Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Agent-Based Models in Finance: Intra-day and Multi Time Horizon Stylized Facts Approximation

Author:
Salomon Lupo

Supervisor:
Dr. Luk Wayne
Dr. Stephen Weston

Submitted in partial fulfillment of the requirements for the MSc degree in MSc in Artificial Intelligence of Imperial College London

September 2021

Abstract

Financial price movements have unique statistical characteristics which are difficult to replicate with standard methods like Geometric Brownian Motions, ARIMA, or GARCH. These financial price properties are called stylized facts. Realistic financial price simulations are essential in the modern financial industry, especially for applications in risk management.

Agent-Based Modelling is a method of simulation that reproduces an environment and the behaviour of the agents within that environment. Researchers have used this method to create a realistic simulation of traders and financial markets that replicate basic stylized facts. However, most papers are limited to approximating some of the stylized facts for daily price returns. The analysis of the goodness of fit is qualitative rather than quantitative and often lacks numerical comparison with the real market price.

This research presents an Agent-Based Model that can accurately replicate stylized facts of the S&P 500 simultaneously across multiple time horizons. The model can also reproduce the intra-day cycle of market prices and accurately represent some differences in behaviour between its different parts. We achieved this by developing a new type of agent that trades disproportionally towards the close of the trading day and according to the intra-day price movement. We call these agents Gamma Traders.

Contents

1 Introduction **1**

2 Background **4**
2.1 Financial Markets Current Context 4
2.2 Implementations in Current Literature 5

3 ABM Market Fit Approach **8**
3.1 Stylized Facts 8
3.2 Target Data 9
3.3 Benchmark 12
3.4 Loss Function 13
3.5 Random Simulator 16
3.6 Exponential Simulator 19
3.7 FCN Simulator 20
3.7.1 Volatility Modification 24
3.8 Optimizer 24
3.9 Gamma Traders 27
3.10 FCN Gamma Simulator 29
3.11 Summary 31

4 Software Implementation **32**
4.1 Code Structure 32
4.1.1 Market 32
4.1.2 Analysis 33
4.1.3 Data 34
4.1.4 Tests 34
4.1.5 Mains & Results 35
4.2 Code Performance 35

5 Evaluation **37**
5.1 Ultra Optimized ABM 37
5.1.1 Optimization 37
5.1.2 Performance 38
5.2 Legal, Social, Ethical and Professional Issues 42

6 Conclusion **44**
6.1 Summary 44
6.2 Limitations 45
6.3 Future Work 45

Chapter 1

Introduction

Realistic asset price simulation is an essential component of the modern financial industry. Monte-Carlo method helps to price complex financial derivatives by simulating multiple times the possible future evolution of the underlying asset on which a derivative is based and simply averaging the payouts. The convergence of the Monte-Carlo method is possible because of the central limit theorem and the law of large numbers [1]. Since the sub-prime crisis of 2008, regulators have placed more stringent risk-management requirements on financial firms. One of these risk-management constraints is the solvency capital requirement, which forces financial firms to hold a certain amount of liquid assets in their safe based on their portfolio's simulated worst-case scenario price movement. Capital requirements exist so that banks and other financial institutions would have enough resources to meet their obligations and perform their essential societal functions in times of crisis. It is easy to imagine the relevance of realistic asset prices simulation as statistical biases in the simulation can cost financial firms billions in the case of the inaccurate pricing of derivatives and can cost the public another severe financial crisis in the case of poorly calculated solvency capital requirements. It is in the bank's best interest to have proper risk management.

Asset prices are not easy to simulate as they follow complex patterns which often differ drastically between assets. However, practitioners have observed some generally accepted and empirically observed stylized facts that can describe the movements of equity prices well. We will illustrate the most relevant ones. The central stylized fact is the lack of auto-correlation of returns. The lack of strong auto-correlation is the most apparent stylized fact as if there were marked temporal patterns in prices, it would be easy for speculators to exploit them and, in doing so, to make the possibility of earning further profits using the same market irregularity disappear. This stylized fact is easy to simulate as we can add *i.i.d.* random variables in a basic ARMA model. The second stylized fact is called volatility clustering, and it is simply the persistence of volatility or the auto-correlation of the square of the returns. This fact models the tendency of market volatility to be persistent or, in other words, correlated with itself. GARCH models have replicated this stylized fact with some success [2]. Thirdly, we have the inverse relationship between prices and volatility of returns. It has been observed that, in times of economic boom and rise in prices,

there is low volatility, and conversely, in times of crises and sudden price decline, we tend to observe a persistent increase in volatility. This phenomenon might exist because people rarely need to think about their portfolios in expansion times, and in times of crisis, many people panic and re-balance their portfolios. Finally, the most critical stylized fact is that financial price returns are not normally distributed but follow a distribution with fat-tails or high Kurtosis. This stylized fact is significant because if we were to simulate prices with normal distribution, we would likely be underestimating the potential losses that could occur.

Agent-based modelling (ABM) is a modelling technique that has been increasing in popularity in recent years. ABM simulates the behaviour of decision-making entities called agents within a system, sometimes referred to as the environment. Increases in computing power have made ABM a powerful method to model complex agent interactions and their effect on the system as a whole. It is a technique best suited to model interactions that would be difficult to model using traditional methods like differential equations. It is also used in cases where it would be intractable to model interactions mathematically as the complexity of the equations increases exponentially with the number of interactions. The most straightforward type of ABM is one in which agents have simplistic, well-defined rules and ways of interacting; however, there are also more complex forms of ABMs where agents are allowed to modify their rules during the simulation, often agents make decisions using neural networks that are trained with the principles of reinforcement learning [3].

There are many benefits of using ABM. The most notable benefit is the ability to capture emergent behaviour and its relationship with the rules of behaviour of the agents. Emergent behaviour is a feature of the system that results from the interaction of the individual agents, and by definition, it cannot be reduced and localized to features of part of the system. Another benefit is that ABM is a natural and intuitive way to describe a system. For example, it is simpler to describe a football match by the functions and behaviour of individual players rather than defining complex mathematical rules for how each player moves around the field. Another significant benefit of using ABM is its flexibility. It is straightforward to observe the effects on emergent behaviour of adding agents, modifying decision rules, creating subgroups of agents, etc. [3]. ABM lends itself well to financial markets as it is an environment in which agents compete and, by doing so, create emergent behavioural patterns of the system. In the context of financial markets, we can use ABM to model traders, which are the agents, and observe the effects of their interactions on the price movement within the asset exchange environment.

This project documents the creation of an ABM that can create simulated exchange prices with stylized facts that closely match the stylized facts of actual price movements, which we defined at the beginning of this section. To check the potential of our ABM, we use real-life minute-frequency S&P 500 price data to check the match in stylized facts. We do not only match daily returns, as most papers in the literature do, but we attempt to match stylized facts of five time horizons (1, 5, 15 and

30 minutes; and 1 day) simultaneously. In addition to the four stylized facts relatively common in the literature, we develop a stylized fact specifically designed to address the idiosyncrasies of price movements towards the end of the trading day. The matching of this stylized fact is the most original contribution of this project, which, as far as we know, has not been attempted yet in the literature. To match the close-of-day stylized fact, we create a particular type of agent called Gamma Trader, which replicates the behaviour of leveraged passive-index providers. Their behaviour has not been described in the context of ABM as far as we are aware.

Creating an ABM that can accurately replicate intra-day market prices movement can enable the creation of tools that traders can use to calibrate their hedging strategy. For example, traders could use the tools to estimate the distribution of the performance of their trading algorithm by simulating an intra-day market many times and letting the algorithm participate in the market simulation. This tool could also test the impact of the trading algorithm on the liquidity of the market. In addition, replicating the stylized facts over multiple time horizons can be helpful to calibrate capital requirements more accurately as capital requirements for each of the banks' positions depend on the position's expiry date.

Chapter 2

Background

2.1 Financial Markets Current Context

Current Monte-Carlo methods that replicate prices as stochastic processes fail to match all of the stylized facts mentioned. Most financial firms use stochastic volatility simulation that can replicate most of these stylized facts except the fat-tails in the distribution of the price returns [4]. However, agent-based simulation methods have shown promise in replicating all these facts and more [5]. Agent-based models attempt to simulate the market in much greater detail than standard time-series stochastic processes as agent-based models simulate singular actions of market participants [6]. To understand the value-added of agent-based models, one must first understand more about the functioning of most financial market exchanges and the Continuous Double Auction (CDA).

CDA is the trading system that most exchanges use to match buyers and sellers of financial assets. Modern asset exchanges have dozens of order types; however, the main types of orders in a CDA are limit orders, market orders and cancel orders. Limit orders are orders submitted to the exchange committing to buying or selling a determined stock volume at an agent-determined price. The aggregation of limit orders is called limit order-book (LBO). Market orders are immediate transaction orders that execute some of the best-prices limit orders available on the exchange. Cancel orders are deletions of some of the previous limit order submissions. The most followed statistic of a CDA exchange is the mid-price, the average between the best bid and the best ask. The best bid is the highest buy limit order price available on the exchange, and the best ask is the lowest sell order available on the exchange. Other vital statistics include best bid and best ask volume and total bid and ask volume.

There are many different categories of actors that submit orders in exchanges. The most relevant actors for this project, that care about the fine details of the limit order book, are called market makers. Market makers provide liquidity by submitting limit orders on both sides of the mid-price, and in return for this service, they try to capitalize on the bid-ask spread, i.e., the difference between the best ask and the

best bid prices.

The primary type of actor that we examine in this project we call Gamma Traders. They are traders whose need to execute towards the close of trading is a function of the day's price movement. In recent years the amount of money in leveraged Exchange Traded Funds (ETF) and derivatives positions has significantly increased. Traders that provide these services need to follow the market trend to replicate the exposure they sell to clients. This need forces them to execute proportionally to the daily return at the close of trading. This need often causes trends to intensify. The effects of these type of actors would be interesting to study in the context of agent-based models as they can increase the realism of the simulation.

2.2 Implementations in Current Literature

Many researchers have attempted to solve realistic price simulation, given its importance within financial markets. The prominence of ABM in recent years has led to several applications on financial prices simulation.

In [7], Majewski et al. extend Chiarella's model to create an ABM where there are three types of agents: fundamental, trend-followers, and noise. They then calibrate their model using real financial prices and observe realistic simulated price movements, including the variable relation between past trends and future returns. They observe that trend-followers cause long periods of divergence from the fundamental price of the asset both in the positive and the negative direction.

Majeski et al. assume that the price change within an interval is proportional to the demand for that asset within that period. i.e $P_{t+\Delta} - P_t = \lambda D(t; t+\Delta)$. They define the dynamics of demand to be described by the following:

$$
\begin{aligned}
D(t, t+\Delta) &= \tilde{k} \int_t^{t+\Delta} (V_s - P_s) ds + \tilde{\beta} \int_t^{t+\Delta} tanh(\gamma M_s) ds + \tilde{\sigma_N} \int_t^{t+\Delta} dW_s^{(N)} \\
dP_t &= k(V_t - P_t)dt + \beta tanh(\gamma M_t)dt + \sigma_N dW_t^{(N)} \\
dM_t &= -\alpha M_t dt + \alpha dP_t \\
dV_t &= gdt + \sigma_V dW_t
\end{aligned}
\tag{2.1}
$$

In Equation 2.1 M_t represents the momentum of the price, V_t represents the fundamental value of the underlying, which, as explained in the equation, follows a Brownian Motion. P_t is the dynamic of the price and dP_t is found by taking the derivative of $\lambda D(t, t+\Delta)$. In $D(t, t+\Delta)$, the first integral represents the impact of fundamental traders as this component will push the price towards the fundamen-

tal value V_t. The second integral represents the trend-followers, pushing the price according to the direction and magnitude of M_t, i.e. the momentum. The third and final component represents the noise traders as it is fully random and independent of past performance or the fundamental value.

The relative importance of the types of traders is given by the relative magnitude of parameters k, β, and σ_N. The authors use the Expectation-Maximization (EM) algorithm to calibrate the parameters to historical financial price movements.

Essentially, the authors develop a Stochastic Partial Differential Equation that represents the dynamics of the price. The price movement is an aggregate of the total demand of each type of agent. This mathematical approach can lead to great generalization of the market; however, the individual agent customization is limited as there is no financial market environment replication, and the authors need to develop a new mathematical for each customization. In addition, it might not always be possible to find a closed-form solution to the mathematical model. A more computational approach to simulating fundamental, trend-following, and noise agents could be used to swiftly test the effects of modifying the base model for a more bespoke implementation. For example, researchers could quickly test the impact of changing the order arrival time distribution on price movements.

In [8], Maeda et al. assume the market is comprised of the same type of traders as in [7] but take a computational approach in replicating the dynamics. The authors simulate a Continuous Double Auction market and the agents trading within the exchange. The agents have characteristics of fundamental, trend-following and noise traders in different proportions. In this case, the authors have more flexibility in how they let the agents interact with one another. This method outputs not only a relatively realistic price movement but also a Limit Order Book, which would otherwise be latent for the technique in [7].

In this paper, the authors are mainly concerned with creating an environment in which to train a Reinforcement Learning agent. The fact that we have an environment for training further explains the advantage of using the computational method over the mathematical. One of the limitations of this paper is that the authors do not use historical data to calibrate the market and have no consideration for a metric to quantify the goodness of fit of the simulated market to a real market. Validating the environment with real data would be helpful to prove that their Reinforcement Learning agent can learn to perform well in a realistic market.

In [9], Mota et al. use a computational method of ABM to replicate stylized facts like the no auto-correlation of returns, volatility clustering and fat-tails. They define fundamental and trend-following strategies, similarly to the previously explained papers. Then they introduce heterogeneity in the system by varying each agent's relative importance of each trading strategy. They achieved a good level of success in replicating the stylized facts. However, this paper presents the potential of their

method to fit stylized facts without offering a method to fit to a custom financial time series. In addition, this paper has no consideration for intra-day characteristic patterns or fitting different time horizons.

In [10], Gilli et al. present a continuous global optimization heuristic for stochastic approximation of an objective function. This approximation can be used to optimize parameters to minimize a loss function in the context of an Agent-Based Modeling of financial markets. Once one has approximated the objective function satisfactorily by evaluating it at many locations in the parameter space, one can optimize the approximation to find the global minimum more efficiently. This method has a huge limitation: one needs to evaluate it at many points in the parameter grids, and therefore the time complexity of this algorithm scales exponentially with the number of parameter dimensions. In practice, when one has many parameter dimensions, it would be better to use Bayesian optimization, which can make smarter decisions and reduce the number of times one has to query the slow objective function.

As far as we know, in the literature, it has not been attempted to create a computational ABM to fit stylized facts across multiple time horizons and toward the close of the day. This model is the contribution of this paper.

Chapter 3

ABM Market Fit Approach

3.1 Stylized Facts

The project aims to create a simulation that matches real data stylized facts, so it is helpful to define these stylized facts mathematically to measure performance effectively.

For the no-auto-correlation of returns stylized facts we need to calculate the following metric:

$$\begin{aligned}
\rho_i^{ac} &= Corr(r_j, r_{j-i}) \\
&= \frac{Cov(r_j, r_{j-i})}{\sqrt{Var(r_j)}\sqrt{Var(r_{j-i})}} \\
&= \frac{\mathbb{E}[(r_j - \mathbb{E}[r_j])(r_{j-i} - \mathbb{E}[r_{j-i}])]}{\sqrt{\mathbb{E}[(r_j - \mathbb{E}[r_j])^2]}\sqrt{\mathbb{E}[(r_{j-i} - \mathbb{E}[r_{j-i}])^2]}} \\
&= \frac{\sum_j (r_j - \frac{1}{n}\sum_j r_j)(r_{j-i} - \frac{1}{n}\sum_j r_{j-i})}{\sqrt{\sum_j (r_j - \frac{1}{n}\sum_j r_j)^2}\sqrt{\sum_j (r_{j-i} - \frac{1}{n}\sum_j r_{j-i})^2}}
\end{aligned} \tag{3.1}$$

where i is the lag and r are the price returns for 1 minute, 5 minutes, 15 minutes, 30 minutes and 1 day intervals

For the volatility clustering stylized fact we use:

$$\rho_i^{vc} = Corr(r_j^2, r_{j-i}^2) \tag{3.2}$$

For the leverage effect stylized fact we use:

$$\rho_i^{le} = Corr(r_j^2, r_{j-i}) \tag{3.3}$$

For the fat tail stylized fact, we look at the distribution of returns for the 5 time horizons:

$$r \sim \Lambda(x) \tag{3.4}$$

where Λ represents the distribution of the returns.

