Cryptocurrency exchange simulation

Kirill Mansurov, Alexander Semenov†, Dmitry Grigoriev†, Andrei Radionov† and Rustam Ibragimov†

Contributing authors: k.mansurov@spbu.ru; asemenov@ufl.edu; d.a.grigoriev@spbu.ru; aradionov@eu.spb.ru; i.rustam@imperial.ac.uk;
†These authors contributed equally to this work.

Abstract

In this paper, we consider the approach of applying state-of-the-art machine learning algorithms to simulate some financial markets. In this case, we choose the cryptocurrency market based on the assumption that such markets more active today. As a rule, they have more volatility, attracting riskier traders. Considering classic trading strategies, we also introduce an agent with a self—learning strategy. To model the behavior of such agent, we use deep reinforcement learning algorithms, namely Deep Deterministic policy gradient (DDPG). Next, we develop an agent–based model with following strategies. With this model, we will be able to evaluate the main market statistics, named stylized–facts. Finally, we conduct a comparative analysis of results for constructed model with outcomes of previously proposed models, as well as with the characteristics of real market. As a result, we conclude that our model with a self–learning agent gives a better approximation to the real market than a model with classical agents. In particular, unlike the model with classical agents, the model with a self—learning agent turns out to be not so heavy—tailed. Thus, we demonstrate that for a complete understanding of market processes simulation models should take into account self—learning agents that have a significant presence at modern stock markets.

Keywords: Agent–based model, Reinforcement Learning, Market simulations, Cryptocurrency

1 Introduction

Today, the share of e-commerce in modern markets has grown significantly. According to data from popular trading platforms, the share of electronic trading on modern equity and financial markets [1] for 2022 is 60-75%. Moreover, research by Finance Feeds[2] indicates that more than 72% of modern participants in the financial sector seek to introduce artificial intelligence (AI) technologies into their work. Given these trends, there is a need for high-quality analytics of modern markets and the ability to conduct simulations.

This topic was extremely interesting to us and we conducted a study on agent simulation of modern markets [1]. Our conclusions allowed us to say that the simulation of modern equity markets works better when adding agents whose strategies are based on artificial intelligence. At the same time, a large proportion of agents with artificial intelligence is more typical for emerging markets. A large proportion of agents with artificial intelligence is more typical for emerging markets. However, as part of our previous research, we were unable to formulate a conclusion for the crypto markets. The patterns obtained for them were not obvious enough. Nevertheless, we decided to include the study of the effect of our method on the crypto markets in our plans for further research.

Indeed, crypto markets have many unique features that make it impossible to apply classical market analysis to them. Siyun He and R. Ibragimov et al.[2] are engaged in identifying the factors that best affect the prediction of cryptocurrency profitability. They come to interesting conclusions that the most useful set of factors for predicting profitability is very different from the set of useful factors for predicting profitability in conventional markets. The authors claim that this is influenced by factors such as the high heaviness of tails, which are characteristic of crypto markets, and the difference in partial correlation parameters.

At the same time, the relevance of studying simulations of the cryptocurrency market is unquestionable. Cryptocurrency is currently a promising direction in the financial sector. The organization of new exchange platforms and the rapid growth of trading volume on them pushes for the study of cryptocurrencies. A very important task may be to simulate a crypto–exchange with the types of agents used in modeling conventional markets. Simulations allow us to understand not only the empirical nature of the market itself but also to learn how to obtain relevant data for testing our own strategies.

The global cryptocurrency market cap today is $1.18 Trillion[3]. Over the past 2 years, the volume of capitalization has grown by 13.72%[4]. For electronic traders on cryptocurrency markets, by estimated standards, it is about 86%. This is more than 20% higher than in classical markets.

[1] https://www.quantifiedstrategies.com/what-percentage-of-trading-is-algorithmic/
[2] https://financefeeds.com/changing-role-ai-financial-markets/
[3] https://www.quantifiedstrategies.com/what-percentage-of-trading-is-algorithmic/
[4] https://www.jpmorgan.com/solutions/cib/markets/etrading-trends

Considering the data mentioned above, we felt that applying our approach to crypto markets could be quite useful.

There are many approaches for modeling market behavior. The key ones are the discrete-event and agent–based approach[3, 4], which we use here.

We developed five types of agents–traders, whose joint interaction was supposed to simulate the exchange process. These include the market–maker agent–trader, liquidity–consumer trader, mean–reversion trader[5], and momentum agent–trader[5, 6]. All of these models have logic characteristic of the strategies of traders of each type. In addition to these agents, we also added noise–traders that were supposed to simulate uncontrolled noises and a self–learning agent that developed on a policy–based reinforcement learning algorithm - **DDPG** (Deep Deterministic Policy Gradient)[7]. This agent's task is to try to describe all traders who trade on the exchange using strategies with self–learning algorithms.

In this paper, our main goal is to evaluate the difference in the quality of market simulation between a model with classical agents and a model where a self–learning agent was added. The task is also to select a model and hyperparameters for it so that it generates market data most similar to real ones.

2 Related research

Christian Oesch et al. [8] present one of the first agent-based models with some classical types of traders, such as market–makers (liquidity providers), liquidity consumers and noise traders. The main aim of the article was to examine the theory of market impact. This research describes the importance of parameters such as volatility clustering, long–memory effect in order flow, autocorrelation of returns and other parameters. In our article, we use these parameters to study our agent-based model and as a subset of features for our self-learning agent.

