Monash University (Clayton Campus)

MTH 5550 - Quantitative trading and market microstructure
Semester 1

Assignment 2

Lecturer: Hassan Fallahgoul

Group member: Ken SO
SID: 34856323
Coding/Computation method: Python (coding)/LaTeX (reporting purpose)

Summary of the Glosten-Milgrom model

Background of Glosten-Milgrom model

Traders' behaviour shaped the market dynamic in the ever-changing Financial World. Glosten, L.R. & Milgrom, P. R. (1985) observed that the traders behave differently under market price dynamics with the existence of information asymmetry. Fama, EF (1970) proposed the efficient market hypothesis ("EMH") and pinpointed that the belief of market participants has 3 variations. Traders who believe the market price is purely random upholding weak form hypothesis. Traders who believe the public information, for instance, news, make impacts on the price upholding semi-strong form hypothesis. Traders who believe and follow the implication of private information upholding strong form hypothesis.

Considering this, Glosten L.R. & Milgrom, P.R. (1985) modelled the trader's behaviour under market price dynamics by taking EMH into assumption and named the model as Glosten & Milgrom model ("GM model"). GM model mimicked the end-to-end process of decision-making of traders, from their beliefs to initial true value hypothesis and subsequent re-estimation, finally to their investment action, backboned by a decision tree with defined probabilities.
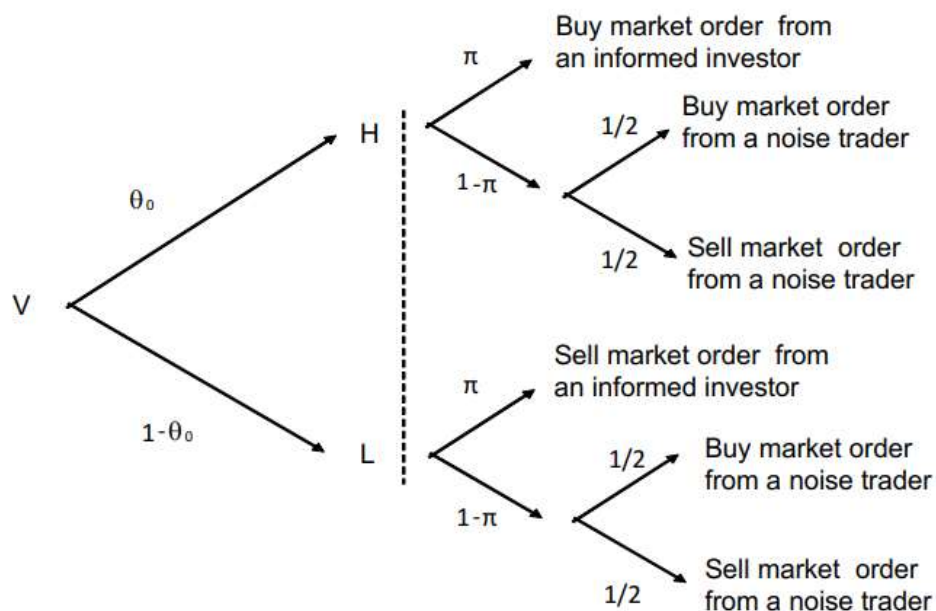
<Decision tree of GM model>
(sourced from your Lecture note, week 4, p.16)

Initial setting of GM model

Pursuant to the book of Malkiel, B.G. (2019), the random walk model ("RW model") is expressed as below:

$$p_t = p_{t-1} + \epsilon_t \text{ , where } \epsilon_t \sim i.i.d.(0, \sigma^2)$$

Econometrically, the ϵ_t is the noise of market and the profit or loss of investments where ϵ_t is independently identically distributed ("i.i.d.") such that assumes that there is no serial correlation between the last noise, current noise and price.

Below are the assumptions of RW model:

$$1. \{p\} \ is \ a \ martingale$$
$$2. \epsilon_t = p_{t+1} - p_t \ , is \ i.i.d.$$
$$3. E(\epsilon_t \epsilon_{t+1}) = E(\epsilon_t)E(\epsilon_{t+1}) = \ 0$$

The model started with the price process and is quantified into a conditional expectation expression below:

$$E(v|\Omega_{t-1}) = \ \mu_t = p_t$$

GM model denotes the security value process as $\{v\}_t$ and $\{\Omega\}_t$ is a set denoted the knowledge accumulated by the trader up to time t. Intuitively, the price process is the expectation of security value given trader's accumulated knowledge.

Types of traders can be diverse. Some may rely on public or even private information, and some may believe in random walk theory and technical analysis. The initial hypothesis on the true value of security is different based on the types of traders and their knowledge and beliefs. v_h denotes the hypothesis on the true value of security that is higher than the current value while v_l , otherwise, denotes that the hypothesis on the true value of security that is lower than the current value. The initial security value v_0 is a constant across the time where $v_0 = \frac{v_h + v_l}{2} = \ \mu_0$.

At the initial state, the hypothesis is a fuzzy logic. GM model assigned θ_0 as the initial hypothesis on v_h and $1 - \theta_0$ as the initial hypothesis on v_l, where $\{\theta\}_t$ is the process of beliefs across time t. It implies that the belief $\{\theta\}_t$ is a recursion across time t. Intuitively, the belief will update across the time when traders accumulated more information. This part will be discussed in Beliefs update and Price discovery section.

So, after the modelling of the initial hypothesis of traders, GM model further models the investment decision making of the traders which aligns with the hypothesises under EMH.

