Gradient-Assisted Calibration for Financial Agent-Based Models

Joel Dyer*
University of Oxford
Oxford, United Kingdom
joel.dyer@cs.ox.ac.uk

Arnau Quera-Bofarull*
University of Oxford
Oxford, United Kingdom
arnau.quera-bofarull@cs.ox.ac.uk

Ayush Chopra
Massachusetts Institute of Technology
Boston, USA
ayushc@mit.edu

J. Doyne Farmer
University of Oxford
Oxford, United Kingdom

Anisoara Calinescu
University of Oxford
Oxford, United Kingdom

Michael Wooldridge
University of Oxford
Oxford, United Kingdom

ABSTRACT

Agent-based modelling (ABMing) is a promising approach to modelling and reasoning about complex systems such as financial markets. However, the application of ABMs in practice is often impeded by the models' complexity and the ensuing difficulty of performing parameter inference and optimisation tasks. This in turn has motivated efforts directed towards the construction of differentiable ABMs, enabled by recently developed effective auto-differentiation frameworks, as a strategy for addressing these challenges.

In this paper, we discuss and present experiments that demonstrate how differentiable programming may be used to implement and calibrate heterogeneous ABMs in finance. We begin by considering in more detail the difficulties inherent in constructing gradients for discrete ABMs. Secondly, we illustrate solutions to these difficulties, by using a discrete agent-based market simulation model as a case study. Finally, we show through numerical experiments how our differentiable implementation of this discrete ABM enables the use of powerful tools from probabilistic machine learning and conditional generative modelling to perform robust parameter inferences and uncertainty quantification, in a simulation-efficient manner.

CCS CONCEPTS

• **Computing methodologies** → *Uncertainty quantification*; **Agent / discrete models**; *Uncertainty quantification*; • **Mathematics of computing** → **Automatic differentiation**; *Variational methods*; *Probabilistic inference problems.*

KEYWORDS

agent-based models, automatic differentiation, parameter inference

ACM Reference Format:
Joel Dyer, Arnau Quera-Bofarull, Ayush Chopra, J. Doyne Farmer, Anisoara Calinescu, and Michael Wooldridge. 2023. Gradient-Assisted Calibration for Financial Agent-Based Models. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (ICAIF '23).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.

ICAIF '23, November 27–29, 2023, New York, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Agent-based models (ABMs) are stochastic simulation models in which a collection of heterogeneous and interacting entities – the "agents" of the system – make decisions and perform actions on the basis of these interactions with other agents and their environment. Such models have gained considerable popularity as a tool to model financial systems, due to their ability to simulate complex systems at a granular level [21]. While the ABM modelling paradigm offers unique advantages, their complexity presents significant challenges, for example in terms of parameter calibration [see e.g. 3, 13, 31]. Multiple factors contribute to the difficulty of these tasks, including the intractability of the ABM's likelihood function, and the often black-box and non-differentiable nature of the ABM [1].

These drawbacks have motivated research into the construction of differentiable ABMs [2, 9], in which an approximation to the gradient of an originally non-differentiable ABM is sought, for example by exploiting automatic differentiation (AD) frameworks. AD – a methodological cornerstone in machine learning, largely underpinning the success of deep learning paradigms due to its ability to accurately compute derivatives within models – circumvents issues present in alternative approaches to model differentiation by applying the chain rule of differentiation at a computational level, resulting in accurate derivatives at a low computational cost. By implementing ABMs within a differentiable programming framework, approximations to the gradient of the ABM's output with respect to its input can be obtained and usefully applied in optimisation and parameter calibration tasks.

In this paper, we consider some of the main challenges faced when constructing differentiable implementations of discrete ABMs in finance, before demonstrating how recent progress in AD and probabilistic machine learning can be leveraged to build and calibrate differentiable ABMs of financial systems. Through experiments with a discrete financial ABM, we provide empirical evidence of the benefits gradient-assisted calibration methods can offer over black-box alternatives, both in terms of simulation-efficiency and the quality of the resulting parameter inferences. Given that ABMs can be extremely expensive to simulate and that decision-makers require robust parameter inferences to appropriately inform their decision-making, these developments have the potential to dramatically improve the ease with which ABMs can be deployed in real-world settings.

2 AUTOMATIC DIFFERENTIATION IN AGENT-BASED MODELS

In many practical settings involving ABMs, it is commonly the case that practitioners are interested in performing optimisation problems. Such problems often take of the following form:

$$\min_{\omega \in \Omega} \mathbb{E}_{z \sim p_\omega} \left[\mathcal{L}(z) \right], \tag{1}$$

where $p_\omega \in \{p_{\omega'} : \omega' \in \Omega\}$ is a probability distribution on domain $\mathcal{Z}$ indexed by a parameter ω belonging to some set Ω, and $\mathcal{L} : \mathcal{Z} \to \mathbb{R}$ is a loss function. For example, a key step in ensuring that the model captures key relevant features of the modeled system is to perform *parameter calibration*, in which the model parameter vector – or a distribution over the parameter vector – is tuned in order to uncover parameter vectors that are consistent with data observed in the real world [31]. For example, some parameter calibration procedures seek to identify the parameters $\boldsymbol{\theta}$ in some set $\boldsymbol{\Theta}$ that minimise some discrepancy $\mathcal{D}(\cdot, \mathbf{y})$ between the model output $\mathbf{x}$ and real-world data $\mathbf{y}$; in many such cases, this task can be cast as the optimisation problem

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{\mathbf{x} \sim p(\cdot \mid \boldsymbol{\theta})} \left[\mathcal{D}(\mathbf{x}, \mathbf{y}) \right], \tag{2}$$

where $p(\cdot \mid \boldsymbol{\theta})$ is the ABM's likelihood function and $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is a time series simulated from the ABM.

