Modelling Stock Markets by Multi-agent Reinforcement Learning

Johann Lussange, Ivan Lazarevich, Sacha Bourgeois-Gironde, Stefano Palminteri, Boris Gutkin

Noname manuscript No.
(will be inserted by the editor)

Modelling stock markets by multi-agent reinforcement learning

Johann Lussange · Ivan Lazarevich · Sacha Bourgeois-Gironde · Stefano Palminteri · Boris Gutkin

Received: date / Accepted: date

Abstract Quantitative finance has had a long tradition of a bottom-up approach to complex systems inference via multi-agent systems (MAS). These statistical tools are based on modelling agents trading via a centralised order book, in order to emulate complex and diverse market phenomena. These past financial models have all relied on so-called zero-intelligence agents, so that the crucial issues of agent information and learning, central to price formation and hence to all market activity, could not be properly assessed. In order to address this, we designed a next-generation MAS stock market simulator, in which each agent learns to trade autonomously via reinforcement learning. We

Johann Lussange
Group for Neural Theory, Laboratoire des Neurosciences Cognitives et Computationnelles, INSERM U960, Département des Études Cognitives, École Normale Supérieure, 29 rue d'Ulm, 75005, Paris, France. E-mail: johann.lussange@ens.fr

Ivan Lazarevich
Group for Neural Theory, Laboratoire des Neurosciences Cognitives et Computationnelles, INSERM U960, Département des Études Cognitives, École Normale Supérieure, 29 rue d'Ulm, 75005, Paris, France. Also at: Lobachevsky State University of Nizhny Novgorod, 23 Gagarina av., 603950, Niznhy Novgorod, Russia.

Sacha Bourgeois-Gironde
Institut Jean-Nicod, UMR 8129, Département des Études Cognitives, École Normale Supérieure, 29 rue d'Ulm, 75005, Paris, France. Also at: Laboratoire d'Économie Mathématique et de Microéconomie Appliquée, EA 4442, Université Paris II Panthéon-Assas, 4 rue Blaise Desgoffe, 75006, Paris, France.

Stefano Palminteri
Laboratoire des Neurosciences Cognitives et Computationnelles, INSERM U960, Département des Études Cognitives, École Normale Supérieure, 29 rue d'Ulm, 75005, Paris, France.

Boris Gutkin
Group for Neural Theory, Laboratoire des Neurosciences Cognitives et Computationnelles, INSERM U960, Département des Études Cognitives, École Normale Supérieure, 29 rue d'Ulm, 75005, Paris, France. Also at: Center for Cognition and Decision Making, Department of Psychology, NU University Higher School of Economics, 8 Myasnitskaya st., 101000, Moscow, Russia.

calibrate the model to real market data from the London Stock Exchange over the years 2007 to 2018, and show that it can faithfully reproduce key market microstructure metrics, such as various price autocorrelation scalars over multiple time intervals. Agent learning thus enables accurate emulation of the market microstructure as an emergent property of the MAS.

Keywords agent-based · reinforcement learning · multi-agent system · stock markets

Acknowledgements 

We graciously acknowledge this work was supported by the RFFI grant nr. 16-51-150007 and CNRS PRC nr. 151199, and received support from FrontCog ANR-17-EURE-0017. I. L.'s work was supported by the Russian Science Foundation, grant nr. 18-11-00294.

1 Introduction

General problem : Due to the steadily increasing role played by financial stock markets in global economics, and the rise of their societal impact at many levels, a main topic of economic research has historically been that of stock market stability. At the heart of such market stability studies lie the processes pertaining to price formation [29, 78]. However, research efforts in this direction have historically been hindered by the structural difficulties inherent to the bottom-up approach to system complexity inference. Yet bottom-up models could open a window on the heart of the price formation, help quantify the learning processes of financial market operators, and hence yield a wealth of information on the underlying nature of stock markets.

Past research : From a broader perspective, economic research has explored various types of quantitative models for its statistical inference of stock market data, among which, two general classes of models stand out as particularly prevalent. The first and most encountered ones are autoregressive time-series models aimed at predicting future values from past history [45]. The second class, MAS [109] (agent-based models and related methods such as order book models [50, 15, 14, 96], and dynamic stochastic general equilibrium models [90]), rather address the causal sources of financial markets activity, and thus take a bottom-up approach to market complexity. The MAS can be applied to both high and low-frequency trading [108, 5] as well as to the study of supply and demand [11] in the form of game theory [37] and the so-called minority game [72]. From a regulatory point of view, the MAS approach has an ever-increasing role to play [17], as it may be applied to model specific macroeconomics phenomena [47].

New trends : Recent technological advances have potentially given MAS approaches in finance a new level of realism, which to our knowledge has received only a limited attention.These trends emerge from the association of two present-day major scientific breakthroughs: the steady advances of cognitive neuroscience and neuroeconomics [34,56], and the progress in reinforcement learning theory due to a resurgence of machine learning methods and especially multi-agent learning [93,95]. This has been accompanied on both ends with the emergence of reinforcement learning algorithms incorporating decision-theoretic features from neuroeconomics [57,82], and neuroscience models approached from the angle of reinforcement learning [33,75]. These developments offer a way to go beyond the former generation of MAS with zero-intelligence agents [42], and their potential financial applications [41,49,79,27] have started to be extended to the class of order book models, coupled with reinforcement learning [98].

Our contribution : In order to explore and exploit the technological impact of these breakthroughs, we have developed a next generation stock market simulator based on a MAS architecture, where each agent represents an economic investor trading via a centralised order book. In such a model, the simulated agents have three novel features: i- each agent learns to both forecast and trade by independent reinforcement learning algorithms in a fully autonomous way; ii- each agent learns a pricing strategy for this forecasting and trading that is more or less chartist (i.e relying on market price) or fundamental (i.e relying on intrinsic economical value); iii- each agent can be endowed with certain traits of behaviour, cognition, and learning gleaned from behavioural economics. The three key aspects can be readily implemented thanks to the reinforcement learning framework and its direct correspondence with decision theory. These features provide a whole new level of realism in simulated data and its emulation of real stock markets data. A basic aspect of the validation of such a model is to emulate market stylised facts.

Structure : In this Paper, we first make a brief literature review (Section 2), where we define what are these stylised facts (Section 2.1), and recall some basic concepts of reinforcement learning (Section 2.2), as the core method we use to model financial stock markets. We proceed to describe in Section 3 the architecture of our MAS model in depth (Section 3.1) and its calibration process to real stock market data (Section 3.2). Finally we display some of its performance results in stock market emulation (Section 4), and end with a sum up of its key aspects (Section 5).

2 Literature review

2.1 Stylised facts

Efficient-market hypothesis : Retrospectively, a major step stone that helped economic research understand the processes pertaining to price formation and

how these impact market stability, has been the efficient-market hypothesis proposed by Fama [38] in 1970, which stated that in a market large enough where information spreads instantaneously, the actors of that market react almost correctly and instantaneously to this incoming information. Under these specific conditions, market prices thus reflect all the information available to investors, and hence stock prices are always at their fair value, so that no consistent profit can be earned over time by investors buying undervalued stocks and selling overvalued ones. The hypothesis proposed by Fama details three levels of tests to gauge such market efficiency and to ensure that stock prices reflect all available information: i- weak (information derived from historical stock prices), ii- semi-strong (information derived from public financial reports beyond market prices), and iii- strong (private information). Hence, a consequence of the efficient-market hypothesis has been that data patterns are voided of any useful exploitation by market operators.

Stylised facts : Yet, beyond how realistic and applicable these tests can be considered, quantitative finance has since ~~then~~ discovered a number of characteristic patterns in financial data, which tend to constrain the domain of the efficient market hypothesis, and indirectly posit the existence of market memory. These were called *stylised facts* and gradually discovered over the nineties: Kim-Markowitz [55], Levy-Levy-Solomon [64,58,59,63,62,60,61], Cont-Bouchaud [24], Solomon-Weisbuch [97], Lux-Marchesi [68,69], Donangelo-Sneppen [30,31,7,8], Solomon-Levy-Huang [51]. Their emulation in MAS financial research has since then been an active topic of research [66,9,19]. What is remarkable is that these stylised facts can generalise to cross-asset markets, and show remarkable time invariance. Understanding such universal market features also pertains to the exogenous or endogenous causes to price formation [29,78]. Implicit consequences of these stylised facts have fed numerous discussions pertaining to the validity of market memory [23,25] or the extension of the efficient market hypothesis [12], and whether such empirical statistical regularities can be exploited by investors to "beat the market." Their definite characteristics has varied ever so slightly over the years and across literature, but the most widespread and unanimously accepted stylised facts can in fact be grouped in three broad, mutually overlapping categories that we here sum up:

i- *Non-gaussian returns*: the returns distribution is non-gaussian and hence asset prices should not be modelled as brownian random walks [87,86], despite what is taught in most text books, and often applied in sell-side finance. In particular the real distributions of returns are dissimilar to normal distributions in that they are: i- having fatter tails and hence more extreme events, with the tails of the cumulative distribution being well approximated [25,87] by a power law of exponent belonging to the interval $[2,4]$ (albeit this is the subject of a discussion [110,35] famously started by Mandelbrot [70] and his Levy stable model for financial returns), ii- negatively skewed and asymmetric in many observed markets [22], with more large negative returns than large positive returns, iii- platykurtic and as a consequence having less mean-centered

events [18], iv- with multifractal k-moments, so that their exponent is not linear with k, as seen in [28,67,103,71].

ii- *Clustered volatilities*: market volatility tends to aggregate or form clusters [36]. Therefore compared to average, the probability to have a large volatility in the near-future is greater if it was large also in the near-past [66,105,81]. Regardless of whether the next return is positive or negative, one can thus say that large (resp. small) return jumps are likely followed by the same [70], and thus display some sort of long memory behaviour [23]. Because volatilities and trading volumes are often correlated, we also observe a related volume clustering. Indirectly, this has long-range implications on the dynamics of meta-orders, and comprises the *square-root impact law* [19] (growth in square-root of orders impact with traded volumes).

iii- *Decaying auto-correlations*: the auto-correlation function of the price returns of financial time series approach zero for any value of the auto-correlation lag, except for shorter lags (e.g. half-hour lags for intraday data) because of a mean-reverting microstructure mechanism for which there is a negative auto-correlation [22,23]. This feeds the general argument of the well-known efficient market hypothesis [38,12] stating that markets have no memory and hence that one cannot predict future prices based on past prices or information [105, 81]. According to this view, there is hence no opportunity for arbitrage within a financial market [25]. It has been observed however that certain non-linear functions of returns such as squared returns or absolute returns display certain steady auto-correlations over longer lags [23].

2.2 Reinforcement learning

2.2.1 Basic concepts

We here outline a brief overview of reinforcement learning theory. Unlike supervised or unsupervised learning, reinforcement learning is another paradigm of machine learning based on the definition of reward or reinforcement. For a thorough study of the subject, we refer the reader to [99,111,101].

Three variables: Three main variables must first and foremost be specified for any reinforcement learning agent: the possible states $s \in \mathcal{S}$ of the environment in which the agent evolves and over which it has no control, the possible actions $a \in \mathcal{A}$ that the agent can effectively control and perform in its environment, and the reward or reinforcement $r \in \mathbb{R}$ proper to that agent, where $\mathbb{R}$ is the set of real numbers. Note that like most other machine learning approaches, reinforcement learning assumes Markov state signals: a state signal that succeeds in retaining all relevant information being said to be Markov, or to have the Markov property if and only if $\forall s', r'$ and histories

$s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \cdots, s_1, a_1, r_1, s_0, a_0$, we have:

$$Pr\{s_{t+1} = s', r_{t+1} = r' | s_t, a_t\} = Pr\{s_{t+1} = s', r_{t+1} = r' | s_t, a_t, r_t, \cdots, r_1, s_0, a_0\} \tag{1}$$

where following [99], we denote $Pr\{X = x\}$ as the probability of a random variable X to take on the value x.

Three functions: Maximising its total reward over time is the ultimate goal of the agent. In order to do so, the agent needs to learn an optimal behaviour in that environment, i.e. what are the best actions a to perform for each given state s. For this, one uses a function called the *policy*, which records the agent's probabilities to select each action a when its environment is in state s:

$$\pi(s, a) = Pr(a|s). \tag{2}$$

This policy is initialised with equiprobable actions in the beginning, but updated via exploration by the agent to find an optimal policy, denoted $\pi^*(s, a)$. Hence at time t, the agent tries a new action a_t in a state s_t and then observes its associated reward, so as to update the probabilities of $\pi(s, a)$ accordingly.

