EXCHANGE AND ALGORITHMIC TRADING – PART 1: UNDERSTAND THE MARKET

Market Details

Market start-time & end-time

- Regular hours: opens at 9:30 a.m. ET and closes at 4:00 p.m. ET, Monday through Friday.
- Extended Pre-Market Trading Hours: 4:00 AM – 9:30 AM ET
- “normal” pre-market trading / "Early Trading Session" : 7:00 AM ET - 9:30 AM ET for NYSE. 4:00 AM – 9:30 AM ET for NASDAQ.

The opening and closing auction rules:

NYSE open:

- **Timeline:**
  - Orders can be entered and canceled until the Designated Market Maker (DMM) opens the stock, even after 9:30 a.m. ET.
  - NYSE starts accepting orders at 7:30 a.m. ET.
- **Process:**
  - NYSE disseminates imbalance information (e.g. more buy orders than sell orders) before the open. The initial imbalance is calculated and published at 8:30 a.m. ET.
  - Imbalance feed and matched size are updated every 5 minutes between 8:30 a.m. and 9:00 a.m. ET, every minute from 9:00 a.m. to 9:20 a.m. ET, and every 15 seconds as the open approaches.

- **Price Determination:** The opening price is set based on supply and demand factors in an auction-style format.

NYSE Close:

- **Timeline:**
    - Orders can be placed as early as 6:30 a.m. ET.
    - Cut-off time for MOC/LOC order entry is 3:50 p.m. ET.
    - No cancels of MOC/LOC/CO after 3:58 p.m. ET.
- **Process:**
    - The closing auction begins at 4:00 p.m. ET.
    - The closing auction determines the day's closing price [1].
    - **Imbalance:** Informational imbalance is disseminated every second until the close, including the paired-off quantity. A Regulatory imbalance is defined as a published imbalance at 3:45 p.m. above 50,000 shares in either direction. In such cases, traders can only trade in the opposite direction of the imbalance with a CLO (Closing-Offset-Order12), until the end of the day.
    - Floor brokers can help traders get better fills at auctions if the stock closes at their limit price

NASDQA open

- **Process:**
    - Buyers and sellers place offers and counteroffers until prices match
    - The goal is to maximize the number of executed trades at a single price
    - Data from trading interest is taken two minutes before the market opens
    - For example, if a buyer offers $100 per share for a given stock and a seller wants $110, the midpoint of the offer is $105. This midpoint is then

multiplied by 10%. The resulting $10.50 is then added to the buyer's offering price, moving it to $110.50 and subtracted from the seller's price, adjusting it to $99.50. This tells investors that the opening price for the shares is between $99.50 and $110.50.

- **Timeline**:
  - Market-on-open (MOO) orders must be received prior to 9:28 a.m. ET
  - Limit-on-open (LOO) orders may be entered until 9:29:30 a.m. ET
- The Opening Cross price determines the Nasdaq Official Opening Price (NOOP)

NASDQA close

- **Process:**
  - Combines on-close orders with auto-ex orders and quotes from the Nasdaq market center into a single print at the price where supply and demand meet.
  - Nasdaq releases indicative prices frequently in the ten minutes before the close
- **Timeline:**
  - Nasdaq accepts on-close orders from all participants before 3:55 p.m. ET
  - Dissemination of closing order imbalance information begins at 3:55 p.m. ET
  - Nasdaq stops accepting entry of LOC orders at 3:58 p.m. ET
- The Closing Cross price determines the Nasdaq Official Closing Price (NOCP)
- **Net Order Imbalance Indicator (NOII):**
  - The NOII gives participants a look at the supply and demand for a stock prior to the opening and closing crosses
  - It is calculated and updated continuously beginning five minutes before the market opens and 10 minutes before the market closes

- **Extended Trading Close (ETC):**
    - NASDAQ has an Extended Trading Close (ETC) session that occurs immediately following the completion of the closing auction.
    - The ETC concludes at 4:05 PM ET (1:05 PM for half trading days), and all unexecuted orders are canceled.
- **Order Handling:**
    - Regular market hours orders received after 9:28 a.m. ET are treated as Imbalance-Only (IO) orders for the Opening Cross.
    - If a firm sends in a MOO/MOC or LOO/LOC order after the cut-off, it is generally rejected

Order types— NYSE Open (from highest priority to lowest)

- Market-on-Open (MOO): They are non-limit market orders, meaning they will be filled at the opening price without a specified limit.
- Limit-on-Open (LOO): To buy or sell shares at the market open if the market price meets the limit's condition. If a LOO order is not executed at the market's open, it is canceled.
- Limit Orders

Order types— NYSE Continuous (from highest priority to lowest)

- Market Orders: execute trades immediately at the present market price.
- Immediate or Cancel Orders (IOC): execute immediately and cancel any unfilled portion.
- Intermarket Sweep Orders (ISO): allow traders to execute trades across multiple trading venues simultaneously.

