GoQuant Real-Time Trade Simulator Documentation

Abhiraj Kumar

1 Model Selection and Input Parameters

In this project, we calculate six real-time trading metrics:

- **Expected Slippage**
- **Expected Fees**
- **Expected Market Impact**
- **Net Cost**
- **Maker/Taker Proportion**
- **Internal Latency**

Each metric was modeled using carefully selected machine learning or rule-based models. Below is a breakdown of the chosen models and their input parameters.

1. Expected Slippage

Model: Quantile Regression (50th percentile)
Inputs:

- **Spread:** Difference between best ask and best bid, indicates immediate execution cost
- **USD Amount:** Order size in USD — larger orders generally cause more slippage
- **Order Book Depth:** Total quantity available in top 10 levels, reflecting liquidity

Why Quantile Regression? Unlike linear regression, quantile regression captures the median slippage, making it robust to outliers — an essential quality in volatile financial markets. Since extreme values in slippage (due to sudden market impact) can bias the mean, we focus on the 50th percentile to better represent the typical outcome.

2. Expected Fees

Model: Rule-based fee model depending on tier & maker/taker classification.
Inputs: Fee tier, order type

3. Expected Market Impact

Model: Almgren-Chriss market impact model using dynamic programming.
Inputs:

- Order size
- Volatility
- Impact coefficients (η, γ)
- Risk aversion

4. Net Cost

Model: Sum of slippage, fees, and market impact

5. Maker/Taker Proportion

Model: Logistic Regression
Inputs: spread,imbalance,volatility,amount,side,order type

6. Internal Latency

Model: Custom timer class using `time.perf_counter()` to measure processing time per tick

2 Regression Techniques and Results

Quantile Regression for Slippage

We used quantile regression to estimate the 50th percentile (median) slippage. This helps predict the most typical slippage value instead of being skewed by large rare events.

Model Coefficients and Results

- **Mean Squared Error (MSE):** 5.39e-05
- **R-squared:** 0.798
- **Coefficients:** [Spread: 0, USD Amount: 0.0109, Depth: 3.46e-07]
- **Intercept:** 0.00538

Interpretation:

- The model explains **80% of the variance** in median slippage.
- Positive coefficient on USD amount shows larger trades incur higher slippage.
- Positive depth coefficient shows that deeper order books slightly reduce slippage.

Why Quantile Regression?

- **Robust to outliers:** Financial slippage distributions are often skewed.
- **No assumption of normality:** Unlike linear regression.
- **Tail quantiles:** Can be extended later to 90th percentile for worst-case slippage.

Logistic Regression for Maker/Taker Prediction

Classification Report:

- **Accuracy:** 100%
- **Precision / Recall / F1:** All perfect

Interpretation: Our features (like imbalance and volatility ,market type) very clearly separate maker vs taker trades. Logistic regression is well-suited for binary classification and interpretable.

3 Market Impact Calculation Methodology

We use the dynamic programming version of the Almgren-Chriss optimal execution model. The total cost is modeled as:

1. Temporary Impact (Execution Price Slippage)

$$\text{Temporary cost at step } i = \eta \left(\frac{x_i}{\Delta t} \right)^{\alpha}$$

Where:

- η: temporary impact coefficient
- α: impact exponent (≈ 1)

2. Permanent Impact (Price Drift)

$$\text{Permanent cost} = \gamma \cdot X$$

Where γ is the permanent impact coefficient.

3. Volatility Risk (Variance Penalty)

$$\text{Risk term} = \frac{\lambda}{2} \sum_{i=1}^{N} \sigma^2 \cdot x_i^2$$

Where λ is the trader's risk aversion.

4. Total Cost

$$\text{Total Cost} = \sum_{t=1}^{T} \left[\eta \left(\frac{x_t}{\Delta t} \right)^{\alpha} + \gamma \left(\frac{x_t}{\Delta t} \right)^{\beta} + \frac{\lambda \sigma^2 \Delta t}{2} \cdot (X_t)^2 \right]$$

- η: Temporary impact coefficient (e.g., due to slippage)
- γ: Permanent impact coefficient (market moves)
- α, β: Empirical impact exponents
- σ: Market volatility
- X_t: Shares remaining to trade at time t
- λ: Risk aversion coefficient

We discretize the total order size into bins and use a bottom-up DP table to compute the minimum cost for each amount over T time steps.

Dynamic Programming Code Snippet

```
dp[t][x] = min over v of:
    dp[t-1][x - v] +
    eta * (v / dt)**alpha +
    gamma * (v / dt)**beta +
    0.5 * lambda * sigma**2 * dt * (x - v)**2
```

Advantage: This avoids brute force optimization over continuous space while still providing optimal strategies and accurate cost modeling.

4 Performance Optimization Approaches

1. Internal Latency Timer

We measure latency per tick using Python's high-resolution timer:

```
import time
start = time.perf_counter()
# process order book
end = time.perf_counter()
latency = end - start
```

2. Discretized Dynamic Programming for Almgren-Chriss

By converting order sizes to discrete bins (instead of iterating over every integer value), memory and time complexity are greatly reduced:

```
bin_size = max(1, order_size // 1000)
bins = np.arange(0, order_size + 1, bin_size)
```

3. Efficient CSV Logging

WebSocket order book updates are saved to CSVs in batches rather than writing to disk on every tick, reducing I/O bottlenecks.

4. Multiprocessing Ready Architecture (Future Work)

Code structure modularized to allow parallel prediction of slippage, market impact, and latency in different processes.

5. Overflow-Safe Dynamic Programming

Memory usage reduced via NumPy arrays with clipping:

```
dp = np.full((steps+1, max_shares+1), np.inf)
dp[0, shares] = 0  # Initialize base case
```

Benefit: Avoids crashes with large trade sizes like 10,000.

6. Fast Feature Computation

Pre-computed order book stats for real-time updates:

```
mid_price = (best_bid + best_ask) / 2
spread = best_ask - best_bid
```

Benefit: Enables sub-millisecond latency.

5 Conclusion

We built a real-time trade simulator using live WebSocket data from GoQuant, and modeled each output parameter using appropriate models:

- Quantile regression for robust slippage estimation
- Logistic regression for maker/taker prediction
- Almgren-Chriss dynamic programming for optimal market impact estimation

- Internal latency timer to benchmark system responsiveness

Our design choices are backed by theory, results, and finance-specific considerations, enabling both accuracy and performance in high-frequency trade simulation.