The strong form EMH believers
Traders who believe the impact on private information are assigned a probability π to buy or sell market orders. Intuitively, π is the probability that traders are privately informed and insisted to what they are informed thus making investment decision in favour of the private information. For example, if the private information is good news to the underlying security, traders will buy the security if they hypothesized that the true value of security is higher than the

current value or, in other words, the security value is currently undervalued, such that they gain on the price rising potential.

Bid-ask price, beliefs and market price
GM model modelled the bid-ask price with following formulae:

The weak and semi-strong form EMH believers
Traders who believe in weak and semi-strong form EMH are assumed to trade market orders under noise. Under the GM model, $1 - \pi$ measured the probability of their investment decision under noise and denoted them as "noise traders". For the subsequent buy or sell decision making, investment decision shares even probability.

Bid-ask price, beliefs and market price
GM model modelled the bid-ask price with following formulae:

Ask-price:

$$a_t = \mu_{t-1} + \frac{\pi \theta_{t-1}(1 - \theta_{t-1})(v_h - v_l)}{\pi \theta_{t-1} + (1 - \pi)/2}$$

Bid-price:

$$b_t = \mu_{t-1} - \frac{\pi \theta_{t-1}(1 - \theta_{t-1})(v_h - v_l)}{\pi(1 - \theta_{t-1}) + (1 - \pi)/2}$$

θ_{t-1} is iterated to the bid-ask price at time t. Also, $\mu_{t-1} = p_{t-1}$ is the last market price which is also in the iteration of bid-ask price at time t. The time of iteration is determined by the length of the vector of $[d_t]$, that is, the maximum of time point t defined.

In deciding whether bid or ask price will be taken as the market price of time t, GM model also assumed the decision rule below:

$$p_t = \begin{cases} a_t = \mu_t^+ = \mu_{t-1} + \frac{\pi \theta_{t-1}(1-\theta_{t-1})(v_h-v_l)}{\pi \theta_{t-1}+(1-\pi)/2}, & if\ d_t = 1 \\ b_t = \mu_t^- = \mu_{t-1} - \frac{\pi \theta_{t-1}(1-\theta_{t-1})(v_h-v_l)}{\pi(1-\theta_{t-1})+\frac{1-\pi}{2}}, & if\ d_t = -1 \end{cases} \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (1)$$

$$p_t = \mu_t = \begin{cases} \mu_{t-1} + s_a \\ \mu_{t-1} - s_b \end{cases} = \mu_{t-1} + S(d_t)d_t \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2)$$

In equation (1), I noted that the order flow is the determinant of whether the bid or ask price is chosen to be the market price. Intuitively, if the order flow at time t is $d_t = 1$ (a.k.a. buy order), traders will sell the security so that their trade will be at ask price, other way around. In the equation (2), the expression in (1) can be rewritten into the last market price $\pm$ the half spread where the d_t decided if it is bid or ask half spread. This information affects traders' behaviour under the GM model as traders are assumed to be in search of the true security price with their oscillating beliefs. In the next section, I will discuss the beliefs update and price discovery mechanic under GMM model and will show how traders' belief iterates across the time and thus recursing the market price in this section.

Beliefs update and Price discovery

Now, I understood the assumptions and initial setting of GM model. The traders trade based on their perceived information. Yet, the question is, what is the source of information? GM model assumed that the π and last belief θ_{t-1} are the parameter affect the belief process $\{\theta\}_t$ of traders. The order flow, again, dominates the calculation of belief.

$$\theta_t^+ = \frac{(1 + \pi)\left(\frac{1}{2}\right)\theta_{t-1}}{\pi\theta_{t-1} + (1 - \pi)\left(\frac{1}{2}\right)} \ , if \ d_t = 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (a)$$

$$\theta_t^- = \frac{(1 - \pi)\left(\frac{1}{2}\right)\theta_{t-1}}{\pi(1 - \theta_{t-1}) + (1 - \pi)\left(\frac{1}{2}\right)} \ , if \ d_t = -1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (b)$$

So, here, I put a focus on the order flow d_t as I observe that the order flow dominates the value of belief. It implies that the order flow affects the belief of trader, and the belief of trader affect the future order flow as per **<the decision tree of GM model>** and thus providing a signal to market maker to adjust their bid-ask spread for making profit. This price adjustment will affect the next belief and bid-ask price. GM model assumed the distribution of order flow d_t:

$$\Pr(d_t = 1|v_h) = (1 + \pi)/2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (A)$$
$$\Pr(d_t = -1|v_h) = (1 - \pi)/2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (B)$$

In equations (A) and (B), I documented that the distribution of the random order flow is computable and π is the sole parameter of the distribution. Recall from the section of the strong form EMH believers, π is the probability that traders are privately informed and insisted to what they are privately informed. So, it implies that the order flow somehow reflects the existence of informed trading, say, (A) is greater than (B). It also makes sense to deduce that traders' belief $\{\theta\}_t$ will increase given their initial hypothesis is v_h with a greater probability in (A). *Finally, if the latest order flow leans to buy side* $(\Pr(d_t = 1) > \Pr(d_t = -1))$, *my hypothesis is that the lean-to-buy order flow becomes a signal to market maker to wider their spread for making profit and thus a higher market price. As such, I further deduce that the traders will tend to buy more in the subsequent trades as they tend to believe that the true value of security is undervalued and thus achieving a belief convergence.* This corollary will be further discussed in Results and Findings section.

Motivation for replicating the model

To achieve a deep understanding on GM model, I replicate the GM model to reproduce the convergence of belief. I also measured the squared pricing error of the price discovery process to show, on the other hand, how the pricing errors measure variates with beliefs convergence. Finally, I measured the averaged pricing errors of 10 simulations to show the speed of price discovery with different π values. By doing so, I expect to realize the learning goal of this course.