A natural approach to solving optimisation problems such as Equation 1 is to perform an iterative gradient-based search of the solution space Ω. Broadly speaking, such algorithms would iterate by making small changes to the parameter ω by taking steps in a direction informed by the gradient

$$\nabla_\omega \mathbb{E}_{z \sim p_\omega} \left[\mathcal{L}(z) \right]. \tag{3}$$

In order to proceed in this manner, however, a necessary prerequisite is the ability to compute – or, where this is not possible, find a Monte Carlo approximation to – the gradient given in Equation 3. Since it is typically extremely difficult to find a closed-form expression for this gradient, one overarching challenge to performing gradient-based optimisation and parameter calibration for ABMs is the task of accurately and efficiently estimating Equation 3.

2.1 Monte Carlo gradient estimation

The task of estimating Equation 3 in the absence of a closed-form expression for the gradient or expectation is known as the problem of Monte Carlo gradient estimation [27], in which samples from the probability distribution p_ω are used to find a finite-sample estimate of Equation 3. Two dominant approaches exist for performing this Monte Carlo estimation:

(1) The first approach is commonly termed a *score-based* gradient estimate, in which minimal assumptions are made about the differentiability of the loss function $\mathcal{L}$. By exchanging the order of the integral and gradient in Equation 3, this estimator is obtained as

$$\nabla_\omega \mathbb{E}_{z \sim p_\omega} \left[\mathcal{L}(z) \right] = \mathbb{E}_{z \sim p_\omega} \left[\mathcal{L}(z) \nabla_\omega \log p_\omega(z) \right] \tag{4}$$

$$\approx \frac{1}{R} \sum_{r=1}^{R} \mathcal{L}(z^{(r)}) \nabla_\omega \log p_\omega(z^{(r)}), \tag{5}$$

where $z^{(n)} \sim p_\omega, n = 1, \ldots, N$. Such a gradient estimate is broadly applicable, since it does not require differentiability of $\mathcal{L}$; however, a drawback of this gradient estimator is that it is known to suffer from high variance in many cases [27].

(2) The second approach is commonly termed a *path* or *pathwise* gradient estimator. This approach makes stronger assumptions about the loss function $\mathcal{L}$ and about the simulator: it typically requires both that $\mathcal{L}$ is differentiable, and that the stochastic sampling procedure $z \sim p_\omega$ can be *reparameterised* [23], which is to say that is can be written as

$$z = g(u, \omega), \quad u \sim p(u) \tag{6}$$

for some parameter-free probability distribution $p(u)$ and function g that is differentiable in its second argument. In this case, the pathwise gradient estimator may be constructed as

$$\nabla_\omega \mathbb{E}_{z \sim p_\omega} \left[\mathcal{L}(z) \right] = \nabla_\omega \mathbb{E}_{u \sim p} \left[\mathcal{L}(g(u, \omega)) \right] \tag{7}$$

$$= \mathbb{E}_{u \sim p} \left[\nabla_\omega \mathcal{L}(g(u, \omega)) \right] \tag{8}$$

$$\approx \frac{1}{R} \sum_{r=1}^{R} \nabla_\omega \mathcal{L}(g(u^{(r)}, \omega)), \tag{9}$$

where $u^{(r)} \sim p(u), r = 1, \ldots, R$ and we use the Law of the Unconscious Statistician [20] in the first line. While this approach imposes more stringent constraints on the form of the simulator, it can result in Monte Carlo gradient estimates that have lower variance than the score-based alternative in Equation 5 [27].

On the basis of this discussion, it can be desirable to ensure that the conditions required to perform the reparameterisation trick and obtain pathwise derivatives are met, in order to obtain lower-variance Monte Carlo gradient estimates and to converge more rapidly onto a suitable minimiser for the optimisation problem in Equation 1.

2.2 Challenges to Monte Carlo gradient estimation for agent-based models

Here, we conclude by considering some of the main challenges that ABMs, both in financial settings and more broadly, pose to the broader challenge of performing Monte Carlo gradient estimation in general optimisation and parameter calibration problems.

2.2.1 Pathwise derivatives in the presence of discrete randomness. By nature, ABMs simulate discrete events, transitions, and interactions, for which meaningful notions of derivatives are difficult to construct due to the intrinsically non-continuous nature of these operations. For this reason, it is typically not immediately the case that ABMs satisfy the constraints required to find pathwise derivatives, and consequently a pathwise Monte Carlo gradient estimator as in Equation 9. One of the central difficulties in performing gradient-based optimisation for ABMs is therefore the issues of differentiating through discrete operations and discrete randomness that are simulated within an ABM, in a manner that preserves the generally low-variance nature of a pathwise gradient estimator.

Initial research efforts towards approaches to differentiating through discrete structures in ABMs have primarily centred on

transforming the ABM's discrete control flow structure with continuous approximations [1, 28]. Such an approach is, however, generally undesirable, since this introduces approximations to both the gradients of the model *and* the underlying model itself. More recently, more desirable approaches to differentiating through discrete randomness that entail only the introduction of approximations to the model's gradients *without* modifying the underlying model have been taken (see e.g. [2, 9]) through the use of the Gumbel-Softmax (GS) reparameterisation trick [22] and StochasticAD [2] in the Julia programming language [6].

In this paper (see section 3 below), we discuss and demonstrate how these efforts can be extended to the case of financial ABMs, and how a discrete financial ABM can be implemented in a differentiable manner using automatic differentiation (AD) frameworks. We will show that the gradient estimates obtained using the GS reparameterisation trick are sufficiently robust for fast and accurate Bayesian inference for a stochastic ABM of trader decisions in an artificial financial market.