Finally, since we want to represents the peculiarities of intra-day price movements towards the close, we developed a stylized fact that looks at the correlation between intra-day returns between the open and a time during the day and returns between that point during the day and the close of trading:

$$\rho^{cc}_{\frac{i}{j}} = Corr(\frac{P_{\frac{i}{j}}}{P_o}, \frac{P_c}{P_{\frac{i}{j}}}) \tag{3.5}$$

Where P_o is the opening price, P_c is the closing price and $P_{\frac{i}{j}}$ is the price at the $\frac{i}{j}$ fraction of the trading day. For example $P_{\frac{1}{2}}$ would be the price at the middle of the training day. We specifically chose 5 points of the trading day which we believe give an indication of near the close price movements. These points are: $\frac{1}{2}$, $\frac{6}{10}$, $\frac{7}{10}$, $\frac{8}{10}$, $\frac{9}{10}$.

3.2 Target Data

As our target market, we chose the S&P 500 (SPX), representing the price of a market-capitalization-weighted basket of the 500 most valuable US companies. This index is probably the most common benchmark used to measure investment funds performance and will do fine for our purposes as it shows the stylized facts we wish to replicate. We use 1-minute frequency SPX prices from 27th of April 2007 until 21st of July 2021.

We analyze the stylized facts across time horizons. In Figure 3.1 we can see the stylized facts for daily returns. The figure has five graphs. The top shows the evolution of SPX prices over the years. The left-centre graph shows the auto-correlation stylized fact. More specifically we show the first auto-correlation of daily returns for the first 10 lags, which if we refer back to equation 3.1 it would be ρ^{ac}_i for $i \in [1, 2, ..., 10]$. We observe that for most of the lags except for lag 1, we get negligible auto-correlation, which we expected. We did not expect such a strong correlation for the first lag; however, our goal is to construct an ABM that can match this information, so we will not concern ourselves with this discrepancy from our expectation of stylized fact. The centre graph shows the volatility clustering information, which is calculated using 3.2. As expected, we observe a strong auto-correlation of volatility, which suggests that the process is non-Markovian and has memory. On the centre-right, we have the leverage effect stylized fact, which shows the negative auto-correlation

between volatility and returns, which is best explained by the fact that in times of crisis, we tend to see increased volatility and vice-versa, in times of good economic performance we tend to see lower volatility. Finally, on the bottom, we see a density histogram of the daily returns. The returns distribution is standardized (we subtract the mean and divide by standard deviation) plotted in comparison with the normal distribution to more clearly observe that returns distribution is fat-tailed and has high Kurtosis. The fat tails of returns is an important stylized fact to replicate since a risk model that does not match this well will underestimate potential losses.

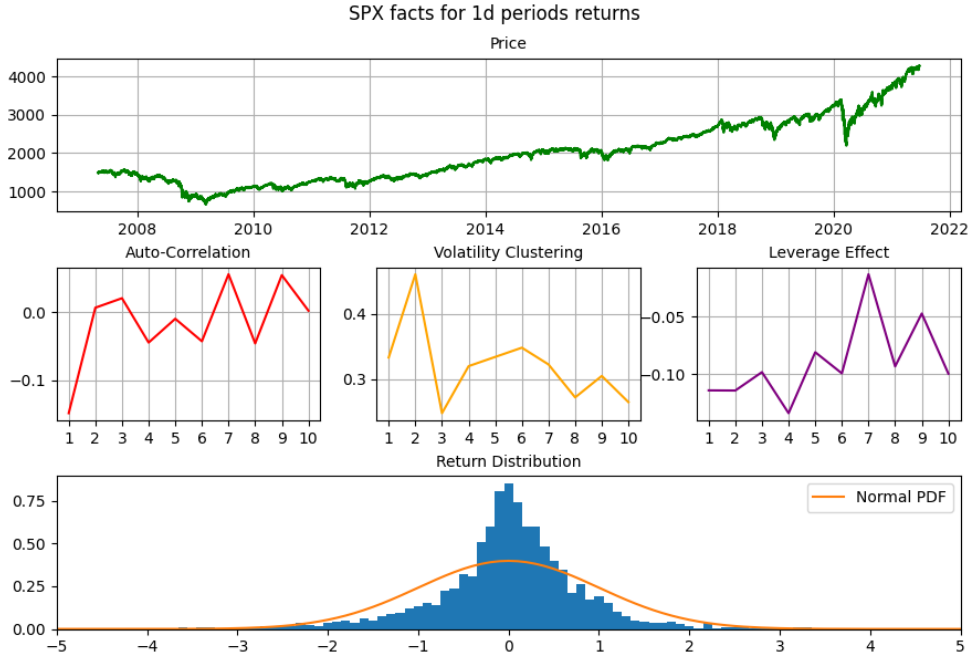

Figure 3.1: Stylized facts for SPX daily returns

In Figure 3.2 we show the stylized facts for another time horizon: The 1-minute returns. We can observe that we have similar stylized facts compared to the 1-day returns. We notice that for the auto-correlation stylized facts, we have a small positive correlation for the first few lags instead of the 1-day returns that have a negative correlation. This feature suggests that the market is trending in the short term and mean-reverting in the long term. In addition, the leverage effect is not very significant for the 1-minute returns. This difference across time horizons shows the value of replicating the stylized facts of multiple time horizons at the same time.

In Figure 3.3 we can see the custom close-correlation stylized fact for the for SPX. We notice that the correlation is positive in all cases, suggesting the presence of an intra-day trend. This presence is consistent with the fact that the auto-correlation of 1-minute returns is positive for the first few lags. However, we notice that the close-correlation is stronger for the last section of the day, which is consistent with our

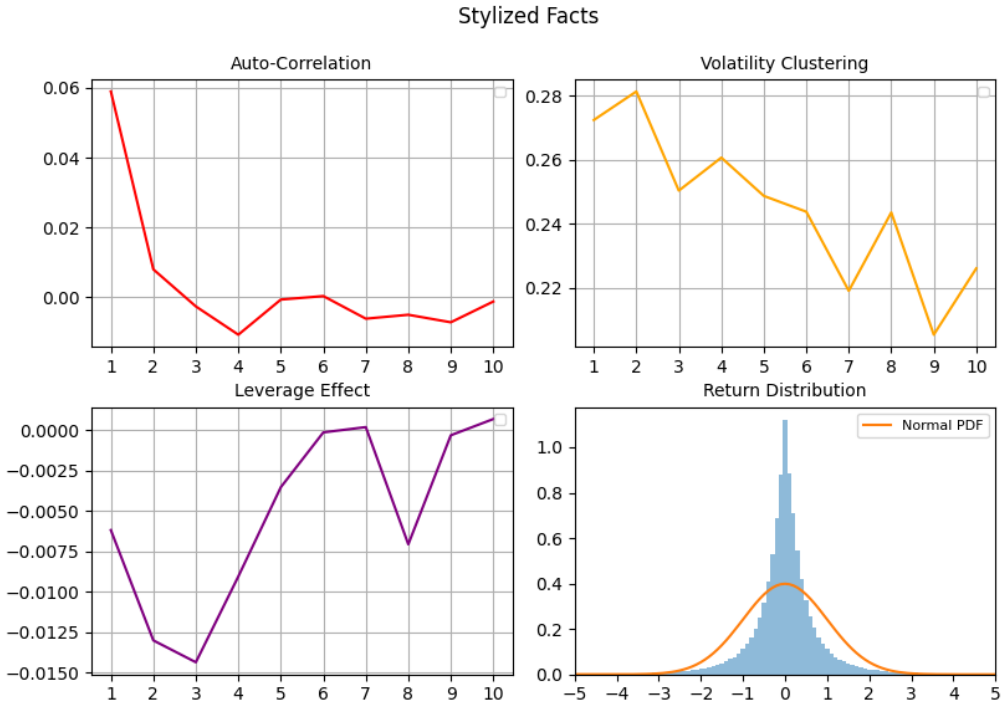

Figure 3.2: Stylized facts for SPX 1-minute returns

hypothesis that Gamma Traders (rebalancers) significantly affect prices towards the close. This effect exists because Gamma Traders are forced to execute large trades in the direction of price movements, exacerbating the trend.

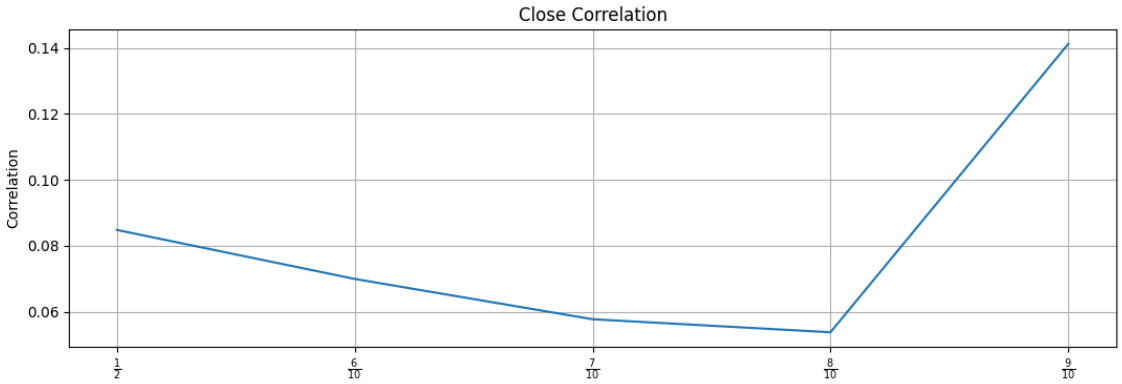

Figure 3.3: Close Correlation for SPX prices

3.3 Benchmark

Now that we have defined the aspects of price movements that we wish to replicate, for the purpose of benchmarking, it is essential to present a current and established method of data simulation that we want to improve upon. We choose a Geometric Brownian Motion.

A Geometric Brownian Motion (GBM) is a stochastic process that is often used to simulate prices to perform Monte-Carlo pricing or risk calculations. The formula for a GBM:

$$S_t = S_0 e^{(r-\frac{1}{2}\sigma^2)t+\sigma W_t} \tag{3.6}$$

Where t is time, r is the price trend, σ is the price volatility, and W_t is a standard Brownian motion. A stochastic process is a standard Brownian Motion if the following conditions are satisfied:

$$\begin{aligned} &(1) \quad W_0 = 0 \\ &(2) \quad \mathbb{E}[W_t - W_s] = 0 \\ &(3) \quad Var[W_t - W_s] = t - s \end{aligned} \tag{3.7}$$

for all t ≥ s

Equation 3.7 defines that a Brownian motion has independent and identically distributed increments.

We can simulate the performance of a GBM at discrete time intervals by using the following formula recursively:

$$\begin{aligned} S_{t+\Delta} &= S_t e^{(r-\frac{1}{2}\sigma^2)\Delta+\sigma\sqrt{\Delta}Z_i} \\ \textit{where} \quad Z_i &\sim \mathcal{N}(0,1) \end{aligned} \tag{3.8}$$

where Z is pseudo-random numbers that replicates sampling from the standard normal distribution.

The independence of increment property of the Standard Brownian motion tells us that ρ_i^{ac}, ρ_i^{vc}, ρ_i^{le}, and ρ_i^{cc} are zero 0 and that the returns are normally distributed without fat tails. This can be easily seen from:

$$ln(\frac{S_t}{S_s}) = (r - \frac{1}{2}\sigma^2)(t-s) + \sigma W_{t-s} \tag{3.9}$$

Which is a Brownian Motion and therefore has has independent increments and is normally distributed. We show an example of the stylized facts of a GBM in Figure

3.4 which confirms the prediction of the GBM theory.

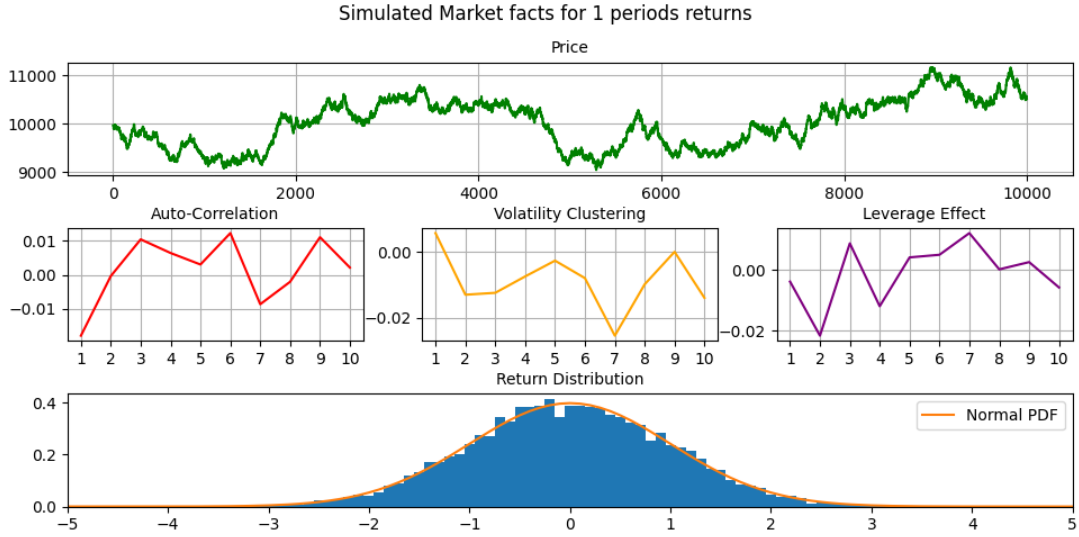

Figure 3.4: Stylized facts for GBM 1-day returns

Understanding the performance of the benchmark in details is important as it enables a more in-depth comparison of the performance of the ABM. In Figure 3.4 we can see the stylized facts for the GBM's 1-minute returns. On the sub-image in the centre-right, we can see that the auto-correlation of return are not significantly different from zero. The same is true for the volatility clustering and the leverage effect, which are in the sub-images on the centre and centre-right. The empirically observed correlation stylized facts are consistent with theory prediction. On the bottom graph, we can see the density of the returns and that they match well with the standard normal distribution, which is also consistent with the prediction.

3.4 Loss Function

Now that we have defined the behaviour of the benchmark, we still need to measure the performance numerically. For this, we need to define a loss function clearly. For the close-correlation stylized fact, we describe the loss to be the mean absolute error

of the correlations.

$$
\begin{aligned}
l^{ac} &= \frac{1}{10}\sum_{i=1}^{10}|\rho_i^{ac} - \bar{\rho}_i^{ac}| \\
l^{vc} &= \frac{1}{10}\sum_{i=1}^{10}|\rho_i^{vc} - \bar{\rho}_i^{vc}| \\
l^{le} &= \frac{1}{10}\sum_{i=1}^{10}|\rho_i^{le} - \bar{\rho}_i^{le}| \\
l^{cc} &= \frac{1}{5}\sum_{j}|\rho_j^{cc} - \bar{\rho}_j^{cc}| \\
\textit{where} \quad & j \in [\frac{1}{2}, \frac{6}{10}, \frac{7}{10}, \frac{8}{10}, \frac{9}{10}]
\end{aligned}
\tag{3.10}
$$

In equations 3.10 the bar represents simulated data.

We choose mean-absolute-error for the correlation loss instead of mean-square-error because we want to prioritize all stylized facts equally. Using mean square error would prioritize the stylized fact that is most difficult to replicate.

