Advanced Mathematical, ML, and Infrastructure Refinements for NEPSE Quant System

> **Purpose:** This addendum captures critical last-mile refinements that materially improve robustness, calibration, and survivability of a probabilistic trading system in a thin, regime-driven emerging market. This document is complementary to the main blueprint and assumes full familiarity with it.

---

1. Mathematical Refinements for Thin-Market Dynamics

1.1 Asymmetric Power-Law Market Impact

In illiquid markets, price impact is asymmetric: sell pressure in stress regimes produces larger and faster price dislocations than buy pressure. Model impact separately for buys and sells.

Let trade size be q_t (signed, positive = buy, negative = sell), ADV be average daily volume.

Define asymmetric impact:

$$\text{Impact}(q_t) = \begin{cases} k_1^+ \left(\frac{q_t}{ADV}\right)^{\alpha^+} + k_2^+ \left(\frac{q_t}{ADV}\right)^2 & q_t > 0 \ k_1^- \left(\frac{|q_t|}{ADV}\right)^{\alpha^-} + k_2^- \left(\frac{|q_t|}{ADV}\right)^2 & q_t < 0 \end{cases}$$

Where typically: - $\alpha^- > \alpha^+$ - $k^- > k^+$ during stress regimes

Regime-conditioned parameters:

$$(k_1^{\pm}, \alpha^{\pm}) = f(s_t)$$

Calibrate parameters using regime-filtered execution data via rolling nonlinear least squares or recursive least squares (RLS).

1.2 Hierarchical Bayesian Shrinkage for Thin Stocks

For stocks with sparse trading, estimate parameters via partial pooling.

Example: volatility or impact coefficient θ_i for stock i:

$$\theta_i \sim \mathcal{N}(\mu_{\text{NEPSE}}, \tau^2)$$

Observed estimate:

$$\hat{\theta}_i \mid \theta_i \sim \mathcal{N}(\theta_i, \sigma_i^2)$$

Posterior mean:

$$\mathbb{E}[\theta_i \mid \hat{\theta}_i] = w_i\hat{\theta}_i + (1 - w_i)\mu_{\text{NEPSE}}, \quad w_i = \frac{\tau^2}{\tau^2 + \sigma_i^2}$$

As data accumulates ($\sigma_i^2 \downarrow$), estimates naturally de-shrink.

Use for: - Impact parameters - Volatility persistence - Regime transition probabilities

---

1.3 CVaR-Constrained Portfolio Optimization

Replace stop-loss logic with tail-aware optimization.

Let portfolio returns be $R_p = w^\top R$.

Define CVaR at level α:

$$\text{CVaR}_\alpha(R_p) = \mathbb{E}[R_p \mid R_p \leq \text{VaR}_\alpha]$$

Optimization problem:

$$\max_w \ \mathbb{E}[R_p] - \lambda \cdot \text{CVaR}_\alpha(R_p)$$

Subject to: - Sector exposure caps - Liquidity constraints - Gross and net exposure limits

Solved via linear programming using scenario simulation from regime-weighted predictive distributions.

---

2. Programming & Infrastructure Enhancements

2.1 Stateful LOB Matching Engine (Numba/Cython)

The LOB simulator must preserve event ordering and queue state.

Key state variables: - Queue depth at each price level - Order arrival timestamps - Agent queue position

Fill probability model:

$$\mathbb{P}(\text{fill within } \Delta t) = 1 - \exp\left(-\frac{V_{\text{ahead}}}{\lambda_{\text{market}}}\right)$$

Where: - V_{ahead} = volume ahead in queue - λ_{market} = arrival rate of opposing market orders

Numba/Cython used to process 10^6+ events/sec in backtests and simulation.

2.2 Feature Parity Auditing (Offline vs Live)

Every feature x_t must satisfy:

$$x_t^{\text{train}} \equiv x_t^{\text{live}}$$

Implementation: - Deterministic rolling windows - Fixed seed ordering - Numerical hash (xxhash / SHA256) on feature vectors

$$H(x_t^{\text{train}}) = H(x_t^{\text{live}})$$

Mismatch triggers hard fail and trading halt.

2.3 Hardware Timestamping & Clock Sync

Queue-sensitive strategies require sub-millisecond sync.

- Use **PTP (IEEE 1588)** with hardware NIC timestamping
- Target clock skew: < 50 microseconds

Execution engine timestamps: - Market data arrival - Feature snapshot - Order submission - Exchange ACK

Latency budget must be explicitly tracked and logged.

---

3. Machine Learning Enhancements

3.1 Regime-Conditional Feature Importance

Estimate importance conditional on regime:

$$I_j^{(k)} = \mathbb{E}[\Delta L \mid s_t = k]$$

Where ΔL is loss increase under permutation of feature j.

Findings often show: - Ownership / promoter features dominate in mean-reversion regimes - Liquidity and volatility dominate in stress regimes

---

3.2 Proper Probabilistic Scoring (CRPS)

Train ML models to optimize distributional accuracy.

Continuous Ranked Probability Score:

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(z) - 1\{z \geq y\})^2 dz$$

CRPS penalizes: - Overconfident wrong predictions - Underconfident correct predictions

Used as primary objective for: - Quantile regression - NGBoost / probabilistic boosting

---

3.3 Ensemble Anomaly Detection

Combine: 1. Deterministic rules (price jumps, zero volume) 2. Isolation Forest trained on clean historical states

Final anomaly score:

$$A_t = \max(A_t^{\text{rules}}, A_t^{\text{IF}})$$

If $A_t > \tau$, data is quarantined and no signal is generated.

---

4. Monitoring & Drift Detection

4.1 Feature Population Stability Index (PSI)

For feature x:

$$\text{PSI} = \sum_i (p_i - q_i) \log\left(\frac{p_i}{q_i}\right)$$

- p_i: training distribution
- q_i: live distribution

PSI > 0.2 triggers investigation; > 0.3 triggers model retraining or shutdown.

---

5. Summary of Enhancements

Category	Enhancement	Benefit
Math	Asymmetric Impact	Captures sell-side fragility
Math	Hierarchical Shrinkage	Stabilizes thin-stock estimates
Risk	CVaR Optimization	Controls tail losses
Infra	Numba/Cython LOB	Realistic execution simulation
Infra	PTP Sync	Queue-position accuracy
ML	CRPS Objective	Better probabilistic calibration
ML	Regime Feature Importance	Structural interpretability
Ops	PSI Monitoring	Early regime drift detection

End of addendum.