Order types— NYSE Close (from highest priority to lowest)

- Market-on-Close (MOC)

- Limit-on-Close (LOC)
- Closing Offset Order (CO):
    - **Discretionary Pricing**: Brokers can set a price range to trade more aggressively near the close.
    - **Auction Participation**: CO Orders are active in the final seconds before the 4:00 PM auction.
    - **Imbalance Response**: They help offset buy/sell imbalances in the closing auction.
    - **Transparency**: Imbalance info, including CO Orders, is published starting at 3:50 PM.
    - **Cutoff Time**: Final entry or modification must be done by 3:59:50 PM.
- NYSE Floor Broker orders: only offered to a select few and it allows the investor to continuously adjust their bid. Can be MOC, LOC, or CO and follows the corresponding priority.

Order types — NASDAQ open: (from highest priority to lowest)

- On-Open (OO) , Market-on-Open (MOO)
- Limit-on-Open (LOO),
- Imbalance-Only (IO):
    - Limit Order Only: You must specify a price. Market IO orders are not allowed.
    - Auction-Only Execution: IO orders are only executed during the opening (9:30 AM) or closing (4:00 PM) auction.
    - Imbalance-Triggered: They only execute if there's an imbalance on the opposite side (after all higher priority orders are triggered). For example:
    - A buy IO order executes only if there's a sell-side imbalance.
    - A sell IO order executes only if there's a buy-side imbalance.

Order types — NASDAQ close:

- On-Close (OC), Market-on-Close (MOC),
- Limit-on-Close (LOC)
- Imbalance-Only (IO)

Alternative Venues & Trading Systems:

Dark pools

i.Why dark pools and how does it work?

- a privately organized financial forum or exchange for trading securities
- allow institutional investors to trade without exposure until after the trade has been executed and reported. This prevents heavy price devaluation, which would otherwise occur. If it were public knowledge, for example, that an investment bank was trying to sell 500,000 shares of a security, the security would almost certainly have decreased in value by the time the bank found buyers for all of their shares.
- a type of alternative trading system (ATS) that gives certain investors the opportunity to place large orders and make trades without publicly revealing their intentions during the search for a buyer or seller.

iii.What issues do dark pools have?

- because of the sheer volume of trades conducted on dark markets, the public values of certain securities are increasingly unreliable or inaccurate.
- mounting concern that dark pool exchanges provide excellent fodder for predatory high-frequency trading.

Smart order routing

Why SOR?

- provide excellent **executions** since the algorithms are very fast and simultaneously scan the routes to meet your execution preferences faster than any human can.

- provide **improved liquidity** as these algorithms can route orders to multiple destinations to fill your requested share size amounts. This results in improved fill prices as the smart route searches for the best fill prices simultaneously.

How does it work?

- **Time-based SOR** – As the name suggests, this method is highly dependent on time. It gives the most importance to the execution speed so that it is possible to take maximum advantage of the opportunity of the market. In this case, speed is essential because the reduction in execution time will help identify the best price, and delay will cause the price to change.
- **Liquidity-based SOR** – In this type, the order execution is based on liquidity, which is the priority over here. Through the algorithm, that order is selected and executed, which impacts the market in a minimum way and has the highest liquidity. In this way, the slippage risk is reduced, and the order gets executed at the best available price.
- **Cost-based SOR** – This method of smart order routing algorithms is the most important to the execution cost. This cost includes all transaction fees that the brokers or exchanges process during the order execution. The algorithm executes the order at a cost that is lowest among all through proper analysis of all the fees of all venues and selecting the venue that has the lowest fee.
- **Dark-pool SOR** – It also gives priority to those order executions that are dark pools. These trading venues are anonymous because they do not publicly show the orders before execution. Such algorithms will aim to execute orders with minimum impact on the market by routing orders through dark pools. This stops or controls the activity of those traders who can take the opportunity of a predictable market to get a good trade.
- **Volume-weighted average price SOR** – VWAP is the route that gives priority to an order with a considerable volume, and the price is calculated based on the formula of VWAP, which is the total value of the trade by the total volume. In this case, such orders are selected whose price is the closest to the VWAP.

