# A GENERATIVE ADVERSARIAL NETWORK APPROACH TO CALIBRATE MULTI-AGENT SIMULATORS


## ABSTRACT

Multi-agent simulators are designed to model and simulate stochastic systems where agents interact with each other and with the environment, and widely applied in finance, supply chain, and healthcare systems. However, multi-agent simulators are complicated and challenging to calibrate with real data or with specific system characteristics. In this note we propose a generative adversarial network approach to calibrate multi-agent simulators. Two examples of applications and one synthetic numerical experiment based on finance applications are presented for illustration.


## 1 Introduction

**GANs.** Generative Adversarial Networks (GANs) (goodfellow2014generative) have emerged in recent years as an efficient alternative to parametric models for pattern simulation with features extracted from complex and high dimensional data sets. GANs are generative models between two competing neural networks: a generator network $G$ and a discriminator network $D$. The generator network $G$ attempts to fool the discriminator network by converting random noise into sample data, while the discriminator network $D$ tries to identify whether the input sample is fake or real. The goal for the generator $G$ is to generate samples that best resemble the true samples under certain criteria.

Recently, GAN-based simulation methods have been proposed to simulate financial market dynamics (koshiyama2020generative,wiese2020quant,marti2020corrgan,ni2020conditional,takahashi2019modeling,buehler2020data,
as well as related stochastic systems (zheng2020doubly,zheng2021continuous). See also the review paper ([Cao and GuoCao and Guo2021]) and the references therein. None of these simulators, however, are designed for stochastic systems *with interacting* agents. Lack of incorporation of agents interaction and understanding of the impact of these interaction leads to over-simplistic and unrealistic stochastic systems. For instance, lack of market impacts among interacting traders leads to unrealistic back-testing performance for trading strategies and huge financial losses for investors.

**MAS.** Multi-agent simulation (MAS) is a paradigm that allows users to incorporate agent interactions in a stochastic system. It provides a framework to understand how the dynamics of the stochastic systems have emerged through interactions among heterogeneous and diverse agents and their environment. The interaction can be co-operative where agents aim to accomplish some common goals as a team, or competitive where each individual agent tries to maximize her own utility function at the expense of other agents. In an interactive stochastic system, agents receive information from other agents and the environment; there are internal rules that represent the cognitive decision process and determine how agents respond. These rules can be simple functions of the received inputs or complex ones incorporating various internal state parameters (payr2002emotions). In addition, these rules can either be fixed (cont2007volatility) or learned adaptively with respect to the stream of new information (vadori2020calibration).

The importance of agent-based simulation models has been recognized and discussed for over a decade, see for instance macal2005tutorial, heath2009survey, and macal2009agent. Nevertheless, when a simulator is used in practice, model parameters have to be set in advance to ensure that it is structurally correct and capable to produce a valid outcome. Calibration of model parameters is an essential step in constructing

a realistic simulator. This step is particularly difficult for MAS. There are two major challenges. First is the consistency issue: a reasonable MAS needs to characterize the complex and multi-faceted interacting system dynamically and consistently. Secondly, MAS needs to be *stable* with respect to parameters changes, and particularly so for agent-class level parameters and when dealing with a large number of agents. This instability issue has been plaguing traditional model calibration methods, including Bayesian approach (grazzini2017bayesian) and simulated minimum distance method (grazzini2015estimation), see also vadori2020calibration. Finally, MAS systems that are capable of overcoming the first two challenges tend to involve a large set of parameters, which are then difficult to calibrate using traditional methods such as Bayesian method and simulated minimum distance as they usually scale poorly with the increasing number of model parameters.

**Our work.** To overcome these challenges, we propose a GAN-based approach for MAS. There are several decisively advantages of this approach over existing ones. First, the adversarial training (i.e., the mini-max game) in the GAN architecture enables more robust parameter calibrations with respect to small perturbations. Moreover, the superior computational power of neural networks allows for more efficient calibrations for a large number of model parameters in a complex MAS.

**Organization of the paper.** This paper is organized as follows. The general framework of the proposed GAN approach to calibrate MAS is presented in Section 2. Two examples of simulating financial markets are introduced in Section 3. One market is with a large number of agents and the other one is with a small number of agents. A brief numerical experiment based on synthetic data is discussed in Section 4 to illustrate the proposed approach. Finally, we discuss some future directions and potential extensions in Section 5.

