Guaranteed VWAP

Dynamic Execution Under Guaranteed VWAP: A Bayesian and Impact-Aware Simulation Framework

Jayesh Khandelwal

May 2025

Abstract

We develop a dynamic simulation framework for Guaranteed VWAP (GVWAP) execution that integrates Bayesian volume modeling, market impact estimation, and trajectory optimization under uncertainty. A Bayesian Dirichlet model provides unconditional volume forecasts, which are updated in real time using a Bayesian Ridge Regression model. Execution prices incorporate nonlinear temporary impact and decaying residual effects. Trade schedules are dynamically adjusted using sinh-shaped trajectories, with κ selected at each step via grid search to minimize risk-adjusted deviation from market VWAP. This framework captures the essential trade-offs in GVWAP contracts between execution cost and market risk.

Contents

1 Introduction

Guaranteed VWAP (Volume-Weighted Average Price) contracts are widely used in institutional trading to transfer execution risk from the client to the broker. Under such contracts, the broker guarantees to execute a target quantity at the realized market VWAP over the day. Any deviation between the broker's execution price and the market VWAP results in a cost borne by the broker, creating the need for a robust and risk-aware execution strategy. This project presents a detailed simulation framework to evaluate and optimize execution under GVWAP constraints, balancing the twin objectives of cost minimization and risk control.

The execution strategy is built upon a two-stage volume modeling pipeline. First, a static volume model based on the Bayesian Dirichlet prior captures the historical average of trading volume across intraday time buckets. It combines a global prior with stock-specific empirical distributions, yielding an unconditional estimate of expected market volume allocation. Second, a dynamic volume model updates this forecast in real-time using a Bayesian Ridge Regression approach. The model conditions on the cumulative market volume and recent intraday volume trends to produce a refined forecast of volume share in each bucket, allowing the agent to adapt to intra-day fluctuations.

To simulate a realistic execution environment, the framework generates market prices using a geometric Brownian motion-like stochastic process and models market volume as log-normal noise centered around the agent's own trade size. Execution prices are impacted by the agent's trading behavior through a temporary impact model, which is nonlinear in trading rate and includes a linear decay of residual impact across buckets. This impact-aware pricing ensures that aggressive trading incurs higher cost, reflecting real-world execution friction.

The agent's trade trajectory is re-evaluated at the end of each bucket using a sinh-shaped trading schedule, parameterized by a control variable κ. A grid search is performed over a range of κ values, encompassing both front-loaded and back-loaded schedules. For each candidate trajectory, the simulation computes the expected deviation from market VWAP and adds a risk penalty proportional to the variance of outcomes. The optimal trajectory is the one minimizing this risk-adjusted cost, reflecting the broker's risk aversion under the GVWAP contract.

This end-to-end framework thus enables realistic testing of execution strategies under intraday uncertainty, market impact, and risk constraints. By incorporating dynamic volume adjustments and trajectory reoptimization, it reflects the iterative nature of real-world trading and provides a valuable testbed for evaluating algorithmic execution methods.

2 Methodology

The key components and steps of the methodology are outlined below:

a) **Static Volume Model (Unconditional)**:

- Implemented using a Bayesian Dirichlet model to estimate the average fraction of daily trading volume across 13 intraday buckets.
- Learns a global prior from all stocks and smooths it with stock-specific history using a blending factor λ_{prior}.
- Provides an initial volume trajectory prior to market open.

b) **Dynamic Volume Model (Bayesian Ridge Regression)**:

- Updates the static volume fractions based on real-time market conditions.
- Predicts the agent's expected volume share in a bucket using:
  - static prediction,
  - cumulative market volume so far,
  - recent volume trends (mean and std over past 3 buckets).

c) **Market Simulation Environment**:

- **Market Price:** Simulated using a geometric Brownian motion-like path with Gaussian noise.
- **Market Volume:** Modeled as log-normal noise centered around the agent's trade size.

d) **Price Impact Modeling**:

- Execution price = market price + temporary impact.
- Temporary impact modeled as:

$$\text{Impact}_t = \eta \cdot \sigma \cdot |v_t|^\beta + 0.5 \cdot \eta \cdot \sigma \cdot |v_{t-1}|^\beta$$

  where:
  - $v_t = \frac{dx}{dt}$ is the optimal trading rate based on a sinh-shaped trajectory,
  - η is the impact coefficient,
  - σ is the volatility,
  - β controls impact curvature (e.g., $\beta = 0.5$).
- Residual impact from the previous bucket decays linearly (50% decay assumed per 30-minute period).