Agency Trading algorithms

Whisper

```
function POV(participation_ratio, market_vol, remaining_vol):
    adjustment_factor = 1 / (1 - participation_ratio)
    raw_child = adjustment_factor * market_vol
    child_total_volume = min(max(raw_child, 0), remaining_vol)  # cap by remain
ing
    return child_total_volume



function Whisper(total_volume, urgency, order_side, limit):
 """
 Order Incompletion: The algorithm may not complete the order
 Inadequate Liquidity: Slows or halts execution
 avoid auctions to stay hidden

 Key Parameters:
    urgency : how closely to peg the spread
    total_volume: Total shares to buy/sell
    order_side : BUY or SELL
    limit : trade price limit

 """

    executed_order =0
    while executed_order< total_volume:
        if get_current_market_time()== NON_NORMAL_TRADING_HRS:
            continue //only trade during normal trading hours
        current_market_data= get_curent_market_data()
        pov= get_expected_participation_rate(historical_makrket_data, current_m
arket_data)
        trade_volume = POV(pov, current_market_data.volume, (total_volume- ex
ecuted_order))
        trade_volume = min(trade_volume, total_volume- executed_order)
        bid, ask = current_market_data.best_bid, current_market_data.best_offer
        spread = ask.price - bid.price
        delta_price = urgency* spread
        if order_side== "BUY":
            trade_price = bid+ delta_price
            trade_price = min(limit, trade_price)
        elif order_side== "SELL":
            trade_price = ask- delta_price
            trade_price = max(limit, trade_price)
        trade_success= false
        while not trade_success:  // halts execution when not enough liquidity
            trade_success = await trade(trade_volume, trade_price)
        executed_order += trade_volume
```

- Other implementations:
  - dynamically determine the best venue such as mid books
  - accurate market condition modelling
  - In high-volatility periods, the external orders might be placed deeper in the book to benefit from price swings.
- Metrics
  - **Implementation Shortfall:** *Difference between decision price and actual execution price.*
  - **Market Impact**: *Change in price caused by order execution, measured from before to after execution.*
  - **Order Completion Rate**:*% of the target order volume executed within the time window.*

TWAP

```
Function TWAP_Execute(total_volume, start_time, end_time, interval, randomis
ed):
    """
    Order Incompletion: may leave order partially filled.
    Inadequate Liquidity: Will post orders as scheduled
    Open & Close Auctions: May send a scheduled slice into the auction if it coi
ncides with scheduled time. No special adjustment for auctions.

    Key Parameters:
    total_volume: Total shares to buy/sell
    start_time, end_time: Execution window
    interval: Time between each child order
    randomised: Adds randomness to order sizes to reduce predictability

    """
    total_duration = end_time - start_time
    number_of_intervals = total_duration / interval
    quantity_per_interval = total_volume / number_of_intervals

    current_time = start_time
    cumulative_volume =0
    For i from 1 to number_of_intervals:
        target= i*quantity_per_interval
        if randomised:
            if (cumulative_volume>=target):
                upper_limit = max(0, quantity_per_interval- (cumulative_volume-targ
et))
                current_trade_volume= random(0, upper_limit)
            else: #target > cumulative_volume
                current_trade_volume= random(quantity_per_interval, quantity_per_i
nterval+ target-cumulative_volume)
            current_trade_volume = min(current_trade_volume, total_volume- curre
nt_trade_volume)
        else:
            current_trade_volume = target
        trade(current_trade_volume)
        cumulative_volume += current_trade_volume
        if cumulative_volume>= total_volume
            return
        Wait until current_time + interval
        current_time = current_time + interval

    End Function
```

- Other implementations:
    - vary order agressiveness based on how ahead/ behind schedule
    - adjust schedule based on market price
- Metrics
    - Difference between actual average execution price and theoretical TWAP.
    - Market Impact
    - Order Completion Rate

VWAP

```
function get_vwap_history(start, stop, price_history, vol_history):
    cumulative_pv={0} #initialize with all 0
    cumulative_vol = {0} #initialize with all 0
    vwap ={0} #initialize with all 0
    for i from start upTo stop:
        cumulative_pv[i] = (price_history[i].close+ price_history[i].high + price_hi
story[i].low)/3 + cumulative_pv[i-1]
        cumulative_vol[i] = vol_history[i] + cumulative_vol[i-1]
        vwap[i] = cumulative_pv[i]/ cumulative_vol[i]
    return vwap[-1] #return the last entry



# Called once to start the VWAP execution
function execute_vwap_order(order_side, total_volume, start_time, end_time,
                            price_history, vol_history, interval):
    """
    Order Incompletion/ Inadequate Liquidity: will attempt catch up logic
    Open & Close Auctions: continue to follow market volume
    Key Parameters:
    order_side : BUY or SELL
    total_volume: Total shares to buy/sell
    start_time, end_time: Execution window
    price_history, vol_history : historical price and volume data
    interval: Time between each child order
    """
    # Loop over time buckets
    for each bucket in time_intervals_between(start_time, end_time, interval):

        vwap_value = get_vwap_history(bucket.start, bucket.end, price_history, v
ol_history)

        target_volume = total_volume * vwap_value

        # Do not exceed remaining capacity, considering outstanding children
        outstanding = get_pending_size()
        remaining = total_volume - get_executed_volume() - outstanding
        send_qty = min(target_volume, remaining)

        if send_qty > 0:
            order_id = send_order(order_side, send_qty)  # non-blocking
            update_pending(order_id, send_qty)

        # Optional early stop if already fully accounted
        if get_executed_volume() >= total_volume:
            cancel_unfilled(get_pending_orders()) # cancel all pending orders
            break

    # Final dispatch for any leftover (non-blocking)
    outstanding = get_pending_size()
    remaining = total_volume - get_executed_volume() - outstanding
    if remaining > 0:
        order_id = send_order(order_side, remaining)
        update_pending(order_id, remaining)
    # Return what is filled so far; further fills will arrive via on_order_update
    return get_executed_volume()
```

