects on other parts of the dYdX ecosystem. However, there is no requirement that rewards be calculated on an epoch basis. Here we introduce trading intervals, which we define as intervals

of time within an epoch on which trader rewards are calculated. With this abstraction, we can construct a new mechanism where the total trader rewards given (currently $R = 3,835,616$ DYDX per epoch) are split among n equal-sized intervals within the epoch, where n is known by traders before the epoch. Instead of allocating R DYDX to the traders for their cumulative trading across the entire epoch, we may for instance allocate R/n DYDX to traders for their trading in each trading interval in the epoch. While R/n of the rewards would be earmarked for each trading interval at the beginning of the epoch, traders will still only be allowed to claim their rewards at the end of the epoch, as is done with the current mechanism.

But how do we determine which intervals should be trading intervals?

The naive approach would be to do the following: let the interval length be $d = T/n$, where T is the length of the entire epoch, and then make the following intervals:

$$I = [(0, d], (d, 2d], \ldots, (T - d, T]].$$

In this approach, $\frac{R}{n}$ would be allocated for trading in each interval. Although this mechanism would certainly incentivize more steady trading throughout the entire epoch, it would still suffer from rewards-profit-maximizing traders placing trades near the end of each trading interval. In other words, it would take the current end-of-epoch trading problem and spread it across n intervals, but it would not resolve the fundamental issue where rewards-maximizing traders wait until the end of the trading interval to pay fees. This is the problem we observed with "Mechanism 1".

But what if traders did not know when the end of each trading interval was? Consider a similar approach to the naive one above, except for the following adjustment: let r be a uniform random variable sampled from $(0, d]$, and let the trading intervals be

$$I = [(r, r+d], (r+d, r+2d], \ldots, (T-2d+r, T-d+r], (T-d+r \mod T, T+r \mod T]].$$

There are a few noteworthy items worth discussing in this change. First, the value of r must be unknown to traders before the epoch is over, at which point it can be generated via smart contract using a random oracle, such as Chainlink's Verifiable Random Function. Second, the last interval has a peculiar " mod " term in it. This is because technically the "last" interval is partially at the end of the epoch (time $T - d + r$ to time T), and partially at the beginning of the epoch (time 0 to time r). This single wrap-around interval has less semantic interpretability, however it does not affect the nice properties of the mechanism, and it provides a way to include the first $(0, r]$ length of the epoch into the trading rewards. For the visually inclined, the epoch endings can be thought of as tick marks on a clock, and r can be thought of as an angle of rotation applied to the clock.

Using this mechanism, we are able to prove that if the DYDX markets are efficient (i.e. expected price of DYDX at any point in the epoch is simply the current price of DYDX), then the expected cumulative USD nominal value of

rewards allocated by the mechanism by time $t \in (0, T]$ grows linearly in t (see proof in appendix).

While the expected USD rewards allocated by the mechanism are linear in time after the time d, how do we know that the cumulative fees paid (which is a proxy for volume) will be linear in time? Well, it turns out that the fees paid *will not* be perfectly linear, due to the time-value of money; for instance, people would rather pay for an asset that they will receive in 8 days than something they will receive in 35 days. However, since the time interval between fee payment and receiving assets is so small, and risk-free interest rates are approximately 0, we approximate the time value of money to be constant. With this assumption, it becomes obvious that a set of rational traders with the ability to efficiently short `DYDX` will have no preference for what time they short the `DYDX`, since their information about other traders does not change over time, and the expected rewards in each time interval of length τ are the same as any other time interval of length τ. In other words, there is no meaningful distinction between any of the time intervals of length τ within the epoch, and thus there would be no reason for strategic changes in the amount paid at those intervals. This mechanism would thus incentivize an equal amount of payment throughout the epoch.