e) **Trade Scheduling and Execution Simulation**:

  i. Use the static model to initialize trade schedule.
  ii. Adapt trade schedule dynamically using the ridge regression model.
  iii. Optimize future execution using a grid search over κ, selecting the one minimizing deviation from market VWAP.
  iv. Execute trades bucket by bucket, adjusting price and volume accordingly.

f) **Execution Cost Evaluation**:

- Compute volume-weighted average price (VWAP) for the agent and the market:

$$\text{My VWAP} = \frac{\sum P_t^{\text{exec}} \cdot Q_t}{\sum Q_t}, \quad \text{Market VWAP} = \frac{\sum P_t^{\text{mid}} \cdot V_t^{\text{market}}}{\sum V_t^{\text{market}}}$$

- Define Guaranteed VWAP Cost as:

$$\text{Cost} = |\text{My VWAP} - \text{Market VWAP}| + \lambda \cdot \text{Risk}$$

  where:
  - λ: Risk aversion coefficient, with unit $\$^{-1}$.
  - Risk: Variance of deviations during κ optimization; unit $\$^2$.

3 Volume Modeling

3.1 Static Volume Model (Unconditional)

The static volume model estimates the typical intraday trading volume distribution for a given stock across 13 equal 30-minute buckets. It assumes stationarity and relies solely on historical volume patterns — independent of real-time market activity.

We implement this using a Bayesian Dirichlet framework, which allows us to encode prior beliefs, perform statistical smoothing, and estimate a robust bucket-wise volume fraction vector.

- **Code:** `Code/Project/src/volume_static.py`
- **Training stock:** `Processed_Data/top_1500_liquid_stocks.txt`; the stocks that were highly liquid in HW 1.
- **Model Type:** Bayesian Dirichlet with smoothing across stock and global priors.
- **Assumption:** Intraday volume curve follows a fixed probabilistic structure for each stock, updated via historical data.
- **Model Saved:** `Processed_Data/bayesian_dirichlet_static_volume.pkl`
- **Unit Test:** `Code/Project/test/volume_static_test.py`

Model Logic and Training

- The model maintains a global prior Dirichlet distribution over 13 buckets, initialized as a uniform pseudo-count vector:

$$\boldsymbol{\alpha}_{\text{prior}}^{\text{global}} = [\texttt{prior\_strength}, \ldots, \texttt{prior\_strength}]$$

- For each stock, it collects daily volume curves (normalized to sum to 1), parses from TAQ trade data.
- A global posterior is computed using all curves:

$$\boldsymbol{\alpha}_{\text{posterior}}^{\text{global}} = \boldsymbol{\alpha}_{\text{prior}}^{\text{global}} + \sum_{\text{all curves}} \text{volume fractions}$$

- For each stock, we compute a smoothed posterior using a convex blend:

$$\boldsymbol{\alpha}_{\text{stock}} = w \cdot \boldsymbol{\alpha}_{\text{posterior}}^{\text{global}} + (1-w) \cdot \sum_{\text{stock curves}} \text{volume fractions}$$

  where:

$$w = \frac{\lambda_{\text{prior}}}{n + \lambda_{\text{prior}}}, \quad n = \#\text{ of stock-specific days}$$

- This blending balances:
  - **Global signal** (from many stocks) when data is sparse.
  - **Stock-specific behavior** when ample history is available.
- Finally, the model converts the Dirichlet $\boldsymbol{\alpha}$ vector into an expected volume fraction:

$$\hat{\mathbf{p}} = \frac{\boldsymbol{\alpha}_{\text{stock}}}{\sum \boldsymbol{\alpha}_{\text{stock}}}$$

Key Parameters

- `prior_strength`: Uniform pseudo-count value for global Dirichlet prior (default = 10).
- `lambda_prior`: Weight controlling how much to trust the global prior versus stock-specific curves (interpreted as equivalent number of prior days).

3.2 Statistical Evaluation

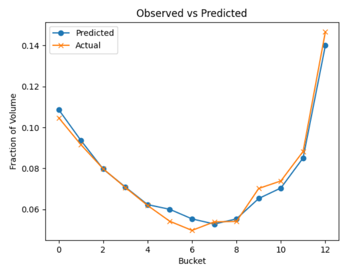
- **Code:** `Code/Project/src/volume_validation.py`
- **Unit Test:** `volume_static_eval_test.py`

Model Fit and Predictive Accuracy metrics are summarized in Table 1, including detailed explanations and numerical results that assess the model's overall performance.