- Other implementations:
  - using different amount of historical data
  - track current short term price and volume trends
- Metrics
  - Difference between actual average execution price and theoretical VWAP.
  - How well the market volume is predicted (consistent Participation Rate)
  - Order Completion Rate

Decipher

```
function POV(participation_ratio, market_vol, remaining_vol):
    adjustment_factor = 1 / (1 - participation_ratio)
    raw_child = adjustment_factor * market_vol
    child_order_size = min(max(raw_child, 0), remaining_vol)  # cap by remainin
g
    return child_order_size

function DECIPHER_POV(order_side, total_volume, urgency_band, start_time, e
nd_time, params):

    """
    Order Incompletion: May pause or slow execution
    Inadequate Liquidity: Seeks alternative venues
    Open & Close Auctions: Adjusts participation level based on expected aucti
on volume and price impact.

    Key Parameters:
    order_side : BUY or SELL
    total_volume: Total shares to buy/sell
    urgency_band: "LOW", "NORMAL" or "HIGH"
    start_time, end_time: Execution window
    price_history, vol_history : historical price and volume data
    params: including volume window, signal window, min clip, alpha, and venu
es
    """
    remaining = total_volume
    last_quote = None

    # Band targets
    if urgency_band == "LOW":
        base_target = 0.15; band_min = 0.10; band_max = 0.20
    elif urgency_band == "NORMAL":
        base_target = 0.25; band_min = 0.20; band_max = 0.30
    else:  # HIGH
        base_target = 0.45; band_min = 0.35; band_max = 0.55

    while now() < end_time and remaining > 0:
        md = read_market_data()  # top-of-book, depth, prints, vol estimates
        bid, ask = md.best_bid, md.best_offer
        market_vol = estimate_market_volume(window=params.vol_window)
        trend = short_term_trend(md.trades, window=params.sig_window)  # + u
p, - down

        # Signal: positive means favorable to execution price
        # For BUY, favorable = price drifting lower; for SELL, favorable = drifting
higher
        drift = -trend if order_side == "BUY" else trend

        target_ratio_unc = base_target + params.alpha * drift
        target_ratio = clamp(target_ratio_unc, band_min, band_max)

        # Child order size from POV
        child_size = POV(target_ratio, market_vol, remaining)
        if child_size < params.min_clip: #halt execution if drift is very -ve
            sleep(params.idle_interval)
            continue
        venue, get_best_venue(params.venues) #get the venue with the best liqui
dity
        trade(child_size, venue)

        # Update remaining and bookkeeping
        executed_qty = get_fills_since_last_tick()
        remaining = max(remaining - executed_qty, 0)

    End Function
```

- Other implementations:
    - greater/ smaller pegging effect
    - venue selection
- Metrics
    - Implementation Shortfall
    - Adverse Selection: How often did the price move against the order after execution?
    - response time /sensitivity to spikes in liquidity or volatility

Iceberg

```
function Iceberg(total_volume, exposed_percent, limit, order_side):
"""
Order Incompletion: May leave part of order unfilled
Inadequate Liquidity: waits for liquidity to refill
Open & Close Auctions: Auctions are anonymous and liquid, so full size can b
e revealed

 Key Parameters:
    total_volume: Total shares to buy/sell
    exposed_percent : Percentage of total order to expose
    limit: will trade up to but not beyond the price
    order_side : BUY or SELL
```

```
"""
    executed_order =0
    while executed_order< total_volume:
        drip = exposed_percent * total_volume
        drip = min(drip, total_volume- exposed_percent)
        if now() == open_auction or close_auction:
            drip= total_volume- exposed_percent #expose fully
        if order_side == BUY and limit> get_curent_market_price():
            await trade(drip)
        if order_side == SELL and limit< get_curent_market_price():
            await trade(drip)
```

- Other implementations:
    - Allows trading venue selection for order routing.
    - data analysis for optimal exposed percentage
- Metrics
    - Implementation Shortfall
    - sudden jumps in opposite flow or price after a tip is refilled
    - order completion rate