**System Overview**

**Market Environment Classification (Tick-Based)**

Every 5000 ticks, calculate:

1. Order Flow Efficiency = |CVD[0] - CVD[5000]| / Sum(|Volume\_i|)

2. Trade Size Distribution = Large\_Trades / Total\_Trades

3. Spread Volatility = StdDev(Bid\_Ask\_Spread) / Mean(Spread)

Market States:

- TRENDING: Order Flow Efficiency > 0.3 AND Large Trade Ratio > 0.4

- RANGING: Bid/Ask Balance within 45-55% AND Spread stable

- VOLATILE: Spread Volatility > 2.0 OR Sweep frequency > normal

- TOXIC: Spread > 2 ticks OR Quote changes > 100/second

**News Time Handling**

News Windows (ET):

- 8:30 AM ± 10 min: Major economic data

- 10:00 AM ± 10 min: Secondary data

- 2:00 PM ± 5 min: Fed minutes (when applicable)

During news windows:

- Increase minimum trade size filter by 2x

- Require 2x normal edge to enter

- Exit all positions 30 seconds before

**Strategy 1: Order Flow Momentum (OFM)**

*Best for: Institutional accumulation/distribution phases*

**Parameters (5):**

1. **CVD\_Period** = 1000 ticks
2. **Imbalance\_Threshold** = 1500 contracts (net delta)
3. **Large\_Trade\_Size** = 10 contracts minimum
4. **Absorption\_Ratio** = 400 (volume per tick movement)
5. **Trail\_Ticks** = 3

**Entry Conditions:**

python

*# Calculate on every tick*

cvd = cumulative\_volume\_delta(CVD\_Period)

trade\_sizes = [t.size for t in last\_n\_trades(CVD\_Period)]

large\_ratio = len([t for t in trade\_sizes if t >= Large\_Trade\_Size]) / len(trade\_sizes)

absorption = volume\_last\_100\_ticks / price\_range\_ticks

*# Microstructure signals*

bid\_pulling = bid\_size\_decreased > ask\_size\_decreased *# Bullish*

ask\_pulling = ask\_size\_decreased > bid\_size\_decreased *# Bearish*

*# LONG Entry (all conditions must be true):*

1. cvd > Imbalance\_Threshold

2. large\_ratio > 0.35 *# Institutional involvement*

3. absorption < Absorption\_Ratio *# Not hitting resistance*

4. bid\_pulling == True *# Market makers bullish*

5. spread <= 1 tick

*# SHORT Entry (opposite conditions)*

*# Execution:*

- Place limit at best\_bid + 1 (aggressive post)

- If not filled in 100ms, market order

- Cancel if spread widens > 1 tick

**Exit Conditions:**

python

*# Initial Stop:*

Long: Largest bid cluster below entry - 1 tick

Short: Largest ask cluster above entry + 1 tick

*# Dynamic Exit based on order flow:*

if position == LONG:

*# Trail when flow continues*

if cvd\_since\_entry > 0:

stop = max(stop, current\_bid - Trail\_Ticks)

*# Exit on flow reversal*

if cvd\_last\_200\_ticks < -Imbalance\_Threshold/3:

exit\_market()

*# Exit on absorption (resistance)*

if volume\_at\_price\_level > Absorption\_Ratio \* 2:

exit\_limit(current\_bid)

**Position Sizing:**

python

*# Tick-based volatility*

tick\_changes\_per\_minute = count\_tick\_changes(60\_seconds)

volatility\_scalar = min(1.0, 20 / tick\_changes\_per\_minute)

Risk\_Per\_Trade = Account\_Value \* 0.01 \* volatility\_scalar

Tick\_Risk = abs(entry - stop) / tick\_size

Contracts = floor(Risk\_Per\_Trade / (Tick\_Risk \* tick\_value))

**Strategy 2: Microstructure Mean Reversion (MMR)**

*Best for: Sweep exhaustion and liquidity gaps*

**Parameters (5):**

1. **Sweep\_Threshold** = 75 contracts (single aggressive order)
2. **Book\_Imbalance** = 3.0 (bid/ask ratio)
3. **Quiet\_Period** = 200 ticks
4. **Reversion\_Percent** = 0.6
5. **Max\_Heat** = 4 ticks