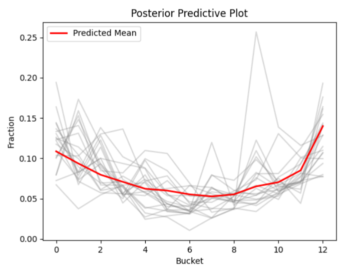
Metric	**Formula / Description**	**Value**	**Inference**
Mean Squared Error (MSE)	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ Sensitive to large deviations. Measures average squared error.	0.001644	Low MSE indicates good fit with few large errors.
Mean Absolute Error (MAE)	$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$ Measures average absolute deviation.	0.027050	Indicates average error magnitude. Less sensitive to outliers.
Log Likelihood	$\log \mathcal{L} = \sum_{i=1}^{n} y_i \cdot \log(\hat{y}_i + \varepsilon)$ Likelihood of observed data under model.	-49051.40	Negative value typical; used comparatively for model fit.
Bayesian R^2	$R^2 = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$ Proportion of variance explained by model.	0.283196	Moderate explanatory power; room for improvement.
Monte Carlo Std. Error (MCSE)	$\text{MCSE} = \frac{\text{Std}(\hat{y} - y)}{\sqrt{n}}$ Error due to sampling noise in posterior estimation.	0.000290	Small MCSE suggests stable posterior estimates.
Sharpness	$\text{Sharpness} = \frac{1}{B}\sum_{b=1}^{B}\left(Q_{97.5}^{(b)} - Q_{2.5}^{(b)}\right)$ Width of 95% predictive interval.	0.143149	Reasonably narrow intervals; reflects model confidence.

Table 1: Model Fit and Predictive Accuracy Metrics

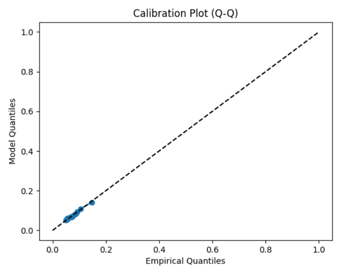
Visual diagnostics such as calibration, residuals, and uncertainty coverage are presented in Figure 1, providing deeper insights into model behavior across time buckets.



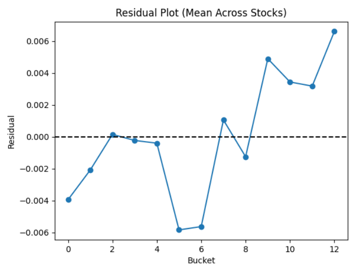
(a) Observed vs. Predicted: Compares mean predicted and actual bucket-wise volume fractions across the test set.
Inference: Close tracking indicates that the model accurately captures the intra-day volume shape.



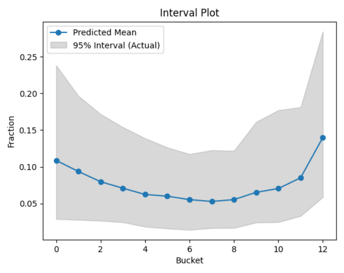
(b) Posterior Predictive: Red curve shows model mean; gray lines are actuals from test days.
Inference: The predicted mean lies near the center of the actual distribution, suggesting good calibration.



(c) Calibration (Q-Q) Plot: Checks distributional alignment between predicted and observed quantiles.
Inference: Alignment with the $y = x$ line confirms well-calibrated distributional outputs.



(d) Residual Plot: Shows bucket-wise mean residuals across stocks.
Inference: Mostly small residuals, with no strong systematic bias, though minor under-prediction in late buckets is noted.



(e) Interval Plot: Predicted mean with 95% interval vs actuals.
Inference: Actual volumes mostly lie within the shaded region, confirming strong coverage and sharpness.

Figure 1: Model Diagnostic Plots: Visual assessment of fit, calibration, residual trends, and uncertainty quantification. The model demonstrates strong predictive alignment and reliable interval estimates.

3.3 Dynamic Volume Model: Bayesian Ridge Regression

To capture the intraday dynamics of market volume, we implement a Bayesian Ridge Regression model that adjusts static volume predictions using real-time market behavior. This allows the agent to adapt its trade schedule based on observed trading conditions.

- **Code:** `Code/Project/src/dynamic_volume.py`
- **Model Saved:** `Processed_Data/bayesian_ridge_dynamic_volume.pkl`
- **Unit Test:**
  - `Code/Project/test/volume_dynamic_data_process.py`
  - `Code/Project/test/volume_dynamic_ridge.py`

Model Overview

The model is trained to predict the volume traded in a given 30-minute bucket using the following features:

- Static model's predicted volume for the current bucket.
- Cumulative market volume up to the previous bucket.
- Mean of market volume in the past 3 buckets.

