# COMP0049
# Market Microstructure Coursework

December 2021

# Question 1

***Note 1:*** The code in this section has been processed following the approach provided in the lecture notes.

***Note 2:*** Where appropriate, the answers to the following questions have been produced as functions to, in principle, allow for similar analysis of alternative datasets.

## (a)

```
# Question 1 - Part (a)

def midprice(OB_dic,ticker,rolling_unit=600) :

    midprice=pd.DataFrame((0.5*OB_dic['Ask Price']['Ask Price 1']+0.5*OB_dic['Bid Price']['Bid Price 1'])
                          ,columns=['Mid-price'])
    midprice_normalized=midprice/10000

    #ax=midprice_normalized.plot()

    midprice_res=midprice_normalized.resample(str(rolling_unit)+'s',label='right').mean()
    midprice_res.columns=[str(rolling_unit)+'S mavg']
    ax=midprice_res.plot(legend=True)

    ax.set_title(f'10-min moving average of {ticker} over the trading day')

    return midprice_normalized
```

```
midprice(OB_dic_amzn,'Amazon',rolling_unit=600)
```

| Time | Mid-price |
|---|---|
| 1970-01-01 09:30:00.017459617 | 223.565 |
| 1970-01-01 09:30:00.189607670 | 223.880 |
| 1970-01-01 09:30:00.189607670 | 223.880 |
| 1970-01-01 09:30:00.189607670 | 223.880 |
| 1970-01-01 09:30:00.189607670 | 223.880 |
| ... | ... |
| 1970-01-01 15:59:59.872741285 | 220.565 |
| 1970-01-01 15:59:59.903989046 | 220.575 |
| 1970-01-01 15:59:59.955241980 | 220.575 |
| 1970-01-01 15:59:59.958244616 | 220.570 |
| 1970-01-01 15:59:59.959359650 | 220.575 |

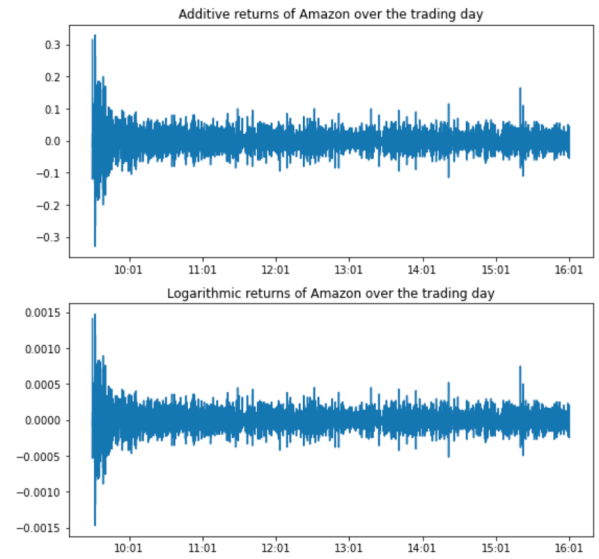
269748 rows × 1 columns

**(b)**

```
# Question 1 - Part (b)

def returns(OB_dic,ticker) :

    # Repeated from part (a) for completeness
    midprice=pd.DataFrame((0.5*OB_dic['Ask Price']['Ask Price 1']+0.5*OB_dic['Bid Price']['Bid Price 1'])
                          ,columns=['Mid-price'])
    midprice_normalized=midprice/10000

    midprice_normalized['Additive Returns']=midprice_normalized.diff()
    midprice_normalized['Logarithmic Returns']=np.log(midprice_normalized['Mid-price']).diff()

    midprice_additive=midprice_normalized['Additive Returns']
    midprice_logarithmic=midprice_normalized['Logarithmic Returns']

    #_____________________________________________________________________
    #
    # Plot 1: Additive Returns: a_{t} = m_{t} - m_{t-1}
    #_____________________________________________________________________

    fig,ax=plt.subplots(2,1,figsize=(9,9))

    ax[0].plot(midprice_additive)

    ax[0].set_title(f'Additive returns of {ticker} over the trading day')

    myFmt=DateFormatter("%H:%m")
    ax[0].xaxis.set_major_formatter(myFmt)

    #_____________________________________________________________________
    #
    # Plot 2: Logarithmic Returns: r_{t} = log( m_{t}/m_{t-1} ) = log(m_{t}) - log(m_{t-1})
    #_____________________________________________________________________


    ax[1].plot(midprice_logarithmic)

    ax[1].set_title(f'Logarithmic returns of {ticker} over the trading day')

    myFmt=DateFormatter("%H:%m")
    ax[1].xaxis.set_major_formatter(myFmt)

    #return midprice_normalized
```