2.2.2 Memory-efficient simulator gradients. An additional consideration that we discuss in this paper is the memory requirements in constructing and employing differentiable financial ABMs in practice. When using AD to compute derivatives within a differentiable simulator, two primary AD techniques are available to the user: reverse-mode AD and forward-mode AD [4]. In reverse-mode AD (RMAD), a computation graph must be stored that records all operations performed within the model, such that the gradients of the model outputs with respect to the input parameters can be obtained. This contrasts with forward-mode AD (FMAD), where the gradients are computed during forward simulation of the ABM.

Two important computational considerations exist when comparing FMAD and RMAD in the context of differentiable ABMs. The first is that the computational time associated with FMAD scales with the number of model inputs, while that of RMAD scales with the number of model outputs. In many machine learning settings, the latter option is more prevalent, since machine learning models often have many more inputs than outputs. However, the computation graph that RMAD must store in (often GPU) memory can be extremely large, hindering the possibility of differentiating through large models. This is particularly pertinent for ABMs: the size of the computation graph grows with the number of agents and time-steps, which can pose a challenge to the use of RMAD for ABMs with a large number of agents and time-steps. For this reason, imprudently using RMAD exclusively during gradient-assisted calibration can result in prohibitively large memory requirements. As we will demonstrate in section 3, this problem can be addressed with a judicious use of both RMAD and FMAD while performing gradient-assisted calibration and optimisation.

3 EXPERIMENTS

In this section, we demonstrate how gradients can be introduced to discrete ABMs and used to perform gradient-assisted calibration.

3.1 The agent-based model

To demonstrate how differentiable programming can be applied to discrete ABMing in finance, we consider the discrete ABM of volatility clustering in financial markets presented in [10], which we briefly describe here. In this model, N agents trade a single asset over discrete periods $t \in \{1, \dots, T\}$, where the price of the asset at time t is denoted S_t. At time t, agent i submits a buy or sell order, represented with $\rho_i(t) = 1$ and -1, respectively. A value of $\rho_i(t) = 0$ implies that agent i is inactive at time t. Agents receive a common signal ϵ_t that forecasts the next time period's log-returns $r_t = \log S_t / S_{t-1}$ in the price of the asset, upon which they base their decisions to place orders; this signal is modelled as a sequence of *iid*, zero-mean Gaussian innovations $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ with standard deviation $\sigma > 0$. On the basis of this signal, agent i decides whether or not to place an order at time t according to the following rule:

$$\rho_i(t) = \mathbb{1}_{\epsilon_t > v_i(t)} - \mathbb{1}_{\epsilon_t < -v_i(t)}, \tag{10}$$

where $v_i(t) > 0$ is the threshold level for agent i at time t, which determines the range of values of the public information ϵ_t that agent i considers to be significant. In this way, each agent receives the same information, but the different agents differ in how this is processed. The initial values $v_i(0)$ are drawn *iid* from some probability density function f_γ indexed by parameters γ. The excess demand

$$Z_t = \sum_{i=1}^{N} \rho_i(t) \tag{11}$$

then determines the actual log-returns according to

$$r_t = \frac{Z_t}{N\eta}, \tag{12}$$

where $\eta > 0$ is a free parameter of the model. Finally, each agent updates their threshold $v_i(t)$ at time t according to

$$v_i(t) = \mathbb{1}_{u_i(t) < s} |r_t| + \mathbb{1}_{u_i(t) \geq s} v_i(t-1), \tag{13}$$

where $u_i(t)$ are *iid* uniformly distributed random variables on the unit interval $[0, 1]$ and $s = 10^{-1}$ is the probability that each agent updates their threshold in a given time period.

In summary, a single model iteration proceeds as follows:

- Each agent receives a common information signal $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$;
- Agent i compares ϵ_t with their (time-varying) threshold $v_i(t)$ to produce purchase order $\rho_i(t)$ according to Equation 10.
- The excess demand Z_t is used to obtain the log-returns r_t at this step according to Equation 12.
- With probability $s = 10^{-1}$, each agent updates its threshold to $|r_t|$, otherwise their threshold is left unchanged.

While the model is a clear simplification of reality – for example, no social learning between agents is introduced, and there is no distinction between different trader types such as chartists and fundamentalists – the model serves as a useful case study for constructing gradients for discrete, stochastic ABMs, which we discuss in more detail in subsection 3.2.

3.2 A differentiable implementation

The model presented in subsection 3.1 is not immediately differentiable, as a result of the following two model properties:

(1) The $\rho_i(t)$ are (non-*iid*) discrete random variables taking values in the set $\{-1, 0, 1\}$ according to whether the ϵ_t falls in

the range $(-\infty, -\nu_i(t))$, $[-\nu_i(t), \nu_i(t)]$, or $(\nu_i(t), \infty)$, respectively. Z_t is consequently also a discrete random variable;

(2) The $\nu_i(t)$ are (non-*iid*) discrete random variables taking values in the set $\{|r_t|, \nu_i(t-1)\}$ with probabilities s and $1-s$, respectively.

To differentiate through the model – for example, to find the partial derivatives of the model outputs with respect to the input parameters – we would like to be able to pass gradients through these discrete random variables (passing gradients through the remaining operations in the model is immediately possible, either due to the continuous nature of the operations or through the use of the reparameterisation trick [23]). Importantly, we would like to do so in a way that does not entail changing the model itself, in order that the modeller remains free to specify the model in the way that they believe is most appropriate.

