Optimal execution on Uniswap v2/3 under transient price impact

Bastien Baude

Joint work with Damien Challet and Ioane Muni Toke

January 26, 2026

CentraleSupélec

Contents

Decentralised Exchanges (DEXs)

- Whereas Centralized Exchanges (CEXs) rely on Limit Order Books (LOBs) to match buyers and sellers, DEXs execute transactions using liquidity pools
- Liquidity Providers (LPs) deposit digital assets into liquidity pools in exchange for a share of the transaction fees
- Liquidity Takers (LTs) trade against these pools according to algorithmic pricing rules: Automated Market Makers (AMMs)

Market

- The Total Value Locked (TVL) across all DEXs is $\$15$b, with a daily trading volume of $\$10$b
- Uniswap accounts for almost 25% of both total TVL and daily trading volume

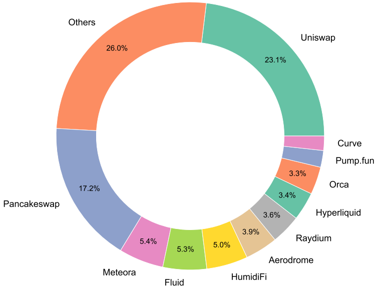

Figure 1. Volume distribution across protocols (in $\%$), source: DeFiLlama

Uniswap v2

- We consider a liquidity pool governed by a Constant Product AMM (CPAMM). The pool consists of ETH and USD, with reserves q^{ETH} and q^{USD} such that:

$$q^{ETH}q^{USD} = L^2$$

  where L is the liquidity of the pool

- The spot price of ETH in USD is:

$$p = \frac{q^{USD}}{q^{ETH}}$$

- Liquidity takers execute trades against the pool in such a way that the product of the reserves remains constant

Uniswap v2

- Suppose a trader wants to sell δ. In return, the AMM determines the amount δ^{USD} to be received by solving:

$$(q^{ETH} + \delta)(q^{USD} - \delta^{USD}) = L^2$$

- The amount δ^{USD} received by the trader is:

$$\delta^{USD} = \frac{\delta p}{1 + \frac{\delta\sqrt{p}}{L}} \approx \delta p\left(1 - \frac{\delta\sqrt{p}}{L}\right)$$

- The post-swap spot price is given by:

$$p^{+} = \frac{p}{\left(1 + \frac{\delta\sqrt{p}}{L}\right)^2} \approx p\left(1 - \frac{2\delta\sqrt{p}}{L}\right)$$

where the approximations correspond to first-order Taylor expansions in $\frac{\delta\sqrt{p}}{L} = \frac{\delta}{q^{ETH}}$

The scheduling problem on Uniswap v2

- We consider the problem of a trader executing a large sell order of size $\xi > 0$ of ETH over a fixed time horizon T. The time interval $[0, T]$ is divided into a regular partition: $0 = t_0 < t_1 < \cdots < t_N = T$, with $\Delta = \frac{T}{N}$
- At each time t_m, the trader sells an amount δ_m of ETH, subject to the volume constraint:

$$\sum_{m=0}^{N} \delta_m = \xi$$

Schematic spot price dynamics

$$f_0$$

$$\downarrow$$

Sell δ_0 at t_0

$$f_0\left(1-\frac{2\delta_0\sqrt{f_0}}{L}\right)$$

$$\downarrow$$

resilience and external factors from t_0 to t_1

$$f_1\left(1-e^{-\rho\Delta}\frac{2\delta_0\sqrt{f_0}}{L}\right)$$

$$\downarrow$$

Sell δ_1 at t_1

$$f_1\left(1-e^{-\rho\Delta}\frac{2\delta_0\sqrt{f_0}}{L}-\frac{2\delta_1\sqrt{f_1}}{L}\right)$$

$$\downarrow$$

$$\dots$$

Spot price

- We model the spot price as the combination of three components:
  - fundamental price process $(f_m)_{m=0}^{N}$
  - cumulative price impact induced by previous trades
  - resilience of the liquidity pool