The main difference is the scale. As we can see from the $y$-axis, the logarithmic returns are considerably smaller than the equivalent additive returns. This is natural given that we are taking the logarithm of a small number which, by definition, produces a smaller number.

```
returns(OB_dic_amzn,'Amazon')
```



## (c)

```
# Question 1 - Part (c)

def centered_returns(OB_dic,ticker) :

    midprice=pd.DataFrame((0.5*OB_dic['Ask Price']['Ask Price 1']+0.5*OB_dic['Bid Price']['Bid Price 1'])
                          ,columns=['Mid-price'])
    midprice_normalized=midprice/10000

    midprice_normalized['Additive Returns']=midprice_normalized.diff()
    midprice_normalized['Logarithmic Returns']=np.log(midprice_normalized['Mid-price']).diff()

    midprice_additive=midprice_normalized['Additive Returns']
    midprice_logarithmic=midprice_normalized['Logarithmic Returns']

    midprice_normalized['Centered Log Returns']=midprice_logarithmic - np.mean(midprice_logarithmic)

    return midprice_normalized
```

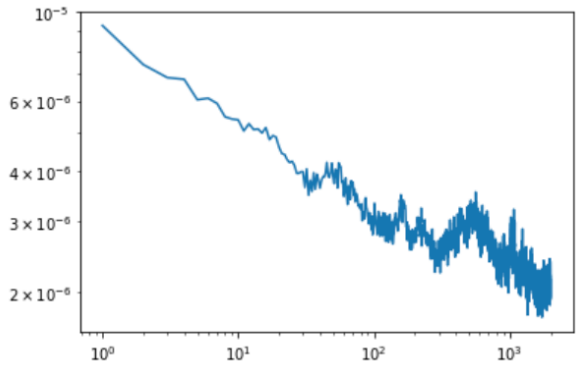
```
centered_returns(OB_dic_amzn,'Amazon')
```

| Time | Mid-price | Additive Returns | Logarithmic Returns | Centered Log Returns |
|---|---|---|---|---|
| **1970-01-01 09:30:00.017459617** | 223.565 | NaN | NaN | NaN |
| **1970-01-01 09:30:00.189607670** | 223.880 | 0.315 | 0.001408 | 1.408044e-03 |
| **1970-01-01 09:30:00.189607670** | 223.880 | 0.000 | 0.000000 | 4.991502e-08 |
| **1970-01-01 09:30:00.189607670** | 223.880 | 0.000 | 0.000000 | 4.991502e-08 |
| **1970-01-01 09:30:00.189607670** | 223.880 | 0.000 | 0.000000 | 4.991502e-08 |
| **...** | ... | ... | ... | ... |
| **1970-01-01 15:59:59.872741285** | 220.565 | 0.000 | 0.000000 | 4.991502e-08 |
| **1970-01-01 15:59:59.903989046** | 220.575 | 0.010 | 0.000045 | 4.538700e-05 |
| **1970-01-01 15:59:59.955241980** | 220.575 | 0.000 | 0.000000 | 4.991502e-08 |
| **1970-01-01 15:59:59.958244616** | 220.570 | -0.005 | -0.000023 | -2.261837e-05 |
| **1970-01-01 15:59:59.959359650** | 220.575 | 0.005 | 0.000023 | 2.271820e-05 |