With the above in mind, we implement the model in `pytorch` [30] and equip $\rho_i(t)$ and $\nu_i(t)$ with gradients by implementing Equation 10 and Equation 12 using the following tricks:

(1) We take the following "straight-through" approach [5] to constructing a differentiable $\rho_i(t)$:

$$\rho_i(t) = \tilde{\rho}_i(t) + \varrho_i(t) - \varrho_i(t)\texttt{.detach()}, \tag{14}$$

where $\tilde{\rho}_i(t)$ is computed according to Equation 10, while

$$\varrho_i(t) = \varsigma_k(\epsilon_t - \nu_i(t)) - \varsigma_k(-\epsilon_t - \nu_i(t)), \tag{15}$$

$$\varsigma_k(x) = (1 + \exp(-kx))^{-1} \tag{16}$$

is a soft-threshold approximation to Equation 10 that uses the sigmoid function ς_k with steepness parameter $k > 0$ as an approximation to the indicator function appearing in Equation 10. In this way, the additional term

$$\varrho_i(t) - \varrho_i(t)\texttt{.detach()}$$

appearing in Equation 14 provides $\rho_i(t)$ with a non-zero gradient via $\varrho_i(t)$, while the term $-\varrho_i(t)\texttt{.detach()}$ ensures that $\rho_i(t) \in \{-1, 0, 1\}$ with the required probabilities *without* destroying the gradient information carried by $\varrho_i(t)$.

(2) We implement Equation 13 similarly using the Gumbel-Softmax (GS) reparameterisation trick [22]. In particular, we once again take

$$\nu_i(t) = \tilde{\nu}_i(t) + v_i^\tau(t) - v_i^\tau(t)\texttt{.detach()}, \tag{17}$$

where $\tilde{\nu}_i(t)$ is computed as in Equation 13, while $v_i^\tau(t)$ is a reparameterised GS random variable with temperature parameter $\tau = 0.1$ [22]. As before, the additional terms in Equation 17 introduce a non-zero approximation to the gradient of the discrete random variable $\nu_i(t)$, but do so without changing the value of $\nu_i(t)$ used internally during the forward simulation of the ABM.

3.3 Gradient-assisted model calibration

Setting $f_\gamma = \text{Gamma}(\alpha, \beta)$ for free parameters $\alpha, \beta > 0$ (i.e. such that $\gamma = (\alpha, \beta)$), we consider in this section the task of calibrating the parameters $\boldsymbol{\theta} = (\log\alpha, \log\beta, \log\sigma, \log\eta)$ with $N = 1000$ agents, $s = 10^{-1}$, and assuming that only the log-returns time series $r_t, t = 1, \ldots, T$ are observed, with $T = 100$. We simulate a pseudo-observation $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T) = (r_1, \ldots, r_T)$ from the ABM described previously at parameter $\boldsymbol{\theta}^* = (\log\alpha^*, \log\beta^*, \log\sigma^*, \log\eta^*)$ = (0.1, 0.5, 0.5, 0.2), and attempt to draw inferences in a gradient-assisted manner about the parameters $\boldsymbol{\theta}$ that are consistent with the observed data $\mathbf{y}$.

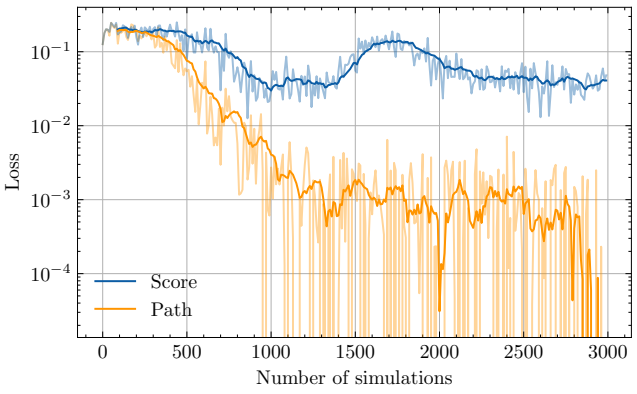

Figure 1: Training loss for the generalised variational inference scheme with score-based (blue) and pathwise (orange) gradient estimators. Dark lines show the moving average loss by averaging over 10 epochs.

3.3.1 Establishing the inference problem. While there exist many different gradient-assisted calibration methods, we focus on a variational approach to Bayesian parameter inference termed Generalised Variational Inference (GVI) [24] – a likelihood-free Bayesian inference approach. that has previously been used to calibrate the parameters of a differentiable ABM [34, 36]. Here, a variational procedure targets a "generalised" posterior [7]

$$\pi_{w,\mathbf{y}}(\boldsymbol{\theta}) \propto e^{-w\cdot\ell(\mathbf{y},\boldsymbol{\theta})}\pi(\boldsymbol{\theta}), \tag{18}$$

where $\pi(\boldsymbol{\theta})$ is a prior distribution over parameters $\boldsymbol{\theta}$, $\ell(\mathbf{y}, \boldsymbol{\theta})$ is a loss function capturing the compatibility between the observed data $\mathbf{y}$ and the behaviour of the ABM at parameter vector $\boldsymbol{\theta}$, and $w > 0$ is a hyperparameter. This posterior can then be approximated by finding a distribution q in some variational family $\mathcal{Q}$ of distributions that minimises the Kullback-Liebler divergence to the generalised posterior given in Equation 18:

$$q^* = \arg\min_{q\in\mathcal{Q}} \left\{ w\,\mathbb{E}_q\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right] + \text{KL}\left(q \| \pi\right) \right\}. \tag{19}$$