- Standard deviation of market volume in the past 3 buckets.

Mathematically, the feature vector at time t is:

$$\mathbf{x}_t = \left[\text{Static}_t,\ \sum_{i=0}^{t-1} \text{Market}_i,\ \text{Mean}(\text{Market}_{t-3:t-1}),\ \text{StdDev}(\text{Market}_{t-3:t-1}) \right]$$

These features are fed into a Bayesian Ridge Regression model, which allows us to quantify uncertainty and prevent overfitting, especially when data is limited.

Data Preparation and Training
The training dataset is generated using historical TAQ quote files for a list of top 1500 liquid U.S. stocks. Each day's market volume curve is parsed, and feature-target pairs are created for all 13 half-hour buckets.
We split available data chronologically:

- **Training Set:** First 80% of trading dates.
- **Test Set:** Remaining 20%, representing unseen market behavior.

The model is then fit on the training set and evaluated using Root Mean Squared Error (RMSE) on both training and test data.

Results
After fitting the model, we obtained the following performance metrics:

Dataset	RMSE
Train	0.039557
Test	0.040451

Table 2: Root Mean Squared Error (RMSE) of the Dynamic Volume Model

These values indicate that the model generalizes well, with only a slight increase in prediction error on unseen data. This level of accuracy supports effective dynamic adjustment of trade schedules during simulation.

4 Execution Simulation

4.1 My Trade Volume Prediction Strategy

- **Code:** `Code/Project/src/stimulation.py`
- **Function:** `vwap_simulation` (It has complete stimulation flow.)
- **Unit Test:** `Code/Project/test/test_vwap_simulation.py`

To determine how many shares to execute in each 30-minute bucket, we combine both static and dynamic volume prediction models described earlier:

- **Static Model (Bayesian Dirichlet):** Provides an unconditional prior estimate of expected trading volume fractions for each bucket, based on historical patterns.
- **Dynamic Model (Bayesian Ridge Regression):** Adjusts the static predictions based on observed market volume behavior during the day.

Volume Decision Logic Per Bucket
The following logic governs how the agent decides how many shares to trade in each bucket:

1. **Bucket 0 (Market Open):** There is no prior market volume information. We rely solely on the static model prediction for this bucket:

$$\text{Volume}_0 = \text{Static}_0 \cdot \text{TotalShares}$$

2. **Buckets 1 to $T-2$:** From the second bucket onward (until the second-last), we incorporate observed market activity by using the dynamic model:

$$\text{Volume}_t = \text{DynamicModel}\left(\text{MarketVolumeFractions}, \text{StaticPred}, t\right) \cdot \text{TotalShares}$$

   Here, the dynamic model uses features like cumulative and recent market volume, along with the static prediction, to adaptively refine the trade schedule.

3. **Final Bucket ($t = T - 1$):** At the end of the trading day, we must trade any remaining shares to complete the order. Thus:

$$\text{Volume}_{T-1} = \text{RemainingShares}$$

To ensure executability, the computed trade volume is rounded to the nearest integer:

$$\text{TradeVolume}_t = \lfloor \text{Volume}_t \rfloor$$

4.2 Market Information

In our simulation framework, we incorporate simple yet realistic models for the market's price and volume evolution throughout the trading day. This provides a reference against which to evaluate our execution quality.

1. Market Price Simulation

- **Code:** `Code/Project/src/stimulation.py`
- **Function:**
  - `compute_arrival_price`
  - `simulate_price_path`
- **Unit Test:**
  - `Code/Project/test/test_arrival_price.py`
  - `Code/Project/test/test_simulate_price_path.py`

We define the starting point of our price trajectory — the **arrival price** — as the average of the first five mid-quote prices observed immediately after the market opens at 9:30 AM. This serves as a fair and unbiased estimate of the initial market price.

$$\text{Arrival Price} = \frac{1}{5}\sum_{i=1}^{5} P_i^{\text{mid}}, \quad \text{where each } P_i \text{ is a mid-quote after 9:30 AM}$$

This price is extracted from TAQ (Trade and Quote) data using a binary reader. If fewer than 5 quotes are available, the function returns the average of however many are found.

To simulate price movement across intraday buckets, we use a stochastic geometric random walk:

$$P_{t+1} = P_t \cdot (1 + \varepsilon_t), \quad \varepsilon_t \sim \mathcal{N}(0, \sigma)$$

where:

- P_t is the price at bucket t.
- ε_t is normally distributed noise, simulating small log returns.
- σ is the volatility parameter (e.g., 0.005).