However, sometimes the reward associated with the action selected by the agent is delayed in time. In the reinforcement learning framework, one thus often scales this reward received by the agent at each time step by a so-called *discount parameter*, denoted γ, which is a real number in the $(0, 1]$ interval that allows one to introduce this concept of delayed reward: the agent then updates its policy not from the realized rewards, but the realized *returns*, denoted R_t, which are defined as the sum of time-discounted future rewards: $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$. By setting $\gamma < 1$, one thus assigns less weight to the longer-term rewards.

Apart from time discounting, another important issue pertaining to rewards is that they are statistical measures. In order to gauge and assess the amplitude of this return, the agent hence works with expectations of such returns. This concept, together with that of time discounting, is found in two major functions through which the agent may assess how rewarding its action selection has been. The first one is the so-called *state-value function*:

$$V(s) = \mathbb{E}[R_t | s_t = s] \tag{3}$$

which is linked with two functions called the transition probability $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$ and the expected value $\mathcal{R}^a_{ss'} = \mathbb{E}[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$. The second major function the agent may alternatively work with is the so-called *action-value function*:

$$Q(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]. \tag{4}$$

Because one has to first compute the transition probability $\mathcal{P}^a_{ss'}$ and the expected value $\mathcal{R}^a_{ss'}$, i. e. to build a model of the environment, in order to derive the state-value function $V(s)$, one may prefer instead to assess the

returns of the action selections via updating the action-value function $Q(s, a)$ instead, as it conveniently allows the agent to explore state-action pairs only, independently of a model.

Three families: The way to learn the optimal policy $\pi^*(s, a)$ practically in reinforcement learning is usually separated in several families of algorithms. In the so-called *model-based* methods (such as *Dynamic Programming*) (or DP), the agent may after each performed action a keep record of its associated transition probability $\mathcal{P}$ and expected value $\mathcal{R}$, then compute its state-value function V, in order to update its policy π. In *model-free* methods (such as *Monte Carlo* (or MC) and *Temporal Difference* (or TD), which can be both united via eligibility traces in the form of TD(λ)), the agent may simply after each performed action a keep record of its action-value function Q in order to update its policy π. Thus model-based simply means that the agent keeps a record of the rewards associated with a model it updates of its environnement, while in model-free reinforcement learning, the agent doesn't keep such a model. Finally, *direct policy search* methods are sometimes used when the agent bypasses the record of state or action-value functions, and updates the probabilities of state-action pairs of the policy after receiving the associated reward directly. Searching the policy space can be done via *gradient-based* and *gradient-free* methods.

Three ~~dilemma~~ dilemmas: Like many other machine learning methods, reinforcement learning draws its inspiration from biology: optimal behaviour is thus not learned directly as such but rather a reward (or reinforcement) is predefined and the behaviour is indirectly learned through trial and error, formally defined as an *exploration vs. exploitation* process. This is because while it learns its policy, the agent eventually faces a dilemma over time: should it keep selecting the actions leading to the best rewards (and hence exploit its policy), or should it select new actions in order to hopefully find better rewards (and hence keep exploring)? Several methods have been proposed to answer this exploration versus exploitation dilemma, among which the ϵ-greedy method (where a chosen probability ϵ is chosen to explore a random action at each time step while exploiting the best action otherwise; this algorithm becomes greedier or bent on exploitation as ϵ is close to 0), the softmax method (where the action to explore is not purely random but graded from known best to worse according to a temperature parameter), or the pursuit method (where continually pursuing the action that is greedy) are most often encountered. The trade-off between exploration and exploitation is not always straightforward, as the agent can never truly know if it may not find a better policy by engaging in more exploration, or conversely, if such new action selection may not produce a large negative reward.

Furthermore a series of actions is sometimes necessary over prolonged periods of time before reaching the reward, and reinforcement learning thus deals with the concept of *delayed reward*. In certain applications of reinforcement learning one thus deals with the sum of discounted rewards over time $\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, which we already saw in the previous state-value $V(s)$ and

action-value $Q(s,a)$ functions. One thus deals with an issue of temporal credit assignment of state-action pairs at each time step.

Finally, another aspect of reinforcement learning is related to the so-called *curse of dimensionality*, namely that the number of state-action pairs that the agent must explore becomes quickly computationally intractable in practical applications. This brings an important dilemma to any reinforcement learning problem, namely how to specify the states and actions in a tractable way. These aspects of exploration versus exploitation, delayed reward, and dimensionality are central features of reinforcement learning, and active domains of research.

2.2.2 Recent fields of research

Most reinforcement learning research has over the years clustered around the three dilemma mentioned above, often with some overlap:

i- *Exploration vs. exploitation*: This dilemma has been addressed by different policy learning methods, such as *policy gradient methods* [100,94] seeking to optimize the control policy with respect to the return by gradient descent, or *actor-critic methods* [46], where an actor controls the behavior of the agent (policy-based reinforcement learning) and a critic evaluates the action taken by the agent (value-based reinforcement learning). Another more recent approach is *meta-reinforcement learning* [106], which deals with "learning how to learn" in order to improve a faster generalization, especially at different timescales [32] (see also *zero-shot* or *few-shots reinforcement learning*).Together with this approach, we should mention *transfer reinforcement learning* [111], which is to transfer the experience gathered on one task to another, and *imitation* or *apprenticeship reinforcement learning*, which is to learn a task from observation of another agent. We can also mention the increasing role played by *multi-agent learning* and *self-play reinforcement learning* [48], which deals with learning a policy by playing against another agent that also learns. *Multitask reinforcement learning* seeks to learn many tasks and exploit their similarities in order to improve single-task performance. It is related to ~~the now famous *asynchronous reinforcement learning*~~ [74], which executes in parallel many instances of an agent while using a shared model in order to obtain data diversification, and *modular reinforcement learning* [6], which learns the policy of a task by dividing it into smaller subtasks and reconstructing their individual policies. Another well-known approach is *Monte Carlo Tree Search reinforcement learning* [92], which determines the best action via a tree search relying on random sampling of the search space. We can also mention the important fields of *lifelong reinforcement learning* [102], which deals with learning a large amount of sequential tasks, and *hierarchical reinforcement learning* [13] which regroups the agent actions in more general tasks. Finally, we can name *hybrid reinforcement learning* [65] (or *human-in-the-loop reinforcement learning*), which deals with human interference in the algorithmic learning process in order to improve it (c.f. intelligent driving).

to the asynchronous RL that has recently gained prominence

ii- *Temporal credit assignment*: This is an issue that deals with the general specification and definition of the agent's reward and return. Apart from the aforementioned hierarchical reinforcement learning approach, ongoing research to address this difficulty includes *shaping rewards* [80], which deals with incorporating background knowledge on sub-rewards in order to improve convergence rates, *inverse reinforcement learning* [4], which seeks to extract the reward function out of the observation of the (optimal) behavior of another agent. We can also mention *homeostatic reinforcement learning* [54,53], which defines the reward via a manifold of many sub-rewards, and a state-dependent approach to the definition of the agent's reward [10].

iii- *Curse of dimensionality*: This issue arises when dealing with the issue of large-scale MDPs, so that the exploration and hence convergence to an optimal policy $\pi^*(s,a)$ becomes quickly intractable. In particular, the so-called *Q-learning* algorithm [107], as part of the more general model-free temporal difference or TD-learning was a breakthrough for reinforcement learning, because it reduced drastically the number of states to explore: only the pairs (s,a) need be explored. This curse of dimensionality points to the more general problem of function approximation, which recently led to the interface of reinforcement learning with artificial neural networks in the form of the now famous *end-to-end* or *deep reinforcement learning* [92]. Related to this is the ongoing work on *partially observable MDP models* [89,52], and *adversarial reinforcement learning* [85], which deals with modeling the noise and uncertainties in state representation via an adversarial agent applying certain specific perturbations to the system.

3 Methodology and data

3.1 Model

We now proceed to describe in details the architecture of our stock market MAS simulator, and the design of its autonomous agents.

3.1.1 General architecture

The main simulation parameters are the number of agents I, the number of traded stocks J, the number of simulation time steps T (we consider a time step to correspond to a trading day, a year to correspond to $T_y = 252$ trading days, a month to $T_m = 21$ trading days, and a week to $T_w = 5$ trading days). Usually, we consider statistical features emerging from a number S of simulations. We also model the friction costs of trade executions via broker fees b applied to each transaction, an annual risk-free rate R applied to the risk-free assets of the agents, and an annual stock dividend yield D applied to the stock holdings of each agent. The real values of these financial parameters have varied over the 2007 to 2018 time span used for model calibration, however for

simplification purposes, we have taken them as constant averages: the broker fees are set at $b = 0.1\%$ based on current industrial broker fees for the London Stock Exchange [2], the annual risk-free rate is set as $R = 1\%$ based on an approximation of the one-year gilt or UK bond yield average between January 2008 and January 2018 [3], and the annual stock dividend yield is set as $D = 2\%$ according to the current dividend impacts of FTSE 250 stocks [1]. The simulation then follows at each time step t the four general steps described below:

i- *Initialisation of agents parameters*: The simulation initialises I agents and all their individual parameters at $t = 0$. Each agent represents an individual or corporate investor or trader managing its own portfolio over time t via interaction with the market. This portfolio is made of specific stock holdings (equity) and risk free assets (bonds). These agent parameters are described in further detail in Section 3.1.2 below.

ii- *Initialisation of fundamentals*: As in other models [39,20], the simulation initialises all market prices at $P^j(t = 0) = £100$, and generates J time series $\mathcal{T}^j(t)$ as jump processes, which correspond to the prices associated with the fundamental values of the stocks. We model the topology of these fundamental values out of a metric that is often seen in corporate finance and company quarterly reports, called the *enterprise value* (which represents the approximative value at which the company would be bought in mergers and acquisitions for instance), and which we divide by the total number of stock outstanding [104]. These are not fully known by the I agents. Instead, each agent i approximates the values $\mathcal{T}^j(t)$ of stock j according to a proprietary rule [76] of cointegration $\kappa^{i,j}[\mathcal{T}^j(t)] = \mathcal{B}^{i,j}(t)$. The time series $\mathcal{B}^{i,j}(t)$ are hence the fundamental values of stock j over time t according to agent i. Together with such modelled fundamental values $\mathcal{T}^j(t)$, we show on Fig. 1 examples of such enterprise values divided by the number of stock outstanding, for various companies traded on the London Stock Exchanges, over the years $2006-2016$. So as to visually explain the concept of cointegration, we also show on Fig. 2, a sample of such modelled fundamental values $\mathcal{T}^j(t)$, together with their approximation $\mathcal{B}^{i,j}(t)$ by some agents. For $S = 20$ simulations, we calculate the average annual number of jumps of $\mathcal{T}^j(t)$ as 12.70 ± 1.85, the average amplitude $(\mathcal{T}^j(t) - \mathcal{T}^j(t-1))/\mathcal{T}^j(t)$ of these jumps as $5.90 \pm 1.84\%$, and the average difference between biased and true values $(\mathcal{T}^j(t) - \mathcal{B}^{i,j}(t))/\mathcal{T}^j(t)$ as $2.37 \pm 1.36\%$ (where $\pm$ terms refer to standard deviations). Each agent thus relies on these two sources of information for its stock pricing strategy: one that is chartist, and one that is fundamental.

iii- *Agents forecasting and trading*: Each agent autonomously uses two distinct reinforcement learning algorithms to interact with the market, each algorithm being described in further detail in Section 3.1.3 below. A first algorithm $\mathcal{F}^i$ learns the optimal econometric prediction function for the agent's investment horizon, depending on specific local characteristics of the market microstructure and the agent's fundamental valuation $\mathcal{B}^{i,j}(t)$. It thus outputs this price forecast, which will in turn enter as input the second reinforcement

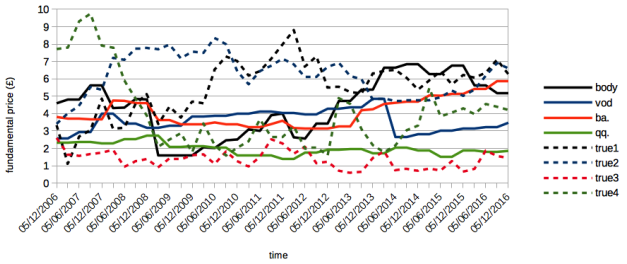

Fig. 1 Examples of fundamental values in the London Stock Exchange (cf. symbols as legend: Bodycote plc as "body," Vodafone Group plc as "vod," Boeing co. as "ba.," QinetiQ Group plc as "qq.") over the years 2006 to 2016 represented by enterprise value divided by the number of stock outstanding (continuous curves), and those generated as $J = 4$ unscaled time series $\mathcal{T}^j(t)$ (dashed curves) by the simulator at time $t = 0$.