3.3.2 Setting w, ℓ, and π. In our experiments, we take $\pi(\boldsymbol{\theta})$ to be a standard multivariate Normal density (i.e. a four-dimensional Normal distribution with mean zero and covariance matrix given by the identity matrix) and $w = 10^3$. The loss function ℓ is taken to be the maximum mean discrepancy (MMD) [8, 19] test statistic between the distributions $\mathbb{P}_{\boldsymbol{\theta}}$ and $\mathbb{P}_T$ of simulated and observed of returns, respectively, defined and computed as

$$\text{MMD}(\mathbb{P}_{\boldsymbol{\theta}}, \mathbb{P}_T) := \|\mathbb{E}_{x\sim\mathbb{P}_{\boldsymbol{\theta}}} k(x, \cdot) - \mathbb{E}_{y\sim\mathbb{P}_T} k(y, \cdot)\|^2 \tag{20}$$

$$\approx \frac{1}{T(T-1)} \sum_{t\neq t'} \left[k_{\mathbf{x}_t,\mathbf{x}_{t'}} + k_{\mathbf{y}_t,\mathbf{y}_{t'}}\right] - \frac{2}{T^2} \sum_{t,t'=1}^{T} k_{\mathbf{x}_t,\mathbf{y}_{t'}}, \tag{21}$$

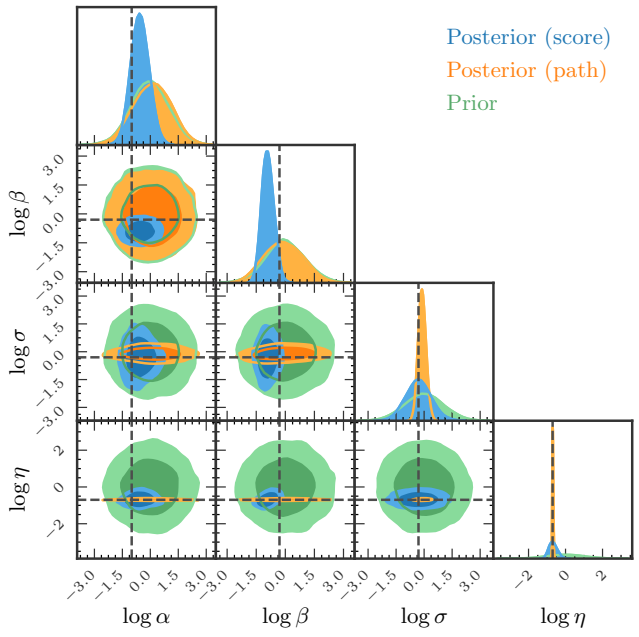

Figure 2: Samples from the prior and the posteriors obtained with pathwise- and score-based derivatives with control variate of 1. Marginal densities are shown on the diagonal, while the joint bivariates are on the off-diagonal. True generating parameters shown with dashed lines.

where k is a Gaussian radial basis function kernel with scale parameter given by the median of the pairwise Euclidean distances between the $\mathbf{y}_t$, and $k_{\mathbf{x}_t,\mathbf{y}_{t'}} := k(\mathbf{x}_t, \mathbf{y}_{t'})$. This relates to prior work on calibrating financial ABMs in which a test statistic measures the similarity between true and simulated time series data [3]. We note that while this estimator for the MMD scales quadratically in T, alternative estimators could be used in place that are of reduced complexity [see, e.g., 29].

3.3.3 Approximating the posterior. To target this posterior, we construct a flexible variational family $\boldsymbol{Q}$ by training a normalising flow q_ϕ [37, 39, 40] with trainable parameters ϕ to approximate $\pi_{w,\mathbf{y}}$. In particular, [1], we use a flow with a standard Normal base density $p(u)$ and 5 transforms consisting of (a) a feedforward network with hidden layer sizes 50, 50, followed by (b) an affine coupling block [11], and (c) a permutation layer. In this case, the minimisation problem given in Equation 19 becomes

$$\phi_{w,\mathbf{y},\pi} = \arg\min_{\phi} \left\{ w\, \mathbb{E}_{q_\phi} \left[\ell(\mathbf{y}, \boldsymbol{\theta})\right] + \mathrm{KL}\left(q_\phi \| \pi\right) \right\}. \tag{22}$$

Throughout, we make use of the BlackBIRDS package [35] to construct and train the generalised posterior estimates in a gradient-assisted manner.

The minimisation problem given in Equation 22 can be performed using gradient-based methods. In particular, to investigate the benefits of using differentiable implementations of discrete ABMs, we consider two gradient-based approaches:

(1) In the first instance, we use an estimator for the gradient of the objective in Equation 22 that does not assume differentiability of the ABM. This entails the use of the score-based/REINFORCE gradient estimator [27] for the first term in Equation 22, which we discussed previously in subsection 2.1, and is used widely in reinforcement learning applications:

$$\nabla_\phi \mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right] \approx \frac{1}{J} \sum_{j=1}^{J} \left(\ell(\mathbf{y}, \boldsymbol{\theta}^{(j)}) - b\right) \nabla_\phi \log q_\phi(\boldsymbol{\theta}^{(j)}), \tag{23}$$

where we introduce a control variate $b \in \mathbb{R}$ as a variance reduction measure [27, 41].

(2) Secondly, we exploit the differentiability of the ABM and of the loss function given in Equation 21 to construct a pathwise Monte Carlo gradient estimator via the reparameterisation trick:

$$\nabla_\phi \mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right] \approx \frac{1}{J} \sum_{j=1}^{J} \nabla_\phi \ell(\mathbf{y}, \boldsymbol{\theta}_\phi(u^{(j)})), \tag{24}$$

where $u^{(j)} \sim p(u)$ is a sample from the base density $p(u)$ and $\boldsymbol{\theta}_\phi(u^{(j)})$ is the transformed sample from the flow q_ϕ. This second approach exploits the differentiability of the ABM in that the dependence of $\ell(\mathbf{y}, \boldsymbol{\theta}) = \mathrm{MMD}(\mathbb{P}_{\boldsymbol{\theta}}, \mathbb{P}_T)$ on $\boldsymbol{\theta}$ is indirect and mediated by $\mathbf{x} \sim p(\cdot \mid \boldsymbol{\theta})$, where $p(\cdot \mid \boldsymbol{\theta})$ is the ABM's likelihood function; thus, by the chain rule, derivatives of $\mathbf{x}$ will appear during the computation of each term $\nabla_\phi \ell(\mathbf{y}, \boldsymbol{\theta}_\phi(u^{(j)}))$, $j = 1, \ldots, J$, appearing in Equation 24.