We are left with defining the loss for the fat-tail stylized fact. To find the difference between two distribution we can use the Wasserstein distance.

$$
l_1(u, v) = \int_{-\infty}^{+\infty} |U - V| \tag{3.11}
$$

Where l_1 is the Wasserstein distance for distributions u and v, while U and V are CDFs of their respective distribution.

The Wasserstein distance is also called the Earth-Mover distance because, if you imagine the distribution to be piles of dirt, it could be an approximation about the amount of work necessary to transform a pile of dirt into the other. We choose this distance because it tries to approximate all moments of the distribution. However, we are not interested in matching the first and second moment, i.e. mean and standard deviation, so we will standardize the distribution before calculating the distance:

$$
\begin{aligned}
\hat{r_R} &= \frac{r_R - \mu_R}{\sigma_R} \\
\hat{r_S} &= \frac{r_S - \mu_S}{\sigma_S} \\
l^{ft} &= l_1(\hat{r_R}, \hat{r_S})
\end{aligned}
\tag{3.12}
$$

Finally, now that we have all the components of the loss we need to aggregate it to define the relative importance to our optimizer. We choose the following aggregation of the losses:

$$L = l^{cc} + \sum_i \frac{1}{4}(l_i^{ac} + l_i^{vc} + l_i^{le} + l_i^{ft}) \quad \textit{where} \quad i \in [\textit{1-min}, \textit{5-min}, \textit{15-min}, \textit{30-min}, \textit{1-day}] \tag{3.13}$$

We can see that L will give equal weights to each time horizon but will consider the close-correlation loss to be the most important single loss since it is the most important stylized fact for Gamma Traders.

We have all the tools to compare simulated markets to the SPX. In Figure 3.5 we can see an overlay of the 1-minute returns for both the real data (SPX) and the benchmark simulated data (GBM), as well as the loss for each of the stylized facts except close-correlation. We can see that the benchmark loss is mainly driven by the fat-tail and the volatility-clustering stylized facts.

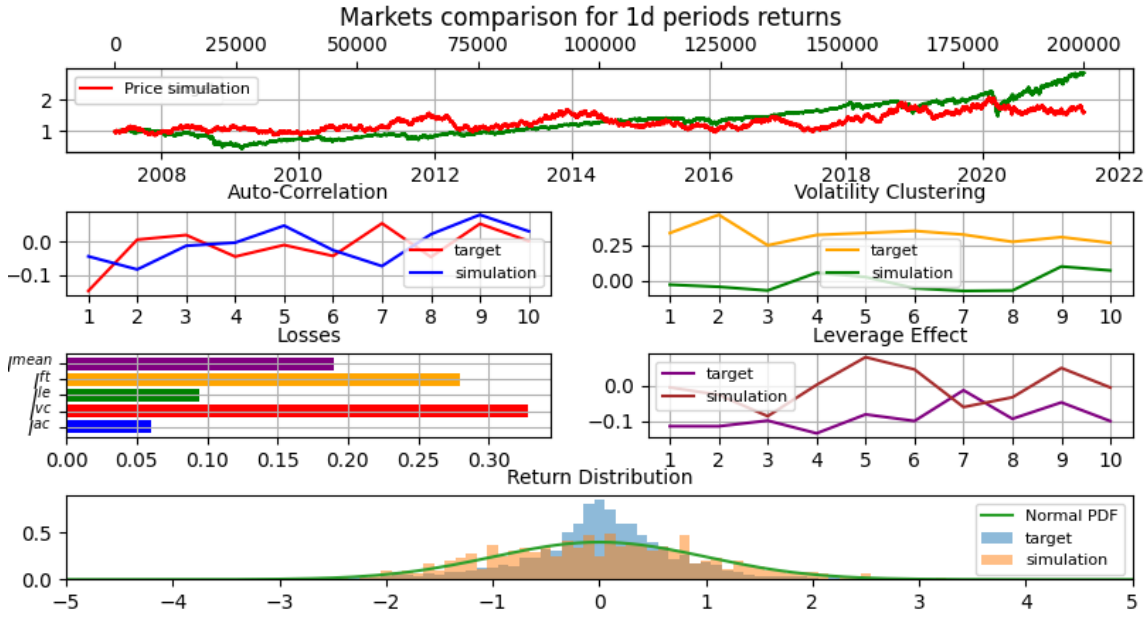

Figure 3.5: Comparison SPX and GBM 1 minute returns

In Figure 3.6 all the components of the loss L in Equation 3.13 for the GBM benchmark. The sub-script on the l for some of the values of the y axis represent the time horizons of the loss (1, 5, 15, and 30 minutes; and 1 day). l_{close} represents the loss

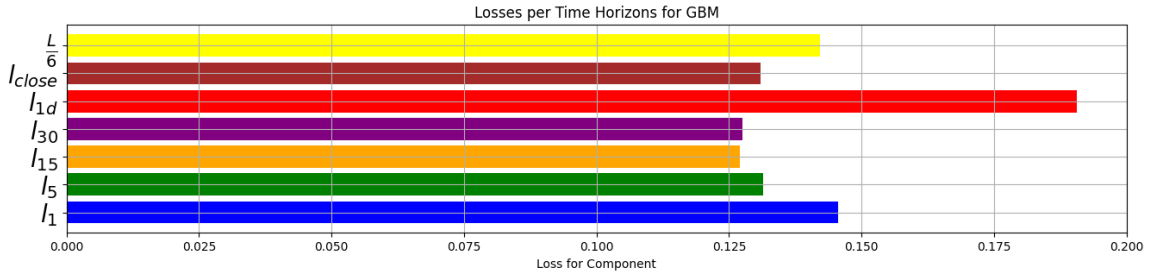

Figure 3.6: Losses for GBM

of the close-correlation stylized fact, and $\frac{L}{6}$ in yellow is the total loss divided by 6 which is the average of all the other components of the loss.

3.5 Random Simulator

We start using an agent based model simulation that is relatively simple, and we will progressively add complexity to arrive at our final ABM simulator.

Algorithm 1 Simulator: Basic Structure

```
trade_times ← sample_trading_times()
cancel_times ← sample_cancel_times()
while time < simulation_time do
    if time in trade_times then
        agent ← sample_agent()
        agent.decide_order()
        agent.submit_order(exchange)
    end if
    if time in cancel_times then
        agent ← retrieve_agent()
        agent.cancel_order(exchange)
    end if
end while
```

The pseudo-code in Algorithm 1 shows the basic structure of all the ABM simulations that this research presents. We start by sampling trading times and order cancellation times. These times can be either deterministic or randomly sampled from a Poisson distribution. Then we let the simulation run for a determined number of steps, and at the trading times, we randomly sample an agent. That agent then decides the order to place; this can be either a market order or a limit order and can use the information available from the exchange at the trade time. After the decision, the agent submits the order to the exchange, which handles the execution. At cancel times, we retrieve the agent that submitted a particular order, and if that

order has not been executed yet, we cancel it from the exchange. It is clear that the most crucial component of the algorithm is the process that the agent uses to decide the order price, volume and type.

In the case of the Random Agent/Simulator, we deterministically choose the trade

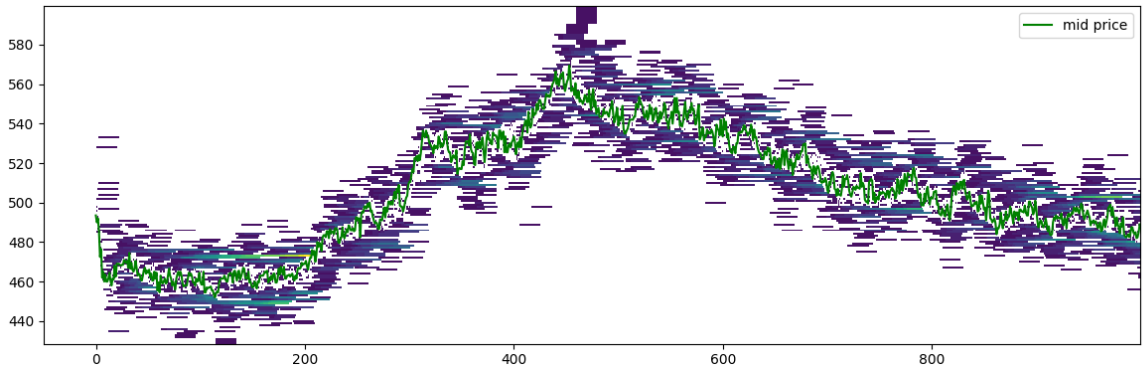

Figure 3.7: Random Agent Simulator Zoomed-In

and cancel times. For example, if we wanted a simulation that lasts 1000 minutes and we wanted to have 6 orders submitted per minute, we would have a trade every 10 seconds. We also cancel orders after a fixed time after the trade origination. The formula we use is to determine the price is:

$$P_o = P_m e^{\mu - \frac{1}{2}\sigma^2 + \sigma Z_i} \tag{3.14}$$

Where P_m is the mid-price of the exchange at the time of the decision, μ and σ are the simulation trend and volatility, and Z_i is a standard normal sample. We choose randomly if the order is a buy or a sell order. If the order is a buy order and the order price is higher than the best ask we submit a market order; otherwise, we submit a limit order. If the order is a sell and the price is lower than the best bid, we submit a market order; otherwise, we submit a limit order.

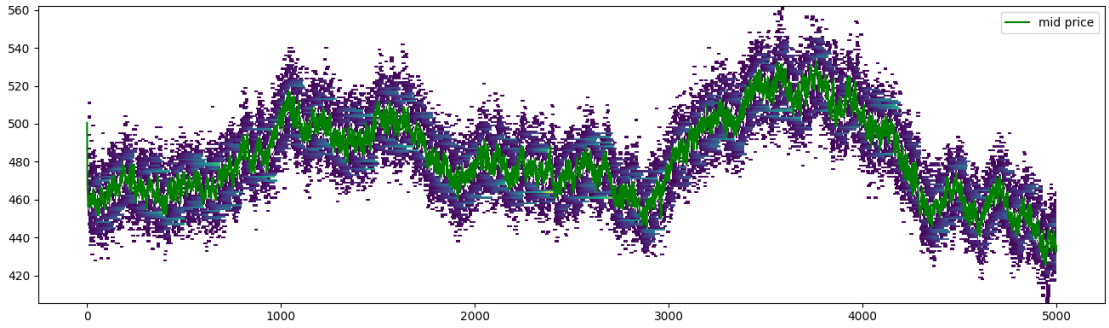

Figure 3.8: Random Agent Simulator Zoomed-Out

In Figures 3.7 and 3.8 we show some examples of the output of the Random Simulation. We have a more detailed view in one image, and in the other, we take a long-term performance view. The most important feature to notice from these figures is the presence of blue lines around the mid-price. The blue lines represent the

presence of limit order on the exchange, and the intensity in their colour represents the volume of the limit orders at that tick level. The ability to get this data from the simulation is an important feature of ABMs.

In Figure 3.9 we show the performance of the Random Simulation for 1-day returns. We can look at Figures 3.6 and 3.10 to compare the difference in performance between the GBM benchmark and basic ABM Random Simulator.

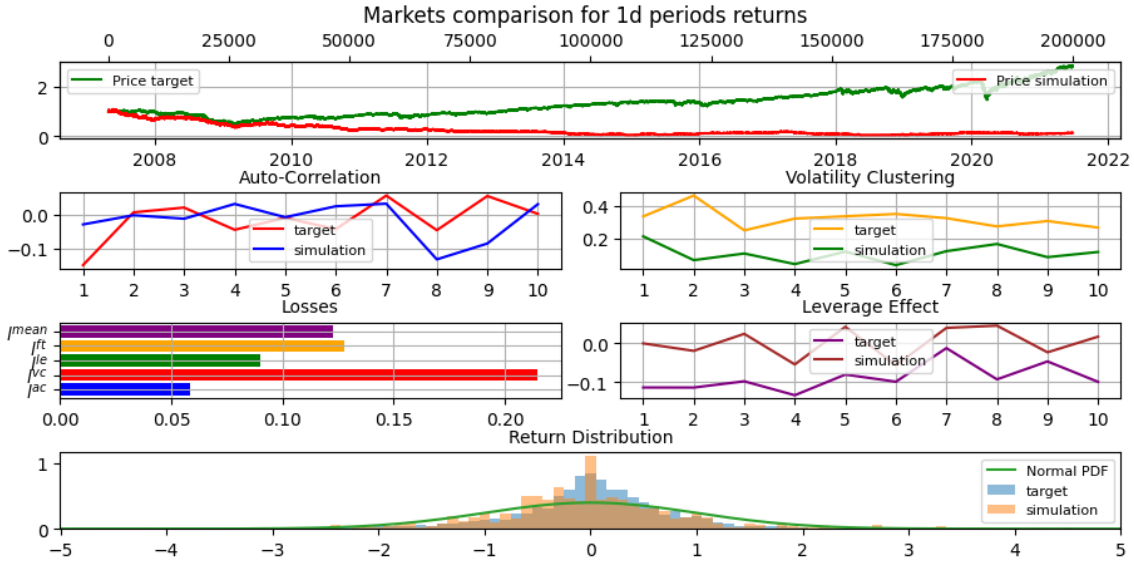

Figure 3.9: Comparison SPX and Random Agent Simulator 1 minute returns

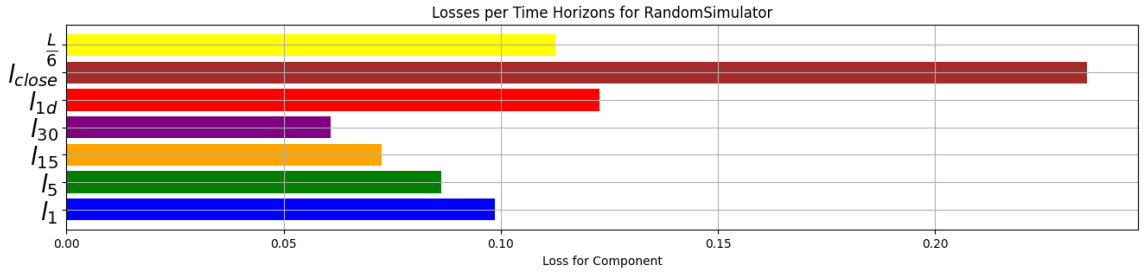

Figure 3.10: Components of Loss for Random Simulator

We can see from the Return Distribution graph in Figure 3.9 that the distribution has a decent level of fat tails as it is more similar to the target than to the normal distribution. We can also see that the simulation achieves a slight volatility clustering effect as the correlation of the square of returns for the first lag is significantly positive (see Volatility Clustering). However, this model has not replicated the leverage effect at all and cannot match the autocorrelation of returns (see Leverage Effect and Auto-Correlation). In Figure 3.10 we see that in terms of time horizons, this model performs well; however, it lacks in replicating the close-correlation patterns. This deficit is not surprising as there is nothing in the model that addresses daily cyclicality.

3.6 Exponential Simulator

We tested an additional ABM, which was primarily inspired by Ponta et al. [5]. In their paper, Ponta et al. explore the effect of the distribution of order arrivals and waiting time on the emergent behaviour of the market. They use homogeneous agents with the same trading strategy, and they observe that if the order's arrival time follow a Weibull distribution, they can observe fat-tailed distribution or returns. We want to improve our RandomSimulator model to include random order times.

In this model, we extend the $sample\_trading\_times()$ function from Algorithm 1 to sample order times that have waiting times that are exponentially distributed. The exponential distribution is a special case of Weibull distribution. The order waiting time has a pdf distribution which is the following:

$$\begin{aligned} \phi(\tau_i) &= \lambda e^{-\lambda \tau_i} \\ t_n &= \sum_{i=1}^{n} \tau_i \end{aligned} \tag{3.15}$$

Where λ is a proxy for the average number of trades per minute and t_n is the time at which the nth trade occurs.