- We write the spot price at time t_m as:

$$p_m = f_m \left(1 - \sum_{n=0}^{m-1} \sum_{j=0}^{J} \omega_j e^{-\rho_j (m-n)\Delta} \frac{2\delta_n \sqrt{f_n}}{L}\right)$$

where $(\rho_j)_{j=0}^{J}$ are resilience parameters and $(\omega_j)_{j=0}^{J}$ the associated weights

The scheduling problem on Uniswap v2

- The execution problem of the trader is:

$$\delta^* = \underset{\delta}{\operatorname{argmax}} \quad \mathbb{E}\Big[\sum_{m=0}^{N} \mathcal{C}_m\Big]$$

$$\text{s.t.} \quad \sum_{m=0}^{N} \delta_m = \xi$$

- The cash-flow at time t_m is:

$$\mathcal{C}_m = \delta_m f_m \Big(1 - \sum_{n=0}^{m-1} \sum_{j=0}^{J} \omega_j e^{-\rho_j (m-n)\Delta} \frac{2\delta_n \sqrt{f_n}}{L} - \frac{\delta_m \sqrt{f_m}}{L}\Big)$$

The scheduling problem on Uniswap v2

General solution

The optimal execution schedule is:

$$\delta^* = \left(\xi - \frac{L}{2}\mathbb{1}^\top A^{-1} B\right)\frac{A^{-1}\mathbb{1}}{\mathbb{1}^\top A^{-1}\mathbb{1}} + \left(\frac{L}{2}\mathbb{1}^\top A^{-1} B\right)\frac{A^{-1}B}{\mathbb{1}^\top A^{-1}B}$$

where $\mathbb{1} = \left(1, \dots, 1\right)^\top$, $B_m = \mathbb{E}[f_m]$ and:

$$A_{mn} = \begin{cases} \sum_{j=0}^{J} \omega_j e^{-\rho_j (m-n)\Delta} \mathbb{E}[f_m \sqrt{f_n}] & \text{if } n \leq m \\ \sum_{j=0}^{J} \omega_j e^{-\rho_j (n-m)\Delta} \mathbb{E}[f_n \sqrt{f_m}] & \text{if } n > m \end{cases}$$

The scheduling problem on Uniswap v2

Martingale case

If the fundamental price process is a martingale, the optimal solution reduces to:

$$\delta^* = \xi \frac{A^{-1}\mathbb{1}}{\mathbb{1}^\top A^{-1}\mathbb{1}}$$

- the solution is independent of the liquidity level L

Geometric Brownian motion case

- We assume the fundamental price to follow a geometric Brownian motion:

$$f_m = f_0 e^{(\mu - \frac{\sigma^2}{2}) m\Delta + \sigma W_m}$$

where:

  - f_0 is the initial fundamental price
  - μ the drift
  - σ the volatility
  - W_m denotes the Brownian motion at time t_m

- The optimal strategy δ^* is unique under the condition:

$$\mu < \frac{3\sigma^2}{4} + 4 \min_j \rho_j$$

The scheduling problem on Uniswap v2

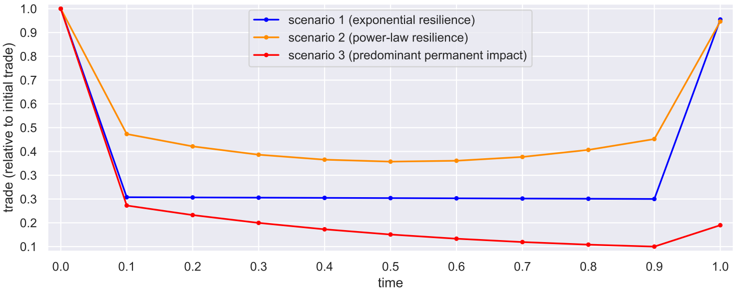

Figure 2. Optimal execution schedule (relative to the initial trade for the sake of comparison) under three scenarios