Another way to conceptualize why this mechanism incentivizes constant rate of fee payment is through a proof by contradiction. For any $\tau \in (0, T]$, consider two time intervals $A = (a, a + \tau]$ and $B = (b, b + \tau]$, and the rewards that will be allocated during those time periods, R_A and R_B. The expected rewards to be paid out in time interval A, as calculated at the beginning of the epoch (assuming $E[p(t)] = p(0)$ for all $t \in (0, T]$), is

$$E[R_A] = \frac{R}{T} \int_a^{a+\tau} E[p(t)]dt = \frac{Rp(0)\tau}{T}.$$

Similarly for the interval B,

$$E[R_B] = \frac{R}{T} \int_b^{b+\tau} E[p(t)]dt = \frac{Rp(0)\tau}{T}.$$

Therefore, at the beginning of the epoch, $E[R_A] = E[R_B]$. The expected rewards allocated in time intervals A and B are the same.

Although the expected rewards are the same for any two equal-length intervals in the epoch, it does not follow that there is a unique Nash equilibrium strategy for trading across the entire epoch. This is largely due to the fact that there are infinitely many repeated "stage games". Trading on interval A is an example of a stage game, and we can use the results from section 2 to show that it has a unique Nash equilibrium, since it is equivalent to an epoch with total length T/n and total reward R/n.

However, there may be opportunities for strategic behavior across games. For instance, trader i might overpay fees by an egregious amount in early trading intervals so as to lower the rewards of other traders, and when they respond with lower fees paid in future epochs, trader i might lower their fees again; by this

strategy, trader i might find a way to pay an overall lower amount in fees and get the same amount (or more) in rewards than they would by adhering to the stage game equilibrium throughout the entire epoch. We do not provide an example of such a Nash equilibrium here, but this would be a natural way to extend the paper to learn more about this mechanism.

4.4.2 Mechanism Properties

1. *Benefit:* Mechanism incentivizes constant rate of fee spend and volume over the entire epoch.

2. *Benefit:* Estimating trading rewards becomes much easier. This will allow dYdX Trading to incorporate better estimates for traders' expected rewards into the user interface.

3. *Benefit:* The complexity required to implement this is very small, as it only requires the creation of a smart contract that uses a random oracle to generate the trading interval start/end times and a slightly modified query to compute traders' rewards with those intervals.

4. *Slight Drawback:* It is slightly more complicated for dYdX Trading to implement this change to the rewards formula. Still, the only new elements of this mechanism that requires thoughtful implementation is the random noise parameter r and the multi-interval trader rewards calculation.

5. *Benefit or Drawback:* Mechanism yields higher rewards for traders who split their trades into multiple, smaller trades. One can check that the value of placing two orders of half the size, but in different trading intervals, can yield a higher trading reward. When taken to the extreme, one could space out trades across an entire epoch to earn on the order of 4x more in rewards than they would get from buying once in the middle of the epoch. This may be seen as a benefit, however, since it encourages users to engage with the protocol more frequently. This multiplier decreases as we increase trading interval length (d) or increase the fee weight in the trader rewards formula (a).

6. *Drawback:* When we relax the assumption that rewards-profit-maximizing traders have the ability to short the rewards token, then our model would require that those traders are risk neutral *and* believe the rewards token is trading at or below what its value will be at distribution time. If we relax these assumptions, then the nice properties of this mechanism, such as incentivizing constant volume across the epoch, break down due to traders' risk aversion to holding the rewards token. However, this can be alleviated by a pre-epoch token sale.

7. *Drawback:* By requiring rewards-profit-maximizing traders to intersperse their trading throughout the epoch, we make it less accessible for less sophisticated traders to maximize their rewards. Thus, we would expect

to observe that although this mechanism incentivizes a continuous rate of fees paid, it would lead to rewards going to an even more concentrated set of token holders than the original trader rewards mechanism.

8. *Drawback:* This mechanism is composed of infinitely many stage games, and it is possible that it thus has multiple Nash equilibria.

Due to the difficulty that this mechanism imposes on rewards-profit-maximizing traders and the lack of clarity for Nash equilibria strategies, we do not recommend this mechanism as a replacement to the current rewards mechanism.