McGroarty et al. [9] extend the model from the previous article and add new agents whose logic is based on high–frequency strategies[10]: momentum traders and mean reversion traders. Authors compare results with new agents: conducting a sensitivity analysis and analysis of stylized facts, they come to the conclusion that the use of new types of agents in such models is justified.

In [11] authors consider one of the possible models for building a multi–agent system for trading between agents. Two agents are linked if one copies the actions of the other. An agent can only copy the actions of one other agent. At the beginning of each turn, each agent looks at the accumulated capital of the others and determines the probability of copying these agents in proportion to it. Next, the choice of their "guru" is made and the agents send various requests that vary within random limits from the price on the previous move and set a new price for the move. The position size is determined randomly, but lies within the agent's capital. The guru determines the direction of the application randomly.

3 Self–learning agent logic

In the following section, we will discuss the technical component of the agent, which we call a self–learning agent. In our model, such an agent acts as an approximation of traders who use adaptive strategies based on machine learning algorithms in their work. The choice of the algorithm that we use in the work of our agent is not obvious. Therefore, we will discuss the motivation for choosing the Deep Deterministic Policy Gradient approach for our task.

3.1 Motivation for choosing an algorithm

As mentioned earlier, our agent should approximate the class of electronic traders who use machine learning algorithms in their strategies. Since the class of machine learning algorithms that traders can use for their strategies is quite large, it would not be correct to use only one of these approaches. However, all machine learning algorithms have one generalizing property: they use the space of market factors as a state to derive approximate values of objective functions using optimization methods. It turns out that we need to build a generalization of possible strategies on a certain space of factors that can be used for trading in the market. Reinforcement learning is best suited for such tasks. C. Packer et al.[12] show how well reinforcement learning generalization works for generalization problems of smaller algorithms.

Next, we assume that our traders have the opportunity to study the environment space in advance and pre–train their strategies to be more successful in the market. As a result, we got an off–policy reinforcement learning algorithm, which may assume that the environment is already known in advance. Hence, we may not always follow our policy greedily. Additionally, given the high dimension of our state and action space, we decided to follow a continuous action–space approach.

Then, among the many off–policy reinforcement–learning algorithms, we chose the Q–learning approach. This approach can best fit the description of the self–learning process for several reasons. Firstly, this algorithm makes good use of such a property of information as relevance. This allows the agent to continuously update their knowledge over time. Secondly, this approach has quite important properties for the market: scalability and transferability[13].

And finally, we chose Deep Deterministic Policy Gradient (DDPG) because this algorithm can be considered as an extension of the Q–learning algorithm to state and action continuous spaces [14].

Taking all of this into account, we chose the DDPG algorithm, as the work [7] shows, it is one of the best solutions among Deep Reinforcement Learning algorithms in multi–agent systems. However, studies on the application of the DDPG algorithm as an approximation of agents using machine learning in their strategies on market have not been conducted yet.

3.2 Technical discussion about DDPG algorithm

Deep Deterministic Policy Gradient (DDPG) is a reinforcement–learning algorithm, that was originally considered as an experiment with deep Q-learning for state and action continuous spaces in [14]. It based on two neural networks: an actor and a critic. First of them tries to approximate the Q-function, while the critic attempts to approximate the loss function. This algorithm is best suited to us, since we can consider both state space and action space as continuous variables because their discrete space is very large.

Next, we will introduce several technical designations to describe the operation of our algorithm.

First of all, recall the Bellman equation, which is the main equation in Q–learning subject:

$$Q^*(s,a) = E_{s' \sim P}[r(s,a) + \gamma \max Q^*_{s'}(s',a')] \tag{1}$$

where $s' \sim P$ and $P(s,a)$ is a distribution for the next state.

And also we can rewrite this with the approximation of Q–function in continuous form:

$$Q^*(\phi, D) = E_{(s,a,r,s',d) \sim D}[(Q_\phi(s,a) - (r + \gamma(1 - d \max Q_{a'}(s',a')))^2] \tag{2}$$

where ϕ is a set of parameters of network, and D is a set of transitions (s,a,r,s',d).

There are two neural networks in the DDPG algorithm: the actor network and the critic network. The former predicts what action needs to be performed now, and the latter predicts next evaluates how good these actions are. In other words, the second neural network is designed to approximate the error. It makes it possible to evaluate and learn not only the Q function, but also the loss function of the system too. Also when we make learning process for our function, we need to make target networks for our actor and critic networks. The term

$$r + \gamma(1 - d) \max Q_{a'}(s',a') \tag{3}$$

is called the target, because when we minimize our loss, we are trying to make the Q-function be more like this target. However, the target depends on the same parameters we are trying to train: ϕ. This makes the minimization process unstable. The solution is to use a set of parameters which comes close to ϕ, but with a time delay—that is to say, a second network, called the target network, which lags the first. The parameters of the target network are denoted ϕ_{targ}.

Our target network updates for each update of the main network by the Polyak rule:

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi \tag{4}$$

In general, this model is given by the following formula:

$$L = E_{(s,a,r,s',d) \sim D}[(Q_\phi(s,a) - (r + \gamma(1 - d) * Q_{\phi_{targ}}(s', \mu_{\theta_{targ}}(s'))))^2] \tag{5}$$

where $\mu_{\theta_{\text{targ}}}$ is the target policy.