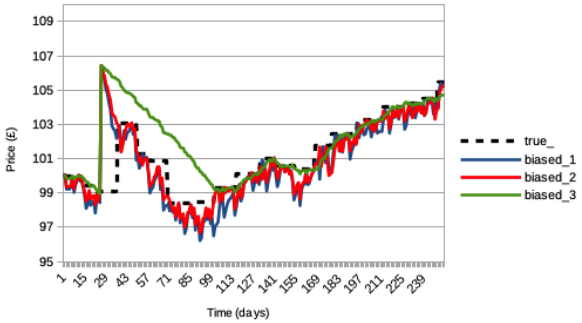

Fig. 2 Examples of fundamental values modelled by $\mathcal{T}^j(t)$ (black dashed curve) and some agent's approximation thereof as $\mathcal{B}^{i,j}(t)$ (continuous blue, red, and green curves) over a simulated time of one year.

learning algorithm $\mathcal{T}^i$. This second algorithm is in charge of sending an optimal limit trading order to a double auction order book [77] at this same time step, based on this prediction and a few other market microstructure and agent portfolio indicators. Notably, the transaction order output by $\mathcal{T}^i$ is filtered by a function $\mathcal{G}^i$, which ensures that the agent waits and sends a transaction order at the optimal time step.

iv- *Order book is filled and cleared*: A number J of order books are filled with all the agents' trading limit orders for stock j at time step t. All buy orders are sorted by descending bid prices, all sell orders are sorted by ascending ask prices, each with their own associated number of stocks to trade. Then the order book is cleared at this same time step t by matching each order at mid-price between buyers and sellers, starting from the top of the order book to the lowest level where the bid price still exceeds the ask price. Importantly, we then define the market price $P^j(t+1)$ of stock j at the next time step t as that last and lowest level mid-price cleared by the order book. We also define the trading volume $V^j(t+1)$ as the number of stocks j traded during that same

time t. Finally, we define the spread $W^j(t+1)$ of stock j at this time step t as the absolute difference between the average of all bids and asks. Notice it is this spread $W^j(t)$ that is used as input to each agent's stock pricing process, and not the traditional bid-ask spread, which is defined as the difference between the highest bid and the lowest ask.

3.1.2 Agents parameters

The model comprises several parameters, both at the agent level and the general architecture. Let $\mathcal{U}(), \mathcal{U}\{\}, \mathcal{N}(), \mathcal{N}\{\}$ respectively denote the continuous and discrete uniform distributions, and the continuous and discrete normal distributions. Each agent i is initialised at *step* 1 with the following parameters:

- Risk-free assets of value $A^i_{\text{bonds}}(t=0) \sim \mathcal{N}(0, 10^4)$. This can be seen as bonds or a bank account out of which the agent may long equity, and that will increase when the agent shorts its stocks.
- A number of stocks $Q^{i,j}(t=0)$, drawn from a discrete positive half-normal distribution $\mathcal{N}^+\{0, 100\}$, amounting to a value of its stock holdings $A^i_{\text{equity}}(t=0) = \sum_{j=0}^{J} Q^{i,j}(t=0) P^j(t=0)$, which the agent may short on the market.
- A drawdown limit $l^i \sim \mathcal{U}(40\%, 50\%)$. If the year-to-date, peak-to-bottom, absolute percentage decrease of the agent's portfolio net asset value exceeds this drawdown limit l^i at any time step t, then the agent is listed as bankrupt and unable to interact with the market anymore. These value may seem unrealistic in the sense that the drawdowns commonly tolerated in the asset management industry rarely exceed 30%. However, contrarily to a real stock exchange, such a large value is a necessity due to the model relying on a constant population of agents, which are not replaced if they become bankrupt.
- A reflexivity amplitude parameter $\rho^i \sim \mathcal{U}(0, 100\%)$, which gauges how fundamental or chartist the agent is in its price valuation, via a weighted average of the agent's technical forecast of the market price $\hat{P}^{i,j}(t)$ and its fundamental pricing $\mathcal{B}^{i,j}(t)$. The value of ρ^i modulates the amplitude of the action $a_2^{\mathcal{F}}$ of the first reinforcement learning algorithm $\mathcal{F}$ (see below).
- An investment horizon $\tau^i \sim \mathcal{U}\{T_w, 6T_m\}$, corresponding to the number of time steps after which the agent liquidates its position. Notice the bounds of this interval correspond to one week and six months in trading days.
- A trading window $w^i \sim \mathcal{U}\{T_w, \tau^i\}$, which will enter as parameter to the function $\mathcal{G}^i$ computing the optimal trading time for longing a certain quantity of stocks j.
- A memory interval $h^i \sim \mathcal{U}\{T_w, T - \tau^i - 2T_w\}$, and corresponding to the fixed size of the rolling interval memorised by the agent for its learning process.
- A transaction gesture $g^i \sim \mathcal{U}(0.2, 0.8)$, and related to how far above or below the value of its own stock pricing the agent is willing to trade and deal

a transaction. The bounds of this uniform distribution interval are selected according to one of the model's hyperparameters, namely the gesture scalar ζ^i, (cf. Tab. 1 of Section 3.2 on calibration).
– A reinforcement learning rate, which in our case is modelled by a parameter $\beta^i \sim \mathcal{U}(0.05, 0.20)$ for both reinforcement algorithms $\mathcal{F}^i$ and $\mathcal{T}^i$. The boundaries of this parameter are drawn from the rich literature in neuroscience on the value of the learning rate [83,57,82].

As we will see in Section 3.2, several of these parameters are optimised as model hyperparameters (like the drawdown limit l^i via $\mathcal{L}$, or the transaction gesture g^i via ζ^i), while others are drawn from previous studies of literature (like the reinforcement learning rate β^i), or enter as learned parameters at the agent reinforcement learning level (like the reflexivity amplitude parameter ρ^i), or are arbitrarily set (like the values of the agents' portfolios A^i_{bonds} and A^i_{equity}, and the agent investment horizon τ^i and intervals w^i, h^i), and can hence be understood as part of the structure of the model architecture.

3.1.3 Agent reinforcement learning: first algorithm $\mathcal{F}^i$

We now describe further *step* 3 and its two reinforcement learning algorithms $\mathcal{F}^i$ (which learns efficient price forecasting) and $\mathcal{T}^i$ (which learns efficient trading based on this forecast). Each algorithm is individually ran by each agent i following a direct policy search, for each stock j, and at each time step t. By direct policy search, we mean each agent selects and updates the probability associated with each action directly from the policy, and not via any action-value function (cf. Generalized Policy Iteration theorem [99]). Each algorithm has 729 and 972 potential action-state pairs, respectively. We define the sets of states $\mathcal{S}$, actions $\mathcal{A}$, and returns $\mathcal{R}$ of these two algorithms according to the following.

Via this first algorithm, the agent continuously monitors the longer-term volatility of the stock prices ($s_0^{\mathcal{F}}$), their shorter-term volatility ($s_1^{\mathcal{F}}$), and the gap between its own present fundamental valuation and the present market price ($s_2^{\mathcal{F}}$). Out of this state, it learns to optimize its price prediction at its investment horizon τ^i by exploring and exploiting three possible actions via a direct policy search: choosing a simple forecasting econometric tool based on mean-reverting, averaging, or trend-following market prices ($a_0^{\mathcal{F}}$), choosing the size of the past time interval for this forecast ($a_1^{\mathcal{F}}$), and choosing the weight of its own fundamental stock pricing in an overall future price estimation, that is both fundamentalist and chartist ($a_2^{\mathcal{F}}$).

States $\mathcal{S}^{\mathcal{F}}$: The first reinforcement learning algorithm $\mathcal{F}^i$ is defined with a state $s^{\mathcal{F}}$ in a state set $\mathcal{S}^{\mathcal{F}} = \{s_0^{\mathcal{F}}, s_1^{\mathcal{F}}, s_2^{\mathcal{F}}\}$, of dimension 27, where each $s_0^{\mathcal{F}}, s_1^{\mathcal{F}}, s_2^{\mathcal{F}}$ individually may take the values 0, 1, or 2. First, each agent computes the variances σ_L^2 and σ_S^2 of the prices $P^j(t)$, over the intervals $[t-3\tau^i, t]$ and $[t-\tau^i, t]$, respectively. Then:

– The value σ_L^2 is computed and recorded at each time step in a time series that is continually sorted in ascending order and truncated to keep a size

corresponding to agent memory interval h^i. The percentile of its present value at time step t sets $s_0^{\mathcal{F}} = 0$ if it is below 25%, $s_0^{\mathcal{F}} = 2$ if it is above 75%, and $s_0^{\mathcal{F}} = 1$ otherwise. The state parameter $s_0^{\mathcal{F}}$ thus gives the agent an idea of the longer-term volatility of the prices $P^j(t)$ of stock j, not in the sense of absolute static thresholds of this longer-term volatility, but of dynamic values depending on the agent's past history.
- The value σ_S^2 is computed and recorded at each time step in a time series that is likewise continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The percentile of its present value at time step t sets $s_1^{\mathcal{F}} = 0$ if it is below 25%, $s_1^{\mathcal{F}} = 2$ if it is above 75%, and $s_1^{\mathcal{F}} = 1$ otherwise. The state parameter $s_1^{\mathcal{F}}$ thus gives the agent an idea of the shorter-term volatility of the prices $P^j(t)$ of stock j, not in the sense of absolute static thresholds of this shorter-term volatility, but of dynamic values depending on the agent's past history. Together, with the longer-term volatility, this shorter-term gives the agent a finer perception of the market price microstructure for its forecasting. The state parameter $s_2^{\mathcal{F}}$ thus gives the agent a sense of distance between present market price and its own fundamental valuation.
- The average of $|P^j(t) - \mathcal{B}^{i,j}(t)|/P^j(t)$ is computed over the interval $[t - 3\tau^i, t]$, and sets $s_2^{\mathcal{F}} = 0$ if it is below 10%, $s_2^{\mathcal{F}} = 2$ if it is above 30%, and $s_2^{\mathcal{F}} = 1$ otherwise.

Actions $\mathcal{A}^{\mathcal{F}}$: Then, the reinforcement learning algorithm $\mathcal{F}^i$ is defined with an action $a^{\mathcal{F}}$ in an action set $\mathcal{A}^{\mathcal{F}} = \{a_0^{\mathcal{F}}, a_1^{\mathcal{F}}, a_2^{\mathcal{F}}\}$, of dimension 27, where each $a_0^{\mathcal{F}}, a_1^{\mathcal{F}}, a_2^{\mathcal{F}}$ individually may take the values 0, 1, or 2. These actions are chosen according to a direct policy search (see below), as the agent is in exploration mode or exploitation mode. First, each agent computes two different averages $\langle P^j_{[t-2T,t-T]}(t)\rangle$ and $\langle P^j_{[t-T,t]}(t)\rangle$ of past stock prices, with $T = (1 + a_1^{\mathcal{F}})\tau^i/2$, and then the econometric tool computes:

$$\hat{P}^{i,j}(t) = P^j(t) + \langle P^j_{[t-2T,t-T]}(t)\rangle - \langle P^j_{[t-T,t]}(t)\rangle \tag{5}$$

$$\hat{P}^{i,j}(t) = \frac{1}{2}\langle P^j_{[t-2T,t-T]}(t)\rangle + \frac{1}{2}\langle P^j_{[t-T,t]}(t)\rangle \tag{6}$$

$$\hat{P}^{i,j}(t) = P^j(t) - \langle P^j_{[t-2T,t-T]}(t)\rangle + \langle P^j_{[t-T,t]}(t)\rangle \tag{7}$$

if $a_0^{\mathcal{F}} = 0, 1, 2$, respectively, and hence corresponding to mean-reverting, moving-average, and trend-following projections. Hence, both $a_0^{\mathcal{F}}$ and $a_1^{\mathcal{F}}$ pertain to technical analysis: action $a_0^{\mathcal{F}}$ determines the nature of the econometric forecasting tool that will be used (mean-reverting, moving-average, and trend-following), and action $a_1^{\mathcal{F}}$ determines the size of the past intervals over which these forecasts are computed. Then, the third action $a_2^{\mathcal{F}}$ enters as a parameter of the weighted average of this latter chosen technical forecast $\hat{P}^{i,j}(t)$ and the agent's fundamental valuation of the stock $\mathcal{B}^{i,j}(t)$, to produce the agent's forecast:

$$H^{i,j}(t) = \alpha\hat{P}^{i,j}(t) + (1 - \alpha)\mathcal{B}^{i,j}(t) \tag{8}$$

for $\alpha \in \mathbb{R}$, which is specified such that if the agent's reflexivity $\rho^i \leq 50\%$, we have $\alpha = 0, \rho^i, 2\rho^i$ for $a_2^{\mathcal{F}} = 0, 1, 2$, respectively, and if the agent's reflexivity $\rho^i > 50\%$, we have $\alpha = 2\rho^i - 1, \rho^i, 1$ for $a_2^{\mathcal{F}} = 0, 1, 2$, respectively. Hence $a_2^{\mathcal{F}} = 2$ allows the agent to learn and gauge the weight it gives to its own chartist vs. fundamentalist pricing.