## 2 Set-Up

Assume there are $N$ agents in a stochastic system. Each agent $i$ is characterized by parameter $\lambda_i \in \Lambda := \{\lambda^{(1)}, \lambda^{(2)}, \cdots, \lambda^{(M)}\}$ which fully describes the behavior of the agent in the system. Agents make decisions at discrete timestamps $t = 1, 2, \cdots$, over a given horizon $T \in \mathbb{N} \cup \{\infty\}$. At each step $t$, the state of agent $i$ is $s_t^i \in \mathscr{S} \subseteq \mathbb{R}^d$ and she takes an action $a_t^i \in \mathscr{A} \subseteq \mathbb{R}^p$. Here $d$ and $p$ are positive integers, and $\mathscr{A}$ and $\mathscr{S}$ are the action space and the state space, respectively. Given the current state profile $\boldsymbol{s}_t := (s_t^1, \cdots, s_t^N) \in \mathscr{S}^N$ and action profile $\boldsymbol{a}_t := (a_t^1, \cdots, a_t^N) \in \mathscr{A}^N$, the state of agent $i$ will change to $s_{t+1}^i$ according to a transition $P^i(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The action $a_t^i$ of each agent $i$ is made according to a decentralized policy $\pi_{\Theta_{\lambda_i}}^{\lambda_i} : \mathscr{S} \to \mathscr{P}(\mathscr{A})$, i.e., $a_t^i \sim \pi_{\Theta_{\lambda_i}}^{\lambda_i}(s_t^i)$. Note that $\pi_{\Theta_{\lambda_i}}^{\lambda_i}$ is a policy that only depends on the type of the agent and is parameterized by $\Theta_{\lambda_i}$. The state of each agent $s_t^i$ can be further decomposed into *exogenous* part (e.g., macro-economic news) and *endogenous* part which depends on the actions made by all agents (e.g., price dynamics which is effected by the trading activities from the investors). With the above specified system configurations, let us denote $\mathscr{G}_{[t_1:t_1+\Delta-1]}(\cdot;\gamma) : \mathscr{S}^N \to \mathscr{S}^{N\times\Delta}$ as a multi-agent simulator (MAS) that maps the initial state profile $\boldsymbol{s}_0$ to $\boldsymbol{s}_{[t_1:t_1+\Delta-1]} := \{\boldsymbol{s}_u\}_{u\in[t_1,t_1+\Delta-1]}$, the state profile of all agents over period $[t_1, t_1+\Delta-1]$ with $\Delta \in \mathbb{N}_+$, under a given set of system parameters $\gamma := (\{\lambda_i\}_{i=1}^N, \{\Theta_\lambda\}_{\lambda\in\Lambda})$.

Given such a stochastic system with interacting agents, our goal is to construct an MAS, by training the simulator parameters $\gamma$ via GAN to match some given stylized facts and/or statistics from a targeted stochastic system. For instance, in financial markets with a large population of investors, key stylized facts include long range dependence of the log-returns and volatility clustering phenomenon with the presence of sustained periods of high or low volatilities; and these features are functionals of agents' states over some period of time.

Suppose $H(\widetilde{\boldsymbol{s}})$ is some targeted statistics (possibly multi-dimensional) of the state profile $\widetilde{\boldsymbol{s}} \in \mathscr{S}^{N\times\Delta}$ over a time period with duration $\Delta$. We define a neural network (NN) $\mathscr{D}(\cdot;\delta)$, parameterized by $\delta$, that takes the

targeted statistics as an input and outputs a single scalar. $\mathscr{D}(H(\widetilde{\boldsymbol{s}});\delta)$ represents the probability that the state profile $\widetilde{\boldsymbol{s}}$ is from the real system rather than the constructed MAS $\mathscr{G}$. We train $\mathscr{D}$ to maximize the probability of assigning the correct label to both training examples and samples from $\mathscr{G}$. And simultaneously, we train $\mathscr{G}$ to minimize $\log(1-\mathscr{D}(\mathscr{G}(\boldsymbol{s}_0)))$. In other words, $\mathscr{D}$ and $\mathscr{G}$ play the following two-player minimax game with value function $V(\mathscr{G}(\cdot;\gamma),\mathscr{D}(\cdot;\delta))$:

$$\min_{\gamma}\max_{\delta} V(\mathscr{G}(\cdot;\gamma),\mathscr{D}(\cdot;\delta)) =$$
$$\min_{\gamma}\max_{\delta}\left(\mathbb{E}_{\boldsymbol{s}_{[0:\Delta]}\sim\mathbb{P}_r}\left[\log\left(\mathscr{D}(H(\boldsymbol{s}_{[0:\Delta-1]});\delta)\right)\right]+\mathbb{E}_{\boldsymbol{s}_0\sim\mathbb{P}_z}\left[\log\left(1-\mathscr{D}(H(\mathscr{G}_{[t_1,t_1+\Delta-1]}(\boldsymbol{s}_0;\gamma));\delta)\right)\right]\right). \quad (1)$$

Here $\mathbb{P}_r$ is the joint distribution of the state profile over a period $\Delta$ from the targeted stationary system and $\mathbb{P}_z$ is the distribution of the initial state profile $\boldsymbol{s}_0$. We denote this GAN-based MAS framework as GAN-MAS .