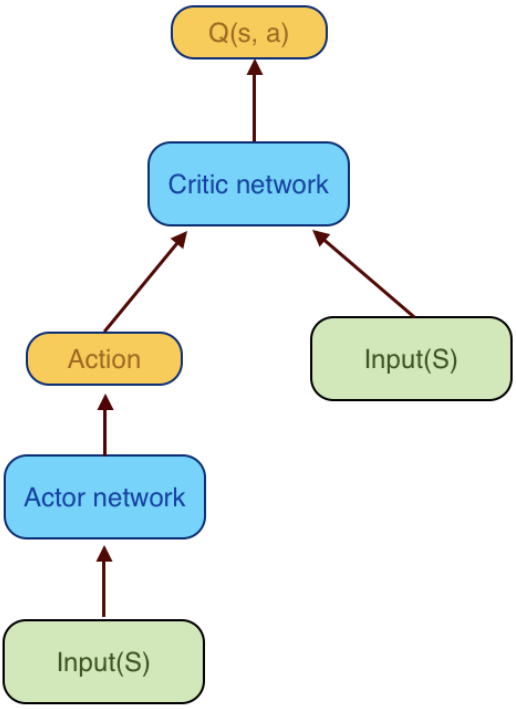

Fig. 1 Architecture of DDPG network model. There are two neural networks in the model, one of which outputs an approximate Q-function value, and the second outputs an estimated quality of this approximation

We can represent the general process of the algorithm in the form of the following procedure. The final strategy of the self–learning agent includes the process of the DDPG algorithm. During the training, the agent makes trades and learns from his mistakes. The target function here is the current market price p_t. As an action a, we take a trade made by an agent to buy or sell an asset. Within our model, the algorithm will work in accordance with the logic of the other agents. So, at each moment of time t_i, the agent with probability p_{DDPG} gets the opportunity to make a trade of a certain size from the acceptable range $(price_{min}, price_{max})$.

Next, in accordance with the DDPG algorithm, the agent computes the target:

$$y(r, s', d) = r + \gamma(1 - d)Q_{\theta_{targ}}(s', \mu_{\phi_{targ}}(s')) \tag{6}$$

Immediate reward size r is calculated as follows:

$$r = (price_t - price_{t-1}) * V_t \tag{7}$$

where $price_t$ is the asset price at current time t, V_t is the agent's asset volume (may be less than zero).

Finally, there is a comparison of the predicted value for the model and the actual price change. Based on the difference between these values, we get the value for policy update.

Algorithm 1 Self–learning trader strategy

Set Policy parameters θ, Q-function parameters ϕ
Set main target parameters: $\theta \leftarrow \theta, \phi_{targ} \leftarrow \phi$
while Trading is avaliable **do**
Observe state s, select action $a = range(a_{low} = pr_{min}, a_{high} = pr_{max}, \mu_\theta(s) + \epsilon)$
if $rand() < p_{DDPG}$ **then**
Make action a, get new state s'
Observe next state s', reward r. Suppose G=(s, a, r, s', d)
Compute targets

$$y(r, s', d) = r + \gamma(1 - d)Q_{\theta_{targ}}(s', \mu_{\phi_{targ}}(s'))$$

Update $Q_{function}$ using step of gradient descent:

$$\bigtriangledown_\phi \frac{1}{\mathsf{G}} \sum_{(s,a,r,s',d') \in B} (Q_\phi(s, a) - y(r, s', d))^2$$

Update policy by step with gradient ascent:

$$\bigtriangledown_\theta \frac{1}{\mathsf{G}} \sum_{s \in B} Q_\phi(s, \mu_\theta)(s)$$

Update target networks with

$$\theta_{targ} \leftarrow \rho\theta_{targ} + (1 - \rho)\theta$$

$$\phi_{targ} \leftarrow \rho\phi_{targ} + (1 - \rho)\phi$$

end if
end while

The state parameter **s** was set by features, which approximately describe the state space of the environment. Thus, for each aggregated time period **t**, **st** was set by several parameters. Such indicators as ***close price***, ***min price***, ***max price***, ***spread***, ***volume value V***, ***volume imbalance*** Δ ***V***, the ***MACD*** indicator [15, 16] with fast–period parameter = 12, slowperiod = 26 and signal–period = 9. The ***MACD signal*** indicator [15] with same

time–period parameters, the ***MACD hist*** [15] indicator, the ***data RSI*** [17] indicator with time–period parameter which equal 14, and the ***data OBV*** indicator [18] were selected.

4 Stylized facts

In this section we will talk a couple of words about stylized facts. By stylized facts we mean a set of statistics which help to estimate various indicators for time–series data. This set of statistics allows us to describe the behavior of the entire market as a whole quite qualitatively. In our research, we use stylized facts to achieve 2 goals. Firstly, with their help, we will be able to compare different markets with each other. So, we are going to compare the classic market, the cryptocurrency market, as well as the agent simulation of the market without a training agent and together with it, respectively. Secondly, we use an error function based on differences in the statistics of stylized facts. With its help, we will select the optimal parameters for the agent simulation model so that it best models the market we need.