Returns $\mathcal{R}^{\mathcal{F}}$: Then, the reinforcement learning algorithm $\mathcal{F}^i$ computes the percentage difference between the agent's former stock price prediction $H^{i,j}(t-\tau^i)$ performed τ^i time steps ago, and its present realization $P^j(t)$:

$$\frac{|H^{i,j}(t-\tau^i) - P^j(t)|}{P^j(t)} \tag{9}$$

recording it at each time step in a time series that is continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The associated percentile corresponding to this value at time step t sets a discrete value of returns $r^{\mathcal{F}}$ in the set $\mathcal{R}^{\mathcal{F}} = \{4, 2, 1, -1, -2, -4\}$ if it respectively belongs to the intervals $[0\%, 5\%(, [5\%, 25\%(, [25\%, 50\%(, [50\%, 75\%(, [75\%, 95\%(, [95\%, 100\%]$.

Policy $\pi^{\mathcal{F}}$: Finally, the reinforcement learning algorithm updates its policy $\pi_t^{\mathcal{F}}(s_{t-\tau^i}^{\mathcal{F}}, a_{t-\tau^i}^{\mathcal{F}})$, at each time step t. This is done according to the agent's own learning rate β, with the equations below being iterated a number $|r^{\mathcal{F}}|$ of times, in order to favour any action that is deemed optimal $a^{\mathcal{F}\star}$ in state $s^{\mathcal{F}}$, by increasing the policy probability associated with this action, compared to the other actions, $\forall a^{\mathcal{F}} \neq a^{\mathcal{F}\star}$:

$$\pi_{t+1}^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}\star}) = \pi_t^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}\star}) + \beta[1 - \pi_t^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}\star})] \tag{10}$$

$$\pi_{t+1}^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}}) = \pi_t^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}}) + \beta[0 - \pi_t^{\mathcal{F}}(s^{\mathcal{F}}, a^{\mathcal{F}})] \tag{11}$$

Added to this, the algorithm uses an off-policy method every $\tau^i/T_m + 2$ time steps, which computes the optimal action that $\mathcal{F}^i$ should have performed τ^i time steps ago now that the price is realised and the forecast accuracy known, and which accordingly updates the policy $\pi^{\mathcal{F}}$ with the agent's own learning rate β, iterated $|r^{\mathcal{F}}| = 4$ times (for the associated action is deemed optimal).

3.1.4 Agent reinforcement learning: second algorithm $\mathcal{T}^i$

Via this second algorithm, the agent continuously monitors whether the stock prices are increasing or decreasing according to former algorithm ($s_0^{\mathcal{T}}$), their volatility ($s_1^{\mathcal{T}}$), its risk-free assets ($s_2^{\mathcal{T}}$), its quantity of stock holdings ($s_3^{\mathcal{T}}$), and the traded volumes ($s_4^{\mathcal{T}}$). Out of this state, it learns to optimize its investments by exploring and exploiting two possible actions via a direct policy search: sending a transaction order to the order book as holding, buying, or selling a

position in a given amount ($a_0^{\mathcal{T}}$), and at what price wrt. the law of supply and demand ($a_1^{\mathcal{T}}$).

States $\mathcal{S}^{\mathcal{T}}$: The second reinforcement learning algorithm $\mathcal{T}^i$ is defined with a state $s^{\mathcal{T}}$ in a state set $\mathcal{S}^{\mathcal{T}} = \{s_0^{\mathcal{T}}, s_1^{\mathcal{T}}, s_2^{\mathcal{T}}, s_3^{\mathcal{T}}, s_4^{\mathcal{T}}\}$, of dimension 108, with $s_0^{\mathcal{T}} = s_1^{\mathcal{T}} = s_4^{\mathcal{T}} = \{0, 1, 2\}$ and $s_2^{\mathcal{T}} = s_3^{\mathcal{T}} = \{0, 1\}$.

- Each agent computes the value $\mu = (H^{i,j}(t) - P^j(t))/P^j(t)$ and records it at each time step in two distinct time series $\boldsymbol{\mu}_-$ and $\boldsymbol{\mu}_+$, depending on whether it is negative or positive, respectively. These two time series are continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The percentile of its present value $\overline{\mu}_-$ at time step t in $\boldsymbol{\mu}_-$ sets $s_0^{\mathcal{T}} = 0$ if it is below 95%, and $s_0^{\mathcal{T}} = 1$ otherwise. The percentile of its present value $\overline{\mu}_+$ at time step t in $\boldsymbol{\mu}_+$ sets $s_0^{\mathcal{T}} = 1$ if it is below 5%, and $s_0^{\mathcal{T}} = 2$ otherwise. Therefore, $s_0^{\mathcal{T}} = 0, 1, 2$ if the econometric forecast μ derived from the previous algorithm $\mathcal{F}^i$ is respectively indicating a decrease, approximate stability, or increase in the price of stock j in τ^i future time steps.
- Each agent records the previously computed variance σ_L^2 of the prices $P^j(t)$ over the interval $[t-3\tau^i, t]$ at each time step, in a time series that is continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The percentile of its present value at time step t sets $s_1^{\mathcal{T}} = 0$ if it is below 33%, $s_1^{\mathcal{T}} = 2$ if it is above 67%, and $s_1^{\mathcal{T}} = 1$ otherwise. $s_1^{\mathcal{T}}$ thus gives a measure of the longer-term volatility in stock prices to the agent.
- Each agent sets $s_2^{\mathcal{T}} = 0$ if the value of its risk-free assets $A^i_{\text{bonds}}(t)$ at time step t is below 60% of its start value $A^i_{\text{bonds}}(t = 0)$, and $s_2^{\mathcal{T}} = 1$ otherwise. Hence each agent likewise continually monitors the size of its risk-free assets in order to adopt the appropriate long or short strategy.
- Each agent sets $s_3^{\mathcal{T}} = 0$ if the value of its stock holdings $A^i_{\text{equity}}(t)$ at time step t is below 60% of its start value $A^i_{\text{equity}}(t = 0)$, and $s_3^{\mathcal{T}} = 1$ otherwise. Hence each agent continually monitors the size of its stock holdings in order to adopt the appropriate long or short strategy.
- Each agent records the trading volume $V^j(t)$ at each time step in a time series that is continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The percentile of its present value at time step t sets $s_4^{\mathcal{T}} = 0$ if $V^j(t) = 0$, $s_4^{\mathcal{T}} = 1$ if it is below 33%, $s_4^{\mathcal{T}} = 2$ otherwise. This gives the agent a sense of market activity that is useful for determining the appropriate bid or ask price to send to the order book.

Actions $\mathcal{A}^{\mathcal{T}}$: Then, the reinforcement learning algorithm $\mathcal{T}^i$ is defined with an action $a^{\mathcal{T}}$ in an action set $\mathcal{A}^{\mathcal{T}} = \{a_0^{\mathcal{T}}, a_1^{\mathcal{T}}\}$, of dimension 9. Here $a_0^{\mathcal{T}}$ and $a_1^{\mathcal{T}}$ both can take the discrete values $\{0, 1, 2\}$, chosen according to a direct policy search (see below). The first action $a_0^{\mathcal{T}}$ corresponds both to the quantity of stocks and the nature of the transaction order (sell, hold, or buy) that the agent chooses to send to the order book. Hence each agent has a long-only

equity strategy: the agent buys at a given price a number of stocks, holds them for a given time, and then sells them, hopefully at a higher price. The second action $a_1^{\mathcal{T}}$ corresponds to the flexibility of the agent with regards to the price at which it is willing to trade. As said, these two actions depend on the evaluation of the price of stock j that was performed by the agent through the first algorithm $\mathcal{F}^i$. First, the agent bid price $P_{\text{bid}}^{i,j}(t)$ is set at:

$$P_{\text{bid}}^{i,j}(t) = \min[H^{i,j}(t), P^j(t)] + g^i W^j(t-1) \tag{12}$$

$$P_{\text{bid}}^{i,j}(t) = \min[H^{i,j}(t), P^j(t)] \tag{13}$$

$$P_{\text{bid}}^{i,j}(t) = \min[H^{i,j}(t), P^j(t)] - g^i W^j(t-1) \tag{14}$$

if $a_1^{\mathcal{T}} = 0, 1, 2$ respectively. Here we recall that g^i is the agent's transaction gesture and $W^j(t-1)$ our market spread of stock j at former time step. The term $\pm g^i W^j(t-1)$ hence specifies the agent's softer or harder stance on the transaction deal, depending on general market conditions of supply and demand, like $W^j(t-1)$ and the traded volumes specified by $s_4^{\mathcal{T}}$. The agent ask price $P_{\text{ask}}^{i,j}(t)$ is set at:

$$P_{\text{ask}}^{i,j}(t) = \max[H^{i,j}(t), P^j(t)] - g^i W^j(t-1) \tag{15}$$

$$P_{\text{ask}}^{i,j}(t) = \max[H^{i,j}(t), P^j(t)] \tag{16}$$

$$P_{\text{ask}}^{i,j}(t) = \max[H^{i,j}(t), P^j(t)] + g^i W^j(t-1) \tag{17}$$

if $a_1^{\mathcal{T}} = 0, 1, 2$ respectively. Then for $Q^{i,j}(t)$ the quantity of stocks j held by agent i at time t, action $a_0^{\mathcal{T}} = 0$ corresponds to a sell order of a quantity $Q^{i,j}(t)$ of stocks j for an ask price of $P_{\text{ask}}^{i,j}(t)$, action $a_0^{\mathcal{T}} = 1$ corresponds to no order being sent to the order book (the agent simply holds its position), and $a_0^{\mathcal{T}} = 2$ corresponds to a buy order of a floored quantity $A_{\text{bonds}}^i(t)/[P_{\text{ask}}^{i,j}(t)J]$ of stocks j for a bid price of $P_{\text{bid}}^{i,j}(t)$. Notice the parameter J is here part of the denominator of this stock quantity to buy, so as to ensure a proper multivariate portfolio management.

Filter $\mathcal{G}^i$: As mentioned earlier, the quantity and price of stock j that agent i sends to the order book at this time step t is conditional on the output of function $\mathcal{G}^i$, that ensures the agent waits and sends a transaction order at the optimal time step and not before. In order to do this, $\mathcal{G}^i$ records in a time series at each time step the value of the action-value function $\arg\max_a \mathcal{Q}_t(s, a)$ maximized by action a. It then sorts this time series in ascending order, and compares its associated percentile $p_{\mathcal{Q}}(t)$ with the ratio of the elapsed time since previous transaction $k^{i,j}(t)$ over the agent's individual trading window w^i. The filter function $\mathcal{G}^i$ lets the trading order chosen by $\mathcal{T}^i$ be sent to the order book only if $p_{\mathcal{Q}}(t) < k^{i,j}(t)/w^i$. Notice this function $\mathcal{G}^i$ thus filters entry

but not exit strategies: the latter are always enacted at the agent's investment horizon τ^i.

Returns $\mathcal{R}^{\mathcal{T}}$: Considering the present realization of stock price $P^j(t)$, the algorithm $\mathcal{T}^i$ then computes the cashflow difference between the present agent's portfolio net asset value, and its present value if the former actions taken τ^i time steps ago had not been taken:

$$Q^{i,j}_{\mathrm{OB}}(t-\tau^i)[P^j(t) - P^{i,j}_{\mathrm{OB}}(t-\tau^i)] \tag{18}$$

where $Q^{i,j}_{\mathrm{OB}}(t-\tau^i)$ and $P^{i,j}_{\mathrm{OB}}(t-\tau^i)$ are respectively the quantity and transaction price of stock j that was cleared by the order book process at time $t-\tau^i$ for agent i and its transaction partner. Notice these may not be those actually sent by agent i at that time, because the quantity of stocks to long or short may not have been entirely cleared at this time (recall the agents send limit orders only), and because the transaction price is set by the order book at mid-price with the transaction partner's order price. These values are then recorded at each time step in a time series that is continually sorted in ascending order and truncated to keep a size corresponding to agent memory interval h^i. The associated percentile corresponding to this value at time step t sets a discrete value of returns $r^{\mathcal{T}}$ in the set $\mathcal{R}^{\mathcal{T}} = \{4, 2, 1, -1, -2, -4\}$ if it respectively belongs to the intervals $[0\%, 5\%($, $[5\%, 25\%($, $[25\%, 50\%($, $[50\%, 75\%($, $[75\%, 95\%($, $[95\%, 100\%]$.