This process results in a synthetic but realistic intraday price path with mild fluctuations. The randomness helps capture the uncertainty faced by an executing agent.

2. Market Volume Simulation

- **Code:** `Code/Project/src/stimulation.py`
- **Function:** `simulate_market_volume_log_normal`
- **Unit Test:** `Code/Project/test/test_simulate_market_volume.py`

To compare the agent's trading volume with overall market activity, we simulate market-traded volume per bucket based on a stochastic model.

Key assumption: Market volume in each bucket is centered around the agent's trade volume but includes realistic dispersion. Specifically, we assume:

$$V_t^{\text{market}} = V_t^{\text{agent}} \cdot \exp(\varepsilon_t), \quad \varepsilon_t \sim \mathcal{N}(0, \sigma)$$

That is, the market volume is generated via a log-normal distribution with multiplicative noise:

- V_t^{agent}: the number of shares the agent plans to trade in bucket t
- σ: log-scale volatility controlling the randomness in market activity (e.g., 0.2)

This formulation ensures positivity and skewness (which is realistic for volume), and mimics the noisy nature of real market participation.
The simulated market volume fraction used for dynamic volume prediction is then computed as:

$$\text{VolumeFraction}_t = \frac{V_t^{\text{market}}}{\text{TotalShares}}$$

This allows the agent's dynamic volume model to adjust future trading based on how the market is behaving.

Summary: The market price path is generated using a noisy geometric process starting from an average of early mid-quotes, while market volume is modeled as a log-normal perturbation of the agent's trade size — reflecting market responsiveness and uncertainty.

4.3 Find Optimal κ and Cost

- **Code:** `Code/Project/src/stimulation.py`
- **Function:** `get_optimal_kappa_and_cost_from_trajectory`
- **Unit Test:** `Code/Project/test/test_optimal_kappa_cost.py`

The Almgren–Chriss model proposes that we optimally schedule our trades over time to minimize a combination of two factors:

- **Market Impact:** The cost of moving the market price due to one's own trading.
- **Execution Risk:** The risk of price changes during the execution horizon. (As Price Impact later continued.)

This trade-off is encoded in a parameter κ (kappa), defined as:

$$\kappa = \sqrt{\frac{\lambda \sigma^2}{\eta}}$$

where:

- λ is the risk aversion coefficient — higher values mean the trader is more concerned with uncertainty.
- σ^2 is the variance (square of volatility) of the asset's price.

- η is the temporary market impact parameter — a measure of how much the price moves when we trade.

Using κ, the optimal remaining shares to trade at time t_k out of a total T periods is given by:

$$x_k = \frac{\sinh(\kappa(T - t_k))}{\sinh(\kappa T)} \cdot X$$

This formula defines a trajectory that tells us how many shares should remain untraded at each point in time. The corresponding trade volume in each period is the difference between x_k and x_{k+1}. For large κ, the schedule front-loads trades to reduce future risk; for small κ, it trades more uniformly.

Objective in Simulation: Since we do not analytically know the best κ under realistic, noisy price paths, we simulate different κ values and select the one that minimizes the following cost:

$$\text{Cost} = |\text{My VWAP} - \text{Market VWAP}|$$

- **My VWAP:** This is the volume-weighted average price of the execution schedule, where prices include market impact. We compute it using:

$$\text{My VWAP} = \frac{\sum_t (\text{Price}_t + \text{Impact}_t) \cdot \text{Trade}_t}{\sum_t \text{Trade}_t}$$

- **Market VWAP:** This is estimated simply as the mean of the future mid-prices, under the assumption of uniform market trading, since in the absence of reliable volume forecasts, the average price approximates the expected market VWAP.
- **Deviation:** For each simulated κ, we compute the absolute difference between these two VWAPs.
- **Risk:** We approximate the risk or uncertainty by computing the *variance* of deviations across all κ values tested. This variance represents how sensitive our execution performance is to the choice of κ.

Special Handling for Final Buckets:

- In the **second-last bucket**, only one bucket remains in the future. Since we must trade all remaining shares in that last bucket, there is no uncertainty in future execution. Hence, the risk is effectively zero.
- In the **last bucket**, the simulation ends, and all remaining shares are traded unconditionally. Thus, again, risk is zero, and no optimization is required.

Optimization Procedure (Numerical):
To find the optimal κ, we use the following numerical process:

- Generate candidate trading schedules using sinh-shaped weights for a grid of κ values.
- For each schedule, compute:
  - My VWAP (with impact).
  - Market VWAP (simple mean of future prices).
  - Deviation = |My VWAP − Market VWAP|
- Select the κ that results in the smallest deviation.
- Estimate risk as the variance of deviations across all tested κ values.