On the other hand, cancel orders are still deterministically determined from the trade time. We also slightly change the $decide\_order()$ agent method. For this simulation, we randomly select if the order is a buy or a sell and depending on this, we determine the price using:

$$\begin{aligned} \textit{if buy:} \quad & p_h = n(t_h) \cdot d(t_{h-1}) \\ \textit{if sell:} \quad & p_h = n(t_h) \cdot a(t_{h-1}) \end{aligned} \tag{3.16}$$

where n is a Gaussian variable, d is the exchange best-bid, and a is the exchange best-ask. According to Equation 3.16 the agents offer a more competitive position half of the time. This Exponential simulation rules lead to the time series and losses shown respectively in Figures 3.16 and 3.12.

In Figure 3.13 on the Return Distribution graph, we can see that we achieve even fatter tails than our target. However, we poorly match the auto-correlation stylized fact. In addition, we can see from Figure 3.13 that the goodness of fit steadily decreases for more distant time horizons. This loss increase is partly due to the fact that we lose the fat-tail distribution of returns.

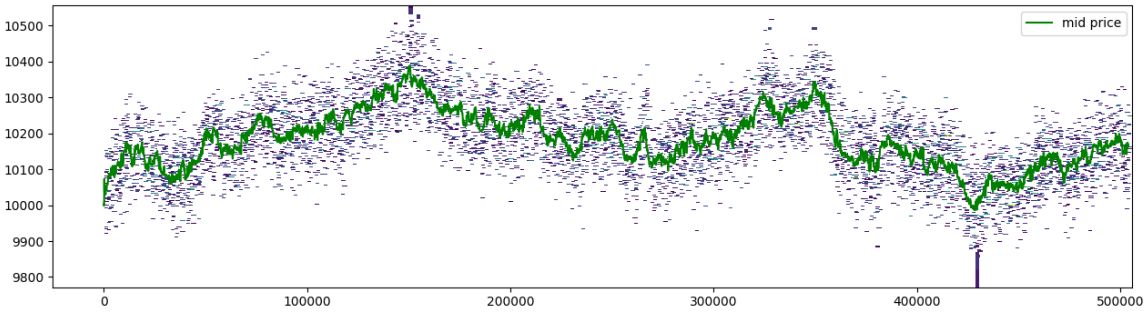

Figure 3.11: Exponential Agent Simulator Performance

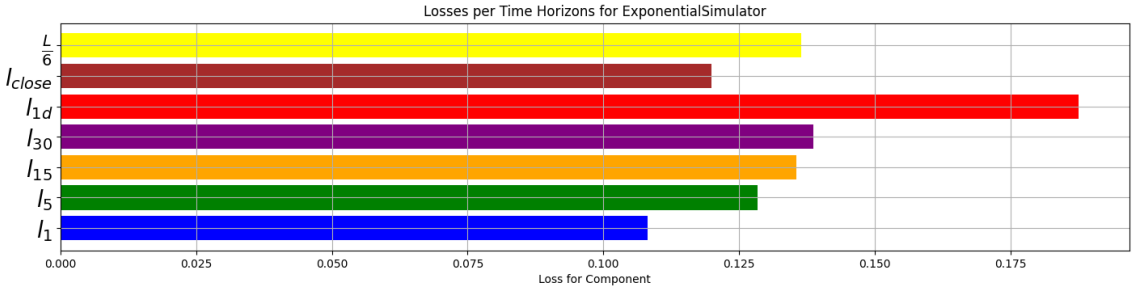

Figure 3.12: Exponential Agent Losses Across Time Horizons

3.7 FCN Simulator

FCN Stands for Fundamental-Chart-Noise. Which are the names of the parameters that each agent can possess in this type of simulation. This type of simulator is largely inspired by [7] and [8]. It follows the assumption that there are three main types of traders operating in this financial market. The fundamental agent (F) would be the agent that trades according to its belief of the fundamental price of the asset, These type of traders could be Equity Analysts and funds that trade on long term predictions, like Warren Buffet's Berkshire Hathaway. The Chart (C) or trend traders act on the trends they see in the realized market price and predict that performing assets will continue to perform and under-performing assets will continue to underperform. These could be medium-to-highly sophisticated price speculators and momentum traders. Finally, Noise traders are random traders that do not follow a well-defined rationale, and these could be considered retail traders. Another way to analyze Noise traders could be by incorporating their strategy in the F and the C as a confounding element to the prediction of the more sophisticated traders as a proxy for the possible difference in interpretation that traders could have of the same data.

We assume that each Agent has a unique combination of the three trading strategies mentioned above for our ABM model. Each parameter is randomly selected from an exponential mean; however, the mean of the three exponential distributions is controlled at the simulation level to test the effect on the exchange price movement of having more or less of a particular trading strategy. This random selection

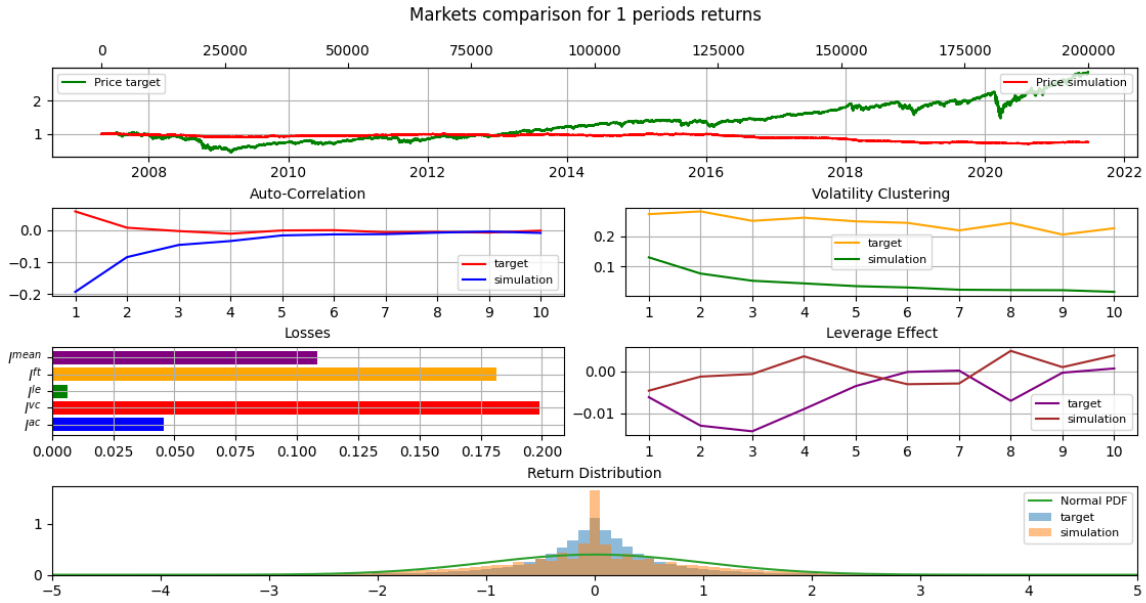

Figure 3.13: Exponential Agent Simulator Stylized facts for 1-minute returns

introduces heterogeneity in the agents. For each agent we sample the following:

$$\begin{aligned} w_F &\sim Exp(\lambda_F) \\ w_C &\sim Exp(\lambda_C) \\ w_N &\sim Exp(\lambda_N) \\ \tau_w &\sim U(\tau_{min}, \tau_{max}) \\ m &\sim U(0, m_{max}) \end{aligned} \tag{3.17}$$

Where t_w is the time window that the agent uses to compute momentum and for which it makes the price prediction, and m is the margin that the agent wants from its target price.

Each time the agent is sampled to make a trade, the agent decides its target price using the following formulas:

$$\begin{aligned} F &= \frac{1}{T} log(\frac{p_t^*}{p_t}) \\ C &= \frac{1}{T} log(\frac{p_t}{p_{t-\tau_w}}) \\ N &\sim \mathcal{N}(\mu, \sigma^2) \end{aligned} \tag{3.18}$$

$$r = \frac{w_F F + w_C C + w_N N}{w_F + w_C + w_N}$$

$$\bar{p}_{t+\tau_w} = p_t e^{r \cdot t_w}$$

Where p_t is the current mid price and p_t^* is the current price of the fundamental value of the asset. For our simulation we define the fundamental value of the asset as the performance of a GBM. We use the predicted return r to calculate the probability that the order is a buy or a sell:

$$\begin{aligned} P(buy) &= \frac{1}{1 + e^{\frac{r}{0.002}}} \\ P(sell) &= \frac{e^{\frac{r}{0.002}}}{1 + e^{\frac{r}{0.002}}} \end{aligned} \tag{3.19}$$

Finally, the price of the order is determined:

Algorithm 2 Order Price Decision

if buy and $\bar{p}_{t+\tau_w} > p_t$ **then**
  $p_o = \bar{p}_{t+\tau_w} \cdot (1 - m)$
else if buy and $\bar{p}_{t+\tau_w} < p_t$ **then**
  $p_o = p_t^b \cdot (1 - m)$
else if $sell$ and $\bar{p}_{t+\tau_w} > p_t$ **then**
  $p_o = p_t^a \cdot (1 + m)$
else$sell$ and $\bar{p}_{t+\tau_w} < p_t$
  $p_o = \bar{p}_{t+\tau_w} \cdot (1 + m)$
end if

Where p_t^a and p_t^b are the current best-ask and best-bid. The volume of the order is determined by a draw from a uniform distribution that is bucketed into integers.

We illustrate the results of the simulator by running the algorithm for parameters $\lambda_F = 0.2$, $\lambda_C = 0.1$ and $\lambda_N = 0.7$. In Figure 3.14 and 3.15 we can see the resulting market performance after running our FCN Simulator. We notice that the mid-price tends to oscillate around the fundamental price with variable amplitude. This effect is most noticeable in Figure 3.15 between time 3400 and 5600. In this graph, the purple line at time 3400 represents a peak in mid-price, the two yellow lines at times 4100 and 5600 represent a convergence of mid-price and fundamental price, and the brown line at 4800 represents the trough of the oscillation.

For this simulation, chart traders have a lookback window randomly chosen between 500 and 1000, so they had enough time to build up demand for the asset before the peak at 3400, as the mid-price had been rising since 2000. We see that the mid-price around the purple vertical line reaches a peak as the upward trend slows down. The slowdown is likely because the fundamental traders' selling pressure has surpassed the buying pressure of the chart traders. The price gets to the first yellow line where the mid-price and the fundamental price cross. At that time, the average trade direction of the fundamental traders switches from selling to buying; however, the time series has built a negative momentum, so the chart traders are still exerting an intense selling pressure. This selling pressure is the main reason the mid-price continues going down until the brown vertical line, where the buying

power of fundamental traders surpasses the selling power of chart traders.

Finally, we see convergence at the final yellow line, which does not lead to a further continuation in the mid-price. The lack of a further upswing in price can be because chart traders did not build enough buying pressure to overpower the fundamental and noise traders. We need not forget that noise traders can have a random powerful effect that can overturn our expectation of behaviour of a pure fundamental and chart market.

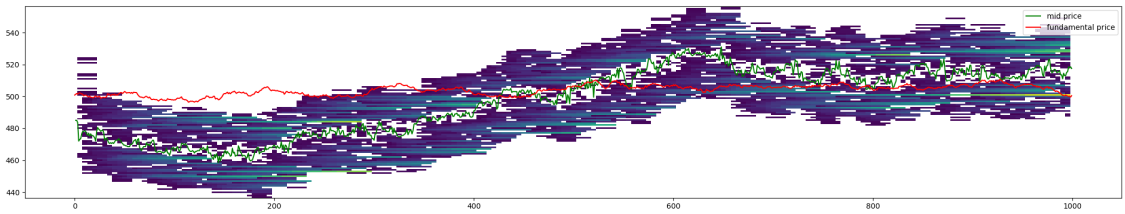

Figure 3.14: FCN Agent Simulator Zoomed-In

This boom and bust cycle that is primarily evident between time 3400 and 5600 can be observed, to a lesser extent, thought the whole simulation. The pattern is less noticeable when the amplitude of the oscillation is smaller because, in that case, the noise traders have a proportionally more significant effect on the price movement. However, we can notice this effect during the first 2000 steps of Figure 3.15 and on Figure 3.14.

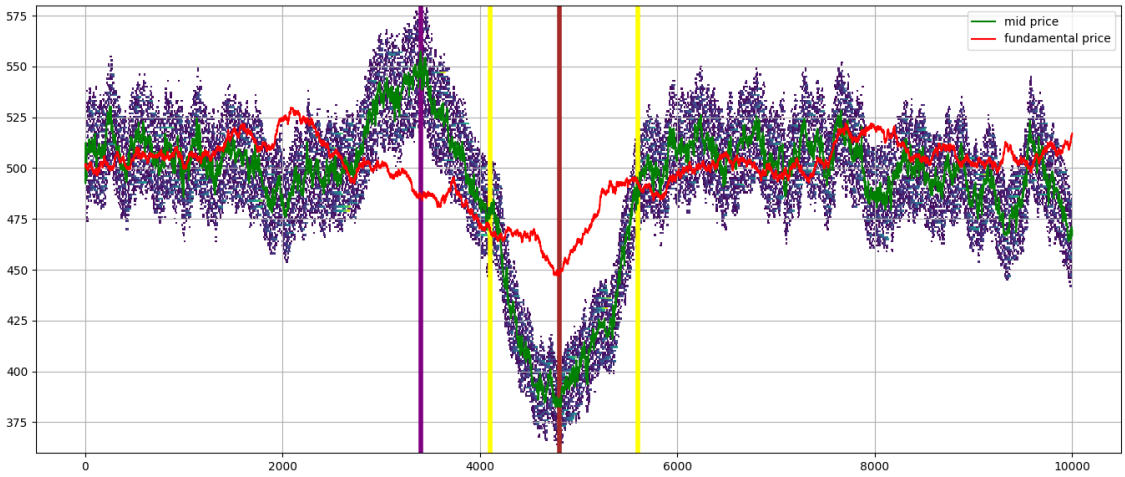

Figure 3.15: FCN Agent Simulator Zoomed-Out

3.7.1 Volatility Modification

We notice that the stylized fact that contributes the most to the loss so far has been the volatility clustering fact. Therefore we address this problem by augmenting the memory of our FCNSimulator.

We make the simulator keep track of the means and standard deviation of the returns for the mid-price using the following recursive formula at each minute:

$$\begin{aligned}\mu_t &= \mu_{t-1} + (\mu_t - \mu_{t-1}) \\ &= \mu_{t-1} + \frac{r_t - \mu_{t-1}}{t}\end{aligned} \tag{3.20}$$

$$\sigma_t^2 = \sigma_{t-1}^2 + \frac{(r_t - \mu_t)^2 - \sigma_{t-1}^2}{t} + \frac{t-1}{t}(\mu_t - \mu_{t-1})^2$$

The formula above allow us to keep track of mean and variance in an efficient way. We then compute an exponentially weighted moving variance using the following:

$$\gamma_t^2 = \alpha(r_t - \mu_t)^2 + (1-\alpha)\gamma_{t-1}^2 \tag{3.21}$$

Where α is between 0 and 1 and represents how much new information we incorporate in our variance estimator We then compute a proportion using:

$$m_t = e^{g(\frac{\gamma_t}{\sigma_t} - 1)} \tag{3.22}$$

Where g is a gearing parameter that enhances the differences in γ and σ.

Finally, we use m_t to scale both the volume traded and the noise component N that each agent receives to decide their future price prediction. This leads agent to incorporate in their prediction if the period in which they are trading has higher or lower volatility than the historical average.

In Figure 3.16, on the Volatility Clustering graph, we can see that this method can approximate the volatility clustering better while not losing goodness of fit on the other stylized facts.

3.8 Optimizer

With the FCN simulator, we have finally defined a simulator with some parameters that we can change for our purposes. We now need to find the right parameters to minimize our loss function and fit the market stylized facts as closely as possible.