Policy $\pi^{\mathcal{T}}$: Finally, the reinforcement learning algorithm updates its policy $\pi^{\mathcal{T}}_t(s^{\mathcal{T}}_{t-\tau^i}, a^{\mathcal{T}}_{t-\tau^i})$, every τ^i time steps following each transaction dealt by the agent. This is done according to the agent's own learning rate β, with the equations below being iterated a number $|r^{\mathcal{T}}|$ of times, in order to favour any action that is deemed optimal $a^{\mathcal{T}\star}$ in state $s^{\mathcal{T}}$, by increasing the policy probability associated with this action, compared to the other actions, $\forall a^{\mathcal{T}} \neq a^{\mathcal{T}\star}$:

$$\pi^{\mathcal{T}}_{t+1}(s^{\mathcal{T}}, a^{\mathcal{T}\star}) = \pi^{\mathcal{T}}_t(s^{\mathcal{T}}, a^{\mathcal{T}\star}) + \beta[1 - \pi^{\mathcal{T}}_t(s^{\mathcal{T}}, a^{\mathcal{T}\star})] \tag{19}$$

$$\pi^{\mathcal{T}}_{t+1}(s^{\mathcal{T}}, a^{\mathcal{T}}) = \pi^{\mathcal{T}}_t(s^{\mathcal{T}}, a^{\mathcal{T}}) + \beta[0 - \pi^{\mathcal{T}}_t(s^{\mathcal{T}}, a^{\mathcal{T}})] \tag{20}$$

Added to this, the algorithm uses an off-policy method every $\tau^i/T_m + 2$ time steps, which computes the optimal action that $\mathcal{T}^i$ should have performed τ^i time steps ago now that the price is realised and the forecast accuracy known, and which accordingly updates the policy $\pi^{\mathcal{T}}$ with the agent's own learning rate β, iterated $|r^{\mathcal{T}}| = 4$ times (for the associated action is deemed optimal).

As one can see, the action-state spaces of these two algorithms $\mathcal{F}$ and $\mathcal{T}$ are highly discretised and handcrafted. The main reason is that such structured state space limits the necessary computational ressources and mitigates one of the central limitations for MAS applied to financial research, namely the requirement for large computational power. Furthermore, the general intuition behind our definition of such state and action spaces has been the *Fundamental*

Theorem of Asset Pricing [26], where present asset prices are estimated from time-discounted future prices expectations: likewise, our agent reinforcement learning framework has been articulated around a forecasting part $\mathcal{F}^i$ and a trading part $\mathcal{T}^i$, in a way not that dissimilar from other recent models such as [98] (cf. Section 3.1.3).

3.2 Calibration to real data

Model hypotheses : The main hypotheses for this model are that: i- the simulated agents faithfully emulate real-world investors, and ii- the simulated transaction limit orders processed by the order book correspond to the dynamics and properties of real stock market orders. Concerning the former, the strength of our approach is the simplification that any agent, regardless of its behaviour and strategy, is structurally bound to interact with the market in three possible ways only, namely by longing, shorting or holding stocks (long-only strategy). Concerning the latter, we recall that the dynamics of order books are extensively documented in literature [50], and hence that their design can be rigorously conducted.

Model limitations : Along with these two major hypotheses, we shall also mention the following model limitations and consistency issues, which are proper to all financial MAS: i- reliance on virtual fundamental generation $\mathcal{T}^j(t)$, ii- absence of portfolio diversification (equity, cross-asset), iii- absence of various trading strategies (e.g. short-selling, leveraging, derivatives, metaorders, market orders, etc.), iv- absence of intraday and seasonal market effects, v- absence of legal and regulatory constraints. Although some of these limitations may seem challenging, their effects and importance impact virtually any other econometric or modelling approach in quantitative finance. Added to this, modelling market activity via a market microstructure emerging from a centralised order book that processes transaction orders sent by many trading agents, has a direct empirical correspondance with real stock markets, and is thus fully epistemologically pertinent.

Training and testing sets : We calibrated the MAS stock market simulator to real stock market data [1]. In order to do so, we used high quality, industry-grade, daily close prices and volumes of $4,313$ stocks from the London Stock Exchange (LSE), between January 15th of 2007 and January 19th of 2018. In order to work on the market microstructure, we have filtered the data as such: i- stock-splits effects have been suppressed, ii- only stocks that have been continuously traded over this time period have been considered. As a consequence of this data curation, our former stock universe has been lowered to 640 stocks. The calibration of the MAS hyperparameters has been conducted on a random

[1] Computations were performed on a Mac Pro with 3,5 GHz 6-Core Intel Xeon E5 processor, and 16 GB 1866 MHz DDR memory

sampling of half of these stocks as training set. We found a remarkable invariance of its main statistical features, when compared to that of the other half. We posit this statistical stability, proper to stock market data, arises from the lack of market arbitrage and relates to the aforementioned stylised facts.

Hyperparameter optimisation : The calibrated hyperparameters are the number of agents I, the agent transaction gesture factor $\zeta^i \in \mathbb{N}$ (which scales the gesture parameter g^i initialised for each agent at time $t = 0$), the fundamental values generation parameter ν (which is the amplitude of the fundamental time series $\mathcal{T}^j$), and the drawdown threshold (which is the higher bound of the drawdown limit initialised at time $t = 0$ for each agent). All hyperparameter combinations were compared with the training set, and are listed on Tab. 1. As one can see from the boundaries and iteration steps of the optimization process, $6 \times 5 \times 8 \times 5 = 1200$ simulations were hence conducted, each simulation consisting of $S = 20$ runs, for statistical consistency.

Table 1 Model hyperparameters, with their intervals for training : lower bound (Low), higher bound (High), and step of incrementation (Step).

Hyperparameter	**Low**	**High**	**Step**
Agent number I	500	5500	1000
Gesture scalar ζ^i	1.0	3.0	0.5
Fundamental amplitude ν	0.1	1.5	0.2
Drawdown treshold $\mathcal{L}$	10	90	20

Sensitivity analysis : During optimisation, such a hyperparameter space has been used to monitor the sensitivity of the model to certain hyperparameter ranges, and especially to highlight areas of non-linearity wrt. calibration to the real data. The sensitivity of the model to the number of agents I is rather linear in the sense that larger values of I gradually diminish short-scale price volatilities. Larger gesture scalar ζ^i and fundamental amplitude ν produce a linear growth in absolute daily price returns. Finally, the model is relatively unaffected by large drawdown thresholds $\mathcal{L} > 30\%$, as these values have a low impact on agent survivability rates.

Model comparison : There is an extensive literature wrt. to [42] on the substitution of markets to individual rationality, namely whether markets eliminate irrational individuals, or whether individuals learn market rules. Fig. 3 shows agent learning curves that can be used for model comparison, notably with recent order book models coupled with reinforcement learning [98], and the former generation of MAS with zero-intelligence agents [42] as baselines.

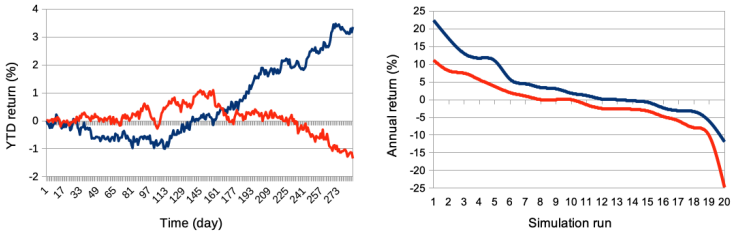

Fig. 3 After 90% of total simulation time, we want to compare the best 10% agents of our MAS stock market simulator (blue curves) with the best 10% agents of a market simulated with noise agents trading randomly (red curves). For this, we check on their performance over the remainder 10% of our total simulation, with averaged equity curves as their year-to-date returns over $S = 20$ simulations (left), and averaged sorted annual returns of each S simulations (right). These simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$.

4 Results

4.1 Statistical features

We show on Fig. 4 to 15, different key market microstructure indicators pertaining to the calibration of the MAS simulator. As one can see, the curves show a qualitative agreement in shape with those from real stock market data, and can be considered state-of-the-art wrt. to agent-based stock market emulation [16,44,88]. This also shows the usefulness of reinforcement learning as a framework for the description of agent learning and trading process in stock markets. Unless otherwise stated, the results discussed below are ran for simulations generated with $I = 500$ agents, $J = 1$ stock being traded, $T = 2875$ time steps during each simulation (accounting for about 11 years of trading days), and $S = 20$ simulation runs.

- Fig. 4 shows the distribution of logarithmic returns of prices $\log[P(t)/P(t-1)]$, for real (dashed black curve) and simulated (continuous red curve) data. One can see a close match between simulated and real logarithmic price returns, but notice the small variability of extreme events found in the distribution tails, revealed by the logarithmic y-axis.
- On Fig. 5, one can see the distributions of the price volatilities over several intervals, namely over two weeks (black), three months (red), and one year (blue), for both real (dashed curves) and simulated (continuous curves) data. These volatilities are defined as standard deviations of prices normalised to the price itself $\sigma/P(t)$. We find that the real volatilities at larger time scales are harder to emulate, and this is most probably because our real data sample covers a rather unique and unusual market regime over the years $2008 - 2009$, namely the financial crisis.
- On Fig. 6, we show the distributions of the correlations of the price logarithmic returns between separated intervals $[t-\Delta,\ t]$ and $[t-2\Delta,\ t-\Delta]$ at each time step t, over values Δ of two weeks (black), three months (red),

and one year (blue). This is for both real (dashed curves) and simulated (continuous curves) data. Despite the great fits, in particular wrt. to the general shape of the distributions, one could wonder as to why real data displays that many zero-autocorrelations; we posit this could be due to particular intraday market activity, or to the absence of trading volumes over long periods of time for stocks of small capitalisation companies.

- In a similar way, we see the simulated data (continuous curves) emulating the real data (dashed curves) in the asymmetric shape of the distribution of the price volatility correlations between separated intervals $[t-\Delta,\ t]$ and $[t-2\Delta,\ t-\Delta]$ for $\Delta = 2T_w$ at each time step t in Fig. 7, and likewise in the shapes of the distributions of the trading volumes correlations between these same separated intervals, for values Δ of two weeks (black), three months (red), and one year (blue) in Fig. 8, with a difference in zero values.
- On Fig. 9, we show the distributions of the correlations in price logarithmic returns at each time step t for the simulated data (continuous curves) and the real data (dashed curves), between the blended intervals $[t-T_w,\ t]$ and $[t-T_w-\delta,\ t-\delta]$, for shifts δ of one day (black), two days (red), three days (blue), four days (green), and five days (yellow), with a similar difference in zero values.
- On Fig. 10, we see the real (blue) and simulated (red) means of blended correlations of the logarithmic returns of prices at each time step t between intervals $[t-T_w,\ t]$ and $[t-T_w-\delta,\ t-\delta]$, for shifts $\delta = 1, 2, 3, 4, 5$. Similarly, Fig. 11 shows close fits for larger intervals of $2T_w$ instead of T_w. This is an important model statistic wrt. the fact that the MAS produces a price microstructure showing cancellation of arbitrage opportunities and market memory, through agent learning. In other words, the agents learn to exploit short-term causality structures in the historical price.
- On Fig. 12, we show the distribution of the number of consecutive days of increasing prices (positive values) and decreasing prices (negative values) at each time step t, for both simulated (continuous curve) and real (dashed curve) data. The number of consecutive days of increasing or decreasing prices is a useful indicator of market regime, or whether the market is "bearish" or "bullish." We find that apart from a few extreme bullish events, the MAS simulates general stock market price dynamics in a way corresponding to real data.

In conclusion, our results show that the simulator is able to faithfully account for the distribution of logarithmic price returns in Fig. 4 and their autocorrelations at different timescales in Fig. 6, 9, 10, 11. These latter autocorrelation metrics are key in the calibration process, because they pertain to the absence of arbitrage and market memory, which are central features of financial markets. ~~Or~~ in other words, beyond the stylised facts, the simulated data should not display price patterns that are more easily identifiable and ready to be exploited by trading than those found in the real data, if any. Finally, we can also underline how the MAS simulator faithfully emulates real stock market regimes of recession and growth, as shown in Fig. 12. This said,

we take note of several avenues for improvement that become apparent for the simulator performance and properties:

- The extremity of the tail distribution of long-term price volatilities in Fig. 5: these are indeed the hardest microstructure effects to capture, as they relate to jump diffusion processes proper to volatile events in the life of a company, industry sector, or full market (here we should mention this LSE data encompasses the financial crisis of $2008 - 2009$).
- The peak in zero autocorrelations for real price returns and volatility in Fig. 6-7: we posit this to be due to the fact that unlike real data, the simulator does not capture intraday market activity, or to the presence of scarcely traded small cap companies.
- Fatter tails in distributions of autocorrelation of trading volumes in Fig. 8: we posit this to be due to seasonal and calendar effects proper to real stock markets.