This allows us to select a κ that not only fits the current price path but also minimizes deviation from the ideal market price, reflecting both cost and risk considerations.

Unit Test:

- `.py`:

4.4 Execution Price (Market_Price + Impact)

- **Code:** `Code/Project/src/stimulation.py`
- **Function:** `compute_execution_impact_with_decay`
- **Unit Test:** `Code/Project/test/est_execution_impact_with_decay.py`

Once the optimal trading trajectory is determined, we must compute the impact of trading on the asset price. In the Almgren–Chriss model, the trader's execution price deviates from the market mid-price due to **temporary price impact**, which grows nonlinearly with the rate of trading.

Trading Rate as a Derivative:
The model assumes that the optimal remaining shares $x(t)$ follow a sinh-shaped curve. Thus, the rate of trading — i.e., how fast we sell shares — is given by the derivative:

$$v_t = \frac{dx}{dt} = -\frac{\kappa \cdot \cosh(\kappa(T-t))}{\sinh(\kappa T)} \cdot X$$

where:

- v_t is the trading rate at time t.
- X is the total number of shares to trade.
- κ is the risk-aversion-weighted shape parameter.

The negative sign reflects that $x(t)$ is decreasing — we are liquidating.

Impact Model:
At any time t, the **temporary price impact** from trading is assumed to follow a power-law of the trading rate:

$$\text{Impact}_t^{\text{current}} = \eta \cdot \sigma \cdot |v_t|^\beta$$

where:

- $\eta = 0.142$
- $\sigma = 0.05$
- $\beta = 0.5$ controls the nonlinearity — typically $\beta = 0.5$ or 1.

This formulation captures the intuition that *larger trades move prices more*, but not necessarily in a linear fashion.

Residual Impact from Previous Trades:
Since market impact doesn't fully disappear after each bucket, we include a residual component from the prior time step's trading rate v_{t-1}. We assume that this impact **decays linearly** over time. Since each bucket is 30 minutes long, we model the residual as being halved in the next bucket:

$$\text{Impact}_t^{\text{residual}} = 0.5 \cdot \eta \cdot \sigma \cdot |v_{t-1}|^\beta$$

This represents the leftover temporary impact from the previous period, which is still affecting the price at the start of the current bucket.

Total Execution Impact:
Hence, the total price impact at time t is the sum of current and decayed prior impact:

$$\text{Impact}_t = \eta \cdot \sigma \cdot |v_t|^\beta + 0.5 \cdot \eta \cdot \sigma \cdot |v_{t-1}|^\beta$$

This impact is then added to the market mid-price to yield the execution price at time t:

$$\text{ExecutionPrice}_t = \text{MidPrice}_t + \text{Impact}_t$$

This model captures both the instantaneous effect of a trade and the residual "footprint" left by past trades, reflecting the reality of how large orders influence the market over time.

4.5 Guaranteed VWAP Cost

t the end of the simulation, we evaluate the quality of our execution using the **Guaranteed VWAP Cost**, which measures how closely we matched the market price while accounting for risk.

1. Agent's VWAP
The agent's VWAP (Volume Weighted Average Price) is computed based on the actual execution prices and volumes in each bucket:

$$\text{My VWAP} = \frac{\sum_t P_t^{\text{exec}} \cdot Q_t}{\sum_t Q_t}$$

where:

- P_t^{exec}: execution price at time t (includes impact).
- Q_t: number of shares traded at time t.

2. Market VWAP
The market VWAP is computed using the simulated mid-prices and simulated market volumes:

$$\text{Market VWAP} = \frac{\sum_t P_t^{\text{mid}} \cdot V_t^{\text{market}}}{\sum_t V_t^{\text{market}}}$$

This serves as a benchmark — the price a "typical" passive trader might have received.

3. Guaranteed VWAP Cost

To capture both execution error and the uncertainty involved in trading, we define the total cost as:

$$\text{Guaranteed Cost} = |\text{My VWAP} - \text{Market VWAP}| + \lambda \cdot \text{Risk}$$

where:

- λ: a risk aversion parameter controlling the tradeoff between price deviation and execution risk.
- Risk: approximated as the variance in potential cost from the execution strategy (estimated during κ optimization).

