Intraday Tick Simulator

Mathematical Foundations

Philippe Damay

di.simtick Module for KDB-X

January 21, 2026

Abstract

This document provides a comprehensive mathematical framework of an intraday tick simulator that generates realistic trade and quote data. The simulator combines a self-exciting Hawkes process for modeling trade arrival times with Geometric Brownian Motion (GBM) and Merton jump-diffusion for price dynamics.

Contents

1 Introduction

Market microstructure simulation requires capturing several stylized facts of high-frequency financial data:

- **Trade clustering**: Trades tend to arrive in bursts rather than uniformly
- **Intraday seasonality**: Trading activity varies throughout the day (high at open/close, low at midday)
- **Price continuity**: Prices follow continuous stochastic processes with occasional jumps
- **Bid-ask dynamics**: Quotes must be consistent with executed trades

Our simulator addresses these requirements through a modular architecture:

- **Arrival times**: Hawkes process with time-varying baseline intensity
- **Price dynamics**: GBM or Merton jump-diffusion at irregular time intervals
- **Trade sizes**: Lognormal distribution
- **Quote generation**: Consistent bid-ask spreads with realistic microstructure

2 Random Number Generation

The simulator requires standard normal random variates. We employ the Box-Muller transform, which converts pairs of uniform random variables to pairs of independent standard normals.

Box-Muller Transform: Let $U_1, U_2 \sim \text{Uniform}(0,1)$ be independent. Define:

$$R = \sqrt{-2\ln U_1} \tag{1}$$

$$\Theta = 2\pi U_2 \tag{2}$$

Then

$$Z_1 = R\cos\Theta \tag{3}$$

$$Z_2 = R\sin\Theta \tag{4}$$

are independent standard normal random variables, i.e., $Z_1, Z_2 \sim \mathcal{N}(0,1)$. The Box-Muller transform produces valid standard normal variates.

Implementation note: The transform requires an even number of uniform inputs. When n normal variates are needed, we generate $m = 2\lceil (n+1)/2 \rceil$ uniforms and truncate the output.

3 Trade Arrival Times: Hawkes Process

3.1 Model Specification

Trade arrivals exhibit *self-excitation*: each trade increases the probability of subsequent trades, capturing the clustering observed in real markets.

Definition 1 (Hawkes Process)**.** *A univariate Hawkes process $N(t)$ is a counting process with conditional intensity:*

$$\lambda(t) = \lambda_0(t) + \sum_{t_i < t} \alpha e^{-\beta(t-t_i)} \tag{5}$$

where:

– $\lambda_0(t)$ *is the baseline (background) intensity*

– $\alpha > 0$ *is the excitation parameter (jump in intensity per event)*

– $\beta > 0$ *is the decay rate*

– $\{t_i\}$ *are the event times prior to* t

3.2 Stability Condition

Proposition 1 (Stationarity)**.** *The Hawkes process is stationary if and only if the* branching ratio *satisfies:*

$$\frac{\alpha}{\beta} < 1 \tag{6}$$

When $\alpha/\beta \geq 1$, the process explodes (intensity diverges to infinity). The ratio α/β represents the expected number of "offspring" events triggered by each "parent" event.

3.3 Recursive Formulation

For computational efficiency, we maintain the excitation term recursively. Define:

$$R(t) = \sum_{t_i < t} \alpha e^{-\beta(t-t_i)} \tag{7}$$

Then between events:

$$R(t + \Delta t) = R(t) \cdot e^{-\beta \Delta t} \tag{8}$$

Upon event arrival at time t:

$$R(t^+) = R(t^-) + \alpha \tag{9}$$

The total intensity is:

$$\lambda(t) = \lambda_0(t) + R(t) \tag{10}$$

3.4 Time-Varying Baseline Intensity

To capture intraday seasonality, we modulate the baseline intensity:

$$\lambda_0(t) = \mu \cdot s\left(\frac{t}{T}\right) \tag{11}$$

where μ is the base intensity parameter, T is the session duration, and $s : [0, 1] \to \mathbb{R}^+$ is a shape function.

