Task 2 — A Rigorous Framework for Execution Scheduling

Setup, data, and notation

We partition the trading day into N buckets $t_1, \dots, t_N$ (e.g., 390 one–minute buckets). At bucket t_i we choose $x_i \geq 0$ shares to execute. The hard completion constraint is

$$\boxed{\sum_{i=1}^{N} x_i \;=\; S} \tag{1}$$

The temporary impact (slippage) at time t_i for trading size x is $g_{t_i}(x)$. From Task 1 we estimate $g_{t_i}(x)$ with a Gradient–Boosting oracle $\widehat{g}_{t_i}(x \mid \mathrm{state}_i)$ using the fields:

`timestamp, size, slippage (=` $g_t(x)$`), vol_ratio (=` x`/depth), spread, depth, imbalance, volatility`

Feature definitions used throughout:

$$\begin{aligned}
&\text{mid } m_t = \tfrac{1}{2}(\text{bid_px_00} + \text{ask_px_00}), \quad \text{spread } = \text{ask_px_00} - \text{bid_px_00},\\
&\text{depth } = \sum_{\ell=0}^{9}(\text{bid_sz}_\ell + \text{ask_sz}_\ell), \quad \text{imbalance } = \frac{\text{bid_sz_00} - \text{ask_sz_00}}{\text{bid_sz_00} + \text{ask_sz_00} + 10^{-9}},\\
&\text{volatility } = \text{std}_{60\text{ s}}(\Delta \log m_t), \quad \text{hour_of_day} = \text{hour}(t), \quad \text{vol_ratio} = x/\text{depth}.
\end{aligned}$$

We regard $g_{t_i}(x)$ as increasing in x with increasing marginal cost over the operative size range.

Objectives

(A) Cost–only objective.

$$\boxed{\min_{x_1,\dots,x_N} \sum_{i=1}^{N} g_{t_i}(x_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} x_i = S, \quad x_i \geq 0} \tag{2}$$

When g_{t_i} is differentiable and convex on the used range, Karush–Kuhn–Tucker (KKT) conditions imply the *equalized marginal cost* rule on used buckets:

$$\boxed{g'_{t_i}(x_i^{)=\lambda \quad \text{for each } i \text{ with } x_i^{>0}}} \qquad \text{together with} \quad \boxed{\sum_{i=1}^{N} x_i^{=S, \quad x_i^{\geq 0}}} \tag{3}$$

We deliberately place the stationarity condition *separate* from the budget constraint to avoid index ambiguity.

(B) Cost + risk–smoothing.

$$\boxed{\min_{x_1,\dots,x_N} \sum_{i=1}^{N} \Big[g_{t_i}(x_i) + \tfrac{\rho}{2}\, x_i^2\Big] \quad \text{s.t.} \quad \sum_{i=1}^{N} x_i = S, \quad x_i \geq 0} \tag{4}$$

KKT becomes

$$\boxed{g'_{t_i}(x_i^{)+\rho\, x_i^{=\lambda \quad \text{on used buckets}, \qquad \sum_{i=1}^{N} x_i^{=S}}}} \tag{5}$$

so $\rho > 0$ discourages lumpy slices.

(C) Time-risk on remaining inventory. Let the remaining inventory *after* executing bucket i be

$$\boxed{R_i \;=\; S - \sum_{k=1}^{i} x_k, \qquad R_0 = S} \tag{6}$$

We penalize carry with nondecreasing weights w_i:

$$\boxed{\min_{x_1,\dots,x_N} \; \sum_{i=1}^{N} g_{t_i}(x_i) \;+\; \tfrac{\psi}{2} \sum_{i=1}^{N-1} w_i\, R_i^2 \quad \text{s.t.} \; \sum_{i=1}^{N} x_i = S, \quad x_i \ge 0} \tag{7}$$

Choosing w_i. We set w_i to grow in time; examples:

$$w_i = \frac{i}{N} \;\text{(linear)}, \quad w_i = \Big(\frac{i}{N}\Big)^{\gamma},\; \gamma \in [1,2] \;\text{(accelerating)}, \quad w_i \propto \widehat{\sigma}_i^2 \;\text{(risk-proportional to forecast volatility)}.$$

In practice we can use $w_i = i/N$ unless a volatility forecast is available.