In our research we consider such statistics as value of fat-tailed distribution of returns, value of autocorrelation of returns[19, 20], Q-Q plot values [21], ECDF plotting, and Hill estimation [22].

4.1 Fat-tailed distribution of returns via kurtosis

The measure of the heaviness of tails is a key indicator for the distribution of returns in financial markets. This value, among other things, allows us to indirectly evaluate such an important indicator as volatility, as well as statistics coming from it, for example VaR. The kurtosis coefficient is the simplest and most obvious way to estimate heavy–tail measures of distributions under consideration.

Let μ_4 denotes the fourth central moment: $\mu_4 = E[(X - EX)^4]$. Then, kurtosis coefficient will look like as:

$$\gamma_2 = \frac{\mu_4}{\sigma^4} - 3 \tag{8}$$

Where $\sigma = \sqrt{D[X]}$ is standart deviation of sample X.

In order to conduct an effective comparative analysis for the severity of tails, it is necessary to take into account the types of markets being compared. In our case, in studies of high–frequency trading, the tails of yield distributions turn out to be quite heavy in relation to the normal distribution ($\alpha \approx 3$), and this indicator is usually higher in the markets for developing exchanges ($\alpha > 3$) [9, 10].

4.2 Autocorrelation of returns

The autocorrelation function for time series is able to detect and evaluate certain patterns of data behavior that repeat over time. In financial analysis

it is used, as a rule, on differentiated values of price changes, which can be interpreted as returns. In general, we can represent autocorrelation function as follows:

$$C_{\alpha,X(t)}(\tau) = corr(\delta X^{(1)}(\mathsf{t}+\tau, \Delta t)^{\alpha}, \delta X^{(1)}(\mathsf{t},\delta t)^{\alpha}) \tag{9}$$

Where $\delta X^{(1)}$ is first–order differential, α – autocorrelation rate.

In modern financial markets, the presence of significant autocorrelation is an abnormal behavior than normal. R. Kont et.al. [19] becomes to conclusion that in markets with lagre timeframes (greater then 15 minutes) any confidence values of autocorrelation should be absent. At the same time, the presence of negative autocorrelation is quite normal in high–frequency markets. This is primarily due to the active presence of mean–reversion traders, where market–makers can also be included. In following research of returns linear independence we investigate α from eq. 9 equal to 1.

In this case, eq. 9 takes the following form:

$$C_2(\tau) = corr(\mathsf{r}(\mathsf{t}+\tau, \Delta t)^2, \mathsf{r}(\mathsf{t},\delta t)^2) \tag{10}$$

Also, the following value can be calculated via power–law coefficient, which can be calculated as 11:

$$C_{\alpha}(\tau) = \frac{A}{\tau^{\beta}} \tag{11}$$

Where β is a coefficient for absolute returns. In our work, we will use the first variant of volatility clustering estimation, because it is better suited for visual representation.

4.3 Q-Q plot

Q–Q (quantile-quantile) plot is a visual estimation method of distributions difference. The key point of the method is to display the corresponding quantiles of two distributions on the x-axis and on the y-axis, respectively.

Firstly, the QQ plot usage as visual comparative method was proposed in [21]. Christofersen et.al [23] makes comparison of the empirical distribution quantiles of returns with the normal distribution. This allows us to test the assumption put forward by Cristofferson earlier that the distribution of stock market data balances will be more heavy-tailed than the normal distribution.

4.4 ECDF plotting

Like the QQ plot, the ECDF (Empirical cumulative distribution function) plot is a graphical method for estimating the difference of distributions. But, unlike QQ plot, this method allows us to most clearly assess the difference in tail heaviness of each of the distributions.

This approach demonstrates the success of using this approach to visualize data in [24], in particular indicators of normalized profitability. With the help of such observations, it is possible to visually assess the difference in data.

In our method, we will compare the available empirical distributions of our observations with normal distributions, in which the first and second central moments will be equal.

4.5 Hill estimator

The Hill estimator is one of the most qualitative ways to assess the shape of the tail distribution. J. Huber and M. Kirchler in [24] consider the approach of constructing a Hill estimate for returns based on trade data. They also explain how to use the resulting score as a stylized fact. The authors propose the following formula for constructing Hill estimates:

$$\alpha_{Hill} = \frac{m}{\sum_{j=1}^{m} \ln(ABSr_{n-j+1}) - \ln(ABSr_{n-m})} \tag{12}$$

where $ABSr_i = \|\frac{r_i - \overline{r}}{\sigma_r}\|$, m indicates ordered statistics, which we want to estimate with Hill estimator.

5 Simulation model

In the exchange model taken as a basis, minute data on the bitcoin quotation on the Bitfinex platform was used as a basis.

We used the multi–agent approach proposed by C. Oesch in [8, 25] as the basis of our implementation. The main idea of the method is to represent each class of agents as representative agents within the current model. Thus, in total, there are six types of agents in our model, each of which has its own strategy. This approach makes it possible to avoid implementing a large set of agents with similar strategies and decision-making parameters, but it does not affect the statistical indicators tracked in the model. Article [26] describes acceptable cases of using representative agents in the framework of building multi-agent systems. After reviewing the content, we can conclude that the use of representative agents is acceptable for our objectives. This allows us to significantly reduce the use of computational resources, which in the future will make it possible to train the model better.

