Liquidity Threshold

December 28, 2021

1 Calculating the Liquidity Ratio Threshold for Parallel Swap

2 Problem Definition

- I have a pool for tokens A and B, with x of token A and y of Token B
- The pool is a Constant-Product Market Maker (CPMM) where $x \times y = k$. Here, k is a constant.
- Fees are a percentage of the amount traded
- We are concerned about the risk of failing transactions due to slippage in a lower liquidity pool during a parallel swap.
  - When performing a parallel swap, if one of the swaps for one or more of the parallel pools fails (due to slippage), the whole transaction reverts.
  - Risk can be assumed as a security vulnerability for a DOS attack
  - A bad actor may use slippage in a low liquidity pool to prevent fellow investors from completing trades
  - The bad actor could have a variety of motivations, but one possible motivation is to prevent other investors from taking advantage of arbitrage opportunities and lower prices in certain pools.

3 Assumptions

- $x \sim y$
- $k \sim x^2 \sim y^2$
- Arbitrage opportunity: price mismatch δ will be the same order of magnitude of the fee percentage
- Arbitrage profit will be on the order of 10% of $\delta \times \sqrt{k}$ (based on x or y, not k)
  - Due to fees
  - Due to the market maker, the *average* price difference will be half of the initial price difference
  - This ratio (here, assumed 10%) is defined as a variable ϵ that must lie between 0 and 100%.

3.1 Other pool

- $x' \times y' = k'$
- If the cost of a DOS attack $\sim$ the anticipated profit of arbitrage or greater, it will be unfavorable to attempt the attack.

- Each iteration of the DOS attack must move the price in the small pool by the slippage tolerance
- Cost $= \sqrt{k'} \times t \times \rho' \times n$ where t is the slippage tolerance and n is the number of iterations

3.2 Result

- if Expected Arbitrage profit = cost of DOS attack:
- $\epsilon \times \delta \times \sqrt{k} = \sqrt{k'} \times t \times \rho' \times n$
- $\frac{\sqrt{k'}}{\sqrt{k}} = \sqrt{\frac{k'}{k}} = \frac{\delta\epsilon}{t\rho' n}$
- $\frac{k'}{k} = \frac{\delta^2\epsilon^2}{t^2\rho'^2 n^2} \approx \frac{\epsilon^2}{t^2 n^2}$

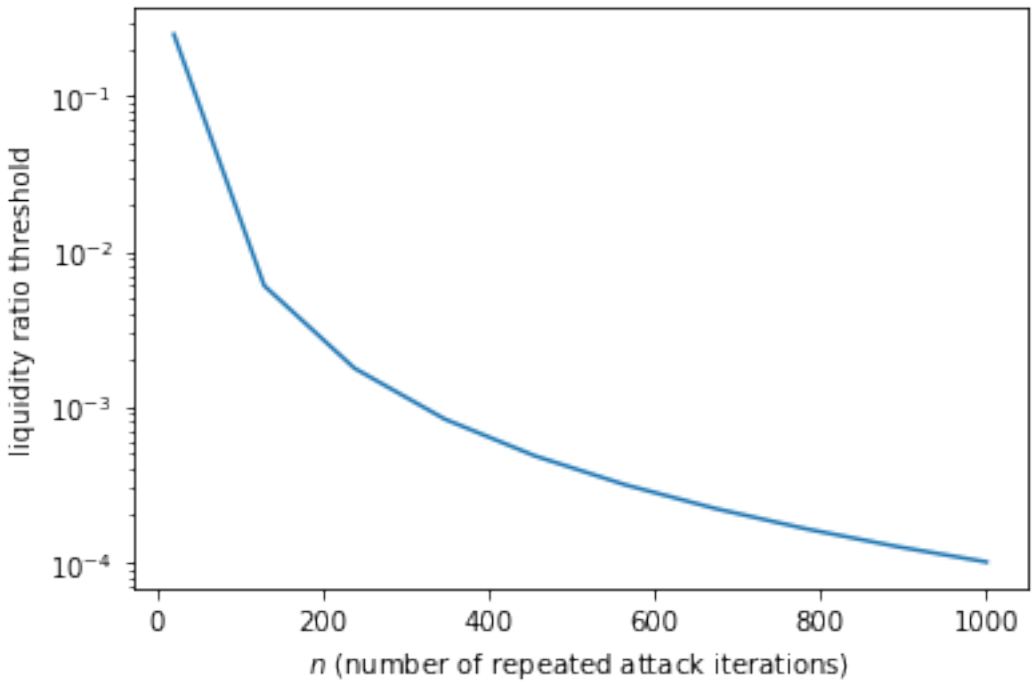
[3]:
```
import numpy as np
import pandas
import matplotlib.pyplot as plt

ns = np.linspace(20, 1000, 10)

def liquidity_ratio(epsilon, t, n):
    return ((epsilon) / (t * n)) ** 2
```

[4]:
```
plt.semilogy(ns, liquidity_ratio(0.10, 0.01, ns))
plt.xlabel('$n$ (number of repeated attack iterations)')
plt.ylabel('liquidity ratio threshold')
```

[4]:
```
Text(0, 0.5, 'liquidity ratio threshold')
```



4 Example

- $x \sim y \sim 10^6$
- $k \sim x^2 \sim y^2 \sim 10^{12}$
- $\rho \sim 0.1\% - 1.0\%$
- $\delta \sim 1\%$
- $\epsilon \sim 10\%$
- $t = 1\%$
- Arbitrage profit ~ 1000

4.1 Solution

- Want to limit liquidity k' to prevent DOS attack
- Use the formula $\frac{k'}{k} \approx \frac{\epsilon^2}{t^2 n^2}$
- With $\epsilon = 10\%$ and $t = 1\%$, $\frac{k'}{k} = \frac{100}{n^2}$

n	k'/k	%
10	1	100%
100	1e-2	1%
316	1e-3	0.1%
1000	1e-4	10^{-2}%
10000	1e-6	10^{-4}%
100000	1e-8	10^{-6}%

Here, we can set the liquidity ratio threshold based on the expected number of subsequent attack iterations. From the table, we can see that a threshold of 0.1% will be sufficient against repeated DOS attacks of up to 316 subsequent iterations.

5 Caveats

- Disclaimer: this heuristic justification is one consideration for the threshold value, apart from real-world testing. If experience in the market shows this threshold value to be too high or too low, it can be updated as part of the front-end.
- This model does not consider the threat vector of an actor wishing to disrupt trades without immediate economic incentive. * That is, we assume here that the bad actor is disrupting trade for some immediate arbitrage benefit.
  - We do not consider an actor who chooses to engage in repeated DOS attacks at cost to himself, for the potential motives of decreasing confidence in the Ref project or for business competition purposes. However, even for such an actor, we can estimate the total cost of engaging in such an attack for a significant amount of time as a function of the liquidity ratio threshold and the absolute liquidity of a given set of pools. This would set a bound on the minimum required financial resources for such an actor.
  - As with the case we considered, the threshold could be re-evaluated in light of real market activity.