4.5 Pre-Epoch Token Sale

4.5.1 Overview

Currently, dYdX distributes 3,835,616 `DYDX` token at the end of each epoch. As evidenced before, a dominant strategy in the current trading rewards regime is to wait until the end of the epoch to pay fees, and one of the most important factors for determining the amount of fees paid in the end of the epoch is the price of `DYDX` near the end of the epoch. By virtue of `DYDX`'s price volatility, this makes it challenging to estimate the dollar value of rewards that will be given in each epoch, and thus risk-averse traders would be less inclined to participate in rewards maximization behavior. This is especially true for traders who do not have the means to short `DYDX` token, or who simply do not want to go through the hassle of shorting the token. Distributing rewards in `DYDX` presents a barrier to entry for rewards-maximizing traders.

A solution to this is to conduct a sale of 3,835,616 `DYDX` into stable assets before the beginning of each epoch. The revenue generated through this sale would be publicly announced as the amount of trading rewards available, denominated in stable assets, such as USDC.

Below we give more specifics on the mechanisms that can be used to conduct this token sale.

4.5.2 Dutch Auction Token Sale

One option would be to conduct a Dutch auction to sell R `DYDX` tokens. Here we briefly describe a Dutch auction.

In a Dutch auction, the auctioneer begins at a price p_{high} above what any bidders would be willing to pay; then, at scheduled time intervals, the auctioneer lowers the price. When the auctioneer's announced price meets the price at which a bidder values an item, the bidder i places a bid for n_i of the auctioned good; this bid is public information. This process continues until the sum of bid sizes reaches the total amount to be auctioned off, i.e. $\sum_i n_i = R$, or the bidding price reaches a minimum "reserve price" set by the auctioneer, p_r. The price at which the last bid was placed, p^*, is the price paid by all auction participants: player i pays a total of $p^* \cdot n_i$ to the auctioneer, and the auctioneer allocates n_i of the good to player i. This is a simple Dutch auction.

Dutch auctions provide a simple mechanism that makes it easy to orchestrate large token sales on-chain. However, a trader's optimal Dutch Auction bidding strategy can be complicated, as it involves predicting how other traders will bid. Furthermore, it has been observed in previous crypto Dutch Auctions that price falls shortly after the auction. This behavior can be understood when we consider the position of a market maker (i.e. liquidity provider): if a public Dutch Auction's clearing price is below the current market price, then liquidity providers will remove their buy offers before the auction participants sell; this leads to lower liquidity, which in conjunction with a collection of auction winners who are willing to sell, leads to large negative price impact on the auctioned token. The other case, where the Dutch Auction's price clears above the current market price, is not a realistic result either: why would a bidder offer more to buy through the Dutch Auction mechanism than through existing spot markets?

It seems that Dutch Auctions do not constitute a great mechanism for conducting large token sales, due to their impact on liquidity / market stability for the token being auctioned.

4.5.3 OTC Desk Sale

A more viable option for DAOs is to conduct a sale through existing OTC desks. An OTC sale would entail the following steps: first, the DAO gives the sale token to the OTC desk, with the agreement that the OTC desk sells the token over a period of time; second, the OTC desk sells the token over the agreed period of time, with the intention of not moving the market; finally, the DAO and the OTC desk agree upon a publicly-verifiable time-weighted average price (TWAP) over the sale period, and the OTC pays back the DAO the value of the token, as calculated by the TWAP, aside from a fee of 0.25%-1%.

This method of sale is the current best practice for large sellers to get a reasonable execution price, and we believe it is currently the best way for dYdX to conduct a token sale. While this form of sale does require trusting a centralized OTC desk, it yields the best execution price for the DAO.

4.5.4 Mechanism Properties

1. *Benefit:* Rewards-profit-maximizing traders for whom shorting the `DYDX` token is an impediment will have no longer have a barrier to maximizing their rewards.

2. *Benefit:* The exchange will be able to predict its volume, and thus revenue, for the next 28 days with significant accuracy.

3. *Drawback:* Selling 3,835,616 `DYDX` is not trivial. This could likely be done through an OTC desk, but at some cost; it could also be done through a Dutch auction, however this mechanism has historically led to deflated token prices following the auction (e.g. the algorand auction).

4.6 Mechanism Recommendations

1. Keep the current rewards mechanism (i.e. Mechanism 0). Although Mechanism 3 provides a solution to the elevated end-of-epoch trading problem, it makes rewards optimization infeasible for less sophisticated, lower-frequency traders. This would lead to lower exchange revenues and a denser concentration of rewards recipients. The current rewards mechanism, despite its high fees near the end of each epoch, is the most accessible option we have seen so far.