To obtain hands-on experience on financial modelling as a student, I pretended to be engaged in a project of financial modelling replication. In planning phase, I understand the model and design the method of simulation. In organize phase, I search for any possible tools and side knowledge to realize my plan. In leading phase, I exercise my coding, Mathematics, Microeconomics and Financial Econometrics knowledge to engineer the algorithm. In control phase, I set up a time limit to deliver the work done and fix any bug arises. By doing so, I expected to sharpen my skills and knowledge in every aspect of quantitative finance discipline.

Objectives of my analysis

This analysis aims to reproduce:

1. Belief convergence
2. Inverse variation between PD measure and Belief convergence
3. Speed of price discovery with different values of π

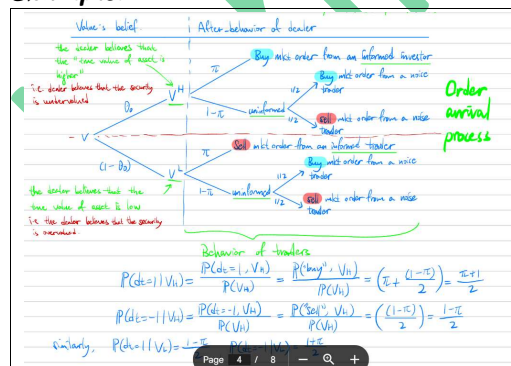by replicating the GM model with algorithms and coding.

Methodologies and assumptions

Planning phase:

My initial idea is to develop an algorithm to fully replicate the simulation in week 4 lecture note slides $30 - 34$ with function and operation method. Python is a more appropriate language to code my idea. Yet, the very first task I need to complete is to ensure my understanding is correct as a financial modeller. <Appendix #2> shows my hand-written note, summarized my initial understanding on the GM model. This hand-written note is a rough work and not 100% accurate and complete. In this task, I aim to provide a solid knowledge base to support my next task.
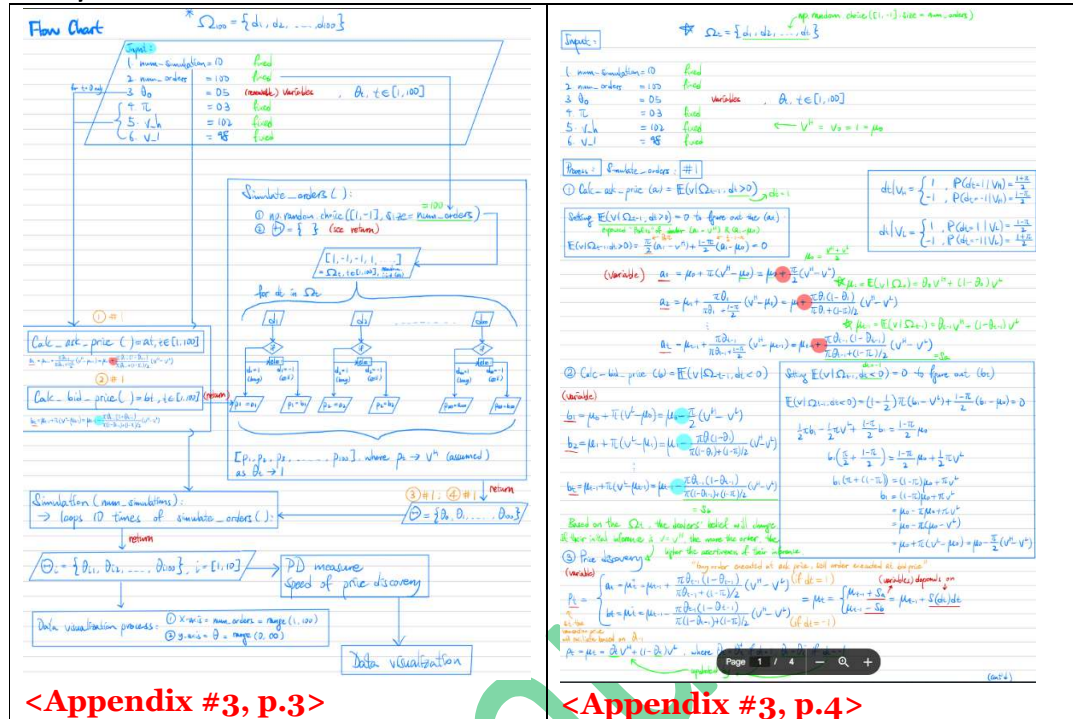
Example:

<Appendix #2, p.4>

<Appendix #2, p.6>

Organizing phase:

My next task is to design an implementation plan for the GM model. Checked to the Assignment 2 requirements, Python is the method approved, and which I preferred. To design the algorithm, I firstly drafted a flow chart in my second
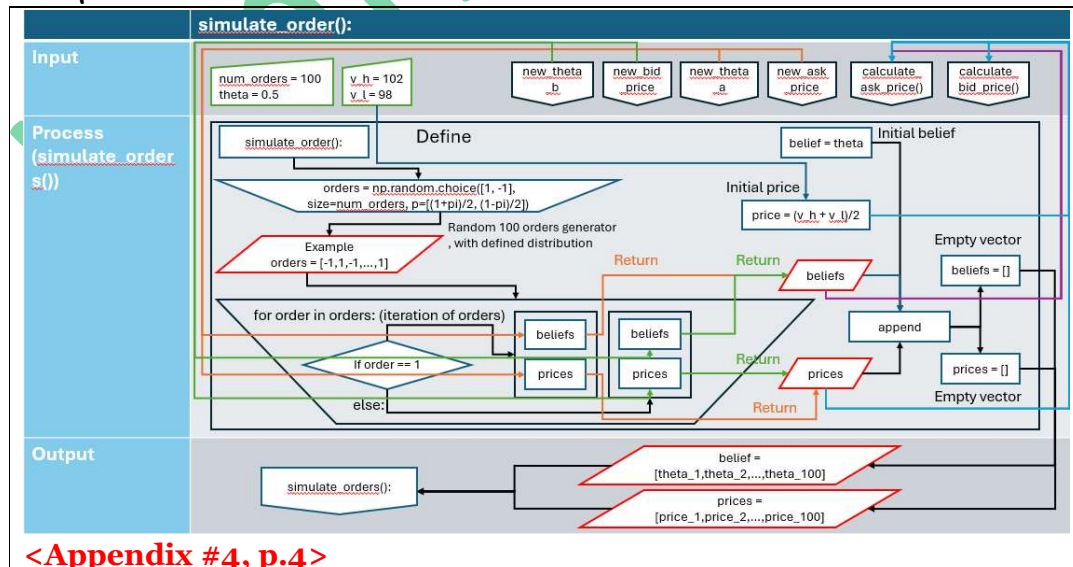
hand-written note **<Appendix #3>** to design my algorithm. Here I exercise my working experience and knowledge learnt from FIT9136 to deliver the draft.

Example:

<Appendix #3, p.3>   **<Appendix #3, p.4>**

<Appendix #4> is the more completed version of flowchart on the algorithm design and the data flow. It is not 100% accurate and complete as it is for self-reference purpose only.

Example:

<Appendix #4, p.4>

Leading phase:

<Appendix #1> is the finalized source code in Python language. Below is the illustration on the source code **<Appendix #1>**:

```
# import modules to be used
import numpy as np
import matplotlib.pyplot as plt
```

Step 2: Set up initial variables
- Set up all ready-to-use global parameters for functions and loops

```
# Parameters
num_simulations = 10
num_orders = 100
theta = 0.5  # Probability that the true value is high
# pi = 0.3    # Probability that a trader is informed
v_h = 102   # High value
v_l = 98    # Low value
```

Step 3: Compute the bid-ask prices
- Compute the bid-ask prices by the operation set in accordance with the hand-written notes **<Appendix #3>**
- Embedded the recursion of ask price a_t , b_t and θ_t^+ , θ_t^- into the functions

```
# Calculate ask_price and theta_a
def calculate_ask_price(mu_last, pi, theta_last, v_h, v_l):
    new_theta_a = ((1 + pi) / 2 * theta_last) / ((pi * theta_last) + ((1 - pi) / 2))
    new_ask_price = mu_last + ((((pi * theta_last) * (1 - theta_last))* (v_h -
v_l)) / ((pi * theta_last) + ((1 - pi)/ 2)))

    return new_theta_a, new_ask_price


# Calculate ask_price and theta_a
def calculate_bid_price(mu_last, pi, theta_last, v_h, v_l):
    new_theta_b = ((1 - pi) / 2 * theta_last) / ((pi * (1 - theta_last)) + ((1 - pi) /
2))
    new_bid_price = mu_last - (((pi * theta_last) * (1 - theta_last))* (v_h -
v_l)) / ((pi * (1 - theta_last)) + ((1 - pi) / 2))

    return new_theta_b, new_bid_price
```

Step 4: Compute the PD measure

- Compute the PD measure with operation in accordance with the formula in lecture note week 4 page 33

```
# Function to PD measure of one set of orders
def pd_measure(price):
    return (price - v_h)**2
```

Step 5: One time simulation
- Please be informed that the function "simulate_order(pi)" is different from "simulate_order**s**(pi, num_orders = num_simulations)" in next step
- To iterate beliefs, prices and PD measures for 100 random orders, the function embedded for-loop in its local operation. Notice that the initial security value v_0 should be included into **<Graph #1>** while there's no need for PD_0 in **<Graph#2>** and PD_{i0} in **<Graph #3>**
- To deliver the latest bid-ask price and belief in every recursion during the simulation, I called out the functions defined in step 3 and 4 with parameters from step 1 then assigning corresponding outputs into the local variables under the for-loop to execute the recursion
- If/else were used for modelling the determinant of d_t while for-loop is used for iterating the simulation results into empty vectors

```
# Function to simulate one set of orders
def simulate_order(pi):
    # Generate 100 random orders
    orders = np.random.choice([1, -1], size=num_orders, p=[(1+pi)/2, (1-
pi)/2])  # 1 for buy, -1 for sell with distribution under GM model
    beliefs = []  # To store updated beliefs
    prices = []  # To store prices

    # Initial belief and price
    belief = theta
    price = (v_h + v_l) / 2

    beliefs.append(belief)

    for order in orders:
        if order == 1:        # Buy order
            belief, price = calculate_ask_price(price, pi, belief, v_h, v_l)
        else:            # Sell order
            belief, price = calculate_bid_price(price, pi, belief, v_h, v_l)

        beliefs.append(belief)
        prices.append(price)

    pds = [pd_measure(price) for price in prices]

    return beliefs, prices, pds ## Solution #1
```