**Setup Identification:**

python

*# Detect aggressive sweep*

sweep = None

for trade in trades:

if trade.size >= Sweep\_Threshold and trade.aggressor:

levels\_taken = count\_price\_levels(trade)

if levels\_taken >= 3: *# Swept multiple levels*

sweep = {

'price': trade.price,

'direction': trade.side,

'size': trade.size,

'levels': levels\_taken,

'timestamp': trade.time

}

*# Monitor post-sweep activity*

if sweep:

quiet\_ticks = 0

total\_volume = 0

for tick in subsequent\_ticks:

quiet\_ticks += 1

total\_volume += tick.volume

if quiet\_ticks >= Quiet\_Period:

avg\_volume = total\_volume / quiet\_ticks

if avg\_volume < sweep.size \* 0.2: *# Volume dried up*

setup\_ready = True

**Entry Conditions:**

python

*# Book analysis at each price level*

def analyze\_book\_support():

bid\_support = sum([level.size for level in bid\_book[:5]])

ask\_pressure = sum([level.size for level in ask\_book[:5]])

return bid\_support / ask\_pressure

*# LONG Entry (fade down sweep):*

if sweep.direction == 'SELL' and setup\_ready:

if analyze\_book\_support() > Book\_Imbalance:

*# Strong bid support after sweep*

entry\_price = sweep.price + 1\_tick

place\_limit\_order(LONG, entry\_price)

*# SHORT Entry (fade up sweep):*

if sweep.direction == 'BUY' and setup\_ready:

if 1/analyze\_book\_support() > Book\_Imbalance:

*# Strong ask resistance after sweep*

entry\_price = sweep.price - 1\_tick

place\_limit\_order(SHORT, entry\_price)

*# Cancel if not filled within 50 ticks*

**Exit Logic:**

python

*# Fixed stops based on sweep range*

Long: sweep.price - (sweep.levels \* tick\_size)

Short: sweep.price + (sweep.levels \* tick\_size)

*# Target: Reversion percentage*

Long: entry + (Reversion\_Percent \* (entry - sweep.price))

Short: entry - (Reversion\_Percent \* (sweep.price - entry))

*# Time stop*

if ticks\_since\_entry > 500 and pnl < 0:

exit\_market()

*# Max heat protection*

if adverse\_excursion > Max\_Heat:

exit\_market()

**Strategy 3: Liquidity Vacuum Breakout (LVB)**

*Best for: Pre-breakout consolidation with dried up volume*

**Parameters (5):**

1. **Consolidation\_Ticks** = 500
2. **Volume\_Reduction** = 0.3 (vs average)
3. **Range\_Ticks** = 5 (max during consolidation)
4. **Breakout\_Volume** = 100 contracts
5. **Target\_Multiple** = 2.5

**Consolidation Detection:**

python

*# Rolling window analysis*

windows = []

for i in range(0, total\_ticks, Consolidation\_Ticks):

window = ticks[i:i+Consolidation\_Ticks]

window\_stats = {

'high': max([t.price for t in window]),

'low': min([t.price for t in window]),

'volume': sum([t.volume for t in window]),

'range': (high - low) / tick\_size

}

*# Check if consolidating*

if window\_stats['range'] <= Range\_Ticks:

avg\_volume\_prior = mean([w.volume for w in windows[-10:]])

if window\_stats['volume'] < avg\_volume\_prior \* Volume\_Reduction:

consolidation = {

'high': window\_stats['high'],

'low': window\_stats['low'],

'mid': (high + low) / 2,

'start\_tick': i

}

**Entry Triggers:**

python

*# Monitor for breakout*

if consolidation:

for tick in subsequent\_ticks:

*# Volume surge detection*

tick\_cvd = tick.ask\_volume - tick.bid\_volume

*# LONG Breakout:*

if (tick.price > consolidation['high'] and

tick.volume >= Breakout\_Volume and

tick\_cvd > 0): *# Buying pressure*

*# Confirm with next tick*

if next\_tick.price >= tick.price:

enter\_long\_market()

*# SHORT Breakout:*

if (tick.price < consolidation['low'] and

tick.volume >= Breakout\_Volume and

tick\_cvd < 0): *# Selling pressure*

*# Confirm with next tick*

if next\_tick.price <= tick.price:

enter\_short\_market()

**Stop and Target:**

python

*# Stops: Other side of consolidation*

Long: consolidation['low'] - 1\_tick

Short: consolidation['high'] + 1\_tick

*# Targets: Multiple of risk*

Long: entry + (Target\_Multiple \* (entry - stop))

Short: entry - (Target\_Multiple \* (stop - entry))

*# Breakout failure protection*

if ticks\_since\_entry > 100:

if position == LONG and price < consolidation['high']:

exit\_market() *# Failed breakout*

if position == SHORT and price > consolidation['low']:

exit\_market()

**Master Control Layer (Tick-Based)**

**Tick Data Quality Checks:**

python

def validate\_tick\_data():