269748 rows × 4 columns

```python
# Question 1 - Part (c) cont.

def c_tau(OB_dic,ticker):

    # Repeated for completeness
    midprice=pd.DataFrame((0.5*OB_dic['Ask Price']['Ask Price 1']+0.5*OB_dic['Bid Price']['Bid Price 1'])
                          ,columns=['Mid-price'])
    midprice_normalized=midprice/10000

    # Compute logarithmic returns
    midprice_logarithmic=np.log(midprice_normalized['Mid-price']).diff()

    # Compute centered log returns
    midprice_normalized['Centered Log Returns']=midprice_logarithmic - np.mean(midprice_logarithmic)
    centered=pd.DataFrame(midprice_normalized['Centered Log Returns'], columns=['Centered Log Returns'])

    # Compute the denominator
    denom=np.sqrt(np.mean(np.square(np.abs(centered))))

    # Compute array
    c_tau = []
    for tau in range(1,2001) :
        c_tau.append( np.mean(np.abs(centered) * np.abs(centered.shift(tau))) / denom )

    x = list(range(1, 2001))
    y = list(c_tau)

    ax=plt.loglog(x, y)
```

```python
c_tau(OB_dic_amzn,'Amazon')
```



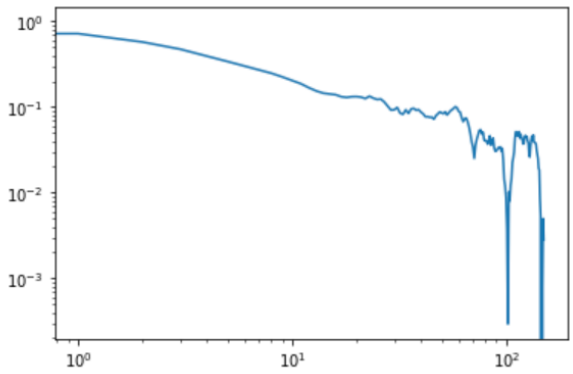

We first note that the function is always positive. This means that, in general, we expect absolute returns to be followed by absolute returns of a similar size, for example smaller absolute returns are followed by smaller, and larger by larger. Of course, this is a proxy for volatility and so periods of lower volatility follow lower volatility, and likewise for higher volatility, in a concept known as *Volatility Clustering*, something that is indeed observed empirically.

Secondly, we can see from the slope of the graph that is does not indicate exponential decay, but rather some lower power log decay. This is an important property as it suggests our absolute returns (or volatility proxy) have a characteristic known as *long-memory*. This means that as we increase the lag of the correlation function, whilst there is indeed a decrease, there is still some dependence on what has happened in the past.

**(d)**

```
# Question 1 - Part (d)

def visible_orders(msg_book,ticker) :
    order_book=msg_book.drop(['Time','OrderID','Size','Price','row_index'],axis=1)

    visible_book = order_book[order_book['Type'] == 4]

    tradesign=visible_book['TradeDirection']

    avg_eps_t = np.mean(tradesign)

    denom= np.sqrt( np.mean( np.square (tradesign - avg_eps_t)))

    c_eps_tau = []
    for tau in range(151) :
        c_eps_tau.append( (np.mean(tradesign * tradesign.shift(tau)) - np.square(avg_eps_t)) / denom )

    ax=plt.loglog(c_eps_tau)

    #return visible_book
```

```
visible_orders(msg_book_amzn,'Amazon')
```




By plotting the graph of the correlation function of the order flow, we find similar properties to those discussed in *(c)*. The correlation function is always positive, which means the direction of one trade tends to follow the direction of the previous trade. Again, the slope of the graph suggests some lower power decrease (rather than exponential) which indicates the long memory property that future events are, to some degree, dependent on past events.

**(e)**

We can consider two potential explanations for the properties of the order flow: Herding and the breaking up of large orders (often referred to as *order-splitting*). Herding can be described as a large group of traders all acting in a similar fashion. This could result from traders basing their orders on very similar or identical information – for instance, if a large group of traders all decide to execute a trade based upon the same Bloomberg data, they are likely to all trade in the same direction. Alternatively, it could arise due to less experienced (or less informed) traders following the lead of others. Both could then lead to the strong auto-correlation of order flows. This hypothesis, however, has been empirically tested and only found to be a minor reason for the empirical properties.

This leads to our second explanation, order-splitting, being the most probable cause. As the name suggests, this is simply the splitting up of a large order into smaller pieces in a bid to reduce the market impact of such a trade. This would then intuitively cause the order flow to have strong auto-correlation as there would be, for instance, ten individual smaller orders all trading in the same direction, as opposed to one single large order. Of course, to confirm this hypothesis one would require access to non-public information (since, by definition, the splitting up of an order is designed to hide the trader's intentions). Such analysis has been conducted in practice and confirms that order splitting plays a considerable role in the empirically observed properties of the order flow.