The shape function interpolates between three multipliers using cosine interpolation:

$$s(p) = \begin{cases} m_{\text{mid}} + (m_{\text{open}} - m_{\text{mid}}) \cos\left(\frac{\pi p}{2\tau}\right) & \text{if } p < \tau \\ m_{\text{mid}} + (m_{\text{close}} - m_{\text{mid}}) \sin\left(\frac{\pi(p-\tau)}{2(1-\tau)}\right) & \text{if } p \geq \tau \end{cases} \tag{12}$$

where:

– $p = t/T \in [0, 1]$ is progress through the trading day

– τ is the transition point (0.5 for symmetric U-shape, 0.3 for asymmetric J-shape)

– $m_{\text{open}}, m_{\text{mid}}, m_{\text{close}}$ are the multipliers at open, midday, and close

Cosine interpolation provides smooth transitions compared to linear interpolation, avoiding discontinuities in the intensity derivative.

3.5 Ogata Thinning Algorithm

Simulating a Hawkes process with time-varying intensity requires the *thinning* (or *acceptance-rejection*) method.

Algorithm 1 Ogata Thinning for Hawkes Process

Input: Parameters μ, α, β, duration T, shape function $s(\cdot)$
Output: Event times $\{t_1, t_2, \ldots, t_n\}$

Compute upper bound: $\lambda_{\max} = \mu \cdot \max(s) \cdot \left(1 + 3\frac{\alpha}{\beta}\right)$
Initialize: $t \leftarrow 0$, $R \leftarrow 0$, times $\leftarrow$ []
while $t < T$ **do**
    Draw $U_1 \sim \text{Uniform}(0, 1)$
    $\Delta t \leftarrow -\frac{\ln U_1}{\lambda_{\max}}$ ▷ Exponential with rate $\lambda_{\max}$
    $t \leftarrow t + \Delta t$
    if $t \geq T$ **then**
        break
    end if
    $R \leftarrow R \cdot e^{-\beta \Delta t}$ ▷ Decay excitation
    $\lambda \leftarrow \mu \cdot s(t/T) + R$ ▷ Current intensity
    Draw $U_2 \sim \text{Uniform}(0, 1)$
    if $U_2 < \lambda/\lambda_{\max}$ **then** ▷ Accept with probability $\lambda/\lambda_{\max}$
        Append t to times
        $R \leftarrow R + \alpha$ ▷ Excitation jump
    end if
end while
return times

The thinning algorithm requires $\lambda_{\max} \geq \lambda(t)$ for all t. We compute:

$$\lambda_{\max} = \mu \cdot \max(m_{\text{open}}, m_{\text{mid}}, m_{\text{close}}) \cdot \left(1 + 3\frac{\alpha}{\beta}\right) \tag{13}$$

The factor $(1 + 3\alpha/\beta)$ accounts for excitation bursts. The expected steady-state excitation is α/β; we use $3\times$ this value as a buffer for transient peaks.

Proposition 2. *The thinning algorithm produces a valid realization of the Hawkes process.*

Proof sketch. The algorithm generates a homogeneous Poisson process with rate $\lambda_{\max}$, then accepts each candidate event with probability $\lambda(t)/\lambda_{\max}$. By the thinning theorem, the accepted points form an inhomogeneous Poisson process with intensity $\lambda(t)$. Since each acceptance updates $\lambda(t)$ via the excitation term R, the self-exciting dynamics are correctly captured. □

4 Price Dynamics

4.1 Geometric Brownian Motion

The baseline price model is Geometric Brownian Motion (GBM), which ensures positive prices and captures the multiplicative nature of returns.

Definition 2 (GBM)**.** *Under GBM, the price process S_t satisfies:*

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t \tag{14}$$

where μ is the drift, σ is the volatility, and W_t is a standard Brownian motion.

Applying Itô's lemma to $\ln S_t$:

$$d(\ln S_t) = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma\, dW_t \tag{15}$$

Integrating from t to $t + \Delta t$:

$$S_{t+\Delta t} = S_t \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,\varepsilon\right] \tag{16}$$

where $\varepsilon \sim \mathcal{N}(0, 1)$.