2. Increase the fee weight parameter to $a = 0.8$, and decrease the open interest weight parameter to $b = 0.15$. This change alone is expected to increase epoch revenues by approximately 20%, which currently comes out to over $3M per epoch.

3. Conduct a sale of 3,835,616 `DYDX` token into stable coins before each epoch begins, and give all of the proceeds as stable coin rewards. This approach can be used to open up `DYDX` supply to the entire market to purchase `DYDX` at a discount, rather than just traders. It is also expected to increase exchange revenues, since risk-averse traders have more guarantees about the market value of stable coin rewards. This will also open up rewards maximization to traders who do not have the means to short the `DYDX` token, thus making rewards more accessible to less sophisticated traders.

5 Conclusion

Since the introduction of dYdX's rewards mechanisms, the exchange has seen a meteoric rise in volume and open interest. Of course, with these economic incentives also come profit maximizing traders, and trading behavior in previous epochs has demonstrated that there is no shortage of traders willing to trade to maximize their trading rewards. This paper was conceived with three objectives: (1) prove results about the nature of the dYdX trader rewards mechanism, (2) 'open source' this rewards profit maximization problem to make it possible for any motivated trader to algorithmically optimize their strategy, and (3) give recommendations for how the mechanism can be improved.

In section two, we tackle objectives (1) and (2). First, we argue that there is an incentive for traders to wait as late as possible into the epoch to pay fees, explaining the observed behavior of amplified late-epoch trading. Next, we provide an algorithm to find the optimal amount of fees paid for each trader using Newton's method, and we prove that it is a Nash equilibrium for all traders to pay that amount in fees. Then we show that this Nash equilibrium is unique, asserting that any rational rewards-maximizing trader would certainly pay fees according to the results of our algorithm. We then show that these results apply to the current trader rewards formula with `stkDYDX` as well as the original rewards formula. We provide all of the math for finding the optimal fees paid in this paper, and we provide open-source code for traders to find

their own optimal amount of fees paid as well. The return on the rewards-profit maximizing strategy is substantial: upwards of 64% annualized return on capital deployed, all compatible with a price-neutral strategy.

In section three, we compare our game theoretical model to real-world historical exchange data. We find that it is *not* valid to assume that traders pay a negligible amount in fees in the beginning of the epoch. However, we also find that under certain mild conditions, the trader rewards incentives alone are sufficient to explain all of the trading volume on the dYdX exchange in more recent epochs. This prompts a chilling question: what would happen if rewards were to stop, or become less valuable? This is an extremely important question, nevertheless it is not within the scope of this paper to speculate on its answer.

In section four, we provide an overview of alternative trading rewards mechanisms. Most trivial changes, such as taking the median trader score on an hourly basis or shortening the length of dYdX's epochs, do not solve the fundamental reason for elevated end-of-epoch trading. Only when we make it impossible for traders to know the precise end of a trading interval will they stop artificially trading to increase their rewards. This is the crux of "Mechanism 3: Mini-Intervals". However, we find that by incentivizing a continuous rate of volume throughout the epoch, we give an advantage to sophisticated traders with consistent exchange up-time. We thus settle for the existing mechanism, albeit with modified Cobb-Douglas weight parameters: $a = 0.8$ and $b = 0.15$, which should increase fee revenues by upwards of \$3M per epoch. In addition to the rewards mechanism parameter changes, we also recommend conducting a sale of `DYDX` tokens before the beginning of each epoch, then using the proceeds of the sale to distribute rewards; this is meant to make trading revenue more predictable, as well as to offer a more fair distribution of `DYDX` near the market rate.

Through this research, we have demonstrated a way for any motivated dYdX trader to maximize their rewards, we have proved important game theoretical properties of the trader rewards mechanism, and we have shown how the dYdX community can improve the rewards mechanism to make it more profitable for dYdX and more fair for all stakeholders.

6 Extensions

Here we outline some potential extensions to this research.