Unit Analysis
Understanding the units involved clarifies why this formulation is coherent:

- **VWAP:** Both My VWAP and Market VWAP are measured in dollars per share ($).
- **Risk:** Defined as a variance of cost, thus has units of dollars squared (^{2}).
- **Risk Aversion Coefficient λ:** Must cancel the units of risk. Hence:

$$\lambda \in \mathbb{R}^+ \quad \text{with units } \$^{-1}$$

Therefore, both terms in the cost function:

$$|\text{My VWAP} - \text{Market VWAP}| \in \$ \quad \text{and} \quad \lambda \cdot \text{Risk} \in \$$$

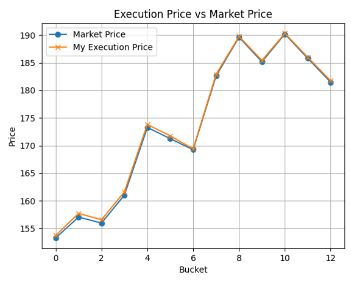
are additive and result in a total cost in dollars — making the Guaranteed VWAP Cost interpretable as a monetary penalty.

Interpretation: The cost quantifies the dollar deviation from market execution, penalized by the uncertainty introduced by trading too quickly or too slowly.

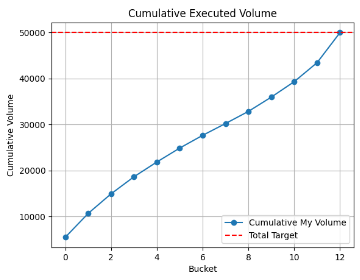
5 Results and Analysis

I ran a simulation for the stock **SPY** using **500,000 shares**. The results are presented below. The stock symbol and the number of shares can be adjusted to simulate different scenarios.

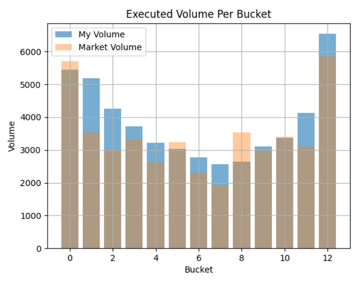
5.1 Visualization Grid



(a) Execution vs. Market Price
Inference: Execution prices closely follow market prices, with minimal slippage. Minor deviations are observed early in the session, but alignment improves as the execution progresses.



(b) Cumulative Executed Volume
Inference: The trade schedule achieves complete order fulfillment with smooth, consistent accumulation throughout the trading horizon.



(c) Executed Volume vs. Market Volume
Inference: Executed volumes adapt to market activity per bucket, demonstrating responsiveness to intra-day volume fluctuations.

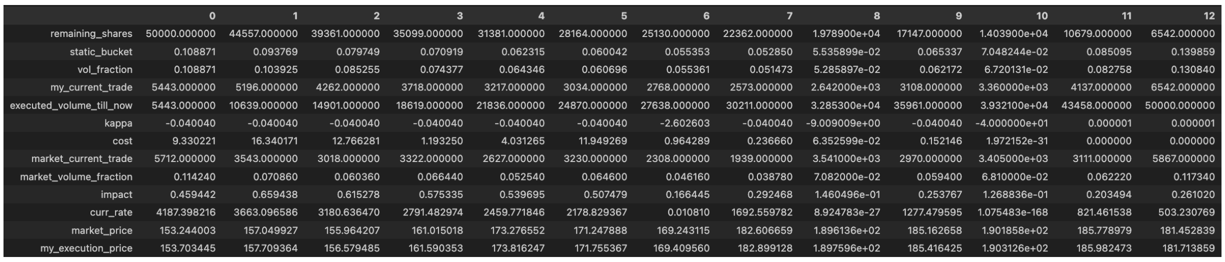
Figure 2: Key visualizations capturing price evolution, execution smoothness, and volume responsiveness across intraday buckets.

5.2 VWAP Execution Simulation Summary

Metric	**Value**
Market VWAP	\$172.7142
My VWAP	\$172.3850
Execution Risk	\$57.0271
Guaranteed Cost	\$57.3563

Table 3: VWAP Execution Summary Metrics.
Inference: The executed VWAP is within \$0.33 of the market VWAP, indicating tight tracking. The total cost is primarily driven by the modeled execution risk, reflecting the price uncertainty associated with dynamic adjustment and market impact.