**(f)**

Let us now consider how we can align our empirical results of the order book with our ideas of market efficiency and the market impact associated with trades. When discussing market efficiency, we must distinguish between the different levels and types of efficiency. One possibility is that markets are strong-form efficient, that is prices reflect all public and non-public information, and an order in any direction will have no impact on price. Of course, this possibility can be eliminated as we have seen that orders do indeed have an impact on the price level; therefore, it is sensible to suggest that markets are only weak-form efficient, they reflect only public information.

Furthermore, as we have shown a strong auto-correlation in the order flow, it seems reasonable to suggest that, in general, the orders are submitted to the LOB in such a way that the observed liquidity becomes asymmetrical. This means traders that are providing liquidity to the market in the form of limit orders, base their prices on the previous orders submitted and the public information available. Consequently, market orders that are submitted in the opposite direction and therefore 'surprise' the market cause a greater impact on the price level. This leads us to conclude that market orders depend on the past and are therefore not 'memoryless' (or in more quantitative terms, they are *non-Markovian*).

We can go further than this though and consider whether certain models provide more insight. Let us begin with a simple propagator model, that is one that assesses market orders to have a constant and permanent impact. However, this model presents challenges as returns would become predictable, which contradicts the empirical results, and therefore the empirical long-memory property discussed in *(d)* directly conflicts with our model.

We now modify the model to account for the fact that surprises impact the price. By doing so we still maintain the view that market orders have a constant and permanent impact, however, there is now a dependence on the past orders (which ties in with our empirical results). This so-called Surprise model provides a much improved match with our empirical results.

An alternative to the above is to extend our simple propagator model to remove the dependence on past transactions and introduce a time varying component. This means the model no longer assumes constant and permanent impact of market orders in any form and instead proposes the impact declines as time passes. This has the advantage of being able to assess the impact of the order-splitting concept discussed in *(e)*. However, there remains a discrepancy between the model and the empirical results.

# Question 2

### (a)

Throughout this question we will use the following result without proof.

**Result:** Let $X$ be a random variable, and $y_i$ be a sequence of $i.i.d$ random variables.

$$\mathbb{E}\left[\sum_{i=1}^{X} y_i\right] = \sum_{X=1}^{\infty} \mathbb{E}\left[\sum_{i=1}^{X} y_i\right] \mathbb{P}(X = x) \tag{1}$$

### (b)

We have taken each $v$ (*order size*), $N$ (*number of orders*) and $p$ (*price*) to be independent. We have subsequently shown that the average depth depends solely on how wide we take our interval $[p_a, p_b]$, that is, we will see a greater average depth as we widen the interval. This makes intuitive sense, however, the prices $p_a$ and $p_b$ are arbitrary which means our model will predict a certain average depth regardless of where the interval is in relation to the best bid – an unrealistic outcome – hence, the model and its assumptions have several limitations in their ability to replicate real LOB dynamics.

Firstly, the number of orders certainly does have some dependence on the price – we would expect the number of orders at or near the best bid price to be far greater than those deep in the LOB. However, our model assumes that at each price the number of orders is Poisson distributed with an activity rate, $\lambda$, in the interval (0,1). This provides a good estimation of the number of orders we might imagine deep in the LOB, as the highest probabilities are zero, or possibly small integers, but does not reflect the realistic number of orders at or near best – which is likely to be substantial. Similarly, the order size, $v$, is likely to vary according to both the price level and number of visible orders (hence market depth) available at a given time.

Furthermore, prices in practice are almost always discrete – that is, there is a tick size $\tau > 0$ imposed by the exchange/regulator – this stems from a general property of LOBs that a tick size of zero allows a market participant to continuously enter an infinitesimally small improvement on the previous bid and can lead to issues such as zero bid-ask spread, computational complexity, and punishment for time priority.

**(c)**

In real LOBs the available depth at best plays an important role through what is known as *selective liquidity taking*. This refers to the trader identifying the visible orders currently available at and near best, then selectively submitting their order to account for that. The intention is to only execute orders at the best possible price and avoid paying the excess costs that would be incurred by executing at the second-best price.