Bayesian Optimization is a method used when we have the following optimization problem:

$$x^* = \underset{x}{\operatorname{argmin}} f(x) \tag{3.23}$$

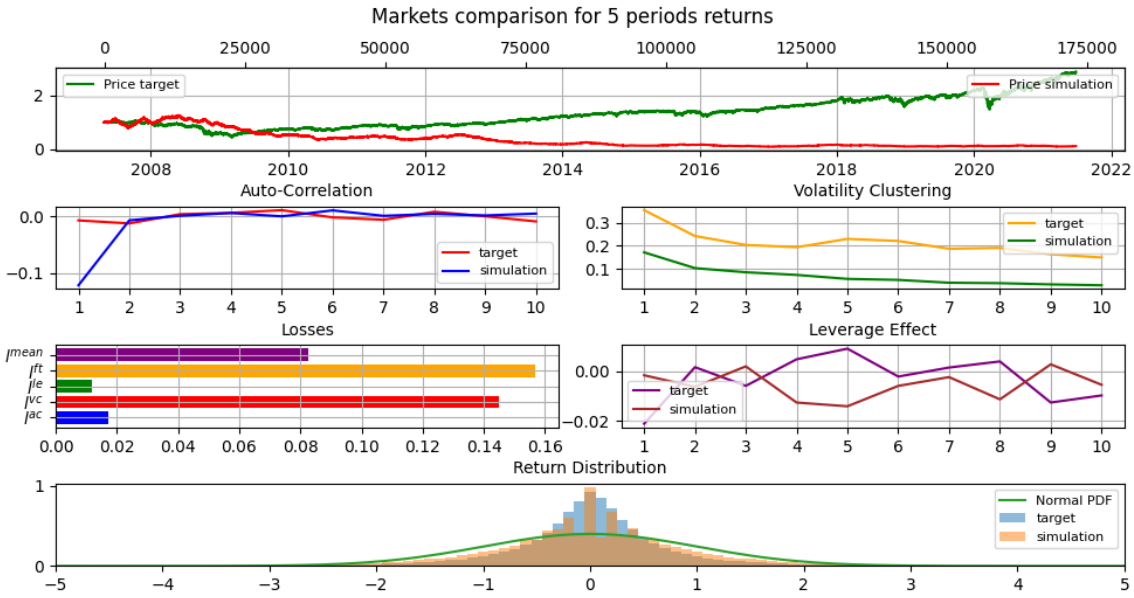

Figure 3.16: Modified FCN Simulator Stylized Facts for 5-minute returns

Where $f(x)$ is slow and expensive to evaluate, it is a black-box function for which we cannot compute gradients, and the evaluations might be noisy. In our case, we fulfil all these criteria since simulations take a few seconds to evaluate, which can quickly become hours when performing grid search across multiple parameter dimensions. This intractability is especially present given that we need to compute many simulations for each set of parameters to get a good estimation for the loss. We also cannot evaluate gradients for this exercise. Therefore, Bayesian Optimization is our method of choice.

Bayesian Optimization creates a Gaussian Process over the parameter space with respect to the loss considering all completed function evaluations so far. It then simplifies the Gaussian Process into an acquisition function that can be queried and optimized quickly to choose the next set of parameters to evaluate which are most likely to lead to an improvement in the loss function [11]. We summarize the optimization process in Algorithm 3.

We start our optimization by gaining a better understanding of the space by randomly evaluating n sets of parameters and using them to create a Gaussian Process with a Matern kernel, which has been shown to work better than a simple Gaussian kernel. We then start the Bayesian Optimization loop and iteratively choose a point to evaluate using an acquisition function. The purpose of an acquisition is to use the Gaussian Process mean and covariance to select the most likely point in the parameter space that will improve on our current best set of parameters. The decision of the next point to evaluate will consider an exploration-exploitation trade-off. We have to optimize an acquisition function for each iteration of the loop, but this is feasible as the acquisition function is quick to evaluate. In our case, we are using the

Algorithm 3 Bayesian Optimization Loop

$\mathcal{D}_0 = \{(X_0, y_0)\}$
for t = 1, 2, ..., n **do**
  Update GP using $\mathcal{D}_{t-1}$
  Sample x_t Randomly
  Evaluate $y_t = f(x_t)$
  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$
end for
for t = n + 1, n + 2, ..., n + m **do**
  Update GP using $\mathcal{D}_{t-1}$
  $x_t = \text{argmax}_x \alpha(x)$
  Evaluate $y_t = f(x_t)$
  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$
end for
Return $x^* = \underset{x}{\text{argmin}} f(x)$

Expected Improvement function:

$$\alpha(x) = -\mathbb{E}[f(x) - f(x^+)] \tag{3.24}$$

Where x^+ is the current best set of parameters.

In Figure 3.17 we can see an example of the log of the loss function versus the iteration number of the Bayesian Optimization Algorithm 3.

In the case of the FCN, we only care about the proportion of the parameter compared to each other; therefore, it is inefficient to sample all the parameters independently as the loss function would be a fractal with respect to the parameters. For this reason, we restrict samples from the surface of a unit sphere in the space using the following transformation:

$$\begin{aligned} \theta &\sim U(0, 2\pi) \\ \phi &\sim U(0, \pi) \\ \lambda_F &= e^{cos\theta sin\phi} \\ \lambda_C &= e^{sin\theta sin\phi} \\ \lambda_N &= e^{cos\phi} \end{aligned} \tag{3.25}$$

Two alternatives to the Bayesian optimization algorithms are the grid search algorithm or a pure form of the random search algorithm. The grid search algorithm divides the parameter space into a grid and evaluates the objective function at each point on the grid to choose the best one. Grid search is a thorough algorithm, but it is intractable for most interesting optimizations as the time complexity of this algorithm scales exponentially with the number of parameters to optimize. Gilli et

al. presented a version of the grid search algorithm, which evaluated the objective function at the grid points to create an approximation of the objective function that is quicker to query and optimize [10]. This form of the algorithm is slightly better than the grid search since it enables to find a better point that is not on the grid but still presents the same challenges in terms of time complexity.

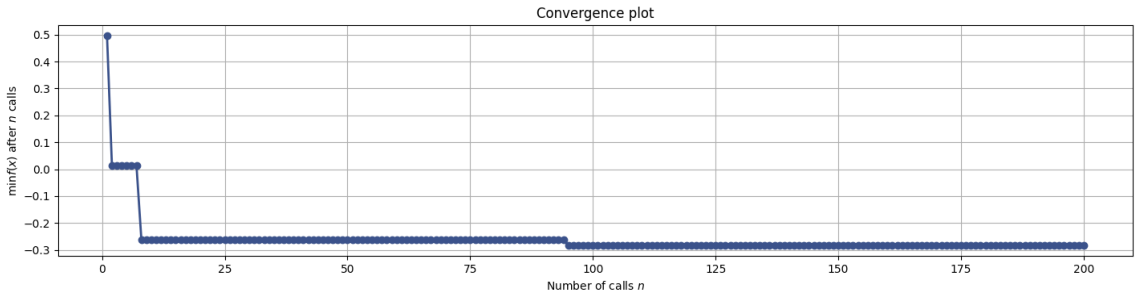

Figure 3.17: Loss convergence for Bayesian Optimization on a basic FCNSimulator

The random search algorithm chooses random points on the parameter space and evaluates them. It is better than grid search as it can capture interesting interaction between parameters with less objective function evaluations. However, the combination of random search and Bayesian optimization is better. The Bayesian optimization can use the information discovered by the random search to guide the choice of the next point to evaluate by considering a custom exploration and exploitation trade-off.

3.9 Gamma Traders

In this section, we explain more deeply the proposed gamma agents. As mentioned, they are agents that trade according to their obligation to provide a particular risk exposure to their clients. The best example of gamma traders are passive leveraged Exchange Traded Funds (ETF) providers. A leveraged ETF is a financial product that provides accelerated exposure to the price movement of specific underlying assets. For example, if the S&P 500 closes the day +2%, a 5X leveraged ETF on the same asset will close the day +10%, and if the S&P 500 closes the day at -3% the leveraged ETF will close at -15%. These returns happen because the leveraged ETF provides 5 times the exposure to the market per unit of principal.

The ETF provider is obliged to hold as much exposure to the underlying as the current price of the ETF multiplied by the leverage of the ETF; therefore, if the price of the ETF increases, the increase in value of the existing shares he holds will not be enough to provide the exposure and he will need to buy more shares. Conversely, if the ETF price decreases, the issuer of the ETF will need to sell shares. The buying or selling of shares usually happens toward the close of the trading day as it is the usual fixing time for the price of the leveraged ETFs.

The quantity of shares that an leverage ETF needs to trade is given by the following formula:

$$\begin{aligned} \textit{Shares Underlying} &= \frac{\textit{Desired Exposure}}{\textit{Price Shares}} \\ S^U &= \frac{P^L \cdot L \cdot N}{P^U} \end{aligned} \tag{3.26}$$

$$\begin{aligned} \textit{Rebalancing} = S_t^U - S_{t-1}^U &= (\frac{P_t^L}{P_t^U} - \frac{P_{t-1}^L}{P_{t-1}^U}) \cdot L \cdot N \\ &= (\frac{(L-1)r_t^U}{1+r_t^U}) \frac{P_{t-1}^L}{P_{t-1}^U} \cdot L \cdot N \end{aligned}$$

Where L is the leverage, N is the number of contracts that need replication, P^L is the price of the ETF and P^U is the price of the underlying asset. For our ABM Gamma Trader we randomly endow the agent with a L and a N.

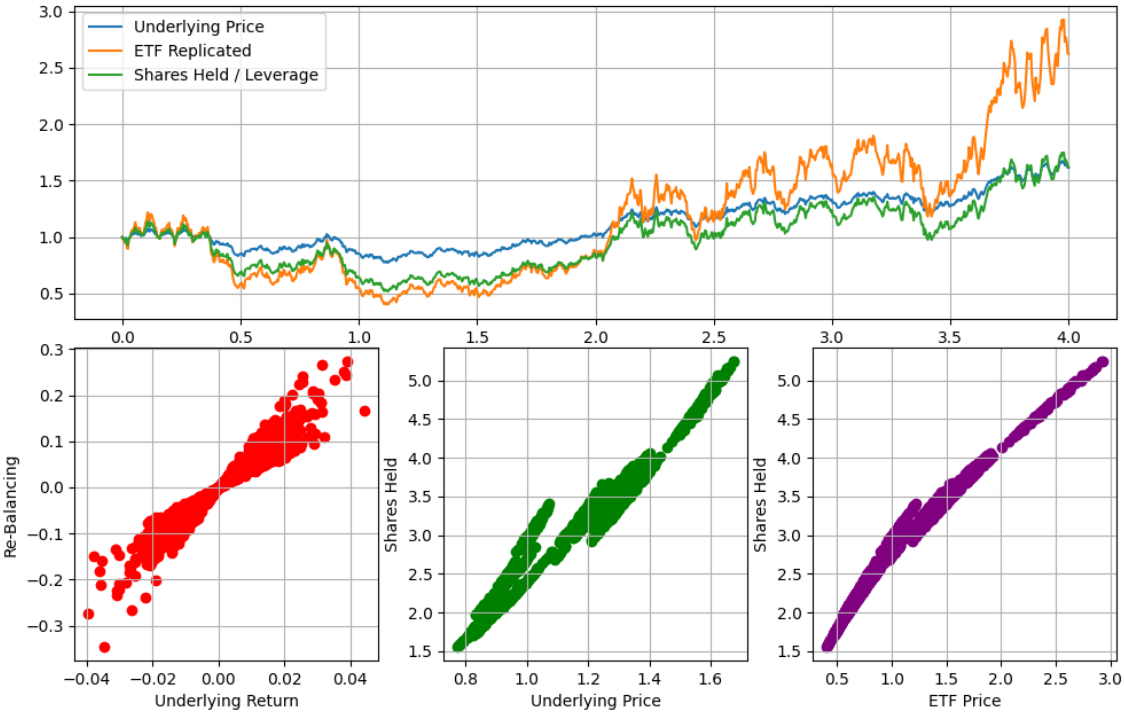

Figure 3.18: Statistics for simulated 3X leverage ETF

In Figure 3.18 we can see some statistics for a simulated performance of a leveraged ETF. On the top graph, we compare the performance of the underlying and the ETF and the number of shares held at any point in time divided by the leverage. We can see in the bottom left the underlying re-balancing needed versus the underlying return, and we can see in practice the clear positive correlation between the two.

3.10 FCN Gamma Simulator

For this simulator, we add Gamma Traders to the population of agents Gamma Traders whose behaviour we explained in the previous section. Since actual Gamma Traders trade near the close of trading, we sample them with a much greater probability towards the close of trading. So far, the simulation does not have an open and closing time; therefore, we define the day to be a cycle of 390 minutes, which is the trading time of the New York Stock Exchange. Gamma traders will be sampled based on a cyclical period of 390 minutes and compare the current price with the price at the start of the period to make their deterministic trading decision. Gamma agents are heterogeneous because we randomly draw the leverage they have to replicate from a set; therefore, they will trade more or less depending on the randomly drawn leverage.

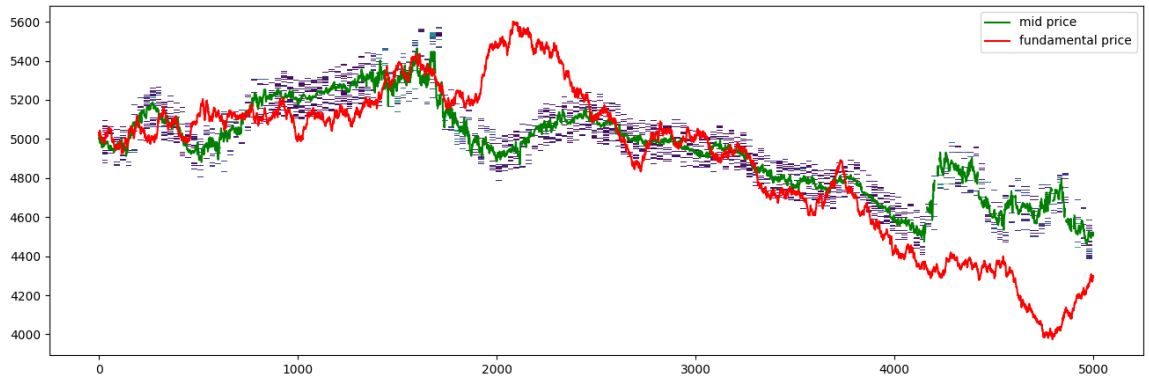

Figure 3.19: Limit Order Book for FCNSimulatorGamma

In Figure 3.19, we can see the performance of the mid-price for this type of simulation. For this simulation, we added 5% of the population as Gamma Traders. We observe a few times during the simulation where the effect of the Gamma Traders was so strong that it fulfilled all the orders on one side of the limit order book and caused a lack of valid mid-prices for a certain period. However, we can quickly fix this problem by calibrating the leverage of the Gamma Traders better.

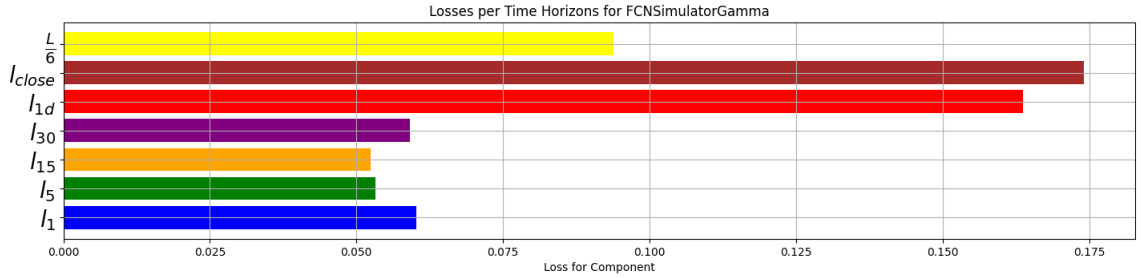

Figure 3.20: Losses for FCNSimulatorGamma

In Figure 3.20 we can see the components of the loss for this type of simulation. We can see that against our expectations, this type of simulation does not perform well with respect to the close-correlation stylized facts, which is the central stylized fact we decided to add Gamma Traders to address. We can see in more detail in Figure 3.22 that the close-correlation does not match the SPX target. However, surprisingly, having Gamma Traders improves the performance for other stylized facts like volatility clustering and fat-tails.