1. **Better approximations of G.** What information could the dYdX foundation provide to smooth the trading process. Total average stakedDYDX? More specifically they could provide $\sum_n (d_k^{0.28} \times g_k^{0.05})$?

2. **Distributions of D and G.** What probability density functions (pdfs) could we use to better simulate the distribution of open interest and stakedDYDX?

3. **Robustness of Volume without Liquidity Mining.** One of the most important results of this paper is that there is evidence that *all* of the dYdX protocol's trading volume can be explained by the trader rewards mechanism. If dYdX were to stop providing trader rewards at all, how much volume would stick around?

4. **Accounting for Slippage.** We ignore slippage cost in this paper, since it is so small. To what extent would taking into account slippage affect our results?

5. **Staking Module Rewards Parameter, c.** We do not dive into the nuance of the safety staking module in this paper, and thus do not propose a change to that parameter. How can we make this parameter more optimal? And does optimizing this parameter require forming an opinion on the safety staking module?

6. **Extending Fee Optimizations to New Mechanisms.** We showed how a trader can compute their optimal fees under the existing mechanisms, however we did not extend this research to any of the new mechanisms that we proposed. What properties can we demonstrate for each of those mechanisms? Can we show that there is a unique Nash equilibrium for them as well? Or that there are multiple Nash equilibria?

7 Appendix

7.1 Proofs

7.1.1 Mini-Intervals Mechanism

Claim: all times in the epoch time range $[0, T)$ have equal probability density of being a trading interval end time.

Proof: Let d be the interval length, and T be the epoch length, and let t_1, t_2 be any two points in $S = (0, T]$.

We know that t_1 must lay in some time interval of the form

$$I_i = ((i-1)d \mod T, id \mod T],$$

for some $i \in \mathbb{N}$, so t_1 is a trading interval ending point if and only if it is the ending point of the ith trading interval. We also know with probability 1 that there is exactly one trading interval ending time $t_1' = \sup I_i$, since there is one ending time at $r, r+d, r+2d \ldots$ and $r \in (0, 1]$. The ending time is of the form $t_1' = r + (i-1)d$. Clearly $t_1' \sim U((i-1)d, id)$, since $r \sim U(0, d)$. Therefore, the probability density that $t_1' = t_1$ is the same as the probability density of t_1 in $U((i-1)d, id)$. Thus, since t_1 is a trading interval ending point if and only if it is interval i's trading interval ending point, it has probability density $1/(id - (i-1)d) = 1/d$ of being a trading interval ending point.

We can apply the same argument to t_2 to find that its probability density function for being the trading interval ending point is $1/d$. Thus, for any two points in S, the probability density of being a trading interval ending point is $1/d$. This is the desired result.

Claim: If the DYDX market is efficient, then the expected cumulative USD nominal value of rewards allocated by the mechanism by time $t \in (0, T)$ grows linearly in t.

Proof: Let n be the number of trading intervals; let R be the total amount of DYDX given in rewards over the entire epoch; let $p(t)$ be the expected market-clearing DYDX ask price at time t; and let T be the length of the epoch. At the end of each trading interval, the mechanism allocates $\frac{R}{n}$ of DYDX, and if this allocation happens at time t it has $\frac{Rp(t)}{n}$ of nominal US dollar value. Also, it is known that $n = T/d$.

Let $E[N(t)]$ represent the expected amount of DYDX rewards allocated by the mechanism by time t.

For a small Δt, we have that

$$\begin{aligned}
E[N(t+\Delta t) - N(t)] &= E[N(t+\Delta t)] - E[N(t)] \\
&= \Pr(\text{rewards are allocated between } t \text{ and } t+\Delta t) \cdot \frac{Rp(t)}{n} \\
&= \frac{\Delta t}{d} \cdot \frac{Rp(t)}{n} \;\; // \text{ see 'equal PDF of end of interval' proof} \\
&= \frac{\Delta t}{d} \cdot \frac{Rp(t)}{T/d} \\
&= \Delta t \cdot \frac{Rp(t)}{T}.
\end{aligned}$$

Therefore, we find that

$$\begin{aligned}
E[N'(t)] &= E\left[\lim_{\Delta t \to 0} \frac{N(t+\Delta t) - N(t)}{\Delta t}\right] \\
&= \lim_{\Delta t \to 0} \frac{E[N(t+\Delta t) - N(t)]}{\Delta t} \\
&= \frac{Rp(t)}{T}.
\end{aligned}$$

If the DYDX market is efficient, then the expected market-clearing DYDX ask price at time t is given by $p(t) = p(0)$ (assuming that the time-value of money is negligible). Suppose this were not the case, then a rational trader could profit by trading DYDX at time 0, which would contradict the efficient market antecedent.