The scheduling problem on Uniswap v2

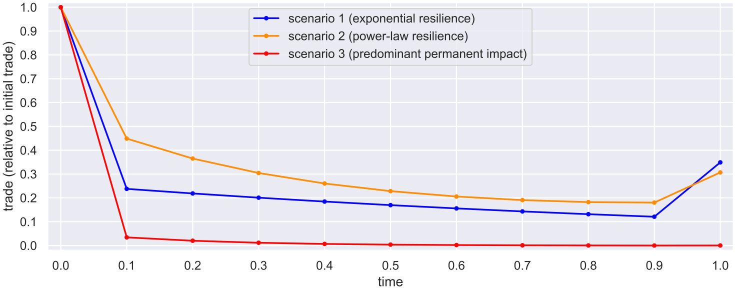

Figure 3. Optimal execution schedule (relative to the initial trade for the sake of comparison) under three stressed scenarios

Dynamic programming framework

- We reformulate the execution problem within a dynamic programming framework
- The state variables are given by the triplet $(x_n, (I_n^j)_{j=0}^J, f_n)$, where x_n is the remaining inventory, I_n^j the cumulative price impacts induced by the j-th resilience factor and f_n the fundamental price (geometric Brownian motion):

$$\begin{cases} x_0 = \xi \\ x_{n+1} = x_n - \delta_n \end{cases} \qquad \begin{cases} I_0^j = 0 \\ I_{n+1}^j = e^{-\rho_j \Delta}\left(I_n^j + \frac{2\delta_n \sqrt{f_n}}{L}\right) \end{cases}$$

Dynamic programming framework

- Let $v_n(x_n, (I_n^j)_{j=0}^J, f_n)$ denote the value function, the associated Bellman equation reads:

$$v_n(x_n, (I_n^j)_{j=0}^J, f_n) = \sup_{\delta_n} \delta_n f_n \big(1 - \sum_{j=0}^{J} \omega_j I_n^j - \frac{\delta_n \sqrt{f_n}}{L}\big) + \mathbb{E}\Big[v_{n+1}(x_{n+1}, (I_{n+1}^j)_{j=0}^J, f_{n+1}) | f_n\Big]$$

- The terminal condition enforces complete liquidation:

$$v_N(x_N, (I_N^j)_{j=0}^J, f_N) = x_N f_N \Big(1 - \sum_{j=0}^{J} \omega_j I_N^j - \frac{x_N \sqrt{f_N}}{L}\Big)$$

Dynamic programming framework

Dynamic programming solution

The value function reads:

$$v_n(x_n, (I_n^j)_{j=0}^J, f_n) = x_n f_n \Big(A_n + \sum_{j=0}^{J} B_n^j I_n^j + C_n x_n \sqrt{f_n} \Big) + \sqrt{f_n} \Big(D_n + \sum_{j=0}^{J} E_n^j I_n^j + \sum_{j_1=0}^{J} \sum_{j_2=0}^{J} F_n^{j_1,j_2} I_n^{j_1} I_n^{j_2} \Big)$$

and the optimal control:

$$\delta_n^*(x_n, (I_n^j)_{j=0}^J, f_n) = \frac{1}{2\phi_{n+1}\sqrt{f_n}} \Big[\theta_{n+1}^1 + \sum_{j=0}^{J} I_n^j \theta_{n+1}^{2,j} + x_n \sqrt{f_n} \theta_{n+1}^3 \Big]$$