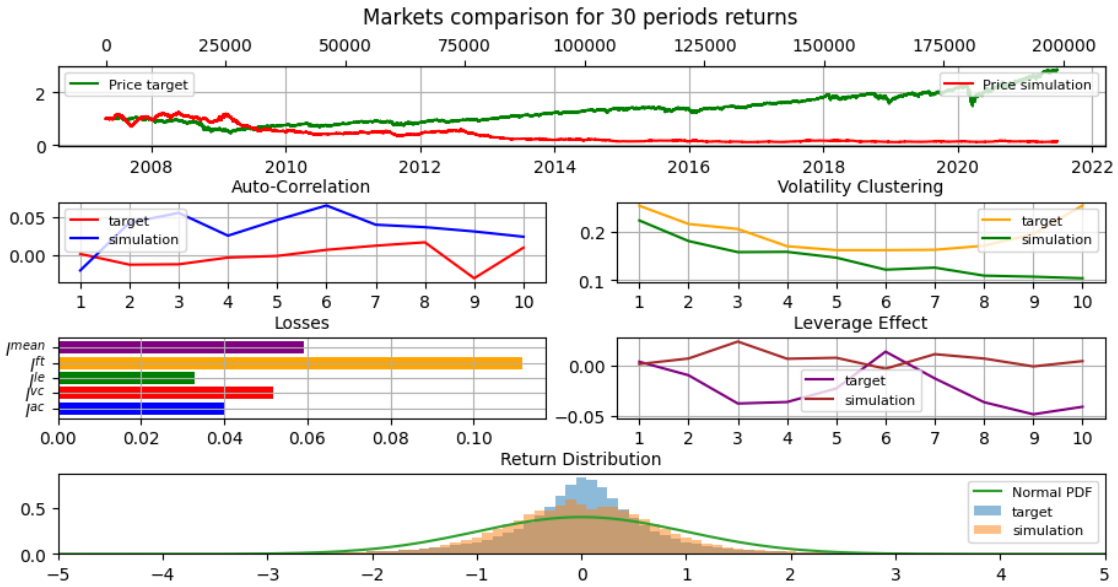

Figure 3.21: Modified FCNSimulatorGamma Stylized Facts for 30-minute returns

We can see in Figure 3.21 on the Volatility Clustering graph that the simulation matches the target closely in a way that we have not observed before for all the other simulations types so far. We can say the same for the fat-tail distribution graph on the bottom of the image. However, we notice a positive auto-correlation that seems persistent and is not in line with the zero correlation target.

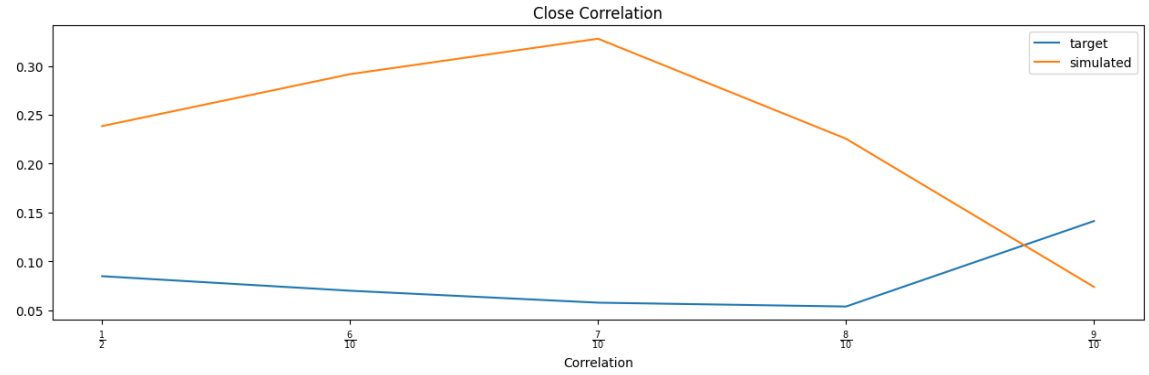

Figure 3.22: Close-correlation for FCNSimulatorGamma vs SPX

3.11 Summary

We have gradually built the complexity of our ABM models and have collected the losses and all of its components in Table 3.1.

$Model$	l_1	l_5	l_{15}	l_{30}	l_{1d}	l_{close}	$\frac{L}{6}$
GBM	0.146	0.132	0.127	0.128	0.191	0.131	0.142
$RandomSimulator$	0.099	0.086	0.073	0.061	0.123	0.235	0.113
$ExponentialSimulator$	0.108	0.128	0.136	0.139	0.187	0.120	0.136
$FCNSimulator$	0.117	0.131	0.137	0.137	0.201	0.045	0.128
$FCNSimulatorMod$	0.080	0.080	0.093	0.103	0.187	0.061	0.101
$FCNSimulatorGamma$	0.060	0.053	0.052	0.059	0.164	0.174	0.094

Table 3.1: Market Simulators Loss Function Comparison

We can see that all the models so far beat the benchmark in terms of total loss $\frac{L}{6}$. The *FCNSimulatorGamma* so far is the best overall, but the FCN Simulator is the best for l_{close}, which is of particular interest to our study. We notice the general trend that longer time horizons are, in general, more challenging to replicate. The best model so far is beats all the others in terms of performance for the first for the intra-day time horizons (l_1, l_5, l_{15}, l_{30}).

Chapter 4

Software Implementation

4.1 Code Structure

The code directory for the ABM simulation is can be found at https://github.com/lupoAI/abm-python. The code is subdivided into several sub-directories:

- Market
- Analysis
- Data
- Tests
- Mains
- Results

4.1.1 Market

This section is the core of the implementation, as here is where we define the basic objects that enable to simulate the market. We define *Agent*, *Exchange*, *Simulator* classes as well as other classes like *Orders* and *OrderReceipts* that help in defining the interactions and interchange of information between the *Agent* and the *Exchange*. The interactions between the elements is represented on Figure 4.1.

The *Simulator* class is the highest level class that manages the interactions between the *Agent* and the *Exchange*, so it decides which *Agent* trades at each point in time and what information is transferred between the two. When when an *Agent* is chosen to trade it decides which *Order* to submit to the *Exchange* which handles it and returns an *OrderReceipt* back to the *Agent* which takes note of the information to keep track of its own portfolio. When a *LimitOrder* is matched the *Exchange* informs both sides of the trade so that they can update their portfolio. The *Agent* is subscribed to an *Exchange* and the structure of the code is such that an *Agent* could subscribe to multiple exchanges.

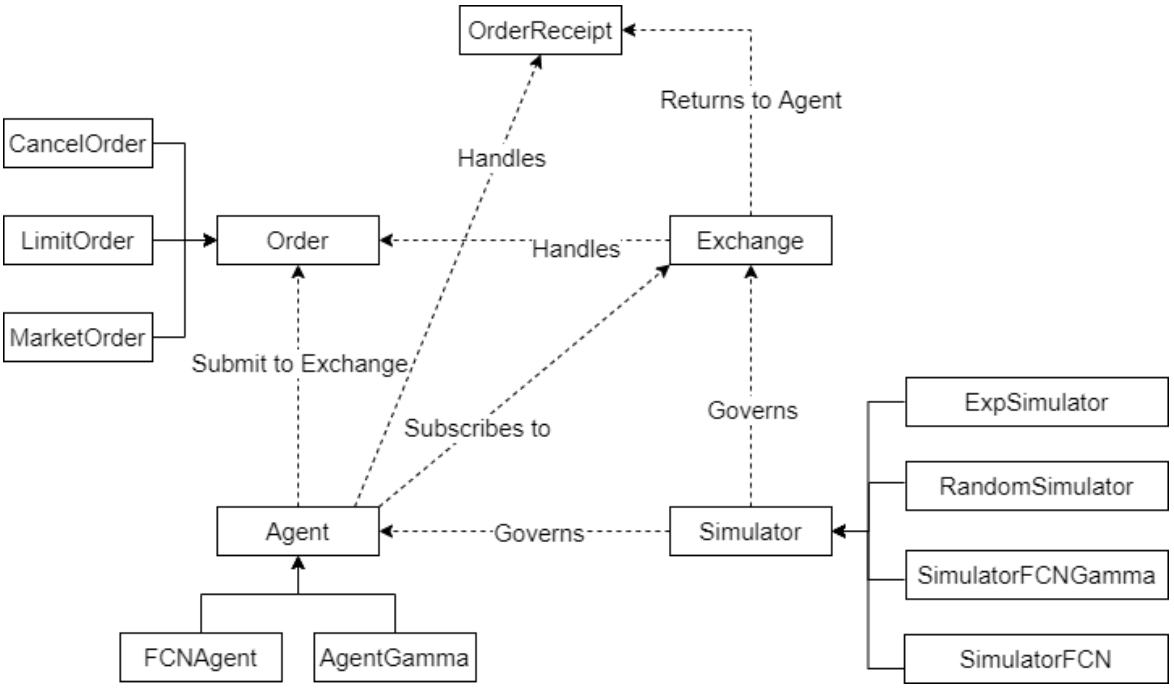

Figure 4.1: Market Sub-directory Class Diagram

4.1.2 Analysis

In this sub-directory we create useful tools to analyze the outputs of the simulator as well as the loss function and the Bayesian optimizer.

We can see in Figure 4.2 how the object within this sub-directories interact with each-other to help us in our analysis. The *VisualizeSimulation* class is able to visu-

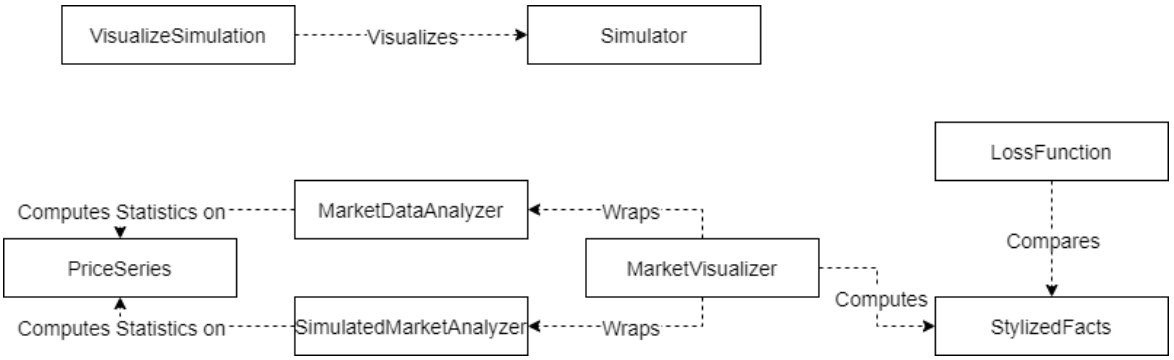

Figure 4.2: Analyzer Sub-directory Class Diagram

alize the evolution of the limit order book of the simulation with time as seen in many images in this report. The *MarketDataAnalyzer* and the *SimulatedMarketAnalyzer* class take as input the *PriceSeries* of real and simulated data respectively. These classes are are able to compute the stylized facts for all time horizons. They are accessed through the *MarketVisualizer* class which computed the *StylizedFacts* of the *PriceSeries*. The *LossFunction* class is able to compare two *StylizedFacts* to compute

the loss function.

In the *optimizer.py* script we implement Algorithm 3. Given that our loss function is stochastic for each set of parameters, we want to run the simulation multiple times. To do this efficiently, we parallelize the simulations for each set of parameters and aggregate the loss function once all of the threads finished computing.

4.1.3 Data

In this sub-directory, we store the raw real minute-frequency price data and a processed version of the stylized facts for that data. We store the processed data because we need to access this information many times during the Bayesian Optimization of our simulator.

4.1.4 Tests

Here we test that the code performs as expected. The interaction between *Exchange* and *Agent* is the main focus of the tests as proper functioning of this relationship is fundamental to the project. We thoroughly test edge cases regarding the use of *LimitOrder*, *MarketOrder*, and *CancelOrder*.

We also test larger interactions within the simulator. We have devised tests for the *FCNSimulator* that show its proper functioning. We can see the results in Figure 4.3.

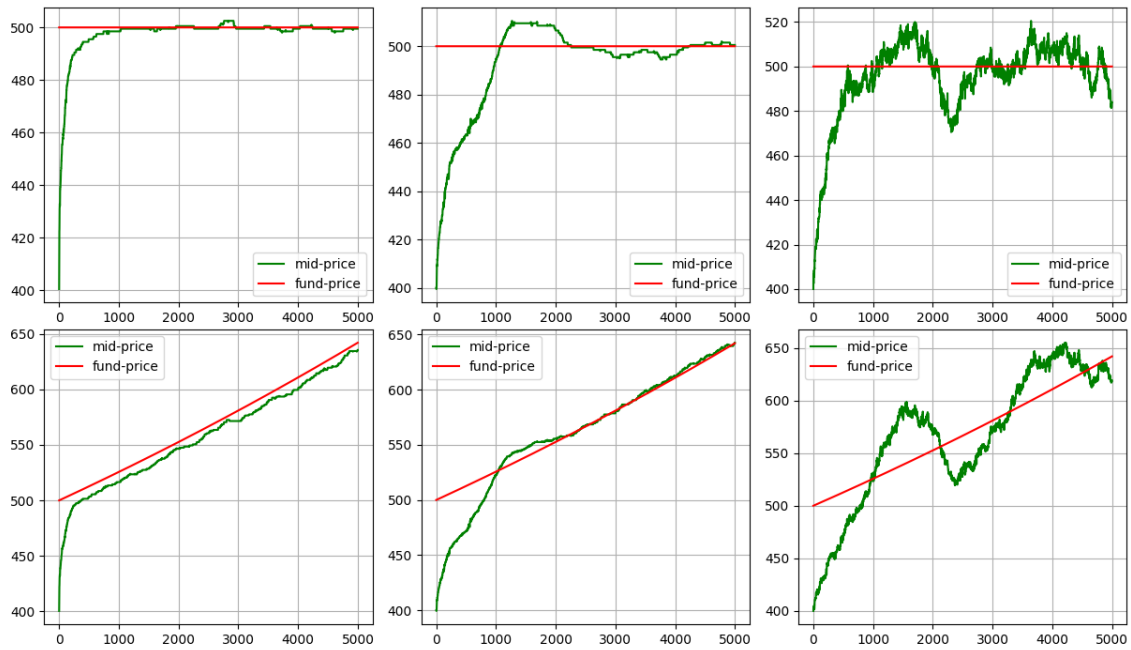

Figure 4.3: Tests for *FCNSimulator*

The sub-image on the top-left shows the evolution of the mid-price when the initial

price is 400, and the fundamental price is 500 if there are only fundamental traders; as expected, the price quickly approaches its fundamental value and stays there. The same can be said for the sub-image in the bottom-left, where the only difference is that the fundamental price is increasing. When we have increasing prices, the mid-price is ever-chasing the fundamental price, but it stays close to it. The images in the top-centre and bottom-centre show the effect of adding chart agents to the simulation. We observe that the mid-price approaches the fundamental value, but it overshoots before reverting to it. We expect overshooting as chart agents will add some inertia to the mid-price movement. Finally, the two images on the left show the effect of adding the noise agents, and we observe that the short term price movements become noisy while the long-term trends stay the same as the images in the centre.

4.1.5 Mains & Results

In the Mains section, we write the mains that we have run to produce all of the graphs and results available in this project, and we save these results in the Results directory.

4.2 Code Performance

The code not only needed to work properly but it was also important for it to be quick as the method of optimization required many long market simulations. It is important for the code not to have a quadratic time complexity with respect to the number of agents and the number of trades, as it would make optimization intractable. In Figure 4.4 we can see the relationship between some important variables and the time it takes to execute both *RandomSimulator* and *FCNSimulator*.