Therefore, when we assume efficient markets, we find that for $t \in (0, T]$,

$$E[N'(t)] = \frac{Rp(0)}{T}.$$

This is not a function of t at all! Thus, it is trivial to find the DE's solution as follows

$$\begin{aligned}
N(t) - N(0) &= \int_0^t \frac{Rp(0)}{T-d} dx \\
&= \frac{Rp(0)}{T} t.
\end{aligned}$$

Since $N(0)$ is 0 (no rewards are allocated within the first moment of the epoch), we have that the cumulative nominal dollar value of DYDX given by time $t \in (0, T]$ is

$$N(t) = \frac{Rp(0)}{T} t.$$

This is the desired result.

7.1.2 Nash Uniqueness Proof

We can prove the uniqueness of our Nash equilibrium using the Banach fixed-point theorem [3]. We will show that our algorithm creates a contraction map [4] g which admits exactly one fixed point, our Nash equilibrium. Using the Banach fixed-point theorem to prove the uniqueness of a Nash equilibrium is a common exercise, and in fact can be used to show that for any twice-differentiable function in a convex metric space for which Newton's method converges, there is a unique fixed point [5] [6] [7].

Define our metric space $(\mathbb{R}^n, d)$ as the Euclidean space with the Euclidean distance metric, where a vector $\vec{F} \in \mathbb{R}^n$ corresponds to our fees vector, and our metric function d is defined by the Euclidean norm, or equivalently, the root mean squared error between two fees vectors. Proofs that the Euclidean space with the Euclidean norm is complete are readily available online, so we will take that for granted. We must now show that the Newtonian update function from Newton's method is a contraction map on $\mathbb{R}^n$.

A contraction map is defined as any function $g : \mathbb{R}^n \to \mathbb{R}^n$ such that there exists a positive number $q < 1$ where $\forall \vec{F_x}, \vec{F_y} \in \mathbb{R}^n$:

$$d(g(\vec{F_x}), g(\vec{F_y})) \leq q d(\vec{F_x}, \vec{F_y}) \tag{11}$$

Notice that it would be sufficient to find and prove that our Newtonian update function g is Lipschitz continuous with Lipschitz constant $0 < K < 1$. However, for an n-dimensional function in an **unbounded** open set Ω, it quite troublesome to analytically determine a finite Lipschitz constant K. Instead, we can limit our metric space to a sufficiently small neighbourhood of $\vec{F}^*$, a fixed point of g in $\mathbb{R}^n$. That is, instead of asserting g is a contraction map in all of $\mathbb{R}^n$, we can take a sufficiently small subset of $\mathbb{R}^n$ wherein our fixed point lies, and show g is a contraction map within this subset. Intuitively this is sufficient since there is a finite amount of capital any trader can pay in fees and can be encapsulated by our choice of δ.

Define our neighbourhood as $N_\delta(\vec{p}) = \{\vec{F} \in \mathbb{R}^n | d(\vec{F} - \vec{p}) \leq \delta\}$. We will show that for sufficiently small $\delta > 0$, g maps $N_\delta(\vec{p})$ into itself and is a contraction on metric space $N_\delta(\vec{p})$. It suffices to show that for such a $\delta > 0, ||\nabla g(\vec{F})|| \leq K$ in $N_\delta(\vec{p})$. Notice that:

$$g(\vec{F}) = \vec{F} - \frac{\vec{P}'(\vec{F})}{\vec{P}''(\vec{F})} \tag{12}$$

$$\nabla g(\vec{F}) = \frac{\vec{P}'(\vec{F})\vec{P}^{(3)}(\vec{F})}{\vec{P}''(\vec{F})^2} \tag{13}$$