The coefficients are determined recursively

Dynamic programming framework

- The coefficients ϕ_{n+1}, θ^1_{n+1}, $\left(\theta^{2,j_1}_{n+1}\right)_{j_1=0}^{J}$ and θ^3_{n+1} read:

$$\phi_{n+1} = \frac{1}{L} + \frac{2}{L}\sum_{j=0}^{J} B^j_{n+1} e^{(\mu-\rho_j)\Delta} - C_{n+1} e^{(\frac{3\mu}{2}+\frac{3\sigma^2}{8})\Delta} - \frac{4}{L^2}\sum_{j_1=0}^{J}\sum_{j_2=0}^{J} F^{j_1,j_2}_{n+1} e^{(\frac{\mu}{2}-\rho_{j_1}-\rho_{j_2}-\frac{\sigma^2}{8})\Delta}$$

$$\theta^1_{n+1} = 1 - A_{n+1}e^{\mu\Delta} + \frac{2}{L}\sum_{j=0}^{J} E^j_{n+1} e^{(\frac{\mu}{2}-\rho_j-\frac{\sigma^2}{8})\Delta}$$

$$\theta^{2,j_1}_{n+1} = -\omega_{j_1} - B^{j_1}_{n+1} e^{(\mu-\rho_{j_1})\Delta} + \frac{4}{L}\sum_{j_2=0}^{J} F^{j_1,j_2}_{n+1} e^{(\frac{\mu}{2}-\rho_{j_1}-\rho_{j_2}-\frac{\sigma^2}{8})\Delta}$$

$$\theta^3_{n+1} = \frac{2}{L}\sum_{j=0}^{J} B^j_{n+1} e^{(\mu-\rho_j)\Delta} - 2C_{n+1}e^{(\frac{3\mu}{2}+\frac{3\sigma^2}{8})\Delta}$$

$$\begin{cases} A_n = A_{n+1}e^{\mu\Delta} + \frac{1}{2\phi_{n+1}}\theta^1_{n+1}\theta^3_{n+1} \\ B^j_n = B^j_{n+1}e^{(\mu-\rho_j)\Delta} + \frac{1}{2\phi_{n+1}}\theta^{2,j}_{n+1}\theta^3_{n+1} \\ C_n = C_{n+1}e^{(\frac{3\mu}{2}+\frac{3\sigma^2}{8})\Delta} + \frac{1}{4\phi_{n+1}}(\theta^3_{n+1})^2 \\ D_n = D_{n+1}e^{(\frac{\mu}{2}-\frac{\sigma^2}{8})\Delta} + \frac{1}{4\phi_{n+1}}(\theta^1_{n+1})^2 \\ E^j_n = E^j_{n+1}e^{(\frac{\mu}{2}-\rho_j-\frac{\sigma^2}{8})\Delta} + \frac{1}{2\phi_{n+1}}\theta^1_{n+1}\theta^{2,j}_{n+1} \\ F^{j_1,j_2}_n = F^{j_1,j_2}_{n+1}e^{(\frac{\mu}{2}-\rho_{j_1}-\rho_{j_2}-\frac{\sigma^2}{8})\Delta} + \frac{1}{4\phi_{n+1}}\theta^{2,j_1}_{n+1}\theta^{2,j_2}_{n+1} \end{cases}$$

with $A_N = 1$, $B^j_N = -\omega_j$, $C_N = -\frac{1}{L}$, $D_N = E^j_N = F^{j_1,j_2}_N = 0$

Two-layer liquidity framework

- We consider a Uniswap v3 pool with two layers of liquidity:
  - L_0 if the spot price is above the threshold $\overline{p}$
  - $L_1 < L_0$ if the spot price is below $\overline{p}$