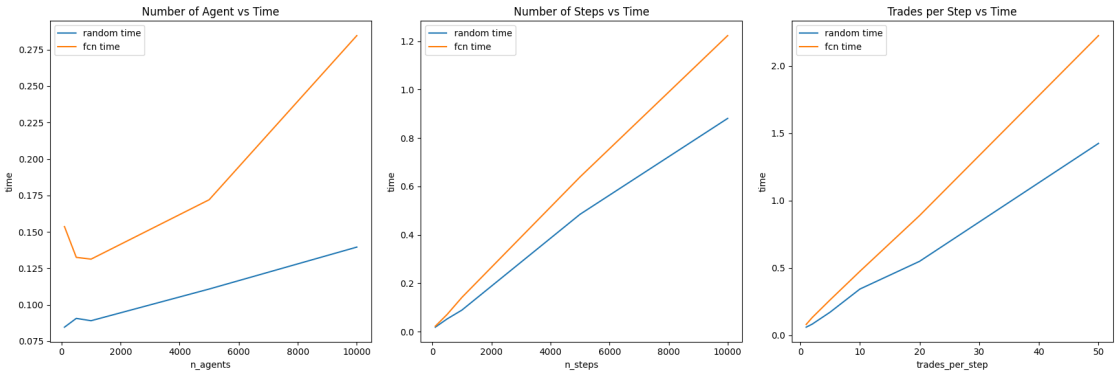

Figure 4.4: Time Complexity Statistics

We can empirically see that the simulation time scales linearly with the number of trades per minute, the number of minutes of trades, and the number of agents. The linearity of the time complexity with these variables means that the exchange checks the possible match of new orders with limit orders that are already present in the

exchange efficiently. If an order matches, the exchange can efficiently check to which agents the orders belong and communicate the success of the trade.

Chapter 5

Evaluation

5.1 Ultra Optimized ABM

5.1.1 Optimization

We used the whole of Chapter 3 to build up the potential of the ABM simulator by adding features and complexity. We explained the elements and the relative benefits of adding specific features to the problem of matching stylized facts. Now it is time to use the built-up potential to fit the SPX market as much as possible and understand the limitations that still exist.

By adding complexity, we introduced many variables for which we did not explore their effects on performance. We present these variables to the parameters space for Algorithm 3 to optimize for them. The variables that we optimize for are:

- λ_F, λ_C, λ_N: Fundamental, Chart, and Noise distribution mean. Chosen according to Equation 3.26.
- p_g: Percentage of the population of agents that are Gamma Traders. Chosen from $[0, 0.05]$ range.
- C_i: Cancel order interval which represents the maximum number of minutes an agents waits before cancelling a limit order that has not been fulfilled. Chosen from $[10, 100]$ range.
- m_{max}: Maximum margin that can be assigned to an agent according to Equation 3.17. Chosen from $[0.005, 0.05]$ range.
- τ_{min}: Minimum lookback time that can be drawn for an agents' chart strategy. Chosen from $[100, 1000]$ range.
- $\tau_{max} - \tau_{min}$: Lookback range for agents' chart strategy. Chosen from $[100, 1000]$ range.
- $atpm$: Average trades per minute which represents λ in Equation 3.15. Chosen from $[2, 6]$ range.

- α: Decay of the exponentially weighted moving variance. Chosen from $[0.005, 0.10]$ range.

Once we perform the Bayesian optimization for 200 steps we achieve the following convergence for the natural log of the loss function:

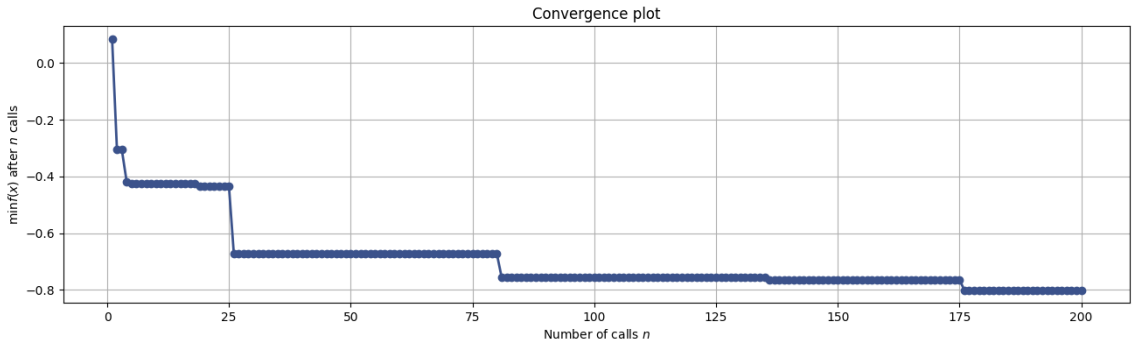

Figure 5.1: Convergence Plot for Ultra Optimized Simulator

The performance of the optimization algorithms for the latest version of the ABM model is much more promising than the performance for the original algorithm on Figure 3.17 that was only trying to optimize λ_F, λ_C, and λ_N.

The optimal parameters that the optimization algorithm found are: $\lambda_F = 0.37$, $\lambda_C = 0.08$, $\lambda_N = 0.55$, $p_g = 0.048$, $C_i = 17$, $m_{max} = 0.042$, $\tau_{min} = 571$, $\tau_{max} - \tau_{min} = 844$, $atps = 5$, and $\alpha = 0.032$.

5.1.2 Performance

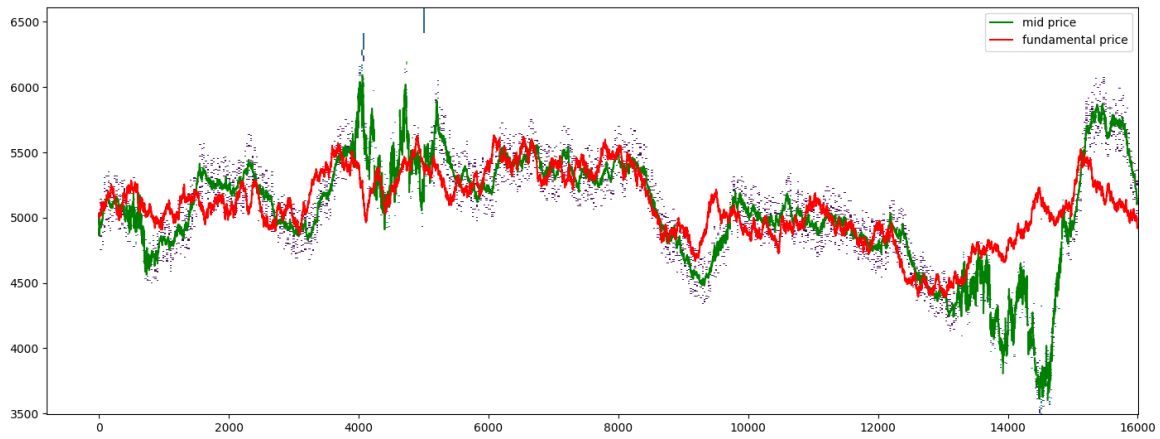

Figure 5.2: Price Evolution for Ultra Optimized Simulator

In this section, we visualize the resulting simulation with all the optimal parameters we found in the previous section. In Figure 5.2, we can see the mid-price movement

of the resulting simulation compared with the fundamental price of the simulation. As λ_F is relatively large, we can see that the mid-price closely follows the fundamental price; however, the mid-price sudden movements are more extreme, and it sometimes diverges by a significant amount before rushing back to the fundamental price. The sudden movements can be partially attributed to the relatively low C_i, as a short cancellation time makes it so that the LBO is not very deep and, therefore, changes in buying and selling pressures have a substantial effect, especially because the $atpm$ is at the high extreme of the spectrum that we tested.

In Figures 5.3, 5.5, 5.6, 5.7, 5.8, and 5.9 we can find detailed information about the performance of the final ABM simulator.

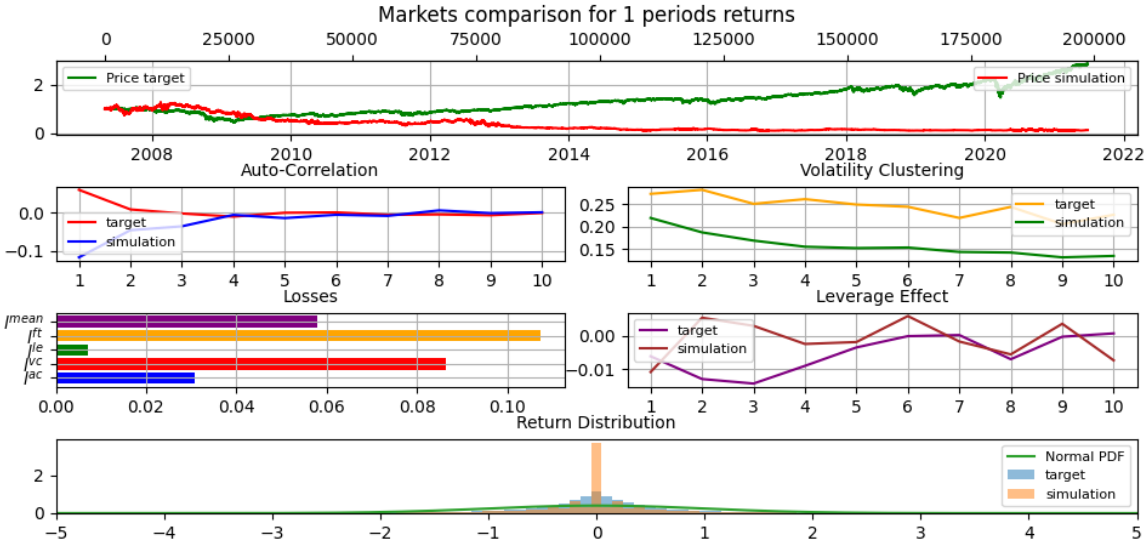

Figure 5.3: Ultra Optimized Simulator Stylized Facts for 1-minute Returns

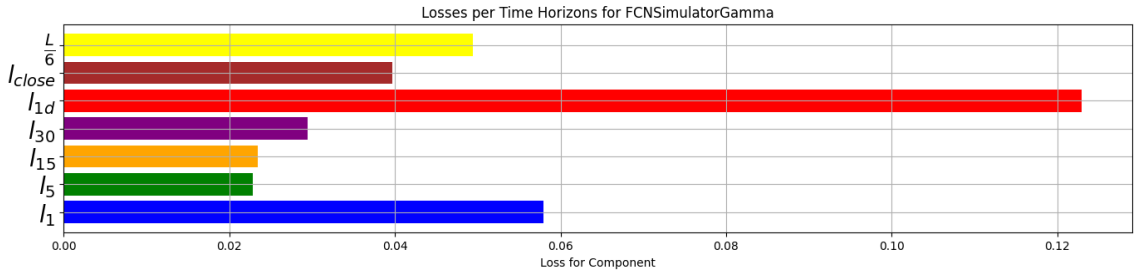

Figure 5.4: Aggregated Losses for Ultra Optimized Simulator

Results are surprisingly good, considering that we only optimize ten parameters. We can see on Figure 5.4 that the losses, especially for l_5, l_{15}, and l_{30}, are very low. In fact when we inspect the information for these loss components more deeply in

Figures 5.6, 5.7, and 5.8, we see on the Return Distribution graph that simulated returns distributions match the target almost exactly. The fat-tails has been the most challenging to replicate; however, it is also the most important in terms of risk management as in risk management tail risk if very important; therefore, it is excellent to have managed to fit it well for some time horizons. Although this model is the best so far, it cannot closely match the return distribution for the 1-day returns.

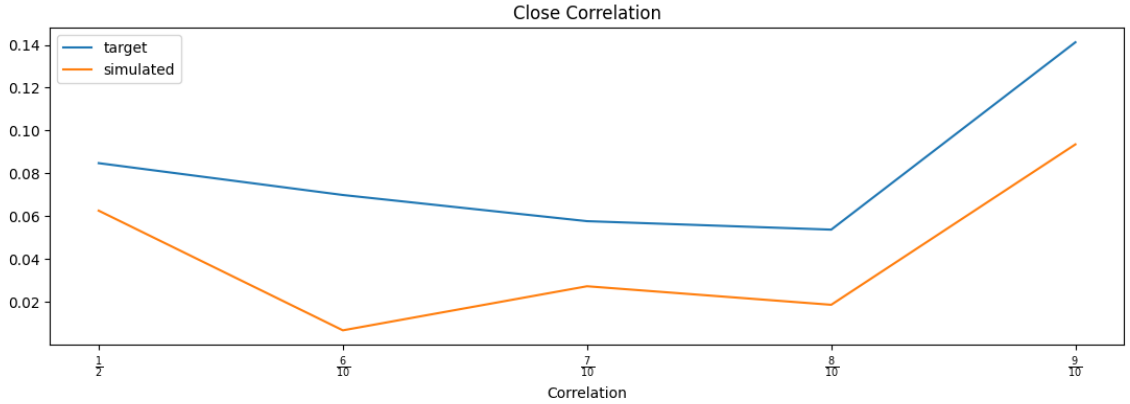

Figure 5.5: Ultra Optimized Simulator Close-Correlation Stylized Fact

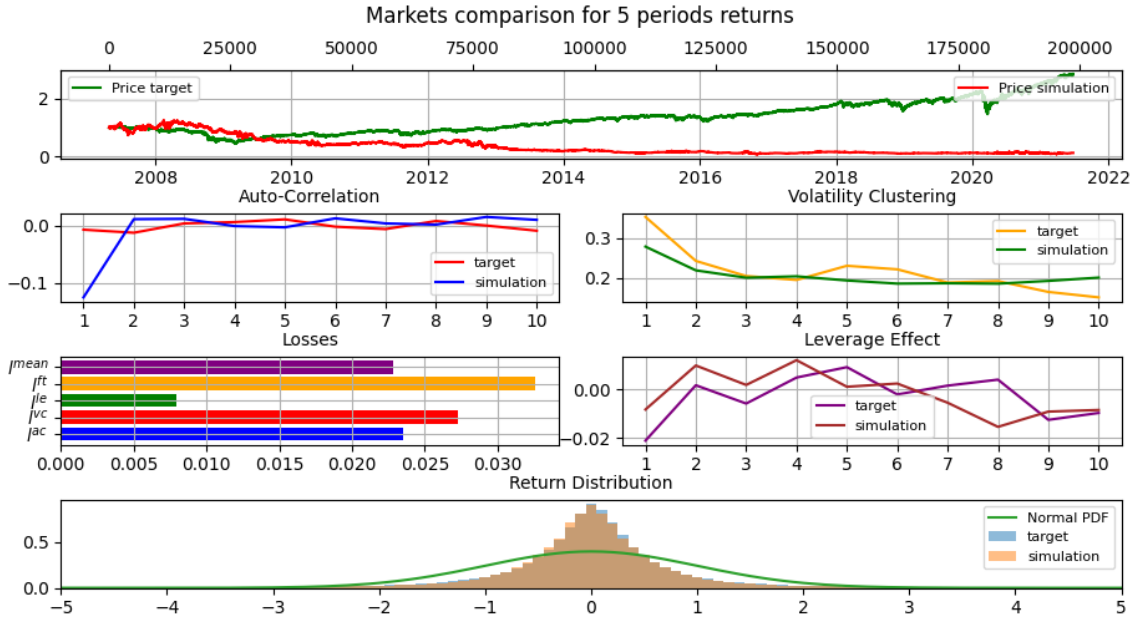

Figure 5.6: Ultra Optimized Simulator Stylized Facts for 5-minutes Returns

Similarly, what used to be the most challenging stylized fact, Volatility Clustering, has been matched to a satisfactory degree as we can see that for most all time-horizons the correlation for the first few time lags approaches the target; however,

we still have a problem in maintaining the correlation for longer time lags. We can observe this deficit in our performance from the fact that in Figure 5.9 in the Volatility Clustering graph, the correlation of the simulated market decays more quickly than the actual market.