One of the simplifications of the model is the removal of money. This simplification was applied based on the following questions:

– The question of the finiteness or infinity of money is a key one in the process of including the latter in the model. If we include a finite amount of money in our model, the final distribution of wealth is obvious, since in our model there are both chartist agents whose goal is to earn money on price fluctuations, and corporates, for whom earning money on the stock exchange is not a priority. The initial distribution of wealth among the agents is also unclear.

– As a rule, money in such models is used to measure the success rate of a particular strategy. However, there are approaches that allow you to assess

the degree of success of trading without resorting to the inclusion of money in the model. So, for example, in our case, for each agent, we use an abstract indicator of reward, which can be calculated as follows:

$$R_{total} = \sum_i r_i \tag{13}$$

where

$$r_i = (p_i - p_{i-1}) \cdot n_i \tag{14}$$

In this case, p_i is price on i_{th} time step, and n_i is current position size on i_{th} step respectively.

Our self–learning agent was based on policy–based reinforcement–learning algorithm - DDPG. This reinforcement–learning algorithm was chosen because it has a number of advantages:

- DDPG is off-policy algorithm. Hence, it is able to learn not only from his strategies, but also from pre-trained datasets. This aproach using by most of traders.
- This approach can be used on continuous–time systems. Exchange markets often considering as continuous time systems.
- DDPG is policy–based algorithm. This reinforcement–learning approach allows us to not study the entire set of states as a whole, but to start studying the system only in more favorable places.

6 Results

After conducting the experiments, some results were obtained on the statistics described above. General results allow us to make conclusion, that approach with DDPG agent is better in market approximation. Also, using this agents in model we get more liquidity and volatility exchange. But also, this exchange has quite less autocorrelation of any order.

6.1 Fat-tailed distribution of returns

Table 1 Kurtosis comparison

results	**mean**	**median**	**min**	**max**
Default model	45.221	44.288	36.912	63.001
With DDPG-agent	40.51	37.264	27.227	58.267
Real data	26.264	22.355	12.272	45.118

Note: Simulation of the values of kurtosis for a sample size of 100 gave a result that allows us to talk about a significant difference in the results of the kurtosis values for all three samples.

As can be seen from Table 1, the differences in kurtosis values are significant, and the initial model simulation has a much heavier distribution of

returns than the default model. The model with a self-learning agent demonstrates an approximation in terms of the value of the kurtosis coefficient to data from a real exchange.

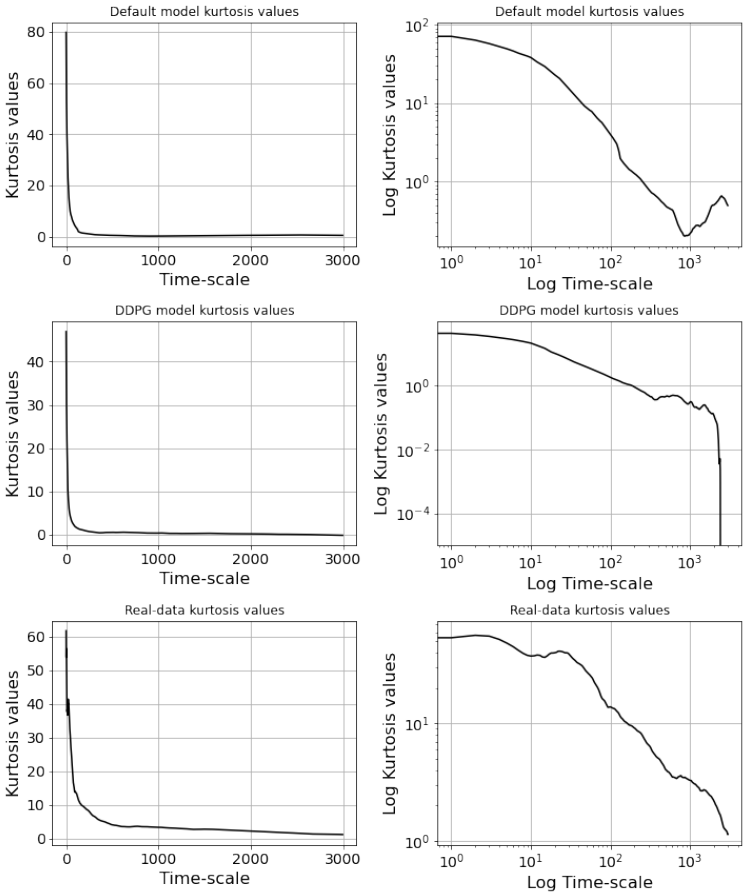

Fig. 2 Comparing kurtosis values in different scenarios. Kurtosis values by time-scale (Left), Log-Kurtosis values by Log-time-scale(Right).

6.2 Autocorrelation of returns

As Table 2 and Figure 3 shows, the most significant autocorrelation values larger than those for real data occur in the standard model. The same result is consistent with the result obtained in [9]. The model with a self-learning agent, on the contrary, produced smaller values. Based on the available data, it can be concluded that the addition of self-learning agents can reduce autocorrelation of returns. This is probably because self-learning agents are able to track intertemporal arbitrage and use it to make profit trades.