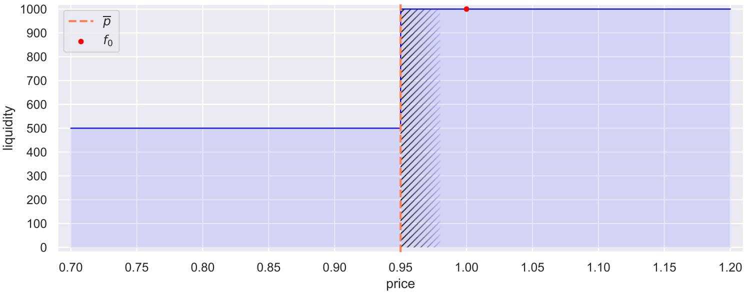

Figure 4. Two-layer liquidity profile

Two-layer liquidity framework

- Dynamics of the cumulative price impacts

$$
I_{n+1}^j = \begin{cases} e^{-\rho_j \Delta}\left(I_n^j + \frac{2\delta_n \sqrt{f_n}}{L_1}\right) & \text{if } p_n \le \overline{p} \\ e^{-\rho_j \Delta}\left(I_n^j + \frac{2\overline{\delta}_n \sqrt{f_n}}{L_0} + \frac{2(\delta_n - \overline{\delta}_n)\sqrt{\overline{p}}}{L_1}\right) & \text{if } p_n > \overline{p} \text{ and } \delta_n > \overline{\delta}_n \\ e^{-\rho_j \Delta}\left(I_n^j + \frac{2\delta_n \sqrt{f_n}}{L_0}\right) & \text{if } p_n > \overline{p} \text{ and } \delta_n \le \overline{\delta}_n \end{cases}
$$

- Bellman equation

$$
v_n(x_n, (I_n^j)_{j=0}^J, f_n) = \sup_{\delta_n} \; \mathcal{C}_n + \mathbb{E}\left[v_{n+1}(x_{n+1}, (I_{n+1}^j)_{j=0}^J, f_{n+1}) | f_n\right]
$$

where:

$$
\mathcal{C}_n = \begin{cases} \delta_n f_n \left(1 - \sum_{j=0}^J \omega_j I_n^j - \frac{\delta_n \sqrt{f_n}}{L_1}\right) & \text{if } p_n \le \overline{p} \\ \overline{\delta}_n f_n \left(1 - \sum_{j=0}^J \omega_j I_n^j - \frac{\overline{\delta}_n \sqrt{f_n}}{L_0}\right) & \\ \quad + (\delta_n - \overline{\delta}_n)\overline{p}\left(1 - \sum_{j=0}^J \omega_j I_n^j - \frac{(\delta_n - \overline{\delta}_n)\sqrt{\overline{p}}}{L_1}\right) & \text{if } p_n > \overline{p} \text{ and } \delta_n > \overline{\delta}_n \\ \delta_n f_n \left(1 - \sum_{j=0}^J \omega_j I_n^j - \frac{\delta_n \sqrt{f_n}}{L_0}\right) & \text{if } p_n > \overline{p} \text{ and } \delta_n \le \overline{\delta}_n \end{cases}
$$

Two-layer liquidity framework

- We define the spread $\overline{s}$ by:

$$\overline{p} = f_0 + \overline{s}$$

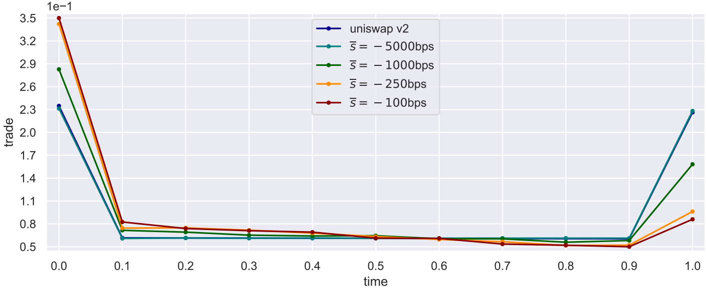

Figure 5. Optimal execution schedule with respect to $\overline{s}$

ETH/USDT liquidity pools

fee	1bp	5bps	30bps
daily volume	$100M	$100M	$25M
tick spacing	1	10	60

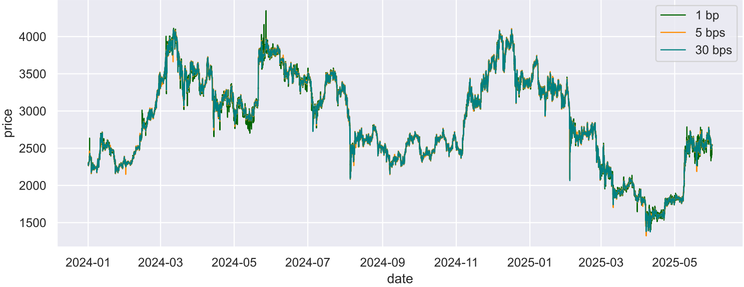

Figure 6. Evolution of ETH/USDT spot prices

ETH/USDT liquidity pools

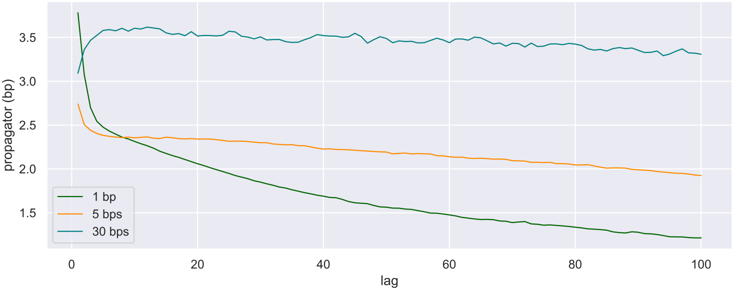

Figure 7. Estimated propagator functions

Conclusion

- We derive closed-form optimal solutions of the scheduling problem on Uniswap v2
- We rely on numerical schemes to estimate the optimal strategy on Uniswap v3
- Model calibration is still ongoing

Questions

Thanks for your attention !

Two-layer liquidity framework

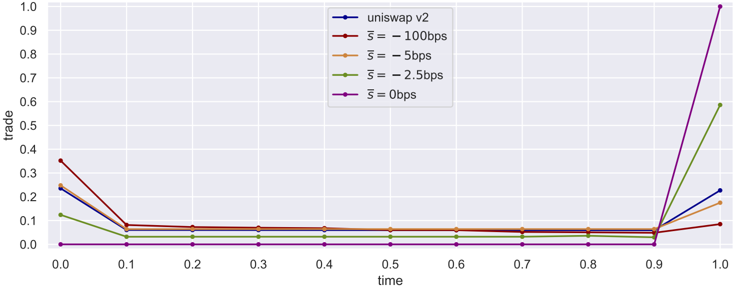

Figure 8. Optimal execution schedule with respect to $\overline{s}$

Two-layer liquidity framework

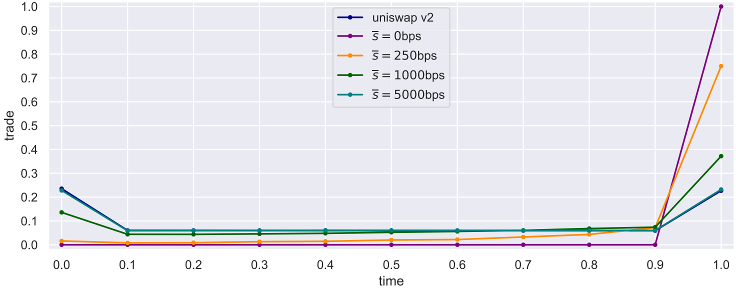

Figure 9. Optimal execution schedule with respect to $\overline{s}$

Transient Impact Model (TIM1)

- From Taranto et al., *Linear models for the impact of order flow on prices I. Propagators: Transient vs. History Dependent Impact*, Quant. Finance 18 (2018), 903–915, the mid-price is given by:

$$m_t = \sum_{t'<t} \left[G(t-t')\varepsilon_{t'} + \eta_{t'} \right] + m_{-\infty}$$

  where:
  - $\varepsilon_{t'}$ is the sign of trade at time t'
  - $\eta_{t'}$ is a noise term at time t'

- The function G is called the propagator and describes the decay of price impact with time