- To workout the 10 times simulation for beliefs, prices and PD measures, I set up 3 empty vectors for iterating vectors (with 100 entries) from the outputs of simulate_order(pi).
- To deliver the speed of price discovery with comparing 3 different pi values, I append all pds vectors (in total 10 vectors with 100 entries of PDs separately) and extract every entry with same i index, to perform the averaging measure of extracted new vectors and then obtaining a new single 1x100 vector of "mean process of PDs"

```python
def simulate_orders(pi, num_orders=num_simulations):
    all_beliefs = []
    all_prices = []
    all_PDs = []

    for _ in range(num_orders):
        beliefs, prices, pds = simulate_order(pi)
        all_beliefs.append(beliefs)
        all_prices.append(prices)
        all_PDs.append(pds)

    # averaging PDs in one curve
    mean_process_PDs = np.mean(all_PDs, axis = 0)

    return all_beliefs, all_prices, all_PDs, mean_process_PDs
```

Step 7: Data Visualization for Graph #1 and Graph #2
- Consider the different π values required (int and list) in **<Graph#1>**, **<Graph#2>** and **<Graph#3>**. The data visualization is separated into 2 functions, plot_1_2() and plot_3().
- This step is to visualize the **<Graph#1>** and **<Graph#2>** where only $\pi = 0.3$ is required.

```python
# Data Visualization for beliefs and PD measure
def plot_1_2():
    # chart 1
    plt.figure(figsize=(12, 6))

    beliefs, prices, pds, mean_process_PDs = simulate_orders(0.3)

    for belief in beliefs:
        plt.plot(belief, marker='o', alpha = 0.5)

    plt.title('Belief Updates Over Time in each run under GM Model (10 runs)')
    plt.xlabel('Trade Number (1 to 100)')
    plt.ylabel('Belief (Theta)')
    plt.ylim(0, 1.2)
    plt.grid()
```

```
# chart 2
plt.figure(figsize=(12,6))

for pd in pds:
    plt.plot(pd, marker='o', alpha = 0.5)

plt.title('Pricing error: each curve shows the evolution of the pricing error in
each run (10 runs)')
plt.xlabel('Trade #')
plt.ylim(0, 16)
plt.grid()
```

```
# Data visualization for average PD measure with 3 different pi_s
def plot_3():
    # chart 3
    plt.figure(figsize=(12, 6))

    pi_s = [0.1, 0.5, 0.9]
    for pi in pi_s:
        _, _, _, mean_pd = simulate_orders(pi)
        plt.plot(mean_pd, label=pi, marker='o', alpha = 0.5)

    plt.title('Squared pricing error, averaged over 10 simulations for pi = 10%,
50% & 90%')
    plt.xlabel('Trade #')
    plt.ylim(0,5)
    plt.legend(
        loc='best',
        title='π',
        title_fontsize=20)
    plt.grid()

# Execution of graph #1, #2 and #3
plot_1_2()
plot_3()
plt.show()
```

Control phase:

<Appendix #5> is a sampled documentation of bug I encountered during the algorithm development.

Example:

```
*Bug#1 - Notepad                                                                                          —  □
File  Edit  Format  View  Help
Bug #1
If I add a new output for simulate_orders(), the data visualzation will also take all_prices into operation and plot it simultaneously with all_beliefs,
resulting a crashed graphical presentation


Working and Solution:

Experiment # 1

line 49
return beliefs, prices ## added , prices into the return of the function simulate_orders()

line 55, 56
beliefs = simulate_orders() ## simulate_order (has 2 returns, 1. belief; 2. prices)
all_beliefs.append(beliefs) ## if you append beliefs here, you append also the prices into beliefs such that the data visualization misinformed

Solution #1

line 53
all_prices = [] ## Newly added an empty vector for iterating all_prices

line 56
beliefs, prices = simulate_orders() ## Newly added , prices , one of the output of simulation_orders() to inform computer to differentiate between these 2 outputs

line 58
all_prices.append(prices) ##
```
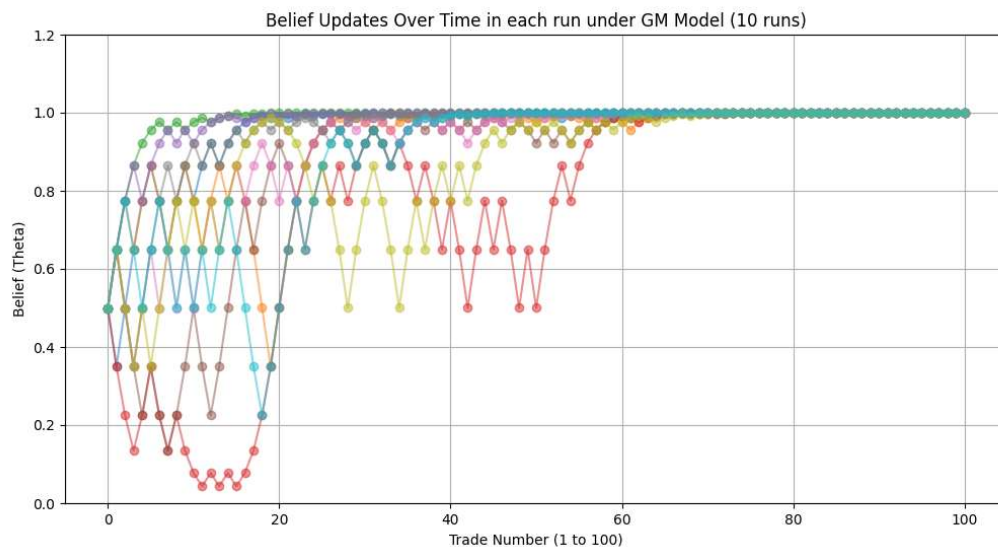
Results and Findings

Beliefs update and Price discovery

In Graph 1, I simulate 10 times the same example with an unbalanced order flow with 100 random orders per simulation to verify my understanding above.