For a sequence of times $\{t_0, t_1, \ldots, t_n\}$ with irregular spacing, we compute:

$$S_{t_i} = S_0 \prod_{j=1}^{i} \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)\Delta t_j + \sigma\sqrt{\Delta t_j}\,\varepsilon_j\right] \tag{17}$$

where $\Delta t_j = t_j - t_{j-1}$ and $\varepsilon_j \sim \mathcal{N}(0, 1)$ are independent.

The drift μ and volatility σ are specified in annualized terms. For a trading session:

$$\Delta t_j^{\text{years}} = \frac{\Delta t_j^{\text{seconds}}}{N_{\text{days}} \times T_{\text{session}}} \tag{18}$$

where N_{days} is the number of trading days per year (typically 252) and T_{session} is the session duration in seconds.

4.2 Merton Jump-Diffusion

To capture discontinuous price movements (e.g., from news events), we extend GBM with compound Poisson jumps.

Definition 3 (Merton Jump-Diffusion)**.** *The price process satisfies:*

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t + (e^J - 1)\, dN_t \tag{19}$$

where:

- N_t *is a Poisson process with intensity* λ_J *(jumps per day)*
- $J \sim \mathcal{N}(\mu_J, \sigma_J^2)$ *is the log-jump size*

For small time intervals, the probability of a jump is:

$$P(\text{jump in } \Delta t) = 1 - e^{-\lambda_J \Delta t} \approx \lambda_J \Delta t \tag{20}$$

The discrete update becomes:

$$S_{t+\Delta t} = S_t \cdot \underbrace{\exp\left[\left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,\varepsilon\right]}_{\text{diffusion}} \cdot \underbrace{\exp\left[\mathbf{1}_{\{U < p_J\}} \cdot (\mu_J + \sigma_J \varepsilon_J)\right]}_{\text{jump}} \tag{21}$$

where:

- $p_J = 1 - e^{-\lambda_J \Delta t^{\text{days}}}$
- $U \sim \text{Uniform}(0, 1)$
- $\varepsilon, \varepsilon_J \sim \mathcal{N}(0, 1)$ independent
- $\mathbf{1}_{\{\cdot\}}$ is the indicator function

5 Trade Quantity Generation

Trade sizes in real markets exhibit a skewed distribution: many small trades and occasional large block trades.

5.1 Lognormal Model

Definition 4 (Lognormal Trade Size)**.** *Trade quantity Q follows:*

$$Q = \max\left(1, \lfloor e^X \rfloor\right), \quad X \sim \mathcal{N}(\mu_Q, \sigma_Q^2) \tag{22}$$

To achieve a target average quantity $\bar{Q}$, we use the lognormal mean formula:

$$\mathbb{E}[e^X] = e^{\mu_Q + \sigma_Q^2/2} = \bar{Q} \tag{23}$$

Solving for μ_Q:

$$\mu_Q = \ln(\bar{Q}) - \frac{\sigma_Q^2}{2} \tag{24}$$

The floor and max operations ensure integer quantities ≥ 1.

5.2 Constant Model

For testing or deterministic scenarios, a constant quantity model is also available:

$$Q = \bar{Q} \tag{25}$$

where $\bar{Q}$ is the `avgqty` parameter. This is selected by setting `qtymodel` to `constant`.

6 Quote Generation

6.1 Methodological Note: Trade-First vs. Quote-First Simulation

Our simulator employs a **trade-first** approach: we generate trade arrivals and prices, then construct quotes that are consistent with these trades. This is a deliberate simplification.

In real markets, the causal flow is reversed:

- Market participants submit **limit orders** to a limit order book (LOB)
- The LOB maintains bid and ask queues at each price level
- **Trades occur** when incoming market orders or aggressive limit orders match against resting orders
- Queue dynamics (arrivals, cancellations, executions) determine price discovery

A fully realistic simulator would model:

- **Order flow**: Poisson or Hawkes arrival of limit/market/cancel orders
- **Queue dynamics**: Orders waiting at each price level, priority by time
- **Price formation**: Mid-price moves when queues deplete
- **Market impact**: Large orders walking through the book
- **Strategic behavior**: Quote stuffing, spoofing, optimal execution

Such models (e.g., Queue-Reactive models, agent-based LOB simulators) are significantly more complex and computationally expensive.