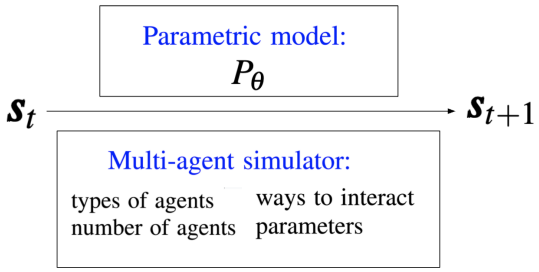



Figure 1: Comparison to classical GAN-based simulator.

As illustrated in Figure 1, in existing GAN-based simulators for stochastic systems, transition dynamics from $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$ is represented by some neural network with trainable weights $\theta$. In GAN-MAS , transition from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ is modeled via activities of a set of agents and the underlying dynamics, where a more complex structure with trainable parameters needs to be specified. See Figure 1 for the comparison.

Admittedly, training for GAN-MAS is potentially harder in practice, in return for the benefit of agent interactions. More importantly, agent strategies can now be tested in the trained GAN-MAS environment with the impact of the interacting agent naturally incorporated.

Due to the randomness of the agent policies and the complex interactions among the signals and the actions, the simulation system may not be stationary at the initial phase. Therefore, we let the simulation system run for $t_1$ rounds before the evaluation. The targeted stochastic system is implicitly assumed to be "stationary" and the theoretical guarantee of reachability of the stationary phase of our simulator $\mathscr{G}$ is left for future study.

## 3 Examples

In this section, we outline two potential applications of GAN-MAS in financial markets, one with a large number of agents and the other one with a small number of agents.

### 3.1 Agent-based Models for Stock Markets

We apply GAN-MAS to construct a simple financial market in which a single liquid asset is available for trading by three different types of traders (kirilenko2017flash,mounjid2019asymptotic): noise traders who invest based on macro-economic information, technical traders who trade based on their future price predictions, and fundamental traders (or value traders) who have knowledge of the fundamental value of

the asset. The goal is not to create a complete realistic market micro-structure. Instead, the agents in the model are built and trained in a way that attempts to capture some essential features and stylized facts from the stock markets such as volatility clustering phenomenon and heavy tails in log return distributions (chakraborti2011econophysics,cont2007volatility).

Similar to the set-up in alfi2009minimal,cont2007volatility,lux2000volatility, each trader may only buy or sell one unit of the asset at a time and trading takes place at discrete points in time. At each time step each trader might buy, sell, or stay inactive. The price is calculated from these trading decisions and all trades are executed at this price. The agents do not learn or adapt their strategies during the simulation.

**Mechanism for the price dynamics.** The price update mechanism is multiplicative. After agents express their trading decisions, the excess demand $D(t)$ is calculated. $D(t)$ is defined as the buying volume minus the selling volume at time $t$, i.e., $D(t) = N_{\text{buy}}(t) - N_{\text{sell}}(t)$. Following [Alfi, Cristelli, Pietronero, and ZaccariaAlfi et al.2009, ContCont2007, Lux and MarchesiLux and Marchesi2000] the price $P(t)$ at time $t$ is then generated as a function of the excess demand $D(t)$ as follows

$$P(t) = \left(1 + \rho \frac{D(t)}{N}\right) P(t-1), \tag{2}$$

where $\rho$ is a parameter limiting the largest proportional change in the price in one iteration and $N$ is the total number of traders which is fixed for the entire simulation. Denote $N_1$, $N_2$ and $N_3$ as the number of noise traders, technical traders and fundamental traders in the system and denote $D_1(t)$, $D_2(t)$ and $D_3(t)$ respectively as the total demand from these groups at time $t$. Then we have $N = N_1 + N_2 + N_3$ and $D(t) = D_1(t) + D_2(t) + D_3(t)$. The parameter $\rho$ measures the impact of trading on the price. $\frac{D(t)}{N}$ is the proportion of traders with the majority opinion ($-1 \leq \frac{D(t)}{N} \leq 1$). Therefore $\rho$ controls how much influence this majority has on the price. Here we assume $0 < \rho < 1$.

**Noise traders.** Noise traders trade on the basis of misunderstanding information and news regarding future prices. They make decisions and trade based on inaccurate analysis of the market. As the key variable, the sentiment of the noise agents implies their reaction to the news they get. Good sentiment means that agents see the news as good news and hope future prices will increase and bad sentiment means that the agents consider the news to be bad and hope prices in the future will decrease. The decision of noise agents is determined by the following process. At each time step t [ContCont2007]:

- Agent $i$ receives public information as a signal $S(t)$ and a private signal $S_i(t)$.
- Agent $i$ calculates the signal

$$\eta_i(t) = b_i S(t) + c_i S_i(t). \tag{3}$$

The inflow of news arrives to the market as a public signal $S(t)$ with $S(t) \sim \mathcal{N}(0, \sigma_I^2)$ and agent receives private information as individual signal $S_i(t)$ with $S_i(t) \sim \mathcal{N}(0, \sigma_P^2)$. Each agent will decide whether the news and private information are significant or not, in each case she will make a decision according to the sign of the combined signal $\eta_i(t)$. Therefore the trading rule of each noise agent $i$ is represented by

$$\phi_i^{\text{noise}}(t) = \mathbf{1}\{\eta_i(t) \geq 0\} - \mathbf{1}\{\eta_i(t) < 0\}, \tag{4}$$

and the demand from all $N_1$ noise traders can be represented as

$$D_1(t) = \sum_{i=1}^{N_1} \phi_i^{\text{noise}}(t). \tag{5}$$

In summary, the trading policy of each noise trader $i$ can be fully characterized by $(b_i, c_i)$.