<Graph #1>

In **<Graph #1>**, the result aligned with the lecture notes week 4 slide 32 and the understanding on the section Beliefs update and Price discovery.

Recall from my corollary under Beliefs update and Price discovery section:

Finally, if the latest order flow leans to buy side $(\Pr(d_t = 1) > \Pr(d_t = -1))$, *my hypothesis is that the lean-to-buy order flow becomes a signal to market maker to wider their spread for making profit and thus a higher market price. As such, I further deduce that the traders will tend to buy more in the subsequent trades as they tend to believe that the true value of security is undervalued and thus achieving a belief convergence*

Mathematically, **<Graph#1>** displayed a favourable result to support my hypothesis as the lean-to-buy probability (=0.65) biases the decision-making of traders and thus converging to v_h. However, is that the all-round evidence to prove my hypothesis? From my perspective as a master student, it can only fail my rejection to my hypothesis but cannot prove the hypothesis. Additionally, the GM model assumed the d_t is i.i.d., which implies that the market order is independent and identically distributed for each d across all time t, or say, serially uncorrelated to its past values. But in real world, we can always observe serial correlation in real stock data. In this case, it worths further research on the process of belief convergence.

PD measure and its variation with belief convergence
In graph 2, I simulate 10 times the PD measures on 100 orders and aim to reproduce the perfectly inverse variation between PD measure and belief convergence.

Pricing error: each curve shows the evolution of the pricing error in each run (10 runs)

<Graph #2>
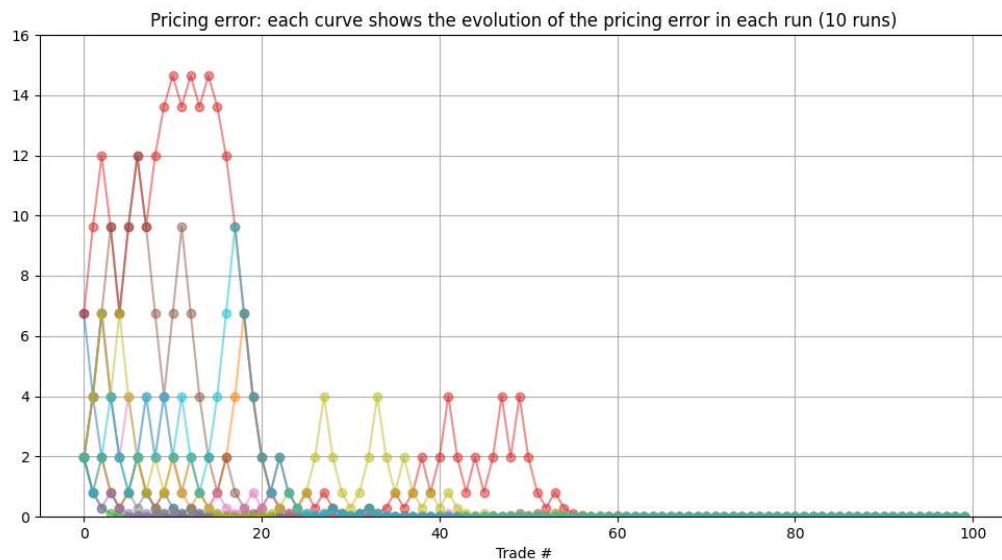
Recall the formula of PD measure.

$$PD_t \equiv (p_t - v)^2$$

In **<Graph #2>**, I observed that it is a mirror of **<Graph#1>** excepting the starting point. It is because PD is a measure of changes so the measure starts with PD_1. I noticed the assumption "The true value of asset is v_h." in lecture note

week 4 slide 30. So, the $v = v_h$ is assumed and $v = v_h$ is a constant. The only stochastic part is the p_t where p_t is determined and computed by the section in Bid-ask price, beliefs and market price. The result aligns with my understanding as discussed. I can observe a perfectly inverse relationship between PD measure and the beliefs convergence. Furthermore, as discussed in the results and findings in beliefs and convergence, the corollary provided an intuition that the sequence of PD measures is as if a pricing error correction process of traders, or say, the process to learn from pricing error and rectify the investment decision based on the current market order flow. With a view to PD measure convergence under the existence of private information reflected by $\pi > 0$, the speed of pricing error correction will converge to 0 with the same speed of belief convergence. Here, I am curious on how the value of π influence the speed of aggregated pricing correction. It will be discussed in next section.

Averaged PD measures and speed of price discovery

<Graph #3>
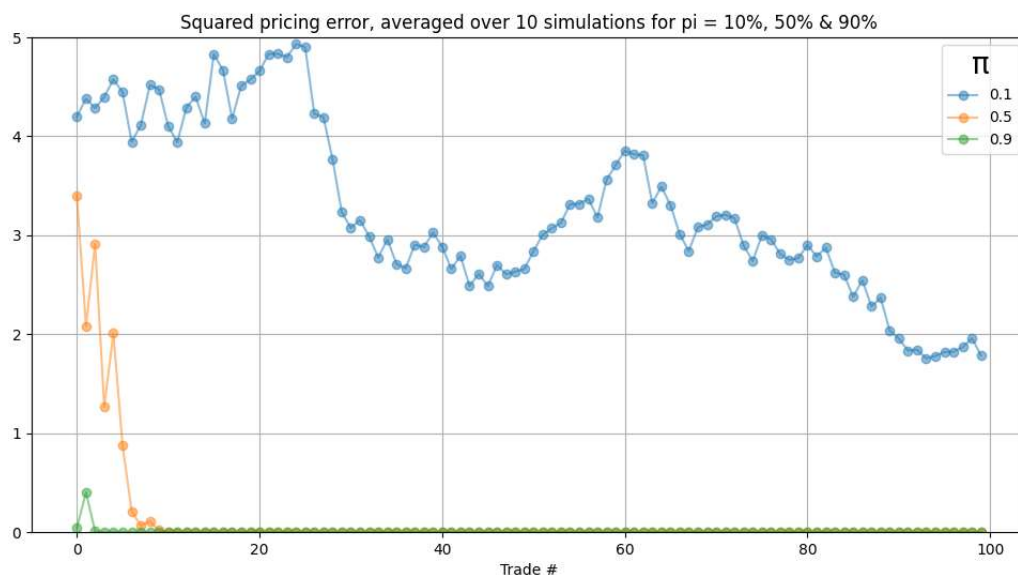
In **<Graph #3>**, it aligns with the chart on lecture note week 4 p.34 and my understanding. By summarizing all pricing error correction process from 10 traders, it is as if aggregating all informed traders in the market to model the price discovery process in a more macro view. I observe that the lower the value of π, the lower the speed of price discovery, vice versa. By this sense, if π is really the same as my understanding below