Table 2 Lags of autocorrelation function

results	1-st lag	max value	min value
Default model	0.0294	0.1925	0.0024
With DDPG-agent	0.0097	0.081	0.0006
Real data	0.0122	0.0875	0.0256

Note: Comparison of the lags of the autocorrelation function for different models showed the greatest significance of the lags for the default model, and the least significance of the lags for the model with a self-learning agent

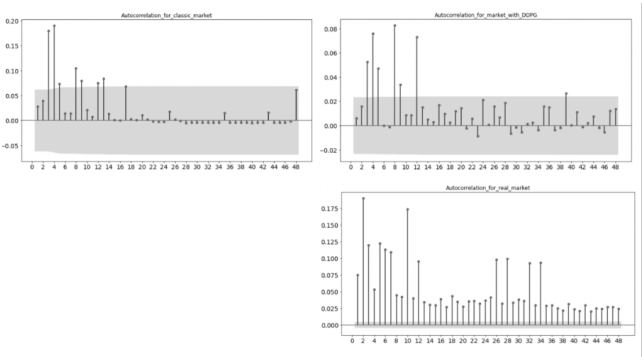

Fig. 3 Autocorrelation function ticks comparing. As can be seen from the figure, the autocorrelation values are quite small and insignificant

6.3 Volatility clustering

Volatility clustering indicates a non-linear relationship between the returns for the simulated data. As a rule, the autocorrelation function of the squares of returns is used to measure the clustering of volatility. Using the method for estimating autocorrelation proposed by Cont in [19], we get results indicating that the model with a self-learning agent has lower values than without it.

This allows us to conclude that self-learning agents are able to detect a nonlinear dependence in the data along with a linear one and use it for their own purposes.

As shown in Table 3, the results of comparing the values for the squares of the autocorrelation of the returns are similar to the results for ordinary returns. This result is quite logical and understandable.

6.4 Q-Q plotting

To assess the heaviness of the tails using q-q plot, it is possible to visually observe these distributions. The shape of the deviations of the quantile tails of the constructed distribution from the quantiles of the normal distribution with a similar mean and variance describes the heaviness of the tail.

Table 3 Autocorrelation of squared returns

results	1-st lag	max value	min value
Default model	0.0068	0.0871	-0.0004
With DDPG-agent	0.0036	0.0548	-0.0001
Real data	0.0002	0.0334	-0.0001

Note: Comparison of the lags of the autocorrelation function for different models showed the greatest significance of the lags for the default model, and the least significance of the lags for the model with a self-learning agent

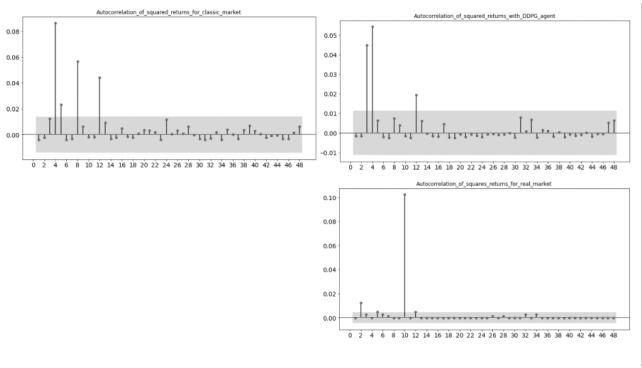

Fig. 4 The volatility clustering test is usually performed as a test for the values of the squares of returns

Tail shapes on Figure 5 can be noted that the deviation forms of the self–learning agent model and the standard model are approximately equal. Data from the real market also show a deviation from the normal distribution, but the form of their deviation is less pronounced than for the previous values.

6.5 ECDF plotting

Comparative estimates of logarithmic tails of distributions with normal distributions having similar averages and variances were carried out. As the results show, the values of the default model yields reach the greatest heavity of the tails. This fact correlates with the conclusions formed from previous results, which allows us to assert the statistical significance of the differences in the heavity of the tails of each of the models.

It is worth noting that the visual results show the difference between the realized samples and the quantiles of the normal distribution. These calculations are statistically confirmed by the Kolmogorov-Smirnov test on the equality of the sample to the normal distribution. The null hypothesis of a normal distribution of returns was rejected at the 95% significance level for all samples.

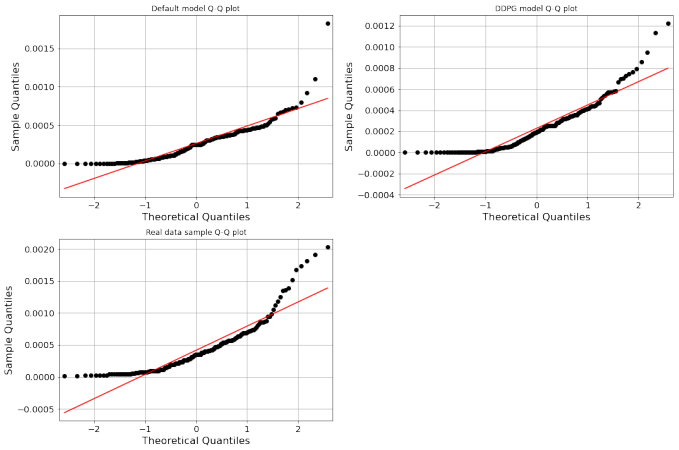

Fig. 5 Q-Q plot values comparison which shows empirical distribution of quantiles relative to the normal distribution. By comparing the data with a normal distribution, it is possible to estimate the heaviness of the tails relative to the latter.

