# Modeling Temporary Market Impact: Execution Scheduling Framework

Mehul Lad

July 27, 2025

## Part 2 – Mathematical Framework for Execution Scheduling

To complement the modeling work I did in Part 1, I now turn my attention to designing a mathematical framework for determining how to optimally split a total order size $S$ across a trading horizon consisting of multiple time points $t_1, t_2, ..., t_N$. The main goal is to minimize the total temporary market impact while ensuring full execution of the order.

### Problem Setting

Let me start by stating the problem formally:

- Let $S$ be the total number of shares I want to buy.
- Let $x_i$ be the number of shares I decide to buy at time $t_i$.
- I need to ensure that the sum of all individual executions equals the total order:

$$\sum_{i=1}^{N} x_i = S \quad \text{with} \quad x_i \geq 0$$

At each time step $t_i$, buying $x_i$ shares incurs a temporary market impact cost $g_{t_i}(x_i)$.

From the empirical results in Part 1, the exponential model

$$g_t(x) = a_t(1 - e^{-b_t x})$$

provided a strong fit across multiple tickers. This model assumes the impact grows quickly at small volumes and tapers off at higher sizes, reflecting saturation of liquidity.

Thus, our goal becomes:

$$\min_{x_1,\dots,x_N} \sum_{i=1}^{N} a_i(1 - e^{-b_i x_i}) \quad \text{subject to} \quad \sum x_i = S$$

### Why I Chose This Model

Empirically, market impact curves are clearly nonlinear. The exponential form does a much better job at capturing the diminishing marginal impact at higher order sizes. It also avoids over-penalizing large trades unrealistically.

### Techniques to Solve It

There are a couple of ways to approach solving this optimization problem:

**1. Convex Optimization Approach** Since the exponential function is convex and we're summing convex functions with a linear constraint, this is a standard convex optimization problem. I could implement this using Python libraries like `cvxpy`, which would let me plug in estimated $a_i$ and $b_i$ for each time period and get the optimal $x_i$ schedule.

**2. Greedy Approximation** As a simpler baseline, I could use a greedy strategy: at each step, look at the marginal cost

$$g'_{t_i}(x_i) = a_i b_i e^{-b_i x_i}$$

and allocate the next unit of volume to the time period with the lowest marginal cost. This method is not optimal but is intuitive and often performs surprisingly well.

**3. Uniform Execution (Benchmark)** As a sanity check, I could compare the optimized cost with the cost of just doing $x_i = S/N$ for all $i$, which would represent a time-weighted average price (TWAP) strategy. This helps to quantify the benefit of optimization.

To summarize, this framework allows me to reason quantitatively about the trade-offs between liquidity and timing. I haven't implemented the solver yet due to scope, but I believe the problem is well-posed, solvable with standard tools, and practically motivated by my Part 1 modeling.