**Technical traders.** Unlike the noise agents, technical agents use historical prices to infer private information and make decisions based on constructed technical indicators [Brown and JenningsBrown and Jennings1989]. Intuitively, technical indicators are designed to identify trends at an early stage and reverse the decision when the trend reversal occurs. Here technical traders are assumed to use the moving average-oscillator (MAO), a commonly used technical indicator in practice, to predict future price moves. The technique involves taking two moving averages of the price of different lengths $l_2^i > l_1^i$, a short-term moving average $A_i(t) = \frac{\Sigma_{s=t-l_1^i}^{t} P(s)}{l_1^i}$, and a long-term moving average $B_i(t) = \frac{\Sigma_{s=t-l_2^i}^{t} P(s)}{l_2^i}$. The moving average crossover is then calculated as the difference $M_i(t) = A_i(t) - B_i(t)$ between these two moving averages. $M_i(t) > 0$ indicates that the price is trending upwards and the agents will make a buy if $M_i(t) > \gamma_i$, i.e., agent $i$ believes the signal is strong enough compared to her threshold $\gamma_i$. Similarly $M_i(t) < 0$ indicates that price is trending down and agent $i$ will sell the stock if $M_i(t) \leq -\gamma_i$. Hence the trading strategy of the technical traders can be parameterized by $(l_1^i, l_2^i, \gamma_i)$. Recall that there are $N_2$ technical traders and the demand from them can be represented as

$$D_2(t) = \sum_{i=1}^{N_2} \mathbf{1}(M_i(t) \geq \gamma_i) - \mathbf{1}(M_i(t) \leq -\gamma_i). \tag{6}$$

**Fundamental traders.** In order to have fundamental traders in the model, there must first of all be a defined "fundamental value" for the traded asset. In a real trading environment, the fundamental value of a stock can be estimated as the current value of expected future dividend payments. For simplicity, all the fundamental traders in this model agree with each other on what the fundamental value $P^*$ is at any time. $P^*(t)$ is set to follow a discretized mean-reverting Heston model with stochastic volatilities,

$$P^*(t) = P^*(t-1) + \nu_1(\mu_1 - P^*(t-1)) + \sqrt{\sigma(t)}\varepsilon_1(t) \tag{7}$$

$$\sigma(t) = \sigma(t-1) + \nu_2(\mu_2 - \sigma(t-1)) + \sqrt{\sigma(t-1)}\varepsilon_2(t) \tag{8}$$

in which $\nu_1$, $\nu_2$, $\mu_1$ and $\mu_2$ are model parameters, and $\varepsilon_1(t), \varepsilon_2(t) \sim \mathcal{N}(0,1)$ are IID. The fundamental traders know a "noisy" version of fundamental value of the asset. At time $t$, fundamental trader $i$ observes $P_i^*(t) = P^*(t) + w_i(t)$ with $w_i(t) \sim \mathcal{N}(0, \sigma_F^2)$ and compares the market price $P(t)$ to $P_i^*(t)$ and decide if the asset represents good value. They will buy if the price is below the fundamental value and sell if it is above. The demand of the fundamental traders at time t is therefore given by

$$D_3(t) = \sum_{i=1}^{N_3} \mathbf{1}\big(P_i^*(t) \geq P(t)\big) - \mathbf{1}\big(P_i^*(t) < P(t)\big), \tag{9}$$

where $N_3$ is the total number of fundamental traders in the model. Their trading strategy pulls the price back towards the fundamental value. They have the opposite effect on prices to the technical traders and help to stabilize the market.

### 3.2 Market Making in Over-the-Counter Market

In over-the-counter (OTC) markets (fermanian2016behavior,ganesh2019reinforcement), dealers try to make profit via buying and selling from their clients and other dealers. When a client requests for a quote (RFQ) from multiple dealers ($i = 1, 2, \cdots, N$) to sell $v_t$ volume of bond $B$ at time $t$, each dealer $i$ will propose a quote price $a_t^i$ for buying and $a_t^i$ is decided based on signals/features including information on recent transactions, current inventory level, recent reference price, approximated bid-ask spread, and some personal information of this client. Then the client will decide whether she wants to make the transaction or not and which offer to take.