In both cases described above, we exploit the reparameterisation trick to construct the following Monte Carlo gradient estimator for the regularisation term $\mathrm{KL}(q_\phi \| \pi)$ appearing in the objective Equation 22:

$$\begin{aligned}\nabla_\phi \mathrm{KL}(q_\phi \| \pi) &= \nabla_\phi \mathbb{E}_{u\sim p(u)}\left[\log q_\phi(\boldsymbol{\theta}_\phi(u)) - \log \pi(\boldsymbol{\theta}_\phi(u))\right] \\ &\approx \frac{1}{R} \sum_{r=1}^{R} \nabla_\phi \left[\log q_\phi\left(\boldsymbol{\theta}_\phi(u^{(r)})\right) - \log \pi\left(\boldsymbol{\theta}_\phi(u^{(r)})\right)\right], \end{aligned} \tag{25}$$

where $u^{(r)} \sim p(u)$ once again. Additionally, we train with the AdamW optimiser [26] using a learning rate of 10^{-3} and using $J = 10$ and $R = 10^4$. Finally, we train for 300 epochs.

3.3.4 Results. In Figure 1 we show the value of the loss function $\mathrm{KL}(q_\phi \| \pi_{w,\mathbf{y}})$ as a function of the number of forward simulations from the ABM for both the score-based and pathwise-based gradient estimators discussed in subsubsection 3.3.3. Translucent lines show the instantaneous loss, while the opaque lines show the moving average of this loss curve by averaging over the previous 10 epochs. Here, we take $b = 1$ as the value of the control variate for the

Table 1: Log-posterior probability density of the true generating parameters by gradient-estimation method

	Score	Pathwise
$\log q_{\phi_{w,\mathbf{y},\pi}}(\boldsymbol{\theta}^*)$	-2.31	0.16

[1] See https://github.com/joelnmdyer/gradient_assisted_calibration_abm.

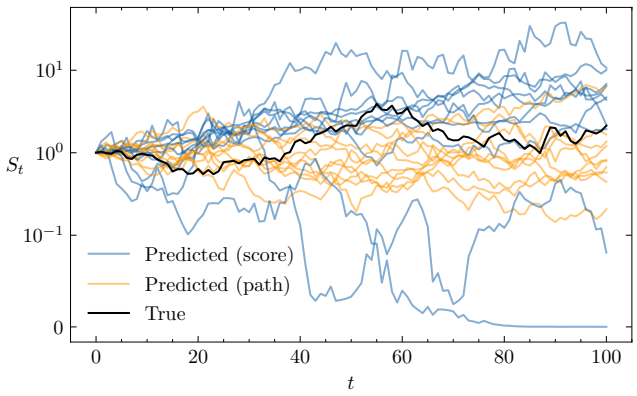

Figure 3: Sample trajectories for the asset price from the posterior predictive distributions obtained from the score-based (blue) and pathwise (orange) gradient estimators. True asset price is shown with the black line.

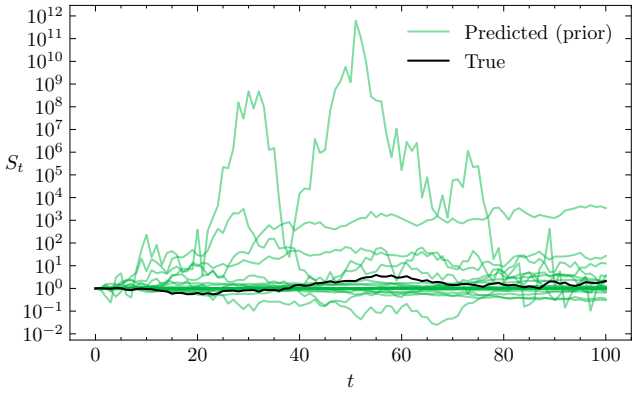

Figure 4: Sample trajectories for the asset price from the prior predictive distribution (green). True asset price is shown in black.

score-based gradient estimator without significant fine-tuning, but observed this to dramatically improve the quality of training over the case of $b = 0$. We furthermore take $k = 5$ in the sigmoid function ς_k appearing in Equation 14, which we observed to result in informative gradient estimates without over-smoothing of the discrete indicator function.

From this plot, we see that the pathwise gradient estimator, enabled through the differentiable implementation of the ABM, results in both a more accurate approximation to the generalised posterior we target than the score-based gradient estimator, in addition to faster convergence to this superior approximation. Indeed, the generalised posterior the algorithm targets is almost perfectly approximated with only roughly 10^3 simulations from the ABM with the use of the pathwise gradient estimator. In contrast, the score-based gradient estimator failed to generate an especially accurate posterior density: the lowest value of the objective function that the score-based optimisation procedure achieves is approximately three orders of magnitude larger than the best value achieved by the pathwise gradient estimator. Thus, while it was able to achieve *some* improvement in the quality of the posterior approximation in comparison to the initial approximation given by the untrained normalising flow, the quality of the approximation did not improve as a dramatically as the optimisation procedure that employed the pathwise gradient estimator.