For many applications, the trade-first approach provides adequate realism:

- **Backtesting strategies**: If the strategy does not model queue position or market impact, trade-level data suffices
- **Statistical analysis**: Trade clustering, volatility estimation, and price dynamics are captured
- **System testing**: Database ingestion, time-series processing, and visualization can be tested with synthetic data
- **Computational efficiency**: Orders of magnitude faster than LOB simulation

The key limitation is that our quotes are *ex-post consistent* with trades rather than *causally generative*. This means:

- We cannot simulate realistic queue depletion or price impact
- Spread dynamics are imposed rather than emergent
- The simulator is unsuitable for market-making or optimal execution research

6.2 Quote Generation Algorithm

Given the trade-first approach, our quote generator ensures:

- Each trade has a valid quote immediately preceding it
- The trade price lies within the bid-ask spread
- Quotes evolve as a random walk between trades

6.3 Bid-Ask Spread Model

The spread is modeled as a fraction of the mid-price:

$$\text{spread}_t = s_{\text{base}} \cdot P_t \cdot m_{\text{spread}}(t) \tag{26}$$

where:

- s_{base} is the base spread (e.g., 0.001 = 10 basis points)
- P_t is the current price
- $m_{\text{spread}}(t)$ is an intraday multiplier (spreads widen at open/close)

The spread multiplier interpolates linearly:

$$m_{\text{spread}}(p) = \begin{cases} m^s_{\text{open}} + (m^s_{\text{mid}} - m^s_{\text{open}}) \cdot 2p & \text{if } p < 0.5 \\ m^s_{\text{mid}} + (m^s_{\text{close}} - m^s_{\text{mid}}) \cdot 2(p - 0.5) & \text{if } p \geq 0.5 \end{cases} \tag{27}$$

6.4 Quote Dynamics

For computational efficiency and full vectorization, intermediate quotes use **price interpolation** rather than a sequential random walk. Given consecutive trades at times t_i and t_{i+1} with prices P_i and P_{i+1}, intermediate quotes are generated at evenly spaced times with mid-prices that interpolate toward the upcoming trade:

$$M_k = P_i + \frac{k}{K+1}(P_{i+1} - P_i) + \delta_M \cdot M_k \cdot \varepsilon_k \tag{28}$$

where:

- $k = 1, \ldots, K$ indexes the intermediate quotes within the gap
- K is the number of intermediate quotes (capped at a maximum, e.g., 10)
- δ_M is a small noise constant (e.g., 0.0001)
- $\varepsilon_k \sim \mathcal{N}(0, 1)$

This approach:

- Ensures quotes naturally converge toward where the next trade will execute
- Eliminates sequential dependencies, enabling full vectorization
- Maintains realistic microstructure (quotes "anticipate" trades)

The bid and ask are then:

$$B_k = M_k - \frac{\text{spread}_k}{2} \cdot (1 + 0.1|\varepsilon'_k|) \tag{29}$$

$$A_k = M_k + \frac{\text{spread}_k}{2} \cdot (1 + 0.1|\varepsilon'_k|) \tag{30}$$

where $\varepsilon'_k \sim \mathcal{N}(0, 1)$ adds randomness to the spread width.

6.5 Pre-Trade Quote

For each trade at time t_{trade} with price P_{trade}, we generate a quote at time:

$$t_{\text{quote}} = t_{\text{trade}} - \Delta t_{\text{offset}} - U \cdot \Delta t_{\text{offset}} \tag{31}$$

where Δt_{offset} is the minimum pre-trade offset and $U \sim \text{Uniform}(0, 1)$.

The quote is centered on the trade price:

$$B_{\text{pre}} = P_{\text{trade}} - \frac{\text{spread}}{2} \tag{32}$$

$$A_{\text{pre}} = P_{\text{trade}} + \frac{\text{spread}}{2} \tag{33}$$

This ensures the trade price is executable given the prevailing quote.

7 References

1. Data Intellect Github. “kdbx-modules“.
   https://github.com/DataIntellectTech/kdbx-modules/tree/main

2. KDB-X Documentation. “Modules“.
   https://code.kx.com/kdb-x/modules/module-index.html

3. Bacry, E., Mastromatteo, I., and Muzy, J.F. (2015). “Hawkes processes in finance.” *Market Microstructure and Liquidity*, 1(1), 1550005.