In most of the OTC markets, we have $N < 10$ [Fermanian, Guéant, and PuFermanian et al.2016] which corresponds to a stochastic system with a small number of agents. Building such a simulator will help dealers to test and further improve their trading strategies. For this application, it is natural to assume the signals/features are exogenous as over-the-counter markets are usually illiquid and there is no strong evidence of temporal correlations over time. The stylist facts/statistics of interest are the overall acceptance rate from the clients (an indirect measure of the information asymmetry) and the difference between the best bid price and the second best bid price (representing an opportunity cost).

**Request for a quote (RFQ).** In each round $t$, a prospective client has identified a specific corporate bond of interest. She sees the prices streamed by dealers for this bond, and believes that it is worth sending a buy RFQ. She also observes the CBBT bid $B(t)$ and ask $A(t)$ prices which are composite prices based on the prices streamed by dealers. We assume $B(t) = P(t) - \Delta$ and $A(t) = P(t) + \Delta$ with $P(t) \sim \text{Unif}[p_{\min}, p_{\max}]$. We denote by $V_t$ the client's view on the value of the bond which follows the following PDF

$$f(v|\mathscr{F}_t) = f_0\left(\frac{v - \frac{A(t)+B(t)}{2}}{\Delta}\right),$$

where $\mathscr{F}_t$ is the set of information up to time $t$ and $f_0$ is the PDF of an exponential power distribution

$$f_0(z;\mu_0,\sigma_0,\alpha_0) = \frac{1}{c_0\sigma_0}\exp\left(-|z|^{\alpha_0}/\alpha_0\right),$$

with $\alpha_0 > 0$, $\mu_0 \in \mathbb{R}$, $\sigma_0 > 0$, $z := (x-\mu_0)/\sigma_0$ and $c_0 := 2\alpha_0^{1/\alpha_0 - 1}\Gamma(1/\alpha_0)$. She then sends a RFQ to $N$ dealers.

**Responses from the dealers.** Dealer $i$ also sees the CBBT bid and offer prices $B(t)$ and $A(t)$ and proposes an offer $O_i(t)$, which is assumed to follow

$$g(o|\mathscr{F}_t) = g_0\left(\frac{o - \frac{A(t)+B(t)}{2}}{\Delta}\right),$$

where $g_0$ is the probability density function (PDF) of an exponential power distribution

$$g_0(z;\mu_1,\sigma_1,\alpha_1) = \frac{1}{c_1\sigma_1}\exp\left(-|z|^{\alpha_0}/\alpha_0\right),$$

with $\alpha_1 > 0$, $\mu_1 \in \mathbb{R}$, $\sigma_1 > 0$, $z := (x-\mu_1)/\sigma_1$ and $c_1 := 2\alpha_1^{1/\alpha_1 - 1}\Gamma(1/\alpha_1)$.

**Matching orders.** If $\min_{1\le i\le N} O_i(t) < V(t)$, she will take the offer with the lowest price $\min_{1\le i\le N} O_i(t)$. Otherwise, she will refuse all offers.

## 4 A Synthetic Numerical Experiment

In this section we conduct an illustrative synthetic experiment based on a modified version of the MAS model described in Section 3.1. The modifications are on the trading rules defined in (3), (5) and (8), with changes from indicator functions to their smoothed approximations. Specifically, the modified trading rules are given by

$$\begin{aligned}
\tilde{D}_1(t) &= \sum_{i=1}^{N_1} \tilde{\phi}_i^{\text{noise}}(t) \triangleq \sum_{i=1}^{N_1} 2\cdot\frac{e^{\eta_i(t)}}{1+e^{\eta_i(t)}} - 1,\\
\tilde{D}_2(t) &= \sum_{i=1}^{N_2} \tilde{\phi}^{\text{technical}}(t) \triangleq \sum_{i=1}^{N_2} 2\cdot\frac{e^{d_i\cdot M_i(t)}}{1+e^{d_i\cdot M_i(t)}} - 1,\\
\tilde{D}_3(t) &= \sum_{i=1}^{N_3} \tilde{\phi}^{\text{fundamental}}(t) \triangleq \sum_{i=1}^{N_3} 2\cdot\frac{e^{(P_i^*(t)-P(t))}}{1+e^{(P_i^*(t)-P(t))}} - 1,
\end{aligned}$$

for $t = 1, 2, \ldots$, where $d_i$'s are positive real numbers and all other notations on the RHS of the equations adopt the definitions in Section 3.1. In the modified trading rule $\tilde{D}_1(t)$ for noise traders and in the modified trading rule $\tilde{D}_3(t)$ for fundamental traders, the original indicator functions are approximated by the sigmoid function. In the modified trading rule $\tilde{D}_2(t)$ for technical traders, the original indicator functions are approximated by the sigmoid function and the original threshold parameter $\gamma_i$'s are replaced with the sensitivity parameter $d_i$'s.

