Modeling Temporary Impact Functions from Limit Order Book Data

Devansh Srivastava

July 31, 2025

Introduction

This report addresses the modeling of the temporary market impact function $g_t(x)$, defined as the slippage (relative to the prevailing mid-price) incurred by a market order of size x at time t. Using depth-10 limit order book data for the three stocks SOUN, FROG, and CRWV, we empirically estimate the temporary impact function and provide a mathematical framework for optimally allocating trades over time to minimize total execution cost.

1 Empirical Modeling of Temporary Impact (Problem 1)

The temporary impact function $g_t(x)$ describes the cost penalty for executing x shares at time t, relative to the prevailing mid price. Our analysis involved:

1. **Order Book Reconstruction:** At one-minute intervals, a full snapshot (10 levels per side) of the LOB was constructed for each ticker.
2. **Market Order Simulation:** For a grid of order sizes x, simulated aggressive market orders consume liquidity, tracking fill prices.
3. **Slippage Calculation:** For each simulated trade, slippage is calculated as

$$\text{Slippage} = \text{VWAP}^1 - \text{Pre-trade Mid Price}$$

4. **Curve Fitting:** To the average slippage versus x, we fit both linear models ($g(x) = \beta x$) and power-law models ($g(x) = kx^{\alpha}$).
5. **Model Evaluation:** R^2 scores are computed for fit quality.

Empirical Results

Interpretation and Model Choice

- **Power-law superiority:** For all tickers, the power-law model $g(x) = kx^{\alpha}$ fits better than linear (all R^2_{lin} are negative: linear is worse than fitting a constant mean).

[1]VWAP (Volume Weighted Average Price): $\text{VWAP} = \frac{\sum(\text{price}\times\text{size})}{\sum \text{size}}$

Ticker	k (Power Law)	α (Power Law)	R^2_{pl}	β (Linear)	R^2_{lin}
SOUN	0.0236	0.07	0.047	0.00001	-13.6
FROG	0.0165	0.35	0.143	0.00007	-0.68
CRWV	0.0757	0.14	0.046	0.00002	-1.56

Table 1: Empirical model fits: mean slippage power-law and linear parameters for three tickers. R^2_{pl}: power law fit, R^2_{lin}: linear fit quality.

- **Sublinear impact:** The exponents α are all in $(0, 1)$, indicating that the impact function is concave—i.e., average slippage rises with order size, but at a decreasing rate.
- **Implication:** Real LOBs for these names on this day are sufficiently deep and liquid; for the tested range, marginal cost per share decreases as size rises. This is typical for high-liquidity environments or when most simulated order sizes are much less than total displayed liquidity.
- **Model recommendation:** For further modeling and optimization, use $g_t(x) = k_t x^{\alpha_t}$, with k_t, α_t locally estimated from LOB data in each time period as needed.

2 Mathematical Framework and Solution for Optimal Scheduling (Problem 2)

Suppose you must buy exactly S shares over a trading day split into N periods, purchasing x_i shares in period i ($i = 1, \ldots, N$). The objective is to minimize total expected slippage using the empirically-fitted cost functions.

Optimization Problem Statement

Define the allocation vector:

$$\mathbf{x} = (x_1, x_2, \ldots, x_N), \qquad x_i \geq 0,$$

with constraint:

$$\sum_{i=1}^{N} x_i = S.$$

The impact cost per interval is modeled as:

$$g_{t_i}(x_i) = k_{t_i} x_i^{\alpha_{t_i}}, \qquad (0 < \alpha_{t_i} < 1,\ k_{t_i} > 0)$$

The total cost to minimize:

$$\text{Total Cost} = \sum_{i=1}^{N} g_{t_i}(x_i) = \sum_{i=1}^{N} k_{t_i} x_i^{\alpha_{t_i}}$$