Empirically we see this behaviour across both small and large tick stocks, but to slightly different degrees. For large tick stock the costs associated with consuming all the available depth at best can be large. Consider a market participant looking to sell $100XYZ$ stock with available bids of $50XYZ$ at £10, and $50XYZ$ at £9. Naively they could execute their entire transaction, but this would impose a cost of £50. Hence, they would likely selectively execute 50 (or less) at the best of £10. Small tick stocks follow the same principle; for illustrative purposes, consider the previous example but with best bids at £10 and £9.90 – clearly the cost is smaller. Hence, small tick stocks see marginally less selective liquidity taking than large tick stocks.

We also note that empirically we see an interesting discretization of the selective consumption of liquidity at best as participants tend to prefer executing 'natural' amounts – orders that are 25%, 50%, 100% of available volume.

# Question 3

### (a)

Firstly, we consider market impact cost – the cost associated with movements in the market due to the size or number of orders being executed. For instance, if XYZ stock has an average daily trading volume (ADV) of 1m shares, then an order for 900k shares (90% ADV) will have a significantly greater market impact cost, compared to a much smaller order of 20k (2% ADV).

Secondly, we consider the urgency of execution for an order. If an order needs to be executed within a short period, then the urgency is high, whereas an order that can be executed over a longer horizon has lower urgency. This itself generates an execution cost. For higher urgency orders, it is unlikely that the price will move away from what was initially sought (ignoring the associated market impact costs). However, for lower urgency, there is a greater risk of the price moving over the trading horizon – compare the movement of a stock over one day, to one month. There are also more obvious costs such as the fees paid to brokers to execute the trades, and the spread between best bid and ask.

These costs affect market participants differently. Directional investors tend to be most concerned with market timing costs, as their typical large orders are split and executed across multiple venues and over a longer period. On the other hand, Market Makers are more affected by market impact costs as they often seek to finish the day with a neutral position, hence, their trading urgency is high – especially if they hold significant inventory as they approach the close of the markets.

### (b)

There are many choices available to traders when seeking to execute multiple orders. The most straightforward, provided the size of the orders are small in comparison to the ADV, is simply to submit multiple orders directly into the market. However, this proves inappropriate in many cases.

Beyond this, algorithms can assist in identifying the opportunities to trade in a single market, such as *Time-Weighted Average Price* (TWAP) which seeks to execute orders equally over a set period of time (e.g., execute an order every hour); or *Percentage-of-Volume* (PoV) which seeks to execute a set percentage of the volume traded during a certain period (e.g., execute an order that is 5% of 10 minute trading volume) which differs from VWAP by considering actual ADV rather than historical averages. Alternatively, algorithms can assist by executing multiple orders across multiple venues through liquidity aggregation – an algorithm that aggregates the current bid and ask orders from multiple LOBs and executes trades accordingly.

The above describe executions on lit venues, but there are also alternative trading venues (e.g., dark pools) as well. These allows participants to execute multiple (often large block) orders without displaying the information associated with the trade to the wider market.

**(c)**

The mathematical representation for total transaction costs for an order (or meta-order) is given in Eq. 1 (Engle et al., 2006).

Let $p_0$ be the fair market price when the order was submitted, and $\tilde{p_t}$ be the actual transaction price for a trade in period $t$.
Let $x_t$ be the number of shares held at the end of time period $t$, such that between period $t$ and $t+1$ the change in shares is given by:

$$\Delta x_{t+1} := x_{t+1} - x_t \tag{2}$$

Then transaction costs are defined as:

$$TC = \sum_{t=1}^{T} \Delta x_t (\tilde{p_t} - p_0) \tag{3}$$

This means for each transaction in a specific period (defined by the change in number of shares) $\Delta x_t$, there will be a cost associated with that trade which is given by the difference in the price identified as fair ($p_0$) and the price that was actually executed ($\tilde{p_t}$).

Under traditional circumstances, when buying the change in shares would be positive ($\Delta x_t > 0$) and the actual price would be greater than the fair price, $\tilde{p_t} > p_0$, leading to positive transaction costs. Similarly, when selling there is a reduction in shares ($\Delta x_t < 0$) and $\tilde{p_t} < p_0$, would again give positive transaction costs. Of course, if the market moves favourably (falls when buying or rises when selling) transactions costs can become negative.