Note that these modifications are to smooth the indicator functions, in order to avoid potential training issues from the discontinuity in the original definition of trading rules. This smoothing step allows us to explicitly focus on whether a simulator that involves agents interactions can be successfully calibrated by adversarial networks. Undoubtedly, discontinuities naturally arise in a range of MAS models and and needs to be appropriately addressed in the future study.

We next discuss how this modified MAS model fits into the set-up described in Section 2. The state profile $\boldsymbol{s}_t$ is given by the market price $P(t)$. We set the duration as $\Delta = 50$ and warm-up period as $t_1 = 50$. The mapping $H$ is set to be an identity mapping. In this experiment, the simulation parameters to train via GAN are given by $\gamma = (\{b_i\}_{i=1}^{N_1}, \{c_i\}_{i=1}^{N_1}, \{d_i\}_{i=1}^{N_2})$. The training data is given by a synthetic data set that is generated from the modified MAS model. For the synthetic data generation, we set $N_1 = 7$, $N_2 = 5$, and $N_3 = 3$. For the noise traders, we further assume that all noise traders share the same value of $b_i$ and $c_i$. That is, $b_1 = b_2 = \ldots = b_{N_1} \triangleq b$ and $c_1 = c_2 = \ldots = c_{N_1} \triangleq c$ with $b = 1$ and $c = 1$. In addition, we take $\sigma_P^2 = 1$ and $\sigma_I^2 = 9$. For each technical trader, we randomly select two different values from the set $\{2, 3, \ldots, 10\}$, independent from all other randomness. For the $i$-th technical trader, the selected smaller value serves as $l_i^1$, and the larger value serves as $l_i^2$. For the fundamental traders, we set $\mu_1 = 100, \mu_2 = 0.01, \nu_1 = \nu_2 = 0.1$, and initialize with $P^*(0) = 100$, $\sigma(0) = 0.01$. The parameter $d_i$'s are all set to be 0.5. The price dynamics at time step 0 is set to be $P(0) = 100$. Noisy traders and fundamental traders start to act from time step 1, while technical trader $i$ starts to act from time step $l_i^2$. We simulate the above specified MAS model for 1,000 replications independently, resulting in 1000 i.i.d. copies of $\boldsymbol{s}_{[0:\Delta]} = (P(0), P(1), \ldots, P(\Delta))$, which serve as the training data. Figure 2 shows one arbitrary sample path from the training data (see the price sequence on the top and the corresponding trading volumes for each trader on the bottom).

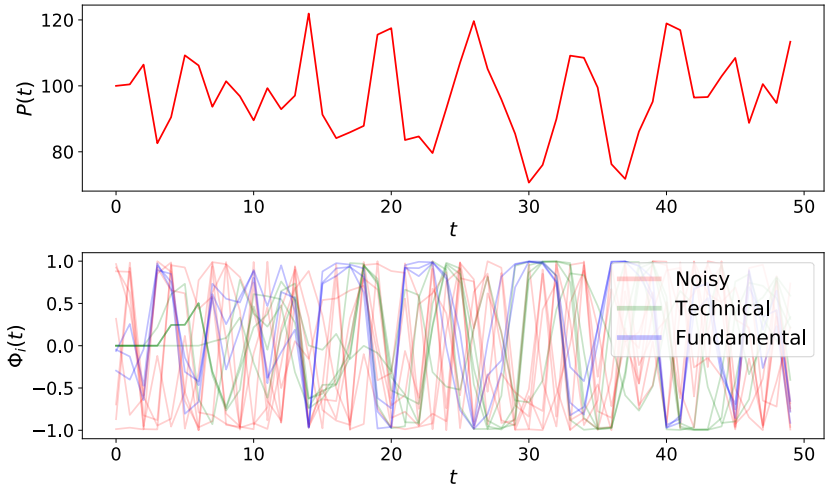



Figure 2: One sample path from training data.

With the simulation parameters $\gamma = (\{b_i\}_{i=1}^{N_1}, \{c_i\}_{i=1}^{N_1}, \{d_i\}_{i=1}^{N_2})$ to be trained via GAN, the discriminator is set with a fully connected feed forward neural network with two hidden layers and each hidden layer has 32 neurons. The activation function for the hidden layers are chosen as Rectified Linear Unit (ReLU) and the activation function for the output layer is set as sigmoid function. The discriminator parameters are summarized as $\delta$. The optimization problem in equation (1) is then solved by iteratively updating $\gamma$ and $\delta$, adopting standard GAN training procedures; see goodfellow2014generative. We use Adam optimizer with learning rate 0.002. The training takes a fixed number of 2000 iterations. The batch size is set to 1000, which means we use the whole training data in each iteration to update the parameter $\gamma$.