Computing the schedule x_i

1) Water–filling (dual bisection) for (2) or (4)

We treat λ as a target marginal cost and solve per-bucket problems, then adjust λ to satisfy (1).
Per-bucket solve. For fixed λ,

$$x_i(\lambda) = \arg\min_{x \ge 0} \big[g_{t_i}(x) - \lambda x\big] \iff \begin{cases} g'_{t_i}(x_i(\lambda)) = \lambda, & \text{for (2)}, \\ g'_{t_i}(x_i(\lambda)) + \rho\, x_i(\lambda) = \lambda, & \text{for (4)}. \end{cases} \tag{8}$$

We evaluate $\widehat{g}_{t_i}(x)$ on a grid $x \in \{0, \Delta x, 2\Delta x, \dots, x_{\max}\}$ with Δx chosen to match execution granularity (e.g., 100 shares or one size bucket). We obtain discrete slopes by forward differences and *monotonize* them if needed:

- **Isotonic (PAVA) projection on $g(x)$.** Apply the pool–adjacent–violators algorithm to enforce nondecreasing $g(x)$ over the grid.
- **Convex regression on $g(x)$.** Solve $\min_{\tilde g} \sum_u (\tilde g(x_u) - \widehat{g}(x_u))^2$ s.t. second differences $\Delta^2 \tilde g(x_u) \ge 0$ (least-squares with nonnegative second-difference constraints; easily done in `cvxpy`).

Given a monotone/convex $\tilde g$, we invert g' by *bisection* on x for each i to find $x_i(\lambda)$ to tolerance ε_x (e.g., 10^{-3} of the spread or a few cents).
Budget match. Define $S(\lambda) = \sum_{i=1}^{N} x_i(\lambda)$ and use bisection on λ until $|S(\lambda) - S| \le \varepsilon_S$ (e.g. 0.1% of S). Illiquid buckets produce $x_i(\lambda) = 0$ naturally.

2) Receding–horizon scheduling (sequential use with real-time state)

At bucket t_i we plan a short window and execute only the first slice, then re-plan. Let $S_{\text{rem}} = S - \sum_{k=1}^{i-1} x_k$ and choose a horizon H (e.g., 10–30 minutes). Query the GB model for $\widehat{g}_{t_j}(x)$, $j = i, \dots, i+H-1$, and solve

$$\boxed{\min_{\{x_j\}_{j=i}^{i+H-1}} \; \sum_{j=i}^{i+H-1} \Big[\widehat{g}_{t_j}(x_j) + \tfrac{\rho}{2} x_j^2\Big] \quad \text{s.t.} \quad \sum_{j=i}^{i+H-1} x_j \;\le\; S_{\text{rem}}, \quad x_j \ge 0} \tag{9}$$

We use "$\le$" within the window to avoid *forcing* all remaining volume into the next H buckets; this preserves flexibility. The *global* equality (1) is enforced because at the final bucket we set $x_N := S_{\text{rem}}$ (or add a terminal equality on the last window). Execute x_i, update $S_{\text{rem}} \leftarrow S_{\text{rem}} - x_i$, slide the window, and repeat.

Practical notes and guardrails

Grid and tolerances. Choose Δx to reflect actual slice granularity; typical tolerances: ε_x a few cents in price, $\varepsilon_S \approx 0.1\%$ of S. **Robustness.** If GB predictions vs. x are noisy, apply isotonic (PAVA) or convex smoothing before inversion. **Weights w_i.** Use linear $w_i = i/N$ by default; switch to risk-proportional $w_i \propto \widehat{\sigma}_i^2$ when intraday volatility forecasts are available. **Terminal enforcement.** Ensure $x_N := S_{\text{rem}}$ to satisfy (1) exactly.

Summary

We minimize $\sum_i g_{t_i}(x_i)$ subject to $\sum_i x_i = S$ using our learned $g_t(x)$. Water–filling (bisection on λ) implements the equal-marginal-cost rule; a short-horizon re-optimization adapts to changing state without violating the global budget. All methods integrate directly with the fields and GB model built in Task 1.