**(d)**

Engle et al. (2006) used order data provided by Morgan Stanley's Benchmark Execution Strategies™ team in 2004 considering NYSE and NASDAQ stock orders. To produce the analysis, they cleaned the dataset in the following ways.

Firstly, they excluded non-completed orders, that is, they removed those that were partially fulfilled and those that were cancelled – this means the individual orders all fell into the same binary category of fully completed. Secondly, any short selling orders were removed due to the associated rules conflicting with the economic model. Essentially, the suggested execution strategy for a short sale may have conflicted with the regulations at the time.

Furthermore, any orders executed before 09:36 were excluded as opening market dynamics tend to be noticeably different to typical intra-day market dynamics and therefore would likely have distorted the analysis. Similarly, orders occurring near the market close were also removed because they could have been executed faster than the selected strategy would suggest and were therefore in direct conflict with the algorithm.

Lastly, to account for small orders, those which were *(a)* less than 1000 shares in size; *(b)* had an arrival price of less than \$5; and *(c)* were fulfilled in less than 5 minutes, were all excluded. This left a dataset containing 233,913 points.
The dataset itself provided several necessary and useful features to conduct the analysis. Whilst it did not offer the identity or reasoning behind the orders, it did provide information on the direction (buy or sell); the exchange used (NYSE vs. NASDAQ); the strategy employed (Arrival – High/Medium/Low Urgency, or VWAP); the order size (in terms of number of shares); the order value (in \$); and spread. All of which can be found in Table 1 (Engle et al., 2006).

**(e)**

Engle et al. (2006) discuss two main strategies for executing orders: *Volume Weighted Average Price* (VWAP) and *Arrival Price* (AP), with the latter be sub-categorised into three separate instances regarding the urgency of the trade: high, medium, and low.

The AP strategy simply attempts to execute the trade at the best possible price with reference to the arrival price ($p_0$) given some predefined willingness to tolerate risk. The urgency of the trade is directly related to the willingness of the participant to tolerate risk. For high urgency trades, the willingness to tolerate risk is low and so the strategy will seek to execute orders quickly, while compromising on market impact costs as these will naturally be higher. For low urgency trades, the participant is willing to tolerate a greater amount of risk – that is, the risk that the price moves away from the original arrival

price ($p_0$) over the trading horizon – however, this delivers the benefit of reducing market impact costs. Medium urgency naturally sits somewhere in between.

The VWAP strategy aims to execute the order proportionally over the trading day based on the typical trading volumes (generally the algorithm uses historical daily volumes). Engle et al. (2006) modify their definition of VWAP slightly to include only those orders that were scheduled to be executed over a single day (rather than a shorter/longer time horizon) and those that made up a very small percentage of daily volume. The intention being that VWAP orders could now be considered as strategy to minimise costs altogether with a willingness to tolerate any amount of risk (risk-neutral). This extends the low urgency AP strategy to a more extreme case.

Hence, our two (four when sub-categorised) order approaches provide an inverse relation between cost and risk and can be summarised in the following table.

| Order Type | Cost | Risk |
|---|---|---|
| Arrival Price – *High Urgency* | Highest | Lowest |
| Arrival Price – *Medium Urgency* | ↓ | ↑ |
| Arrival Price – *Low Urgency* | ↓ | ↑ |
| Volume Weighted Average Price | Lowest | Highest |

**(f)**

The novelty of the paper is the generation of the 'Efficient Frontiers' for expected transaction costs conditioned on the market conditions at that time. Their work results in: a simple, yet flexible model for analysing the costs and risks associated with trades; an empirical view of the links between market impact costs and the size of an order for different strategies; an empirical view of the links between risk, as measured by standard deviation, and the size of an order for different strategies; and a cost-risk frontier for different strategies.

Firstly, we note the construction of the model, using an exponential distribution, allows for convenient interpretations with respect expected cost and expected risk since they are both anticipated to be positive. Furthermore, the inclusion of i.i.d standard normal random variables associated with the risk (standard deviation) allows for straightforward modelling and ease of explanation, whilst their use of White standard errors allows for some non-normality.