From Figure 2, we observe the practical consequences of this difference in training performance. In Figure 2 we overlay contour plots for samples from (a) the prior density (green), (b) the posterior obtained with the pathwise gradient estimator (orange), and (c) the posterior obtained with the score-based estimator (blue). The marginal densities are shown on the diagonal, while the off-diagonal panels show the bivariate joint densities. From this, we see that while both the score-based and pathwise derivatives result in posterior densities that are a significant update to the prior density, the pathwise gradient estimator results in a posterior that correctly assigns relatively high credibility to the true generating parameters $\boldsymbol{\theta}^*$; see Table 1 for the values of these log probability densities. In contrast, the score-based estimator exhibits overconfidence in its estimates of various parameters, resulting in the assignment of an incorrectly (by the comparatively high loss in Figure 1) low credibility to $\boldsymbol{\theta}^*$; see also Table 1. This Figure also demonstrates that the pathwise Monte Carlo gradient estimator assists the posterior approximator to correctly update the initial belief distribution (the prior density) more or less strongly according to the evidence in favour of different values for $\boldsymbol{\theta}$: the pathwise approach correctly identifies that the parameters σ and η are strongly constrained by the data $\mathbf{y}$ and loss function ℓ, but that the parameters α and β are far less strongly constrained by this new information.

Additionally, it is apparent that the dynamics present in the observed data $\mathbf{y}$ are better captured when the generalised posterior is targeted in a way that exploits the differentiable implementation of the ABM, rather than by using the score-based training procedure. Figure 3 shows sample price trajectories $\tilde{\mathbf{x}}$ from the posterior predictive distribution

$$p_{w,\mathbf{y},\pi}(\tilde{\mathbf{x}}) = \int p(\tilde{\mathbf{x}} \mid \boldsymbol{\theta})\, q_{\phi_{w,\mathbf{y},\pi}}(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \tag{26}$$

obtained using the pathwise (orange) and score-based (blue) Monte Carlo gradient estimates. As before, $p(\cdot \mid \boldsymbol{\theta})$ denotes the probability density (likelihood) function of the stochastic ABM at parameters $\boldsymbol{\theta}$. The observed price trajectory is also shown in Figure 3 with the black curve. From this, we see that the sample trajectories generated by the pathwise posterior predictive distribution appear visually to deviate less significantly from the ground truth data than those from the score-based posterior predictive distribution. It is noteworthy, however, that both the pathwise and score-based approaches have seen considerable success in eliminating inappropriate regions of the parameter space during training: Figure 4 shows sample price trajectories $\tilde{\mathbf{x}}$ from the prior predictive distribution,

$$p(\tilde{\mathbf{x}}) = \int p(\tilde{\mathbf{x}} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}, \tag{27}$$

which we see to deviate considerably from the true price trajectory. Thus while the introduction of approximations to the gradients within the ABM appears to offer clear advantages in this case –

such as a higher quality approximation to the targeted posterior density and a faster rate of convergence to this posterior – both the score-based and pathwise gradient estimators are able to result in reasonably accurate inferences.

3.3.5 Reducing memory consumption with forward-mode automatic differentiation. As discussed in subsubsection 2.2.2, the use of reverse-mode AD (RMAD) can often be hindered by the memory requirements of storing the computation graph in GPU memory. This is accentuated when running the simulation for a large number of time-steps, since the computation graph size grows linearly with the number of time-steps. Given that financial ABMs often require the simulation of a large number of agents over long time horizons, the exclusive use of RMAD can pose a challenge to gradient-assisted calibration and optimisation of differentiable ABMs.

To address this, we propose to employ a hybrid AD technique in which forward-mode AD (FMAD) is used to obtain the Jacobian $J_{\boldsymbol{\theta}}$ of the ABM outputs with respect to $\boldsymbol{\theta}$, and RMAD is used to obtain the gradient through the variational posterior q_ϕ, yielding

$$\nabla_\phi \mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right] = J_{\boldsymbol{\theta}}(\mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right]) \cdot \nabla_\phi \boldsymbol{\theta}, \quad \text{with} \tag{28}$$

$$J_{\boldsymbol{\theta}}(\mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right]) = \frac{\partial \mathbb{E}_{q_\phi}\left[\ell(\mathbf{y}, \boldsymbol{\theta})\right]}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1\times d}. \tag{29}$$

Here, (29) is the Jacobian obtained through FMAD and $\nabla_\phi \boldsymbol{\theta} \in \mathbb{R}^{d\times F}$ is the gradient, obtained with RMAD, of the d ABM parameters generated by the normalising flow q_ϕ with parameters $\phi \in \mathbb{R}^F$.

In Figure 5, we plot the memory consumption observed during computation of the Jacobian given in Equation 29, using both RMAD and FMAD for a fixed number of 10^6 agents in the ABM we consider. From this, we see that the memory consumption of FMAD indeed remains constant at merely 17 MB and is determined by the allocation cost of the agents' data. On the other hand, RMAD scales linearly with the number of time steps simulated, growing to over 30 GB for 10^3 time-steps. We highlight that this memory requirement is likely to be a limitation when employing GPUs, and that our proposed use of a hybrid FMAD-RMAD approach to Monte Carlo gradient estimation may be used to scale gradient-assisted calibration procedures to agent-based simulations involving large populations for long periods of simulated time, while minimising overall memory consumption.

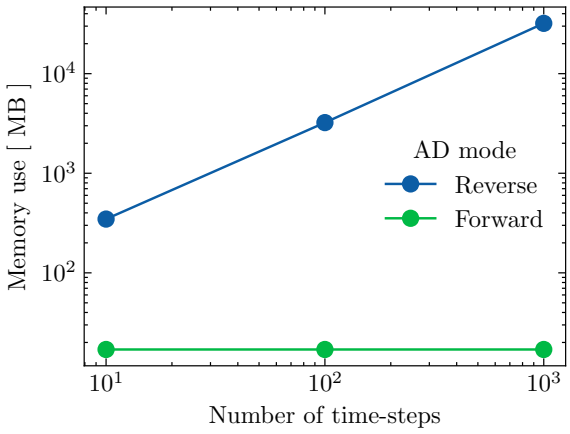

Figure 5: Memory consumption as a function of the number of time-steps for a simulation with 1M agents. Blue and green show respectively the memory cost of reverse- and forward-mode automatic differentiation, respectively.