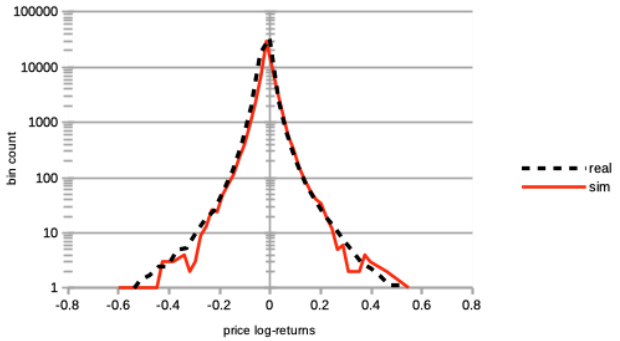

Fig. 4 Distribution of logarithmic returns of prices $\log[P(t)/P(t-1)]$ of real (dashed black curve) and simulated (continuous red curve) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

4.2 Classification of main statistics

Random forest classification: In order to evaluate how well the model-generated log-return signals match the real market stock dynamics, we aimed to build a binary classification model which could take a fixed-size time series sample, and predict whether the sample comes from simulated or real stock market data. The whole dataset was split into non-overlapping training and validation subsets. Then, the time series samples of fixed size (the number of timestamps ranging from 5 to 50, but fixed for each experiment) were generated by applying a sliding window to the full price logarithmic-return time series from both simulated and real data. Both training and validation subsets were constructed

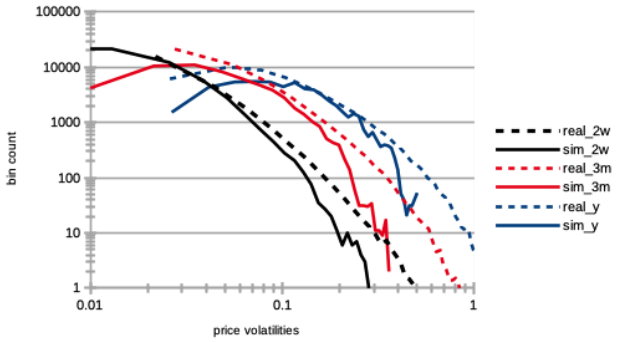

Fig. 5 Distribution of volatilities (defined as standard deviations of price normalised to price itself $\sigma/P(t)$) computed over lags of two weeks (black), three months (red), and one year (blue) intervals for both real (dashed curves) and simulated (continuous curves) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

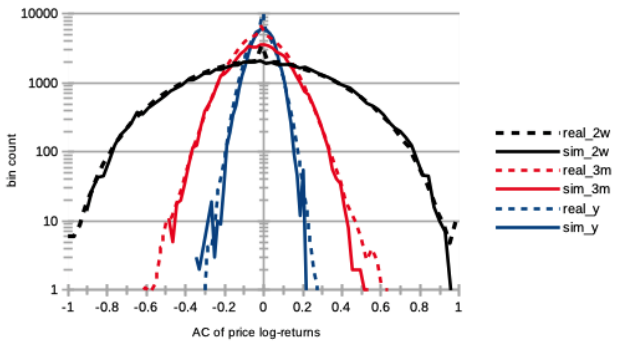

Fig. 6 Distribution of autocorrelations of the logarithmic returns of prices at each time step t between intervals $[t - \Delta,\ t]$ and $[t - 2\Delta,\ t - \Delta]$, over lags Δ of two weeks (black), three months (red), and one year (blue) intervals for both real (dashed curves) and simulated (continuous curves) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

so that they were balanced in terms of class distribution. Before training a classifier on the collected data, a preprocessing step was taken to map the time series samples to a vector feature space by computing a set of 787 predefined signal properties for each sample [2]. A standard scaling procedure was then applied to the dataset with mean and variance statistics estimated from the training subset. The low-variance features were discarded as an additional filtering step. This normalised feature-vector representation of the dataset was then used to train random forest classification models [3]. Accuracy scores were

[2] We used the time series feature extraction functions implemented in the *tsfresh* Python package [21]

[3] We used the implementation from the *scikit-learn* Python package [84], with 200 estimators, maximal tree depth equal to 5 and default values for other hyperparameters.

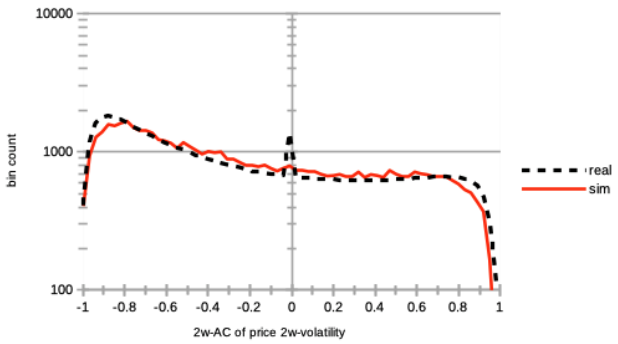

Fig. 7 Distribution of autocorrelations of two weeks-interval volatilities at each time step t between intervals $[t - \Delta,\ t]$ and $[t - 2\Delta,\ t - \Delta]$ for $\Delta = 2T_w$, for both real (dashed black curve) and simulated (continuous red curve) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

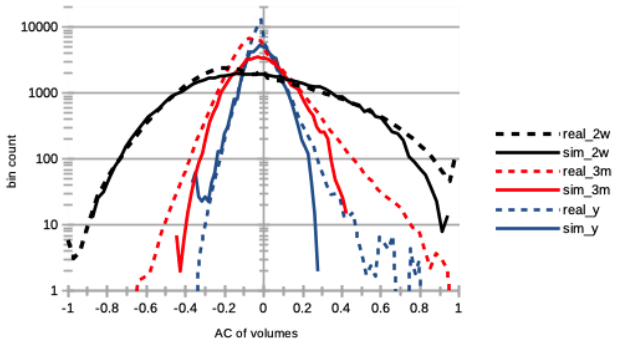

Fig. 8 Distribution of autocorrelations of the trading volumes at each time step t between intervals $[t - \Delta,\ t]$ and $[t - 2\Delta,\ t - \Delta]$, over lags Δ of two weeks (black), three months (red), and one year (blue) intervals for both real (dashed curves) and simulated (continuous curves) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

measured for multiple runs of model training (with different splits of the training/testing subsets, $N = 50$ runs total for each experiment) and for the different number of timestamps initially taken for each sample. The obtained accuracy score value distributions are shown in Fig. 13 (see more details on Fig. 16, in the Supplementary Material section 6). The accuracy reported for the "value distribution" feature set is the accuracy of classifiers trained on a subset of all 787 features available in $tsfresh$ only pertaining to the properties of the in-sample value distributions. These features include distribution statistics which do not depend on the order of values in the time series like mean, median, variance, kurtosis, quantile values and so on. The accuracy for "all time-series" feature set in Fig. 13 corresponds to the full set of features of $tsfresh$ being used to generate vector embeddings for the time series

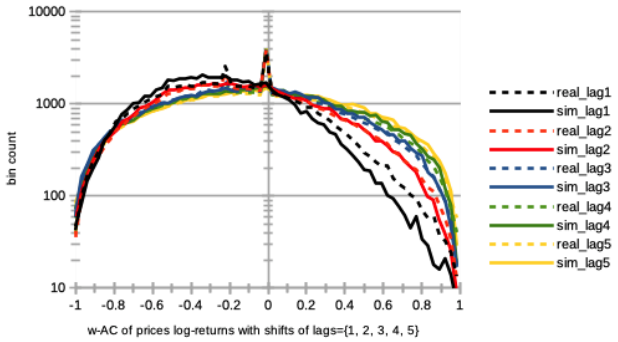

Fig. 9 Distribution of autocorrelations of the logarithmic returns of prices at each time step t between intervals $[t - T_w,\ t]$ and $[t - T_w - \delta,\ t - \delta]$, for shifts δ of one day (black), two days (red), three days (blue), four days (green), and five days (yellow). This is for both real (dashed curves) and simulated (continuous curves) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

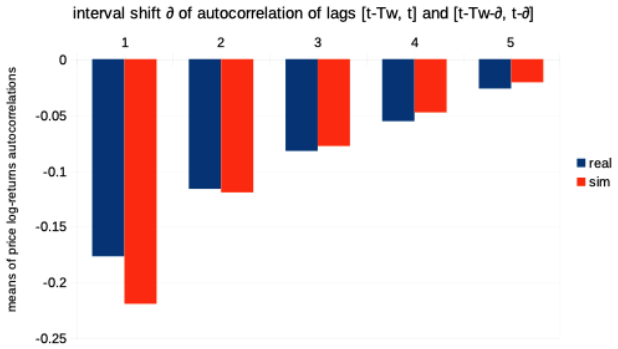

Fig. 10 Means of autocorrelations of the logarithmic returns of prices at each time step t between intervals $[t - T_w,\ t]$ and $[t - T_w - \partial,\ t - \partial]$, for shifts $\partial = [1, 2, 3, 4, 5]$. This is for both real (blue) and simulated (red) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

samples. Low "value distribution"-based classification accuracy achieved for random forest models signifies that the value distibution of log-return series is well simulated in our case. The temporal structure of the log-return series, as encoded by the corresponding time series features, is also modelled fairly well (see Fig. 13).

Dimensionality reduction: Another useful metric that was extracted from the trained random forest classifiers is the measure of feature importance that was averaged over multiple training runs ($N = 20$ total runs with different training/testing subset splits). This procedure allowed us to determine a subset of features which turn out as the most discriminative for the defined binary classification task. The features which were found in the top-20 importance score list could be grouped into several categories: i- value distribution

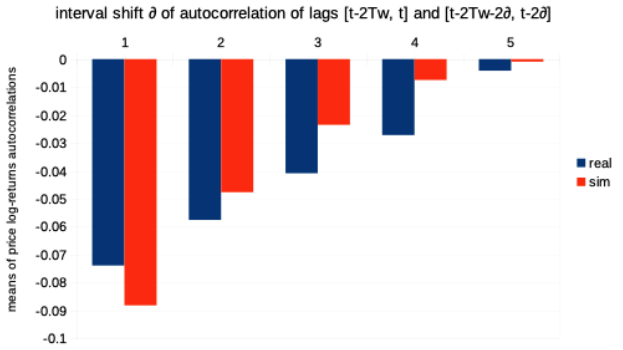

Fig. 11 Means of autocorrelations of the logarithmic returns of prices at each time step t between intervals $[t-2T_w,\ t]$ and $[t-2T_w-\partial,\ t-\partial]$, for shifts $\partial = [2, 4, 6, 8, 10]$. This is for both real (blue) and simulated (red) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

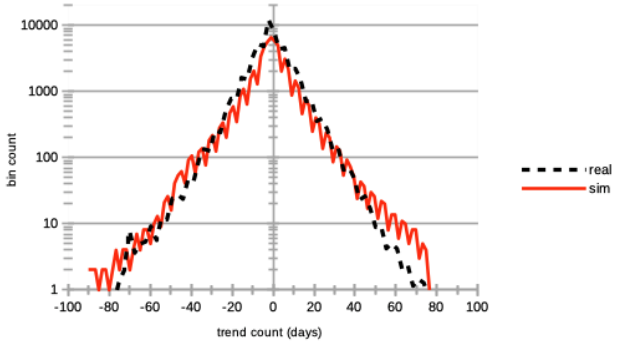

Fig. 12 Distribution of the number of consecutive days of increasing prices (positive values) and decreasing prices (negative values). This is for both real (dashed black curve) and simulated (continuous red curve) data. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

properties (e.g. kurtosis, skewness, 60th, 70th, 80th percentiles of the value distribution within the sample), ii- fast Fourier transform (FFT) spectrum statistics (centroid, skew and kurtosis of the fast Fourier transform coefficient distributions; absolute values of the first two FFT coefficients), iii- features related to the temporal structure of the series, e.g. autocorrelation and coefficients of a fitted autoregressive $AR(k)$ process. Such features are discussed in more details in the Supplementary Material section 6. In order to visualise the distribution of classes across time series samples in the feature-vector space, we applied several dimensionality reduction algorithms to the dataset, results of which are shown in Fig. 14 and 15. The first algorithm is generic principal component analysis (PCA), which was applied to the reduced set of the top-10 features, ranked by feature importance scores, as seen on Fig. 14. One can notice that the produced linear mapping to the 2-dimensional

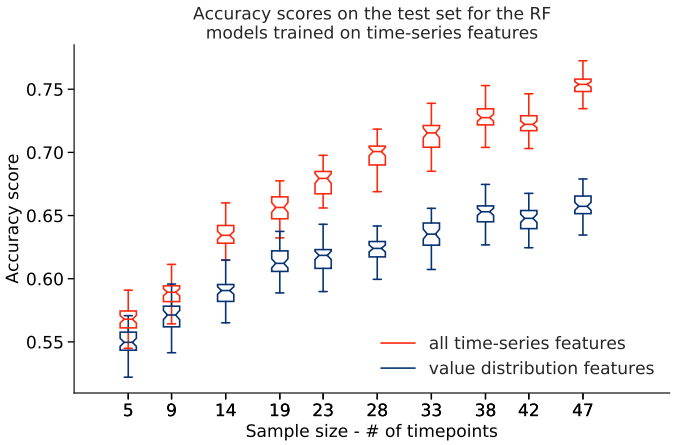

Fig. 13 Accuracy as a function of time-series sample size. The value distribution features include only features that do not depend on the order of values in the time-series (e.g. mean, median, variance, kurtosis, skewness of the value distribution, etc.), whereas all time-series features correspond to the total set of time-series features including those that depend on the temporal structure of the series (e.g. autocorrelation, entropy, FFT coefficients, etc.). The testing subsets are balanced in terms of class distribution. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

space does not guarantee high separability of point clouds corresponding to different classes (simulated vs. real data). This means that a linear combination of basic time-series features is not sufficient to effectively discriminate simulated data from the real signals. Once we applied a supervised nonlinear mapping, namely Uniform Manifold APproximation (UMAP) [73], visual class separability between point clouds increased. We then measured the point cloud separability for both PCA and UMAP embeddings by training a logistic regression classification model on the two dimensional representations of the samples. We did not find a significant improvement in class separability for UMAP vs. PCA (classification accuracy of 64% vs. 62%, respectively), with classification accuracy being low for both embedding methods.