- The response function is defined by:

$$\mathcal{R}(\ell) = \mathbb{E}\big[(m_{t+\ell} - m_t)\varepsilon_t\big]$$

MSFT and AAPL

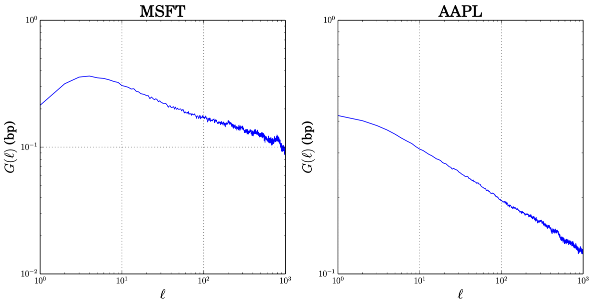

Figure 10. Estimated propagator functions from Taranto et al., *Linear models for the impact of order flow on prices I. Propagators: Transient vs. History Dependent Impact*, Quant. Finance 18 (2018), 903–915

ETH/USDT liquidity pools

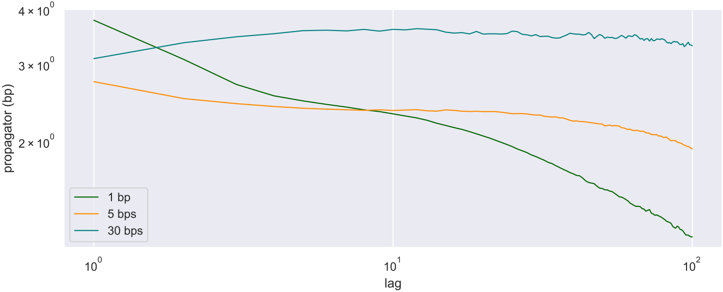

Figure 11. Estimated propagator functions

MSFT and AAPL

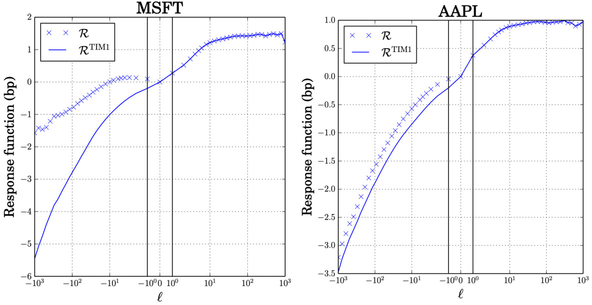

Figure 12. Response functions from Taranto et al., *Linear models for the impact of order flow on prices I. Propagators: Transient vs. History Dependent Impact*, Quant. Finance 18 (2018), 903–915

ETH/USDT liquidity pools

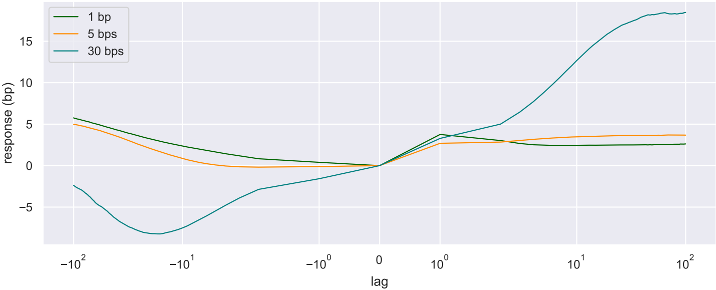

Figure 13. Response functions

ETH/USDT liquidity pools

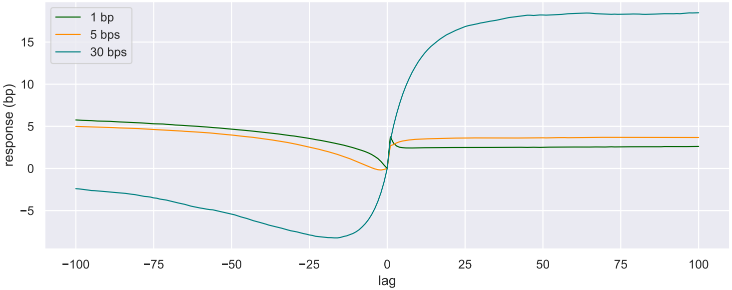

Figure 14. Response functions