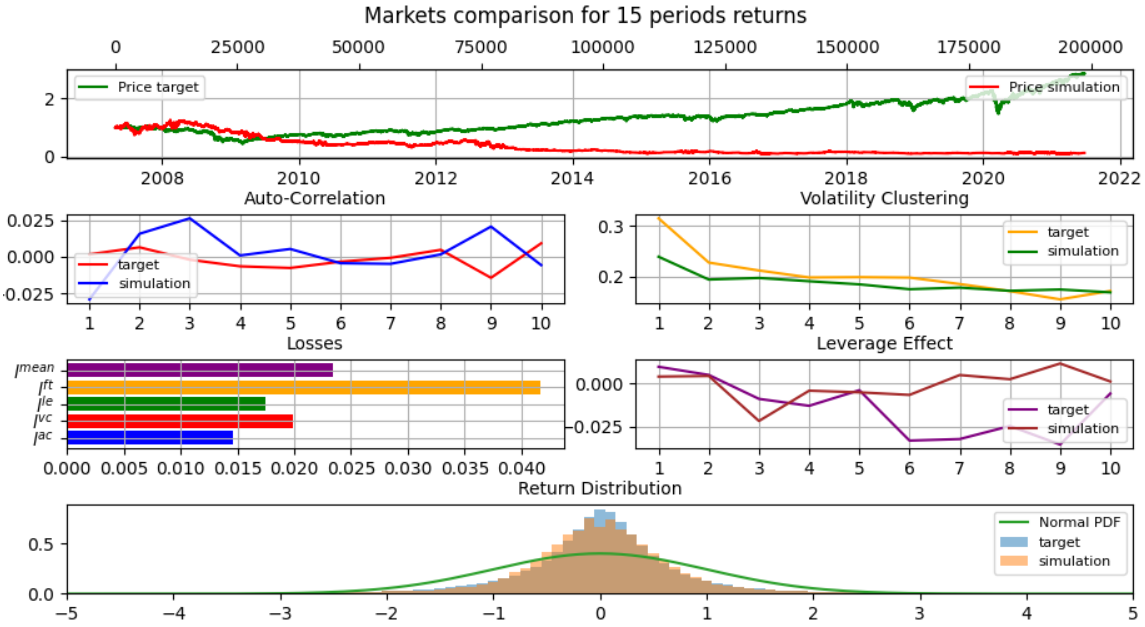

Figure 5.7: Ultra Optimized Simulator Stylized Facts for 15-minutes Returns

In addition, we can see from Figure 5.5 that we have a decent fit to the close-correlation stylized fact as well.

To compare our best simulator (*FCNSimulationUltra*) with the previous attempts we can see Table 5.1. We see that the loss L is almost half of the second-best attempt (*FCNSimulationGamma*) and one-third of our GBM benchmark. We achieve best in class results for each of the loss components, including the close-correlation loss.

Finally, we notice that this model, like all the others, has problems approximating the 1-day return stylized facts. In Figure 5.9 we see from the Losses graph that this loss is primarily due to the volatility clustering and the fat-tails stylized facts.

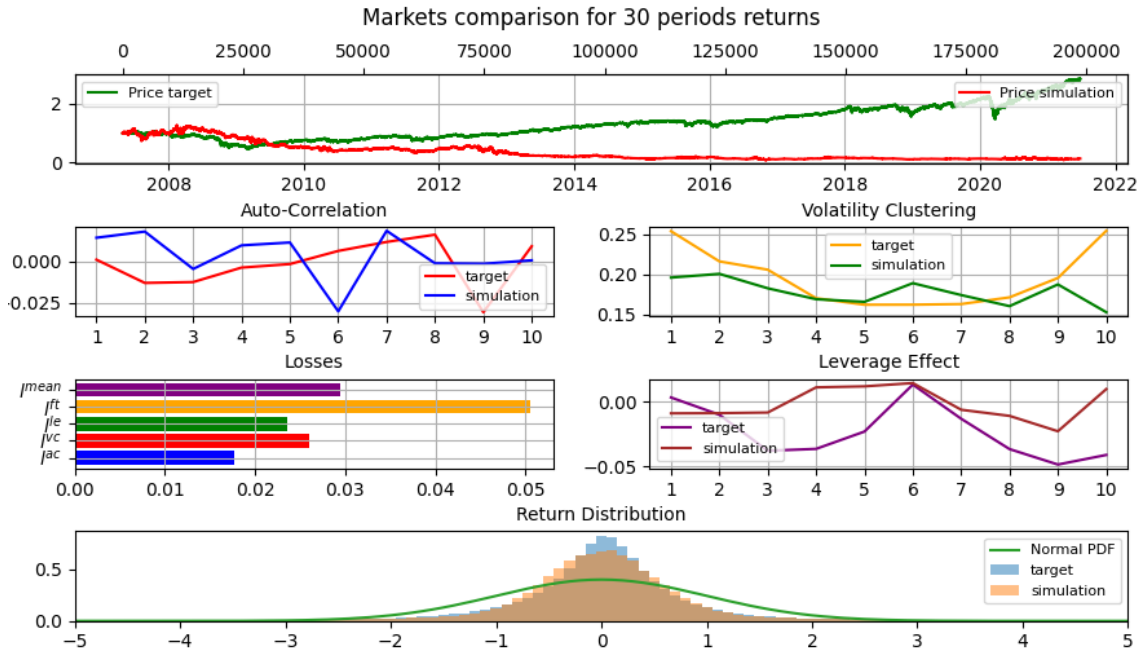

Figure 5.8: Ultra Optimized Simulator Stylized Facts for 30-minutes Returns

$Model$	l_1	l_5	l_{15}	l_{30}	l_{1d}	l_{close}	$\frac{L}{6}$
GBM	0.146	0.132	0.127	0.128	0.191	0.131	0.142
$RandomSimulator$	0.099	0.086	0.073	0.061	0.123	0.235	0.113
$ExponentialSimulator$	0.108	0.128	0.136	0.139	0.187	0.120	0.136
$FCNSimulator$	0.117	0.131	0.137	0.137	0.201	0.045	0.128
$FCNSimulatorMod$	0.080	0.080	0.093	0.103	0.187	0.061	0.101
$FCNSimulatorGamma$	0.060	0.053	0.052	0.059	0.164	0.174	0.094
$FCNSimulatorUltra$	0.058	0.023	0.023	0.029	0.123	0.040	0.049

Table 5.1: Market Simulators Loss Function Comparison

5.2 Legal, Social, Ethical and Professional Issues

A potential problem that could arise from using ABM models to calibrate risk models is regulation. Regulators across the world are starting to require the explainability of machine learning models. The rationale behind this requirement is that explainability leads to the accountability of a model's decision which makes it easier to control and fix it. If a deep neural network makes a decision, it is tough to localize the decision making to a subset of the model. Although ABMs are relatively more explainable since they replicate the behaviour of natural components of a system, we cannot represent most computational ABM in a closed-form solution, and it could be challenging to attribute performance to the individual decision rules of the agents. The regulations might make it a problem if one would try to calculate a capital requirement by simulating the future values of a portfolio using an ABM. Regulators

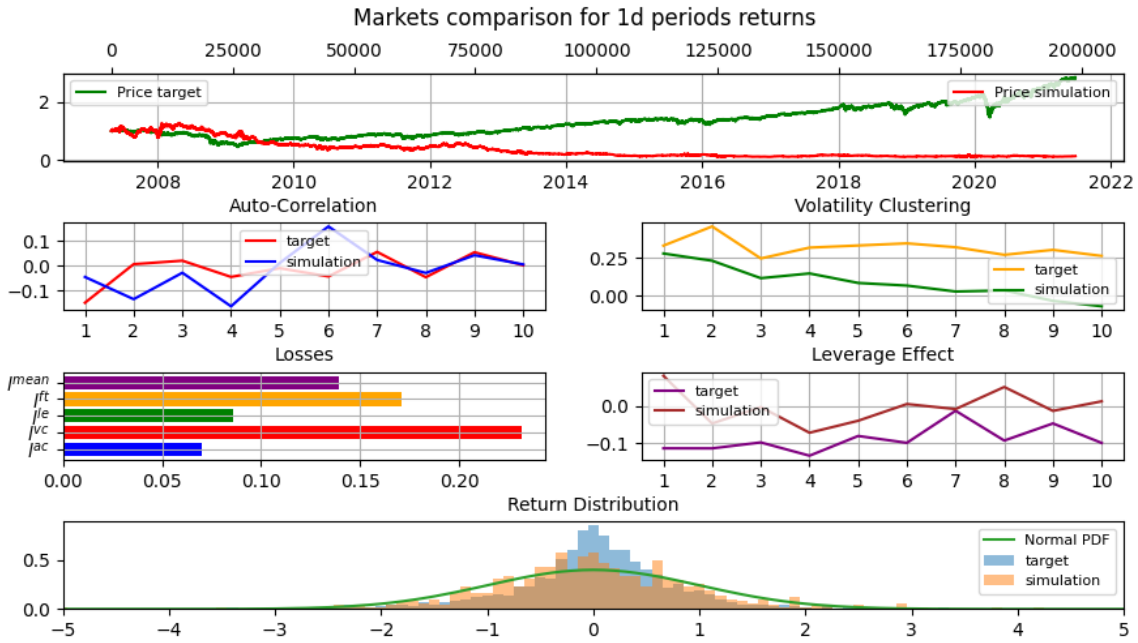

Figure 5.9: Ultra Optimized Simulator Stylized Facts for 1-day Returns

would expect more than just the fact that the price simulation matches historical stylized facts.

Chapter 6

Conclusion

6.1 Summary

We have established that financial prices have specific non-normal statistical properties such as the fat-tails the distribution of the price returns and the auto-correlation of the square of the returns (Volatility Clustering), which we called stylized facts. Simulating these stylized facts well is vital to have realistic evaluation of the potential future value of portfolios that we can use to calibrate trading strategies and calculate banks risk management requirements. In this project, we have presented an Agent-Based Model that can match the stylized facts of any market. As the target for our tests, we have chosen the minute-frequency prices for the S&P 500 (SPX). We have defined a standard method of price simulation (GBM) as a benchmark for our performance, and we have defined a metric to measure the goodness of fit of the simulated data to the target market (see Equation 3.13).

The ABM model that matched the target market more closely had four types of agents that trade in a Continuous Double Auction exchange environment: Fundamental, Trend-Following/Chart, Noise, and Gamma Traders. Fundamental agents chose their orders accordingly to the fundamental value of the asset. Chart agents trade according to the recent price trend of the market. Noise agents deal randomly. Finally, Gamma Traders trade towards the close of the daily trading cycle in the same direction as the intra-day price movement and a size proportional to the same price return.

We optimized the final ABM parameters using a combination of random search and Bayesian optimization (see Algorithm 3). The optimized model achieves much better performance than the benchmark and any other model tested during this project. The model is significantly better than other models at fitting the volatility clustering and the fat-tail stylized facts across time horizons. This level of performance signifies that the model has achieved its goal of matching the behaviour of real-life prices for multiple time horizons and could realistically be used to calibrate risk management requirements and trading strategies within specific markets.

6.2 Limitations

Although the final model matches the stylized facts well and has a much lower loss than the second-best model tried, we could still improve the model. When looking at the loss components in Figure 5.4, we see that most of the current loss is coming from the 1-day time horizon. The most significant deficit in the fit is that the volatility clustering of the simulation decays too quickly and that the leverage effect match is not satisfactory as the simulated market's leverage effect is not significantly away from zero. Stylized facts for 1-day returns are the most difficult but the most important to replicate as close-to-close returns are conventionally the most used to calculate statistics and measure portfolio performance and, consequently, to calculate risk management requirements.

Another limitation of this model is the speed of training. Training the final model took a few hours. This time is not trivial considering that we only optimized for ten parameters. The model would be a strain on the computational resources of a bank if we were to re-optimize these models every day for multiple markets, which is something that a bank would need to do for ABM models in production in their system. We might also want to have more parameters to increase the realism of the simulation.

Finally, the current model lacks limit order book realism. We expect this deficit because we had no actual limit order book data to match. The lack of LBO realism is a significant limitation as if we wanted to test the effect of a trading strategy, we would find it valuable to replicate a realistic limit order book as this would estimate the slippage of the trading strategy and the impact on the liquidity.

6.3 Future Work

We have only scratched the surface of the application of ABM methods on financial price replication. This research was limited in finding the methods to approximate the stylized facts. It would be interesting to extend this research by testing its effectiveness in real-life applications. For example, we could further explore the concept presented by Maeda et al. and use the ABM as a training environment for a reinforcement learning agent that could learn from a simulated environment and perform well in the real world [8].

We would use the market data to train an ABM to fit a market's stylized facts, including stylized facts about the liquidity and the limit order book. Then we would simulate many markets while injecting in the system a Reinforcement Learning agent that would learn to outperform its peers. We could test the same agent in the real market and see if it could learn and generalize enough to make money in a live trading environment.

Bibliography

[1] Galin L. Jones. On the Markov chain central limit theorem. *Probability Surveys*, 1(none):299–320, January 2004. ISSN 1549-5787, 1549-5787. doi: 10.1214/154957804100000051. URL https://projecteuclid.org/journals/probability-surveys/volume-1/issue-none/On-the-Markov-chain-central-limit-theorem/10.1214/154957804100000051.full. Publisher: Institute of Mathematical Statistics and Bernoulli Society. pages 1

[2] Rama Cont. Volatility clustering in financial markets: Empirical facts and agent-based models. In Gilles Teyssière and Alan P. Kirman, editors, *Long Memory in Economics*, pages 289–309. Springer. ISBN 978-3-540-34625-8. doi: 10.1007/978-3-540-34625-8_10. URL https://doi.org/10.1007/978-3-540-34625-8_10. pages 1

[3] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. 99:7280–7287. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.082080899. URL https://www.pnas.org/content/99/suppl_3/7280. Publisher: National Academy of Sciences Section: Colloquium Paper. pages 2

[4] E. Cisana, L. Fermi, G. Montagna, and Oreste Nicrosini. A Comparative Study of Stochastic Volatility Models. *arXiv.org, Quantitative Finance Papers*, September 2007. pages 4

[5] L. Ponta, E. Scalas, M. Raberto, and S. Cincotti. Statistical Analysis and Agent-Based Microstructure Modeling of High-Frequency Financial Trading. *IEEE Journal of Selected Topics in Signal Processing*, 6(4):381–387, August 2012. ISSN 1941-0484. doi: 10.1109/JSTSP.2011.2174192. Conference Name: IEEE Journal of Selected Topics in Signal Processing. pages 4, 19

[6] Agent-based Modeling and Investors' Behavior Explanation of Asset Price Dynamics on Artificial Financial Markets | Elsevier Enhanced Reader. pages 4

[7] Adam Majewski, Stefano Ciliberti, and Jean-Philippe Bouchaud. Co-existence of Trend and Value in Financial Markets: Estimating an Extended Chiarella Model. *arXiv:1807.11751 [q-fin]*, July 2018. URL http://arxiv.org/abs/1807.11751. arXiv: 1807.11751. pages 5, 6, 20

[8] Iwao Maeda, David deGraw, Michiharu Kitano, Hiroyasu Matsushima, Hiroki Sakaji, Kiyoshi Izumi, and Atsuo Kato. Deep Reinforcement Learning in Agent Based Financial Market Simulation. *Journal of Risk and Financial Management*, 13(4):71, April 2020. doi: 10.3390/jrfm13040071. URL https://www.mdpi.com/1911-8074/13/4/71. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute. pages 6, 20, 45

[9] Roberto Mota Navarro and Hernán Larralde. A detailed heterogeneous agent model for a single asset financial market with trading via an order book. *PLoS ONE*, 12(2), February 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0170766. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5330465/. pages 6

[10] M. Gilli and P. Winker. A global optimization heuristic for estimating agent based models. 42(3):299–312. ISSN 0167-9473. doi: 10.1016/S0167-9473(02)00214-1. URL https://www.sciencedirect.com/science/article/pii/S0167947302002141. pages 7, 27

[11] Apoorv Agnihotri and Nipun Batra. Exploring bayesian optimization. 5(5):e26. ISSN 2476-0757. doi: 10.23915/distill.00026. URL https://distill.pub/2020/bayesian-optimization. pages 25