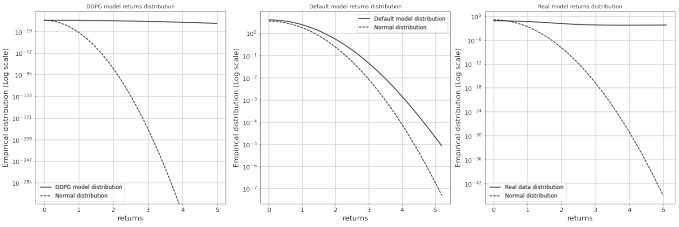

Fig. 6 Empirical cumulative distribution function (ECDF) with vertical axis log-scaling. Using a solid line, the graphs show the tails of the empirical distribution of returns. The dotted graph shows the values of a normal distribution with a similar mean and variance. These graphs clearly show the asymptotic behavior of the tails and their heaviness relative to the normal distribution.

6.6 Hill Estimator

Hill's estimators also allow us to evaluate the parameters of the heaviness of the tails of distributions. Unlike many previous methods, it allows to numerically describe parameter statistics values.

Figure 7 shows the dynamics of the behavior of the Hill coefficient relative to the ordinal statistics of the distribution of returns. As can be seen from the behavior of the curves, their general asymptotics and shapes are similar to each other. This behavior can be explained by a similar reaction of models to similar price changes. However, the distribution of returns for a model without a self-learning agent shows, on average, a more heavy-tailed distribution than the returns for models with DDPG agent.

Figure 8 shows the values for 10%, 5% and 2.5% ordinal statistics. These numerical results correlate with graphical estimates of the heaviness of the tails, since the values with the largest tail weights have lower values of the Hill

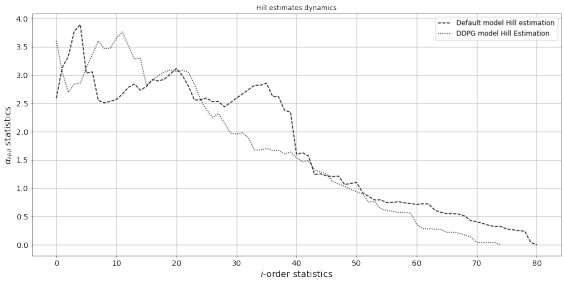

Fig. 7 Comparison of Hill coefficients with respect to ordinal statistics. As can be seen from the comparative analysis, Hill estimates for the usual model have a greater heaviness of distribution.

coefficients. Such results are credible because they are comparable to the results obtained when evaluating real markets, as well as when modeling markets using agent-based models [27, 28].

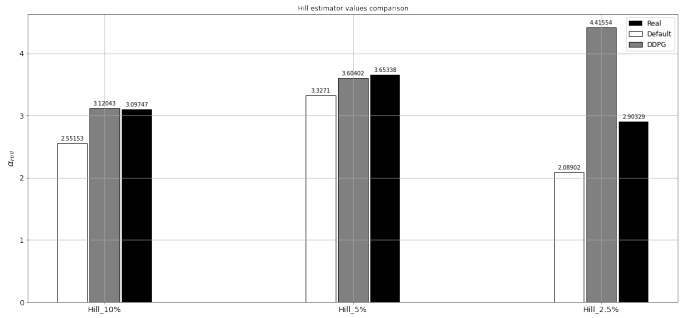

Fig. 8 10%, 5% and 2.5% Hill Estimator for each model. Models: default model, model with DDPG agent, Real data from S&P 500 futures index.

7 Conclusions

Within the current research, we introduce a new class of agents for agent–based model approach, which is based on adaptive optimization approach instead of static algorithms. The main idea of introducing this agent and studying its influence is the assumption that agents of similar types are quite common in real markets today. Our paper's key contribution is to formulate a conclusion about the justification for including agents of similar types in modern agent–based models. Having trained a standard model on SP500 futures data, we show that the inclusion of DDPG agents improves most of the statistics of the stylized facts.

We demonstrated this results on cryptocurrency market, on Bitcoin stock values from Binance exchange. Further research may be directed to study a more general application of such models to the cryptocurrency markets.

Acknowledgement

Financial support from the Saint-Petersburg University, Russia (D. Grigoriev and K. Mansurov, project ID: 101748259) and the Russian Science Foundation (R. Ibragimov, Project No. 20-18-00365) for various and non-overlapping parts of this research is gratefully acknowledged.

Imperial College Business School, London, United Kingdom; the Center for Econometrics and Business Analytics, Saint Petersburg University, St. Petersburg, Russian Federation; New Economic School, Moscow, Russian Federation.

Herbert Wertheim College of Engineering, University of Florida, FL, USA.