"π is the probability that traders are privately informed and insisted to what they are privately informed thus trader makes investment decision in favour of the private information."

the π would be a key indicator of existence of private information.

Reviews and Conclusion

This assignment exposed me to financial modelling replication with GM model. After finishing this assignment, I have:

1. achieved a deep understanding on GM model
2. obtained hands-on experience on financial modelling
3. experienced an algorithm design with flowcharting for financial economics
4. sharpened my coding skill in Python
5. consolidated my mathematics and analysis skills

To summarize my understanding and after-feeling in one paragraph:

"Persist what you have asserted at the beginning, the answer you are looking for will converge to your insight in the foreseeable future even if you make mistake during your journey. Be brave and explore."

This assignment is very inspirational to me. GM model brings me to the new horizon of economics and market microstructure. I believe this assignment will be the cornerstone of my future career.

Appendix

<Appendix #1>

Source code (Python) (It is also attached in the zip file)

```python
# import modules to be used
import numpy as np
import matplotlib.pyplot as plt

# Parameters
num_simulations = 10
num_orders = 100
theta = 0.5  # Probability that the true value is high
# pi = 0.3    # Probability that a trader is informed
v_h = 102    # High value
v_l = 98     # Low value

# Calculate ask_price and theta_a
def calculate_ask_price(mu_last, pi, theta_last, v_h, v_l):
    new_theta_a = ((1 + pi) / 2 * theta_last) / ((pi * theta_last) + ((1 - pi) / 2))
    new_ask_price = mu_last + ((((pi * theta_last) * (1 - theta_last))* (v_h - v_l)) / ((pi * theta_last) + ((1 - pi)/ 2)))

    return new_theta_a, new_ask_price

# Calculate ask_price and theta_a
def calculate_bid_price(mu_last, pi, theta_last, v_h, v_l):
```

```python
    new_theta_b = ((1 - pi) / 2 * theta_last) / ((pi * (1 - theta_last)) + ((1 - pi) / 2))
    new_bid_price = mu_last - (((pi * theta_last) * (1 - theta_last))* (v_h - v_l)) / ((pi * (1 - theta_last)) + ((1 - pi) / 2))

    return new_theta_b, new_bid_price

# Function to PD measure of one set of orders
def pd_measure(price):
    return (price - v_h)**2

# Function to simulate one set of orders
def simulate_order(pi):
    # Generate 100 random orders
    orders = np.random.choice([1, -1], size=num_orders, p=[(1+pi)/2, (1-pi)/2])  # 1 for buy, -1 for sell with distribution under GM model
    beliefs = []  # To store updated beliefs
    prices = []  # To store prices

    # Initial belief and price
    belief = theta
    price = (v_h + v_l) / 2

    beliefs.append(belief)

    for order in orders:
        if order == 1:        # Buy order
            belief, price = calculate_ask_price(price, pi, belief, v_h, v_l)
        else:                 # Sell order