In principle, effective task-specific time series features could be learned if a suitable deep neural network model for time series was used in the classification task (e.g. a deep 1D convolutional neural network, 1D-CNN). These considerations open further directions for model tuning/calibration, for instance by imposing a penalty term which would depend on classification accuracy achieved by a CNN classifier (akin to a generative adversarial model setup [43]) in order to minimise it.

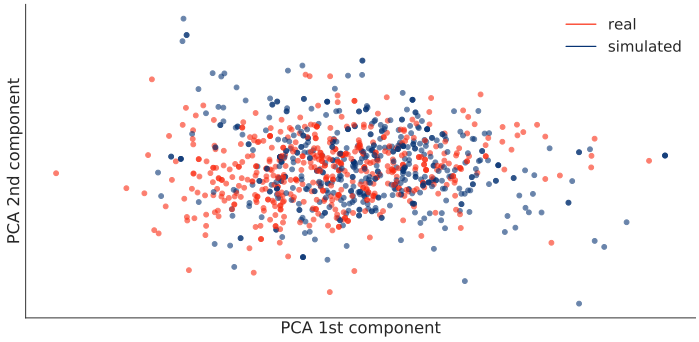

Fig. 14 Principal component analysis scatter plot on top 10 statistical features of real (red) and simulation (blue) data. PCA mapping was fitted on 6000 time-series samples each containing 50 timestamps. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

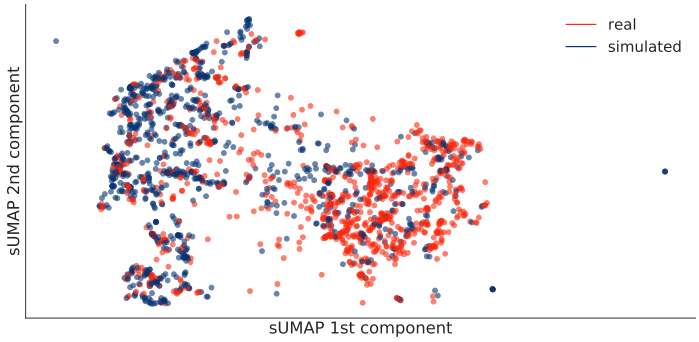

Fig. 15 Supervised UMAP trained on 3000 samples for top 10 statistical features of real (red) and simulation (blue) data, and applied to another 3000 samples (each sample containing 50 timestamps). The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

5 Conclusion

We ~~thus~~ modelled a stock market via an intelligent MAS, where the agents autonomously perform portfolio management via long-only equity strategies, based on autonomous reinforcement learning algorithms performing price forecasting and stock trading. In ~~such a~~ this model, each agent also learns to gauge how fundamentalist or chartist it will be in its approach to price estimation. We have calibrated this MAS simulator to real stock market data from the London Stock Exchange between the years 2007 and 2018, and achieved state-of-the-art performance in price microstructure emulation [16,44,88]. In particular, it emulates key market statistics of real stock exchanges wrt. price returns, volatilities at different time-scales, diverse auto-correlation metrics and market regimes. We posit this model could be a powerful tool for both financial industry and academic research, to further explore stock market microstruc-

ture and price formation by a bottom-up approach. Also, we posit that the reinforcement learning framework of the agents could be used to implement psychological traits of decision theory and behavioural economics, and hence to study the influence of agent learning and cognition on financial markets at the macroscale. Finally, we also see the following immediate and natural extensions of our model: i- using the multivariate feature of the agents trading coupled with portfolio risk management in order to study and simulate covariance structure across several stocks, and ii- extension of the agent reinforcement learning framework to the continuous domain (cf. policy gradients, deep Q-learning, etc.).

References

1. Current dividend impacts of FTSE-250 stocks. URL `https://www.dividenddata.co.uk`. Accessed: 2020-05-19
2. IG fees of Contracts For Difference. URL `https://www.ig.com`. Accessed: 2020-05-19
3. UK one-year gilt reference prices. URL `https://www.dmo.gov.uk`. Accessed: 2020-05-19
4. Abbeel, P., Coates, A., Ng, A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. The International Journal of Robotics Research (2010)
5. Aloud, M.: Agent-based simulation in finance: design and choices. Proceedings in Finance and Risk Perspectives '14 (2014)
6. Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. International Conference on Machine Learning (2017)
7. Bak, P., Norrelykke, S., Shubik, M.: Dynamics of money. Physical Review E **60**, 2528–2532 (1999)
8. Bak, P., Norrelykke, S., Shubik, M.: Money and goldstone modes. Quantitative Finance **1**, 186–190 (2001)
9. Barde, S.: A practical, universal, information criterion over nth order markov processes. University of Kent, School of Economics Discussion Papers **04** (2015)
10. Bavard, S., Lebreton, M., Khamassi, M., Coricelli, G., Palminteri, S.: Reference-point centering and range-adaptation enhance human reinforcement learning at the cost of irrational preferences. Nature Communications **4503** (2018)
11. Benzaquen, M., Bouchaud, J.P.: A fractional reaction–diffusion description of supply and demand. The European Physical Journal B **91(23)** (2018)
12. Bera, A.K., Ivliev, S., Lillo, F.: Financial Econometrics and Empirical Market Microstructure. Springer (2015)
13. Bhatnagara, S., Panigrahi, J.R.: Actor-critic algorithms for hierarchical decision processes. Automatica **42** (2006)
14. Biondo, A.E.: Learning to forecast, risk aversion, and microstructural aspects of financial stability. Economics **12(2018–20)**, 1–21 (2018)
15. Biondo, A.E.: Order book microstructure and policies for financial stability. Stud Econ Finance **35(1)**, 196–218 (2018)
16. Biondo, A.E.: Order book modeling and financial stability. Journal of Economic Interaction and Coordination **14(3)** (2018)
17. Boero, R., Morini, M., Sonnessa, M., Terna, P.: Agent-based models of the economy, from theories to applications. Palgrave Macmillan (2015)
18. Bouchaud, J., Cont, R., Potters, M.: Scale Invariance and Beyond, Proc. CNRS Workshop on Scale Invariance, Les Houches. Springer (1997)
19. Bouchaud, J.P.: Handbook of Computational Economics **4** (2018)
20. Chiarella, C., Iori, G., Perell, J.: The impact of heterogeneous trading rules on the limit order book and order flows. arXiv:0711.3581 (2007)

21. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time series feature extraction on basis of scalable hypothesis tests, tsfresh–a python package. Neurocomputing **307**, 72–77 (2018)
22. Cont, R.: Empirical properties of asset returns: stylized facts and statistical issues. Quantitative Finance **1**, 223–236 (2001)
23. Cont, R.: Chapter 7 - Agent-Based Models for Market Impact and Volatility. A Kirman and G Teyssiere: Long memory in economics, Springer (2005)
24. Cont, R., Bouchaud, J.P.: Herd behavior and aggregate fluctuations in financial markets. Macroeconomic Dynamics **4**, 170–196 (2000)
25. Cristelli, M.: Complexity in Financial Markets. Springer (2014)
26. Delbaen, F., Schachermayer, W.: What is a free lunch? Notices of the AMS **51(5)** (2011)
27. Deng, Y., Bao, F., Kong, Y., Ren, Z., Dai, Q.: Deep direct reinforcement learning for financial signal representation and trading. IEEE Trans. on Neural Networks and Learning Systems **28(3)** (2017)
28. Ding, Z., Engle, R., Granger, C.: A long memory property of stock market returns and a new model. Journal of Empirical Finance **1**, 83–106 (1993)
29. Dodonova, A., Khoroshilov, Y.: Private information in futures markets: An experimental study. Manag Decis Econ **39** (2018)
30. Donangelo, R., Hansen, A., Sneppen, K., Souza, S.R.: Modelling an imperfect market. Physica A **283**, 469–478 (2000)
31. Donangelo, R., Sneppen, K.: Self-organization of value and demand. Physica A **276**, 572–580 (2000)
32. Duan, Y., Schulman, J., Chen, X., Bartlett, P.L., Sutskever, I., Abbeel, P.: Rl-squared: Fast reinforcement learning via slow reinforcement learning. arXiv:1611.02779 (2016)
33. Duncan, K., Doll, B.B., Daw, N.D., Shohamy, D.: More than the sum of its parts: A role for the hippocampus in configural reinforcement learning. Neuron **98**, 645–657 (2018)
34. Eickhoff, S.B., Yeo, B.T.T., Genon, S.: Imaging-based parcellations of the human brain. Nature Reviews Neuroscience **19**, 672–686 (2018)
35. Eisler, Z., Kertesz, J.: Size matters: some stylized facts of the stock market revisited. European Physical Journal B **51**, 145–154 (2006)
36. Engle, R.F.: Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. Econometrica **50(4)**, 987–1007 (1982)
37. Erev, I., E.Roth, A.: Maximization, learning and economic behaviour. PNAS **111**, 10818–10825 (2014)
38. Fama, E.: Efficient capital markets: A review of theory and empirical work. Journal of Finance **25**, 383–417 (1970)
39. Franke, R., Westerhoff, F.: Structural stochastic volatility in asset pricing dynamics: Estimation and model contest. BERG Working Paper Series on Government and Growth **78** (2011)
40. Fulcher, B.D., Jones, N.S.: Highly comparative feature-based time-series classification. Knowledge and Data Engineering, IEEE Transactions **26**, 3026?3037 (2014)
41. Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., Veloso, M.: Reinforcement learning for market making in a multi-agent dealer market. arXiv:1911.05892 (2019)
42. Gode, D., Sunder, S.: Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. Journal of Political Economy **101(1)** (1993)
43. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
44. Green, E., Heffernan, D.M.: An agent-based model to explain the emergence of stylised facts in log returns. arXiv:1901.05053 (2019)
45. Greene, W.H.: Econometric Analysis. Pearson, 8th Edition (2017)
46. Grondman, I., Busoniu, L., Lopes, G., Babuska, R.: A survey of actor-critic reinforcement learning: standard and natural policy gradients. IEEE Transactions on Systems Man and Cybernetics **42**, 1291–1307 (2012)
47. Gualdi, S., Tarzia, M., Zamponi, F., Bouchaud, J.P.: Tipping points in macroeconomic agent-based models. Journal of Economic Dynamics and Control **50**, 29–61 (2015)

