Modeling Temporary Impact g(x) from Market Data-

I modeled temporary price impact gt(x) as the execution-price premium (vs. mid) paid to trade size x at time t. Using top-10 MBP-10 snapshots for three tokens across 21 days, I constructed observations of g(x) and compared non-linear, state-dependent models against classical parametric curves. I explored more flexible approaches that account for the concavity of the impact curve induced by the depth ladder and the strong dependence on spread, depth, imbalance, volatility, and time-of-day. Out-of-sample tests (GroupKFold by day/ticker) revealed that tree ensembles (Gradient Boosting, Random Forest) provided the most accurate predictions ($R^2 \approx 0.77$).

A. Constructing gt(x) from the order book (`h.py`)

Goal. For each snapshot t, I computed the slippage a market buy of size x would have incurred by sweeping the visible ask ladder (analogous for sells).

1. **Book features.** For each snapshot I computed the mid mt, the best-level spread, the total top-10 depth Vt (both sides), the best-level imbalance, a rolling intraday volatility proxy, and the hour-of-day.

2. **Simulated sweep.** For sizes x∈{100,200,…,2000, I accumulated fills from L1 upward and computed the average execution price p¯t(x).

3. **Temporary impact.** I defined gt(x)=p¯t(x)−mt.

Output. I wrote an “enhanced slippage” CSV for each ticker-day with columns:
`timestamp, size, slippage (=gt(x)), vol_ratio (=x/depth),spread, depth, imbalance, volatility, hour_of_day`
These files fed the modeling stage.

Performance. I vectorized the sweep in a Numba-compiled loop over levels and sizes, which allowed me to process tens of millions of snapshots efficiently and write compact per-snapshot/per-size observations.

Non-linear, state-dependent modeling of gt(x) (`I.py`)

How I designed the study-

- **Target.** I modeled gt(x)=p¯t(x)−mt in price units.
- Used the liquidity ratio rt=x/Vt.
- **Features.** I engineered
  {x/V^0.5, x/V, logx, spread, depth, imbalance, volatility, hour_of_day}
- **Aggregation.** I averaged each ticker-day into **20 size buckets** and tagged each with a `file_id`. Concatenating all days/tickers yielded ~1,280 rows.
- **Validation.** I used **GroupKFold (5 folds) by `file_id`**, so each test fold was a completely unseen day/ticker and there was no leakage.

Models I compared-

- **Parametric, non-linear in size/liquidity**
  - Square-root: $g = \sigma\sqrt{x/V}$
  - Logarithmic: $g = a + b\log x$
  - Quadratic in size: $g = b_1 x + b_2 x^2$
  - **Power-law (liquidity-scaled):** $g = \alpha\,(x/V)^{\beta}$ (non-linear fit for α, β)
- **Linear w/ regularization on full state**
  - Ridge, ElasticNet
- **Non-parametric, state-aware**
  - Random Forest, Gradient Boosting, XGBoost, KNN, SVR

Equations for Random Forest, Gradient Boost-

$$\hat{g}_t(x) = \frac{1}{B}\sum_{b=1}^{B}\sum_{m} c_{b,m}\,\mathbf{1}\{\phi_t(x) \in R_{b,m}\} \quad \text{(RF)}$$

$$\hat{g}_t(x) = F_0 + \sum_{k=1}^{K}\sum_{m} \eta\,\gamma_k\,d_{k,m}\,\mathbf{1}\{\phi_t(x) \in R_{k,m}\} \quad \text{(GB)}$$

with $\phi_t(x) = [\sqrt{x/V_t}, x/V_t, \log x, \text{spread}_t, V_t, \text{imbalance}_t, \text{vol}_t, \text{hour}_t]$.

Results-

```
Cross-validated Model Performance:
        model  mean_test_mse  std_test_mse  mean_test_r2  std_test_r2
GradientBoost       2.389680      0.835836      0.774102     0.074274
 RandomForest       2.463870      1.038403      0.766079     0.100258
      XGBoost       2.834322      0.827539      0.725054     0.099264
        Ridge       4.420681      1.165765      0.589537     0.054985
   Linear x/V       5.726270      1.539792      0.470070     0.070683
 PowerLaw x/V       5.733184      1.579140      0.469843     0.075209
  Square-root       6.728375      1.438590      0.373448     0.039820
    Quadratic       6.944874      1.335428      0.349971     0.041279
   ElasticNet       7.683910      1.540505      0.284034     0.026587
  Logarithmic       8.325828      1.400881      0.218657     0.031837
          SVR      10.848014      2.022484     -0.012478     0.035985
```

Interpretation and Model Choice-
Top predictive accuracy-

- **Gradient Boosting** achieved the **lowest MSE (2.39)** and **highest mean $R2=0.774$**, with a moderate fold-to-fold variability ($\sigma R2=0.074$).
- **Random Forest** was a close second ($R2=0.766$, but higher variance), and **XGBoost** followed ($R2=0.725$).

Gradient Boosting is the best out-of-sample predictor of gt(x). It flexibly captures both the concave size-dependence and the interactions with spread, depth, imbalance, volatility, and time-of-day. To guard against overfitting, I used 5-fold GroupKFold cross-validation by day/ticker and kept tree depth and learning-rate settings moderate (e.g.100 trees, default max depth), ensuring the model generalizes across unseen days rather than memorizing noise. Its CV variability is modest, giving confidence in its stability.