Using Figures 3–6 (Engle et al., 2006) we can examine the trade-off between the relative size of an order and its expected cost or expected risk. We start with the expected cost (Figure 3 – NYSE; Figure 5 – NASDAQ), both of which are concave functions. Whilst we would expect the costs associated with an order to increase with the size of the order, it does so in a non-linear fashion, perhaps

highlighting economies of scale. As anticipated, the expected costs increase as we increase the urgency of the order executions. What is interesting, however, is the relatively small range (c.10bps) between all strategies.

We see a similar, albeit inverted, story for the expected risk (Figure 4 – NYSE; Figure 6 – NASDAQ). What is worth highlighting is the size of the range between the different strategies. For orders on both the NYSE and the NASDAQ, the VWAP standard deviation is some 50bps higher than the AP – Low Urgency strategy, but with very little difference in the expected cost. This suggests there is little reason to choose a VWAP approach over AP – Low Urgency.

The most critical part of the paper, however, is the construction of the Efficient Frontiers in Figures 7–10 (Engle et al., 2006). In Figures 7 and 8, they construct the frontier for both NYSE and NASDAQ executions, which confirms the earlier point that there is little benefit derived from taking a VWAP approach over an AP – Low Urgency one. This is indicated by the near horizontal line connecting point 3 and point 4, showing an increase in expected risk ($x$-axis) with a negligible decrease in expected cost ($y$-axis). Unfortunately, there is no indication of what their definition of 'typical stock' and 'typical day', so we cannot comment on this further.

However, we can see from Figures 9 and 10 that the efficient frontier is non-linear with respect to the size of the order, with the 75th percentile (black) and 90th percentile (teal) lines moving considerably further towards the North-East, indicating worsening expected outcomes on both the cost and risk fronts.

### (g)

The analysis performed by Engle et al. (2006) allows us to draw several conclusions. Let us start by considering the market conditions. We can see from the regression parameters listed in Tables 3–6 (Engle et al., 2006) the spread coefficient is positive for all cases. This makes intuitive sense as we would expect a wider spread to increase both the costs associated with executing trades and the expected risk, since wider spreads are often associated with a lack of liquidity – recall the sparsity of the LOB for small tick stocks.

Similarly, the coefficient of (log)-volatility is positive for all cases, which again makes intuitive sense as increased volatility is often associated with periods of market turbulence or uncertainty and therefore, we would expect costs to execute an order to increase and of course risk to increase as this is, by definition, the standard deviation.

There are also some noticeable differences in terms of the markets the orders are executed in. There is a 0.095 increase from the coefficient of the VWAP strategy (given by const. in Table 4) to the coefficient Low Urgency for NYSE

expected costs. Whereas there is only a 0.025 increase for the same change in NASDAQ expected costs. Similarly, between Medium and High Urgency there is around a 0.1 increase for NYSE expected costs versus a 0.05 increase for NASDAQ orders. This clearly shows that whilst the overall interpretation of each parameter is the same for both markets, there are differences in the emphasis on specific parameters and strategies.

We now turn our attention to the order characteristics. The volume of an order plays an interesting role. The coefficient of the (log)-volume for the expected cost of both the NYSE and the NASDAQ are negative, which as noted before, helps produce the concave function of expected costs against order size (Figures 3 and 5) and is traditionally what we would expect. What is curious to note, however, is that the same is true for the risk. The negative coefficient of (log)-volume for the expected risk, suggests that as the size of an order increases the standard deviation will decrease, all things equal. This is exactly what we saw from our concave function of expected risk against order size (Figure 4 and 6).

Naturally, the (log)-value coefficient is positive in all cases and indicates that an increase in the value of the trade will, all things equal, increase the expected cost and expected risk with a trade. We have seen there are a variety of relationships between parameters and expected cost and expected risk, which can be summarised in the following table.

| Parameter | Impact on Expected Cost | Impact on Expected Risk |
|---|---|---|
| *Market Conditions*<br>*An increase in...* | | |
| Spread (inverse liquidity) | Increases | Increases |
| Liquidity (inverse spread) | Decreases | Decreases |
| Volatility (indirectly market turbulence and uncertainty) | Increases | Increases |
| *Order Characteristics* | | |
| Order size (volume) | Decreases | Decreases |
| Order value | Increases | Increases |