Including practical limitations (e.g., finite available liquidity L_i at each t_i):

$$0 \leq x_i \leq L_i \qquad \forall i.$$

Lagrangian and First-Order Conditions

Define the Lagrangian:

$$\mathcal{L}(\mathbf{x}, \lambda) = \sum_{i=1}^{N} k_{t_i} x_i^{\alpha_{t_i}} + \lambda \left(\sum_{i=1}^{N} x_i - S \right)$$

The first-order condition (for unconstrained x_i):

$$\frac{\partial \mathcal{L}}{\partial x_j} = k_{t_j} \alpha_{t_j} x_j^{\alpha_{t_j} - 1} + \lambda = 0$$

which leads to:

$$x_j^{\alpha_{t_j} - 1} = -\frac{\lambda}{k_{t_j} \alpha_{t_j}}$$

This implies, in the unconstrained case, that the optimizer tries to allocate as much as possible to the interval(s) with the lowest k_{t_j}, but practical caps L_j typically bind first, so allocation is distributed up to those caps.

Numerical Solution Approach

Because: - $0 < \alpha_{t_i} < 1$ (concave power law), - liquidity caps L_i may be present, - k_{t_i} (and α_{t_i}) may vary over time,

the problem is **non-convex**, and must be solved numerically.

Implementation: - Set N, S according to your trading window and shares to execute. - Fit or set k_{t_i}, α_{t_i} from empirical slippage curves or use daily averages if only overall fits are available. - Set liquidity caps L_i by inspecting order book depth (or as a practical conservative limit). - Solve optimization:

$$\min_{0 \leq x_i \leq L_i} \left\{ \sum_{i=1}^{N} k_{t_i} x_i^{\alpha_{t_i}} \;\middle|\; \sum_{i=1}^{N} x_i = S \right\}$$

- In Python, you can use optimization packages such as `scipy.optimize.minimize` with method='SLSQP' for nonlinear constraints.

Empirical Solution Example: Assuming constant k and α as in your fitted results, symmetric liquidity caps, and $N = 390$ (for minute-by-minute allocation over a typical trading day), the optimizer evenly spreads trades:

$$x_i^* \approx \frac{S}{N}, \quad \forall i$$

This follows from sublinear cost and uniform cost/time.

Interpretation and Real-World Insights

- **Concave model ($0 < \alpha < 1$):** In the absence of risk or liquidity constraints, the optimizer would theoretically concentrate execution in the most favorable (lowest k_{t_i}) intervals. Constraints on x_i (liquidity, practical market risk, maximum order size per window, or operational risk limits) typically force more even allocations.
- **Uniform allocation as optimal:** Given symmetric cost parameters and caps, optimal execution is nearly uniform, as found in your code output.
- **Extending the model:** For more realistic uneven costs, you can fit k_{t_i} and α_{t_i} for each window and repeat the optimization.

Practical Algorithm Pseudocode

Given S, N, fitted (k_{t_i}, α_{t_i}), and liquidity caps L_i, the solution method is:

1. Set up the cost:
$$C(\mathbf{x}) = \sum_{i=1}^{N} k_{t_i} x_i^{\alpha_{t_i}}$$
2. Solve numerically:
$$\begin{aligned} \min_{\mathbf{x}} \quad & C(\mathbf{x}) \\ \text{s.t.} \quad & \sum_{i=1}^{N} x_i = S \\ & 0 \le x_i \le L_i \end{aligned}$$
(using, e.g., SLSQP optimizer)
3. If k_{t_i} differ or more constraints are required, repeat per time window for adaptive scheduling.

3 Summary

- For all stocks, empirical cost functions are concave ($0 < \alpha < 1$): slippage increases with order size, but sublinearly.
- Practically, the optimizer allocates as much volume as possible to the cheapest intervals, but in your data's symmetric setup, spreads uniformly due to identical parameters and liquidity caps.
- This framework can be extended to, and solved for, time-varying parameters or further real-world constraints.

Note: All analysis, simulation code, and fit results were computed as shown in the attached notebook. Refer to the github repo at link for all code and analysis.