48. Heinrich, J.: Deep rl from self-play in imperfect-information games. Ph.D. thesis, University College London (2017)
49. Hu, Y.J., Lin, S.J.: Deep reinforcement learning for optimizing portfolio management. 2019 Amity International Conference on Artificial Intelligence (2019)
50. Huang, W., Lehalle, C.A., Rosenbaum, M.: Simulating and analyzing order book data: the queue-reactive model. Journal of the American Statistical Association **110**, 509 (2015)
51. Huang, Z.F., Solomon, S.: Power, lévy, exponential and gaussian-like regimes in autocatalytic financial systems. European Physical Journal B **20**, 601–607 (2000)
52. Katt, S., Oliehoek, F.A., Amato, C.: Learning in pomdps with monte carlo tree search. Proceedings of the 34th International Conference on Machine Learning (2017)
53. Keramati, M., Gutkin, B.: A reinforcement learning theory for homeostatic regulation. NIPS (2011)
54. Keramati, M., Gutkin, B.: Homeostatic reinforcement learning for integrating reward collection and physiological stability. Elife **3** (2014)
55. Kim, G., Markowitz, H.M.: Investment rules, margin and market volatility. Journal of Portfolio Management **16**, 45–52 (1989)
56. Konovalov, A., Krajbich, I.: Gaze data reveal distinct choice processes underlying model-based and model-free reinforcement learning. Nature communications **7**, 12438 (2016)
57. Lefebvre, G., Lebreton, M., Meyniel, F., Bourgeois-Gironde, S., Palminteri, S.: Behavioural and neural characterization of optimistic reinforcement learning. Nature Human Behaviour **1(4)** (2017)
58. Levy, M., Levy, H., Solomon, S.: A microscopic model of the stock market: cycles, booms, and crashes. Economics Letters **45**, 103–111 (1994)
59. Levy, M., Levy, H., Solomon, S.: Microscopic simulation of the stock market: the effect of microscopic diversity. Journal de Physique I **5**, 1087–1107 (1995)
60. Levy, M., Levy, H., Solomon, S.: New evidence for the power-law distribution of wealth. Physica A **242**, 90–94 (1997)
61. Levy, M., Levy, H., Solomon, S.: Microscopic simulation of financial markets: from investor behavior to market phenomena. Academic Press, New York (2000)
62. Levy, M., Persky, N., Solomon, S.: The complex dynamics of a simple stock market model. International Journal of High Speed Computing **8**, 93–113 (1996)
63. Levy, M., Solomon, S.: Dynamical explanation for the emergence of power law in a stock market model. International Journal of Modern Physics C **7**, 65–72 (1996)
64. Levy, M., Solomon, S.: Power laws are logarithmic boltzmann laws. International Journal of Modern Physics C **7**, 595–601 (1996)
65. Liang, H., Yang, L., Tu, H.C.W., Xu, M.: Human-in-the-loop reinforcement learning. 2017 Chinese Automation Congress (2017)
66. Lipski, J., Kutner, R.: Agent-based stock market model with endogenous agents' impact. arXiv:1310.0762 (2013)
67. Lobato, I.N., Savin, N.E.: Real and spurious long-memory properties of stock-market data. Journal of Business and Economics Statistics **16**, 261–283 (1998)
68. Lux, T., Marchesi, M.: Scaling and criticality in a stochastic multi-agent model of a financial market. Nature **397**, 498–500 (1999)
69. Lux, T., Marchesi, M.: Volatility clustering in financial markets: a microsimulation of interacting agents. Journal of Theoretical and Applied Finance **3**, 67–70 (2000)
70. Mandelbrot, B.: The variation of certain speculative prices. The Journal of Business pp. 394–419 (1963)
71. Mandelbrot, B., Fisher, A., Calvet, L.: A multifractal model of asset returns. Cowles Foundation for Research and Economics (1997)
72. Martino, A.D., Marsili, M.: Statistical mechanics of socio-economic systems with heterogeneous agents. Journal of Physics A **39**, 465–540 (2006)
73. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
74. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. arXiv:1602.01783 (2016)

75. Momennejad, I., Russek, E., Cheong, J., Botvinick, M., Daw, N.D., Gershman, S.J.: The successor representation in human reinforcement learning. Nature Human Behavior **1**, 680–692 (2017)
76. Murray, M.P.: A drunk and her dog: An illustration of cointegration and error correction. The American Statistician **48(1)**, 37–39 (1994)
77. N, R.M., Larralde, H.: A detailed heterogeneous agent model for a single asset financial market with trading via an order book. arXiv:1601.00229 (2016)
78. Naik, P.K., Gupta, R., Padhi, P.: The relationship between stock market volatility and trading volume: evidence from south africa. J Dev Areas **52(1)** (2018)
79. Neuneier, R.: Enhancing q-learning for optimal asset allocation. Proc. of the 10th International Conference on Neural Information Processing Systems (1997)
80. Ng, A.Y., Harada, D., Russell, S.: Theory and application to reward shaping. International Conference on Machine Learning (1999)
81. Pagan, A.: The econometrics of financial markets. Journal of Empirical Finance **3**, 15–102 (1996)
82. Palminteri, S., Khamassi, M., Joffily, M., Coricelli, G.: Contextual modulation of value signals in reward and punishment learning. Nature communications pp. 1–14 (2015)
83. Palminteri, S., Lefebvre, G., Kilford, E., Blakemore, S.: Confirmation bias in human reinforcement learning: Evidence from counterfactual feedback processing. PLoS computational biology **13(8)** (2017)
84. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn, machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
85. Pinto, L., Davidson, J., Sukthankar, R., Gupta, A.: Robust adversarial reinforcement learning. arXiv:1703.02702 (2017)
86. Plerou, V., Gopikrishnan, P., Amaral, L.A., Meyer, M., Stanley, H.E.: Scaling of the distribution of fluctuations of financial market indices. Physical Review E **60(6)**, 6519 (1999)
87. Potters, M., Bouchaud, J.P.: More stylized facts of financial markets: Leverage effect and downside correlations. Physica A **299**, 60–70 (2001)
88. Preis, T., Golke, S., Paul, W., Schneider, J.J.: Multi-agent-based order book model of financial markets. Europhysics Letters **75(3)**, 510–516 (2006)
89. Ross, S., Pineau, J., Chaib-draa, B., Kreitmann, P.: A bayesian approach for learning and planning in partially observable markov decision processes. Journal of Machine Learning Research **12**, 1729–1770 (2011)
90. Sbordone, A.M., Tambalotti, A., Rao, K., Walsh, K.J.: Policy analysis using dsge models: An introduction. Economic policy review **16(2)** (2010)
91. Schreiber, T., Schmitz, A.: Discrimination power of measures for nonlinearity in a time series. Physical Review E **55**(5), 5443 (1997)
92. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–489 (2016)
93. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi and go through self-play. Science **362**(6419), 1140–1144 (2018)
94. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. Proceedings of the 31st International Conference on Machine Learning **32** (2014)
95. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**, 354–359 (2018)
96. Sirignano, J., Cont, R.: Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance **19(9)** (2019)
97. Solomon, S., Weisbuch, G., de Arcangelis, L., Jan, N., Stauffer, D.: Social percolation models. Physica A **277(1)**, 239–247 (2000)
98. Spooner, T., Fearnley, J., Savani, R., Koukorinis, A.: Market making via reinforcement learning. Proceedings of the 17th AAMAS (2018)

99. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press Cambridge MA (1998)
100. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems **12**, 1057–1063 (2000)
101. Szepesvari, C.: Algorithms for Reinforcement Learning. Morgan and Claypool Publishers (2010)
102. Tessler, C., Givony, S., Zahavy, T., Mankowitz, D.J., Mannor, S.: A deep hierarchical approach to lifelong learning in minecraft. arXiv:1604.07255 (2016)
103. Vandewalle, N., Ausloos, M.: Coherent and random sequences in financial fluctuations. Physica A **246**, 454–459 (1997)
104. Vernimmen, P., Quiry, P., Dallocchio, M., Fur, Y.L., Salvi, A.: Corporate Finance: Theory and Practice. John Wiley and Sons, 4th Edition (2014)
105. de Vries, C., Leuven, K.: Stylized facts of nominal exchange rate returns. Working Papers from Purdue University, Krannert School of Management - Center for International Business Education and Research (CIBER) (1994)
106. Wang, J.X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J.Z., Hassabis, D., Botvinick, M.: Prefrontal cortex as a meta-reinforcement learning system. Nature Neuroscience **21**, 860–868 (2018)
107. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine learning **8(3-4)**, 279–292 (1992)
108. Way, E., Wellman, M.P.: Latency arbitrage, market fragmentation, and efficiency: a two-market model. Proceedings of the fourteenth ACM conference on Electronic commerce pp. 855–872 (2013)
109. Wellman, M.P., Way, E.: Strategic agent-based modeling of financial markets. The Russell Sage Foundation Journal of the Social Sciences **3(1)**, 104–119 (2017)
110. Weron, R.: Levy-stable distributions revisited: Tail index ¿ 2 does not exclude the levy-stable regime. International Journal of Modern Physics C **12**, 209–223 (2001)
111. Wiering, M., van Otterlo, M.: Reinforcement Learning: State-of-the-Art. Springer, Berlin, Heidelberg (2012)

6 Supplementary material

Classification accuracy. We show on Fig. 16 the accuracy of both the testing (left) and training (right) sets, as functions of time-series sample size, for samples containing larger numbers of timestamps than in the Fig. 13. The saturating accuracy dynamics can be observed for both testing and training sets: for the value distribution feature set, the former does not exceed 70% and the latter 75%, while for the full time-series feature set, the former saturates above 90% and the latter above 95%. One can notice that the accuracy values on the training set are generally higher than for the testing set, and do not show such a pronounced saturation dynamic. The accuracy on the training set is not too large because the trees in the random forest have been regularized (with maximal depth equal to 5), since we found it is necessary for a good generalization on the testing set.

Top statistical features. We provide a general grouping and examples of the top statistical features used in the dimensionality reduction performed in section 4.2. The exact ranking of particular features found in our experiments together with their importance metric value Θ is as follows. The imporance metric Θ is summed from 30 random forest models trained on different random splits of the training/testing sets.

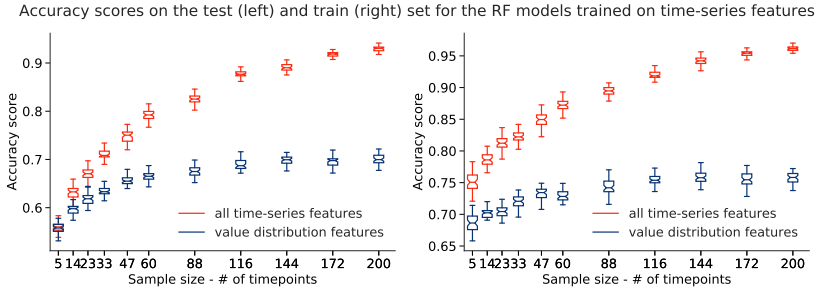

Fig. 16 Accuracy of the testing set (left) and training set (right), as a function of time-series sample size. The value distribution features include only features that do not depend on the order of values in the time-series (e.g. mean, median, variance, kurtosis, skewness of the value distribution, etc.), whereas all time-series features correspond to the total set of time-series features including those that depend on the temporal structure of the series (e.g. autocorrelation, entropy, FFT coefficients, etc.). Both testing and training subsets are balanced in terms of class distribution, and their respective accuracy is achieved with samples containing up to 200 timestamps. The simulations are generated with parameters $I = 500$, $J = 1$, $T = 2875$, and $S = 20$.

1. Partial autocorrelation value of lag 1, $\Theta = 1.2240$.
2. First coefficient of the fitted $AR(10)$ process, $\Theta = 1.0777$.
3. Kurtosis of the FFT coefficient distribution, $\Theta = 1.0214$.
4. Skewness of the FFT coefficient distribution, $\Theta = 1.0001$.
5. Autocorrelation value of lag 1, $\Theta = 0.9861$.
6. 60th percentile of the value distribution, $\Theta = 0.9044$.
7. Kurtosis of the FFT coefficient distribution, $\Theta = 0.7347$.
8. Mean of consecutive changes in the series for values in between the 0th and the 80th percentiles of the value distribution, $\Theta = 0.6349$.
9. Variance of consecutive changes in the series for values in between the 0th and the 20th percentiles of the value distribution, $\Theta = 0.5948$.
10. Approximate entropy value (length of compared run of data is 2, filtering level is 0.1), $\Theta = 0.5878$.
11. 70th percentile of the value distribution, $\Theta = 0.5589$.
12. Variance of absolute consecutive changes in the series for values in between the 0th and the 20th percentiles of the value distribution, $\Theta = 0.5584$.
13. Mean of consecutive changes in the series for values in between the 40th and the 100th percentiles of the value distribution, $\Theta = 0.4755$.
14. Ratio of values that are more than 1 standard deviation away from the mean value, $\Theta = 0.3282$.
15. Median of the value distribution, $\Theta = 0.2957$.
16. Skewness of the value distribution, $\Theta = 0.2894$.
17. Measure of time series nonlinearity from [91] of lag 1, $\Theta = 0.2867$.
18. Second coefficient of the fitted $AR(10)$ process, $\Theta = 0.2726$.
19. Partial autocorrelation value of lag 1, $\Theta = 0.2575$.
20. Time reversal symmetry statistic from [40] of lag 1, $\Theta = 0.2418$.

The top-10 features referenced in section 4.2 are the first 10 features taken from the list above. The PCA and UMAP mappings of the top-10 features onto a two-dimensional space demonstrated some separability between the two classes (real vs. simulated data), as measured by training a linear classifier on these two-dimensional data representations (see section 4.2 for details), as well as by calculating the Kolmogorov-Smirnov (KS) statistic for each embedding component. The KS statistic value between the two classes is 0.24 and 0.11 for PCA (for the first and second component, respectively) and 0.30 and 0.25 for UMAP.