            belief, price = calculate_bid_price(price, pi, belief, v_h, v_l)

        beliefs.append(belief)
        prices.append(price)

    pds = [pd_measure(price) for price in prices]

    return beliefs, prices, pds ## Solution #1

def simulate_orders(pi, num_orders=num_simulations):
    all_beliefs = []
    all_prices = []
    all_PDs = []

    for _ in range(num_orders):
        beliefs, prices, pds = simulate_order(pi)
        all_beliefs.append(beliefs)
```

```python
        all_prices.append(prices)
        all_PDs.append(pds)

    # averaging PDs in one curve
    mean_process_PDs = np.mean(all_PDs, axis = 0)

    return all_beliefs, all_prices, all_PDs, mean_process_PDs


# Data Visualization for beliefs and PD measure
def plot_1_2():
    # chart 1
    plt.figure(figsize=(12, 6))

    beliefs, prices, pds, mean_process_PDs = simulate_orders(0.3)

    for belief in beliefs:
        plt.plot(belief, marker='o', alpha = 0.5)

    plt.title('Belief Updates Over Time in each run under GM Model (10 runs)')
    plt.xlabel('Trade Number (1 to 100)')
    plt.ylabel('Belief (Theta)')
    plt.ylim(0, 1.2)
    plt.grid()

    # chart 2
    plt.figure(figsize=(12,6))

    for pd in pds:
        plt.plot(pd, marker='o', alpha = 0.5)

    plt.title('Pricing error: each curve shows the evolution of the pricing error in
each run (10 runs)')
    plt.xlabel('Trade #')
    plt.ylim(0, 16)
    plt.grid()

# Data visualization for average PD measure with 3 different pi_s
def plot_3():
    # chart 3
    plt.figure(figsize=(12, 6))

    pi_s = [0.1, 0.5, 0.9]
    for pi in pi_s:
        _, _, _, mean_pd = simulate_orders(pi)
        plt.plot(mean_pd, label=pi, marker='o', alpha = 0.5)
```
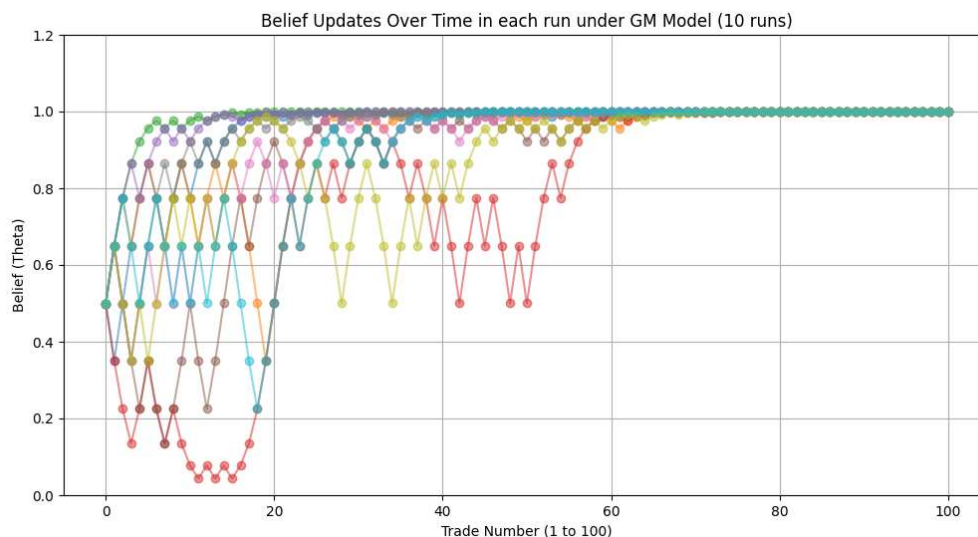
```
    plt.title('Squared pricing error, averaged over 10 simulations for pi = 10%,
50% & 90%')
    plt.xlabel('Trade #')
    plt.ylim(0,5)
    plt.legend(
        loc='best',
        title='π',
        title_fontsize=20)
    plt.grid()

# Execution of graph #1, #2 and #3
plot_1_2()
plot_3()
plt.show()
```
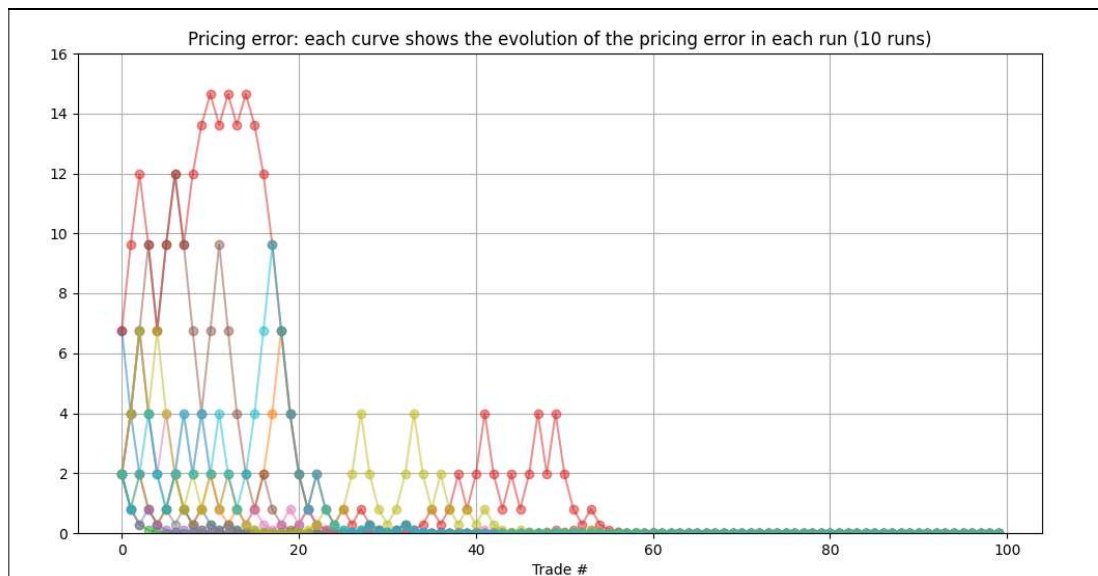
<Appendix #2> hand-written notes (It is attached in the zip file)
<Appendix #3> hand-written notes 2 (It is attached in the zip file)
<Appendix #4> flowchart (It is attached in the zip file)
<Appendix #5> bug#1 (It is attached in the zip file)

<Graph #1>

Belief Updates Over Time in each run under GM Model (10 runs)

<Graph #2>

Pricing error: each curve shows the evolution of the pricing error in each run (10 runs)

<Graph #3>

Squared pricing error, averaged over 10 simulations for pi = 10%, 50% & 90%

References

1. Glosten, L. R. & Milgrom, P. R. (1985). Bid, ask and transaction prices in a specialist market with heterogeneously informed traders, *Journal of Financial Economics* **14**, 71–100.
2. Fama, EF (1970), 'Efficient Capital Markets: a Review of Theory and Empirical Work', Journal of Finance, 25(1), pp 383–417.
3. Malkiel, B. G. (2019). *A random walk down Wall Street* (12th ed.). WW Norton.