To train the GAN, we first need to set the initial value of $\gamma$. We initialize $b_i$, $c_i$ and $d_i$ with $\mathcal{N}(0,1)$. With the initialized parameter $\gamma$, we train GAN to calibrate the value of $\gamma$. We present the learning curve under three different initializations in Figure 3, and compare them with their respective true value. Each row of Figure 3 represents one initialization of $\gamma$. Note that in the third initialization, the parameter $b_i$ converges to $-1$ instead of 1. Recall that in equation (3) that describes the noisy trader, the parameter $b_i$ serves as a scalar parameter of a normal distribution with mean zero, which results in the values of $-1$ and 1 being equivalent for $b_i$. The training is implemented with PyTorch on a Macbook Pro 2015 with 2.7 GHz Dual-Core Intel Core i5 CPU, and takes about 40 minutes for each experiment.

We further compare the price sequences sampled from the calibrated MAS model (under the first initialization shown in Figure 3(a)) and the true model (i.e., the targeted stochastic system). Specifically, we compute one price sequence from each of the true model and the calibrated MAS model, under the same set of initialization and exogenous noises. The corresponding sample path (from the true model) is shown in Figure 2 and the sample from the calibrated MAS model is provided in Figure 4 (top). Denote respectively $P_r(t)$ and $P_c(t)$ the price sequence sampled from the true model and the calibrated model. We provide the relative error, denoted as $e(t) \triangleq |P_r(t) - P_c(t)|/P_r(t)$, on the bottom of Figure 4. The difference is uniformly smaller than 1.5% throughout the sample path and hence presents the effectiveness of our GAN-MAS framework.

**Insights from numerical experiments.** 1) In terms of the convergence speed of each estimated parameter as the number of iterations increases, there is a significant difference among parameters. The estimated parameter for $c$ tends to converge to the true value slower than the other two trainable parameters. Recall that this parameter $c$ represents the sensitivity of each noisy trader's combined signal with respect to their individual/private signal. Because this individual signal is different for each noisy trader and is not observed by other traders, there seems to be an intrinsic difficulty in learning this sensitivity, which is reflected by the convergence speed as shown in Figure 3(a). 2) The training performance is generally robust with respect to the initialization of estimated parameters, but occasionally the estimated parameter converge to a value that is different from the true value. In those cases, even though the converged value is different from the true value, their corresponding simulators generate close outputs in terms of sample paths. There may be multiple local optimal solutions that yield similar performances. 3) In the framework of GAN-MAS , the input of the discriminator is generated from an MAS model with explicitly defined inner structure, and the parameters learned have meaningful interpretations. The equations in the MAS model that describe the behavior of traders can be viewed as some prior knowledge. From the perspective of a broader generation task with specific data structure, we believe the incorporation of the prior knowledge of the inner structure can improve GAN's performance and training stability, while meanwhile brings better interpretability for the learned parameters.

## 5 Discussion and Future Directions

**Convergence to the equilibrium.** In the set-up of the GAN-MAS framework (1), the stochastic system is warmed up for $t_0$ rounds before the evaluation, with the hope that the system will converge to the stationary

phase by then. The convergence result in our experiment (see Figure 3) supports this heuristic design with $t_1 = 100$. In the future, we would like to provide some theoretical supports for the existence and reachability of some stationary equilibria or the invariant measure. See for example [Cao and GuoCao and Guo2020].

**Different loss function.** Other than the loss function used in (1) where the discriminator provides a score approximating the probability that an input sample is from the training data (targeted stochastic system), one can also adopt other loss functions with different metrices to measure the difference between the training data and the generated data. Examples include Wasserstein distance arjovsky2017wasserstein, Bregman divergence guo2017relaxed, total variation zhao2016energy, and quantile divergence [Ostrovski, Dabney, and MunosOstrovski et al.2018].

**Learning agents.** In the set-up of the GAN-MAS framework, we assume each agent adopts a prefixed policy. It will be interesting to include reinforcement learning agents in the sense that the agents can learn and improve their strategies overtime while interacting with other agents and the environment hu1998multiagent,guo2019learning.

## REFERENCES


[Alfi, Cristelli, Pietronero, and ZaccariaAlfi et al.2009] Alfi, V., M. Cristelli, L. Pietronero, and A. Zaccaria. 2009. "Minimal agent based model for financial markets I". *The European Physical Journal B* 67(3):385–397.

[Brown and JenningsBrown and Jennings1989] Brown, D. P., and R. H. Jennings. 1989. "On technical analysis". *The Review of Financial Studies* 2(4):527–551.

[Cao and GuoCao and Guo2020] Cao, H., and X. Guo. 2020. "Approximation and convergence of GANs training: an SDE approach". *arXiv preprint arXiv:2006.02047*.