[3] https://en.wikipedia.org/wiki/Banach_fixed-point_theorem
[4] https://en.wikipedia.org/wiki/Contraction_mapping#Locally_convex_spaces
[5] https://www2.math.upenn.edu/ kazdan/508F10/palais.pdf
[6] https://pages.cs.wisc.edu/ sifakis/courses/cs412-s13/lecture_notes/CS412_29_Jan_2013.pdf
[7] https://schoolbag.info/mathematics/advanced/18.html

The proof becomes very simple. Notice that by the definition of our fixed point: $g(\vec{F}^*) = \vec{F}^*$, so $\vec{P}'(\vec{F}^*) = 0$. It follows from equation (9) that $\nabla g(\vec{F}) = 0$. By the evident continuity of ∇g, it must be that for any $K > 0$, $\nabla g(\vec{F}) \leq K$ for $g : N_\delta(\vec{p}) \to N_\delta(\vec{p})$. Hence, there must exist δ that ensures g is a contraction map in the neighbourhood of our fixed point $\vec{F}^*$. By the Banach fixed point theorem: A contraction mapping $T : X \to X$ admits one unique fixed-point x^* in X. Hence, our fees vector $\vec{F}^*$ is a unique Nash equilibrium in $g(\vec{F}^*) = \vec{F}^*$.

Notice that the key assumption was creating a sufficiently small neighbourhood using $\delta > 0$. This does not undermine the proof that g is a contraction map, and in fact we could use any finite $\delta > 0$ in our proof above.

7.1.3 Allocating stkDYDX

Let's begin by examining the optimal allocation of capital between open interest and stkDYDX. Denote l as your average leverage ratio throughout the epoch, and M as your total capital available to put toward a rewards profit maximization strategy at the beginning of the epoch. Then our optimization is to maximize:

$$(lx)^{0.28} \times \max(10, M - x)^{0.05}, \tag{14}$$

where x is the amount of equity in dYdX trading protocol, as opposed to staking (note that average open interest $d = lx$). We can easily compute what the optimal allocation of x is by taking the derivative of the above expression. Let's begin by considering the case where $g > 10$:

$$\frac{d}{dx}((lx)^b \times (M - x)^c) = 0 \tag{15}$$

Using the product rule and solving for x:

$$x = \frac{b}{c + b}M \tag{16}$$

We can then compute the exact initial capital required for $g > 10$ to be an optimal allocation:

$$(lM)^b \times 10^c = (l\frac{b}{c + b}M)^b \times (M - \frac{b}{c + b}M)^c \tag{17}$$

Solving for M:

$$\begin{aligned} M^*(b, c) &= \frac{10(b + c)}{c} \times (\frac{b}{b + c})^{-\frac{b}{c}} \\ &\approx 166 \end{aligned} \tag{18}$$

Hence, for $M > M^* \approx 166$, one should allocate stkDYDX according to equation 11, otherwise one should hold no stkDYDX.

7.1.4 Approximate Return on Deployed Capital

Let us use the closed form approximation from equation 6 and the result of return-on-fees-spent to approximate an investor's return on deployed capital to this strategy. Let r_{oi} be the return on deployed capital, l the trader's average leverage ratio, c the trader's account equity, r_f their return on fees spent, R the total `DYDX` rewards, p the price of `DYDX`, D the total open interest, f the trader's total fees spent, d the trader's open interest, and P the trader's profit from fees spent.

We know the trader's profit from fees spent is is given by $P = r_f f$. From equation 6 we know that $f = d\frac{0.7Rp}{D}$. Also by the definition of the average leverage ratio, we have $d = lc$. Thus $f = (lc)\frac{0.7Rp}{D}$. So profit is $P = r_f(lc)\frac{0.7Rp}{D}$. Thus, total return on capital deployed is

$$\begin{aligned} r_{oi} &= \frac{P}{c} \\ &= \frac{r_f(lc)\frac{0.7Rp}{D}}{c} \\ &= \frac{0.7Rpr_fl}{D}. \end{aligned}$$