5.3 Bucket-Wise Execution Log



	0	1	2	3	4	5	6
remaining_shares	50000.000000	44557.000000	39361.000000	35099.000000	31381.000000	28164.000000	25130.000000
static_bucket	0.108871	0.093769	0.079749	0.070919	0.062315	0.060042	0.055353
vol_fraction	0.108871	0.103925	0.085255	0.074377	0.064346	0.060696	0.055361
my_current_trade	5443.000000	5196.000000	4262.000000	3718.000000	3217.000000	3034.000000	2768.000000
executed_volume_till_now	5443.000000	10639.000000	14901.000000	18619.000000	21836.000000	24870.000000	27638.000000
kappa	-0.040040	-0.040040	-0.040040	-0.040040	-0.040040	-0.040040	-2.602603
cost	9.330221	16.340171	12.766281	1.193250	4.031265	11.949269	0.964289
market_current_trade	5712.000000	3543.000000	3018.000000	3322.000000	2627.000000	3230.000000	2308.000000
market_volume_fraction	0.114240	0.070860	0.060360	0.066440	0.052540	0.064600	0.046160
impact	0.459442	0.659438	0.615278	0.575335	0.539695	0.507479	0.166445
curr_rate	4187.398216	3663.096586	3180.636470	2791.482974	2459.771846	2178.829367	0.010810
market_price	153.244003	157.049927	155.964207	161.015018	173.276552	171.247888	169.243115
my_execution_price	153.703445	157.709364	156.579485	161.590353	173.816247	171.755367	169.409560

	7	8	9	10	11	12
remaining_shares	22362.000000	1.978900e+04	17147.000000	1.403900e+04	10679.000000	6542.000000
static_bucket	0.052850	5.535899e-02	0.065337	7.048244e-02	0.085095	0.139859
vol_fraction	0.051473	5.285897e-02	0.062172	6.720131e-02	0.082758	0.130840
my_current_trade	2573.000000	2.642000e+03	3108.000000	3.360000e+03	4137.000000	6542.000000
executed_volume_till_now	30211.000000	3.285300e+04	35961.000000	3.932100e+04	43458.000000	50000.000000
kappa	-0.040040	-9.009009e+00	-0.040040	-4.000000e+01	0.000001	0.000001
cost	0.236660	6.352599e-02	0.152146	1.972152e-31	0.000000	0.000000
market_current_trade	1939.000000	3.541000e+03	2970.000000	3.405000e+03	3111.000000	5867.000000
market_volume_fraction	0.038780	7.082000e-02	0.059400	6.810000e-02	0.062220	0.117340
impact	0.292468	1.460496e-01	0.253767	1.268836e-01	0.203494	0.261020
curr_rate	1692.559782	8.924783e-27	1277.479595	1.075483e-168	821.461538	503.230769
market_price	182.606659	1.896136e+02	185.162658	1.901858e+02	185.778979	181.452839
my_execution_price	182.899128	1.897596e+02	185.416425	1.903126e+02	185.982473	181.713859

Figure 3: Bucket-wise execution trace showing per-bucket volume, kappa, cost, impact, and price evolution.

The table above provides a detailed snapshot of the internal state of the simulation across all 13 half-hour trading buckets. Notable trends include:

- A consistent decrease in `remaining_shares`, confirming orderly execution over time.
- Gradual adaptation of `vol_fraction` based on static and dynamic predictions.

- `kappa` values adjusting over time to minimize risk-adjusted deviation from market VWAP.
- The `impact` and `curr_rate` diminish as we approach the final buckets, consistent with decelerating trade rates.
- Execution prices (`my_execution_price`) closely track market prices, with controlled deviations reflecting modeled impact.

These logs provide interpretability for model decisions at each step of the execution and serve as a diagnostic tool for debugging, tuning, or extending the framework.

6 Conclusion

This project presents a comprehensive and modular simulation framework for Guaranteed VWAP execution that realistically captures the key elements of institutional trade execution under uncertainty. By integrating a static Bayesian Dirichlet volume model with a dynamic Bayesian Ridge adjustment mechanism, the framework allows for adaptive estimation of market volume across intraday time buckets. Market dynamics are further modeled through stochastic price evolution and log-normal volume perturbations, while execution prices incorporate nonlinear temporary market impact and residual decay.

A key feature of the framework is the dynamic replanning of trade trajectories using sinh-shaped execution profiles. At each decision point, the agent evaluates a range of κ-parameterized schedules and selects the one that minimizes a risk-adjusted cost function, defined as the deviation from market VWAP plus a risk penalty proportional to the variance of potential outcomes. This reflects the broker's objective in GVWAP contracts: to balance the cost of guaranteeing execution against the uncertainty of market behavior.

Overall, the framework successfully models the essential trade-offs in execution strategy design, including volume predictability, price impact, risk aversion, and real-time trajectory adjustment. It provides a valuable foundation for testing, analyzing, and optimizing algorithmic execution under guaranteed pricing constraints. The modular structure also supports extensions such as permanent impact modeling, alternative risk metrics, and reinforcement learning-based trajectory control.