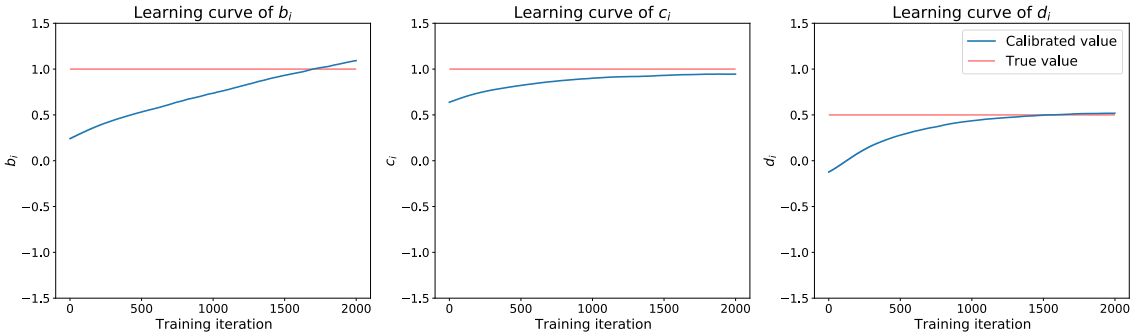
[Cao and GuoCao and Guo2021] Cao, H., and X. Guo. 2021. "Generative Adversarial Network: Some Analytical Perspectives". *arXiv preprint arXiv:2104.12210*.

[ContCont2007] Cont, R. 2007. "Volatility clustering in financial markets: empirical facts and agent-based models". In *Long memory in economics*, 289–309. Springer.

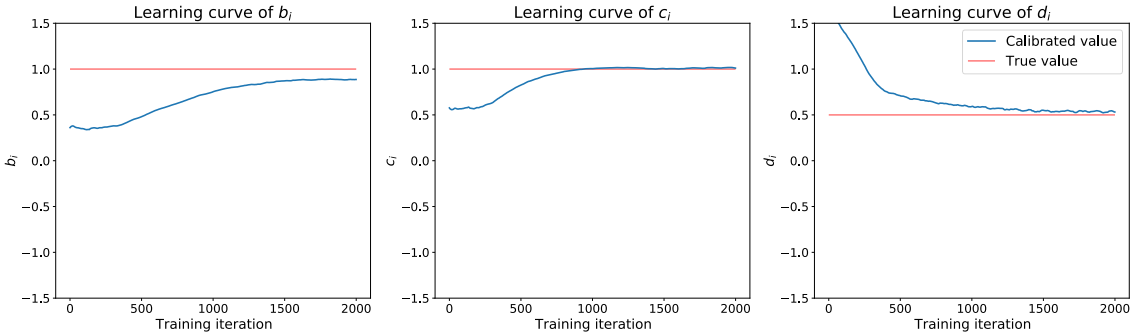
[Fermanian, Guéant, and PuFermanian et al.2016] Fermanian, J.-D., O. Guéant, and J. Pu. 2016. "The behavior of dealers and clients on the European corporate bond market: the case of Multi-Dealer-to-Client platforms". *Market microstructure and liquidity* 2(03n04):1750004.

[Lux and MarchesiLux and Marchesi2000] Lux, T., and M. Marchesi. 2000. "Volatility clustering in financial markets: a microsimulation of interacting agents". *International journal of theoretical and applied finance* 3(04):675–702.

[Ostrovski, Dabney, and MunosOstrovski et al.2018] Ostrovski, G., W. Dabney, and R. Munos. 2018. "Autoregressive quantile networks for generative modeling". In *International Conference on Machine Learning*, 3936–3945. PMLR.




(a) First initialization.




(b) Second initialization.


(c) Third initialization.

Figure 3: The convergence of parameters $b_i$, $c_i$ and $d_i$ to their true values.

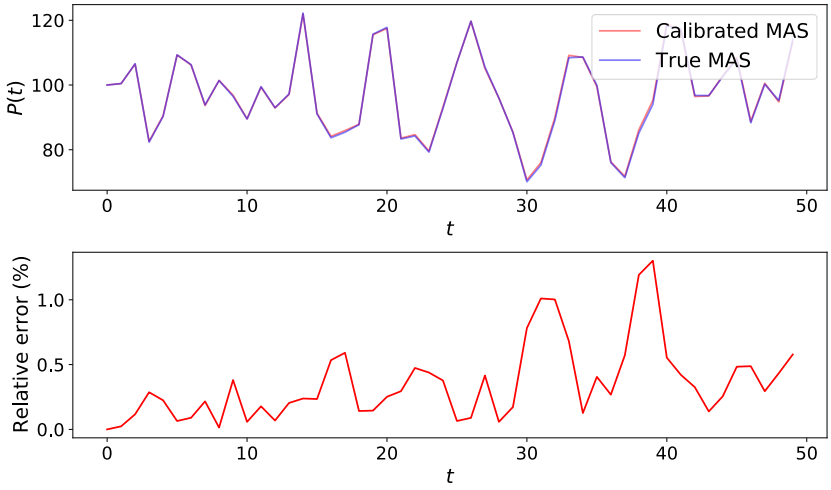



Figure 4: Comparison of price sequence.