1 TWAP Oracle Manipulation Derivation

We want to determine the number of ticks τ that must be crossed to manipulate the TWAP Oracle by δ percent.

1.1 Basic TWAP Oracle Definition

In Uniswap v3, the TWAP oracle doesn't actually track the price. It tracks the time weighted tick of the pool. Let the tick at time t be τ_t

Uniswap v3 creates an oracle update whenever the tick $\tau_{t-k} \neq \tau_t$. The oracle then adds $v\tau$ to a rolling counter. v is the amount of time since the last oracle update.

Let the accumulator at time t equal

$$\sum_{i=0}^{t} v\tau_i$$

To calculate the TWAP from $t-N$ to t, you subtact the accumulator at time t from time $t-N$ and divide by N

$$TWAP_{t-N,t} = \frac{\sum_{i=0}^{t} v\tau_i - \sum_{i=0}^{t-N} v\tau_i}{N}$$

Notice that this simplifies to

$$TWAP_{t-N,t} = \frac{\sum_{i=t-N}^{t} v\tau_i}{N}$$

1.2 Definition of Manipulation

The manipulation of length k between time $t-N$ to time t can be defined as

$$M_{(t,k)} = \frac{1.0001^{TWAP_{t-N,t}} - 1.0001^{TWAP_{t-N,t-k}}}{1.0001^{TWAP_{t-N,t-k}}}$$

1.3 What does a manipulator control

Oracle manipulators would have access to at least one oracle update without arbitragers. We want to know how large this update needs to be to make $M(t,k) = .2$

This final k slots for a N length TWAP can be defined as

$$\sum_{t-k}^{t} v\tau_i$$

This is the only thing that a potential manipulator has control over, so this is what needs to be solved for

1.4 Solving for the size of the last update required for a δ manipulation

We want to know how large a final update is needed to cause a manipulation of δ size defined as

$$\sum_{t-k}^{t} v\tau_i$$

Use $M_{(t,k)} = \delta$

$$\delta = \frac{1.0001^{TWAP_{t-N,t}} - 1.0001^{TWAP_{t-N,t-k}}}{1.0001^{TWAP_{t-N,t-k}}}$$

$$\frac{\ln 1 + \delta}{\ln 1.0001} = TWAP_{t-N,t} - TWAP_{t-N,t-k}$$

Apply the previously calculated TWAP formula

$$TWAP_{t-N,t} = \frac{\sum_{t-N}^{t} v\tau_i}{N}$$

$$TWAP_{t-N,t-k} = \frac{\sum_{t-N}^{t-k} v\tau_i}{N-k}$$

And plug back into the equation

$$\frac{\ln 1 + \delta}{\ln 1.0001} = \frac{\sum_{t-N}^{t} v\tau_i}{N} - \frac{\sum_{t-N}^{t-k} v\tau_i}{N-k}$$

Notice that $\sum_{t-N}^{t+k} v\tau_i = \sum_{t}^{t+k} v\tau_i + \sum_{t-N}^{t} v\tau_i$ and simplify

$$\frac{\sum_{t-k}^{t} v\tau + \sum_{t-N}^{t-k} v\tau_i}{N} - \frac{\sum_{t-N}^{t-k} v\tau_i}{N-k}$$

$$\frac{(N-k)\sum_{t}^{t+k} v\tau_i}{(N-k)t} - \frac{k\sum_{1}^{150-k} v\tau_i}{(N-k)N}$$

Replace back into the equation above

$$\frac{\ln 1 + \delta}{\ln 1.0001} = \frac{\sum_{t-k}^{t} v\tau_i}{N} - \frac{k\sum_{t-N}^{t-k} v\tau_i}{(N-k)N}$$

$$\sum_{t-k}^{t} v\tau_i = N\frac{\ln 1 + \delta}{\ln 1.0001} + kTWAP_{t-N,t-k}$$

1.5 Optimal Manipulation

While Uniswap v3 only makes oracle updates when a tick is updated, this is only for gas efficiency. It is the same value if you make an oracle update every block. We can simply the equation by making an oracle update every block.

Optimally, a manipulator would first swap up to the needed tick for manipulation and do nothing else until the needed time has passed to secure the desired TWAP.

Let $v = 1$, which assumes that an oracle update is fired every block, and the price is moved to τ ticks during the first block of the manipulation

$$k\tau = N\frac{\ln 1 + \delta}{\ln 1.0001} + kTWAP_{t-N,t-k}$$

$$\tau = \frac{N\frac{\ln 1+d}{\ln 1.0001}}{k} + TWAP_{t-N,t-k}$$