*# Check for data issues*

if bid > ask:

return False *# Crossed market*

if spread > 5 \* normal\_spread:

return False *# Wide spread*

if time\_since\_last\_tick > 1000ms:

return False *# Stale data*

if bid\_size == 0 or ask\_size == 0:

return False *# No liquidity*

return True

**Strategy Selection:**

python

def select\_strategy(market\_state, tick\_metrics):

*# Toxic market - no trading*

if market\_state == "TOXIC" or not validate\_tick\_data():

return None

*# Clear institutional flow*

if abs(cvd\_5000) > 5000 and large\_trade\_ratio > 0.4:

return "OFM"

*# Post-sweep opportunity*

if recent\_sweep\_detected and ticks\_since\_sweep > 100:

return "MMR"

*# Consolidation breakout setup*

if volatility\_contraction and volume\_decline:

return "LVB"

return None *# No clear opportunity*

**Risk Management:**

python

*# Tick-based risk metrics*

MAX\_TICKS\_PER\_SECOND = 50 *# Above this = news/unusual*

MIN\_BOOK\_DEPTH = 100 *# Contracts each side minimum*

MAX\_SPREAD = 2 *# Ticks*

*# Pre-trade checks*

if ticks\_per\_second > MAX\_TICKS\_PER\_SECOND:

disable\_trading(60\_seconds)

if book\_depth < MIN\_BOOK\_DEPTH:

reduce\_size\_by\_half()

if spread > MAX\_SPREAD:

cancel\_all\_orders()

flatten\_positions()

**Mathematical Validation Framework (Tick Data)**

**1. Microstructure Statistics:**

python

*# Tick-level metrics*

- Fill\_Rate = Filled\_Orders / Total\_Orders

- Adverse\_Selection = mean(price\_1000\_ticks\_later - fill\_price)

- Implementation\_Shortfall = (fill\_price - decision\_price) / decision\_price

*# Acceptance criteria:*

Fill\_Rate > 0.85

Adverse\_Selection < 0.5 \* average\_profit

Implementation\_Shortfall < 0.1 \* tick\_size

**2. Execution Quality:**

python

*# Slippage analysis*

Slippage\_Distribution = histogram(fill\_price - intended\_price)

Expected\_Slippage = mean(slippage)

Slippage\_95th = percentile(slippage, 95)

*# Queue position analysis*

Queue\_Position = (our\_order\_id - first\_order\_id) / total\_orders

Time\_To\_Fill = timestamp\_fill - timestamp\_place

*# Targets:*

Expected\_Slippage < 0.5 ticks

Queue\_Position < 0.3 *# Front of queue*

median(Time\_To\_Fill) < 500ms

**3. Market Impact:**

python

*# Temporary impact*

Temp\_Impact = (price\_at\_fill - price\_at\_decision) / order\_size

*# Permanent impact*

Perm\_Impact = (price\_10min\_later - price\_at\_decision) / order\_size

*# Model: Impact = α√size + β\*participation\_rate*

Run regression to find α, β

*# Capacity constraint*

Max\_Order\_Size = Daily\_Volume \* 0.001 *# 0.1% participation*

**4. Information Ratio:**

python

*# Tick-based Sharpe*

tick\_returns = [exit\_price/entry\_price - 1 for each trade]

tick\_sharpe = mean(tick\_returns) / std(tick\_returns) \* sqrt(trades\_per\_year)

*# Information coefficient*

IC = correlation(predicted\_move, actual\_move)

Breadth = number\_of\_independent\_bets

Information\_Ratio = IC \* sqrt(Breadth)

Target: IR > 0.5

**Implementation Requirements**

**Technology Stack:**

* **Data Feed**: Direct exchange connections (not consolidated)
* **Processing**: In-memory tick database (kdb+ or Arctic)
* **Execution**: Native exchange protocols (FIX/OUCH)
* **Latency**: Sub-100ms tick-to-trade

**Minimum Data Requirements:**

python

*# Per tick:*

- Timestamp (microsecond precision)

- Bid/Ask price and size

- Trade price, size, and aggressor flag

- Book depth (5 levels minimum)

*# Calculated in real-time:*

- CVD (cumulative volume delta)

- Trade size distribution

- Book imbalance ratios

- Microstructure features

**Critical Success Factors:**

1. **Data Quality**: Direct feeds, no aggregation
2. **Fast Execution**: Limit orders with quick cancels
3. **Risk First**: Many "no trade" conditions
4. **Small Edge**: 0.1-0.3 ticks per trade after costs
5. **High Frequency**: 10-50 trades per day
6. **Tight Stops**: 3-5 ticks maximum

This tick-based architecture captures real order flow and microstructure inefficiencies that time-based bars completely miss.