References

[1] Kirill Mansurov, A.S.A.R.R.I. Dmitry Grigoriev: Impact of self-learning based high-frequency traders on the stock market **33** (2022). https://doi.org/10.1609/aaai.v33i01.33014213

[2] Siyun He, R.I.: Predictability of cryptocurrency returns: evidence from robust tests **20** (2022). https://doi.org/10.1515/demo-2022-0111

[3] Linda Ponta, S.C.: Traders' networks of interactions and structural properties of financial markets: An agent-based approach. Complexity **9** (2018) https://doi.org/10.1155/2018/9072948. https://doi.org/10.1155/2018/9072948

[4] E Samanidou, D.S. E Zschischang, Lux, T.: Agent-based models of financial markets. Reports on Progress in Physics **40** (2007). https://doi.org/10.1088/0034-4885/70/3/R03

[5] Serban, A.F.: Combining mean reversion and momentum trading strategies in foreign exchange markets. Journal of Banking Finance **34**, 2720–2727 (2010). https://doi.org/10.1016/j.jbankfin.2010.05.011

[6] Balvers, R.J., Wu, Y.: Momentum and mean reversion across national equity markets. Journal of Empirical Finance **13**(1), 24–48 (2006). https://doi.org/10.1016/j.jempfin.2005.05.001

[7] Shihui Li, X.C.H.D.F.F.S.R. Yi Wu: Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient **33** (2019). https://doi.org/10.1609/aaai.v33i01.33014213

[8] Oesch, C.: An agent-based model for market impact. In: 2014 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr), pp. 17–24 (2014). https://doi.org/10.1109/CIFEr.2014.6924049

[9] McGroarty, F., Booth, A., Gerding, E.e.a.: High frequency trading strategies, market fragility and price spikes: an agent based model perspective. Annals of Operations Research **282**, 217–244 (2019). https://doi.org/10.1007/s10479-018-3019-4

[10] Eric M. Aldrich, K.L.V.: Experiments in high-frequency trading: comparing two market institutions. Experimental Economics **23**, 322–352 (2019). https://doi.org/10.1007/s10683-019-09605-2

[11] Grilli, R., Tedeschi, G.: Modeling financial markets in an agent-based framework, 103–155 (2016). https://doi.org/10.1007/978-3-319-44058-3_3

[12] Charles Packer, J.K.P.K.V.K.D.S. Katelyn Gao: Assessing generalization in deep reinforcement learning **17** (2019). https://doi.org/10.48550/arXiv.1810.12282

[13] Sharma, J., Andersen, P.-A., Granmo, O.-C., Goodwin, M.: Deep q-learning with q-matrix transfer learning for novel fire evacuation environment. IEEE Transactions on Systems, Man, and Cybernetics: Systems **51**(12), 7363–7381 (2021). https://doi.org/10.1109/TSMC.2020.2967936

[14] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015). https://doi.org/10.48550/ARXIV.1509.02971

[15] Appel, G., Dobson, E.: Understanding MACD vol. 34. Traders Press, ??? (2007)

[16] Aguirre, A.A.A., Medina, R.A.R., Méndez, N.D.D.: Machine learning applied in the stock market through the moving average convergence divergence (macd) indicator. Investment Management & Financial Innovations **17**(4), 44 (2020)

[17] Rosillo, R., De la Fuente, D., Brugos, J.A.L.: Technical analysis and the spanish stock exchange: testing the rsi, macd, momentum and stochastic rules using spanish market companies. Applied Economics **45**(12), 1541–1550 (2013)

[18] Tsang, W.W.H., Chong, T.T.L., *et al.*: Profitability of the on-balance volume indicator. Economics Bulletin **29**(3), 2424–2431 (2009)

[19] Cont, R.: Empirical properties of asset returns: stylized facts and statistical issues. Quantitative Finance **1**(2), 223–236 (2001) https://doi.org/10.1080/713665670. https://doi.org/10.1080/713665670

[20] Smith, E., Farmer, J.D., Gillemot, L., Krishnamurthy, S.: Statistical theory of the continuous double auction **3**(6), 481–514 (2003). https://doi.org/10.1088/1469-7688/3/6/307

[21] Marie, K., I, R.S.: The qq-estimator and heavy tails. Communications in Statistics. Stochastic Models **12**, 366 (1996). https://doi.org/10.1080/15326349608807407

[22] Hill, B.M.: A simple general approach to inference about the tail of the distribution. The Annals of Statistics **3**(6), 1163–1174 (1975). https://doi.org/10.1088/1469-7688/3/6/307

[23] Christofersen, P.: Elements of Financial Risk Management. https://doi.org/10.1016/B978-0-12-374448-7.00008-7

[24] Huber, J., Kleinlercher, D., Kirchler, M.: The impact of a financial transaction tax on stylized facts of price returns—evidence from the lab. Journal of economic dynamics control **36**, 1248–1266 (2012). https://doi.org/10.1016/j.jedc.2012.03.011

[25] Roşu, I.: A Dynamic Model of the Limit Order Book. The Review of Financial Studies **22**(11), 4601–4641 (2009) https://academic.oup.com/rfs/article-pdf/22/11/4601/24429168/hhp011.pdf. https://doi.org/10.1093/rfs/hhp011

[26] Galán, J.M., Izquierdo, L.R., Izquierdo, S.S., Santos, J.I., del Olmo, R., López-Paredes, A., Edmonds, B.: Errors and artefacts in agent-based modelling. Journal of Artificial Societies and Social Simulation **12**(1), 1 (2009)

[27] Plerou, V., Stanley, H.E.: Stock return distributions: Tests of scaling and universality from three distinct stock markets. Phys. Rev. E **77**, 037101 (2008). https://doi.org/10.1103/PhysRevE.77.037101

[28] Wagner, F.: Estimation of agent-based models: The case of an asymmetric herding model. Computational Economics **26** (2005). https://doi.org/10.1007/s10614-005-6415-1