4 RELATED WORK

Substantial research effort has been directed towards calibration methods for financial agent-based models [12]. Much of the original effort in this area focused on approaches resulting in parameter point estimates, for example through the use of Indirect Inference [17] and the Simulated Method of Moments [16]. More recently, this has included the use of Bayesian optimisation to minimise a test statistic capturing the discrepancy between the observed data and model output [3]. These approaches can be contrasted with distribution-centric approaches, in which distributions over parameterisations of the financial simulator are instead obtained, rather than single point estimates. Examples of such approaches include: the use of approximate Bayesian computation [15, 18, 32] to sample from the posterior density over parameters with e.g. Markov chain Monte Carlo; the use of conditional generative adversarial networks to obtain an implicit distribution over parameterisations of financial simulation models [33]; or neural conditional density estimators and probabilistic classifiers to obtain surrogate posterior densities over the parameters of financial agent-based models [13, 14]. Our work differs substantially from these methods however: we focus on using variational inference to target a *generalised* posterior – rather than targeting the classical Bayesian posterior, or point estimates – and compare and contrast approaches that do or do not make use of the differentiability of the financial agent-based simulator, rather than assuming that gradient information is never available within the mechanistic agent-based simulator.

5 DISCUSSION & CONCLUSION

In this paper, we discussed the use of differentiable programming to implement agent-based models in finance. We discussed some of the primary difficulties in implementing differentiable versions of discrete financial agent-based models, and demonstrated on a simple agent-based model of an artificial market how discrete randomness may be manipulated to propagate gradients through the internal operations of the agent-based model. Through experiments, we demonstrated that the introduction of approximations to the gradient within the discrete agent-based simulator facilitated a higher quality and more quickly convergent gradient-assisted Bayesian inference procedure, enabling us to more accurately recover the ground-truth parameter values and price dynamics.

While our experiments made use of a Bayesian inference procedure, in which a generalised posterior [7] was targeted with a variational approach [8, 24], we note that the use of gradients in calibration tasks is not limited to this approach. For example, one could remove the prior regularisation term in the objective Equation 22 and assume an adversarial approach similar to Storchan et al. [38], in which the agent-based model would assume the role of the generator in a GAN training framework. Furthermore, while in our experiments we defined the loss $\ell(\mathbf{y}, \boldsymbol{\theta})$ to be a discrepancy

between the real and simulated log-returns alone, this loss function may be adapted to the particular interests of the modeller. For example, the loss ℓ may be extended to capture additional or more specific stylised facts, such as the absence of significant linear autocorrelations in returns, or long-memory behaviour in trading volume and volatility [25].

There exists various additional avenues for future work. In constructing approximate gradients for discrete operations and discrete randomness, additional hyperparameters are often introduced, such as the parameter k used in the sigmoid function ς_k introduced in subsection 3.2 and the so-called "temperature" hyperparameter in the categorical Gumbel-Softmax reparameterisation trick. It would be of value to the community to develop principled and robust approaches to assigning values to these hyperparameters. Extensive empirical investigations into the relative informativeness and stability of different gradient approximations (e.g. Gumbel-Softmax [22] vs. StochasticAD [2]), how to tune their hyperparameters optimally, and the difficulty with which these hyperparameters can be tuned, would also be useful for providing guidance to practitioners within the agent-based modelling community. Further work should also investigate whether, or the degree to which, differentiable programming principles can be applied to other common components of financial agent-based models, such as limit order book simulators and market clearing mechanisms, for which extensions to existing techniques for differentiating through discrete operations may be required. Finally, while we believe that the techniques used in our differentiable implementation of the agent-based model are generically applicable, we note that other common components of ABMs may also require adaptation; for example, agent-agent interactions may need to be recast as message passing procedures on a graph to ensure scalability of the gradient computation (see e.g. [9]).

ACKNOWLEDGMENTS

This research was supported by a UKRI AI World Leading Researcher Fellowship awarded to Wooldridge (grant EP/W002949/1). M. Wooldridge and A. Calinescu acknowledge funding from Trustworthy AI - Integrating Learning, Optimisation and Reasoning (TAILOR), a project funded by European Union Horizon2020 research and innovation program under Grant Agreement 952215. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Philipp Andelfinger. 2021. Differentiable Agent-Based Simulation for Gradient-Guided Simulation-Based Optimization. *arXiv:2103.12476 [cs, eess]* (March 2021). arXiv:2103.12476 [cs, eess]

[2] Gaurav Arya, Moritz Schauer, Frank Schäfer, and Christopher Rackauckas. 2022. Automatic differentiation of programs with discrete randomness. *Advances in Neural Information Processing Systems* 35 (2022), 10435–10447.

[3] Yuanlu Bai, Henry Lam, Tucker Balch, and Svitlana Vyetrenko. 2022. Efficient Calibration of Multi-Agent Simulation Models from Output Series with Bayesian Optimization. In *Proceedings of the Third ACM International Conference on AI in Finance*. 437–445.

[4] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research* 18 (2018), 1–43.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[6] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A Fresh Approach to Numerical Computing. *SIAM review* 59, 1 (2017), 65–98.

[7] Pier Giovanni Bissiri, Chris Holmes, and Stephen G Walker. 2016. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78, 5 (2016), 1103.

[8] Badr-Eddine Cherief-Abdellatif and Pierre Alquier. 2020. MMD-Bayes: Robust Bayesian Estimation via Maximum Mean Discrepancy. In *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference (Proceedings of Machine Learning Research, Vol. 118)*, Cheng Zhang, Francisco Ruiz, Thang Bui, Adji Bousso Dieng, and Dawen Liang (Eds.). PMLR, 1–21.

[9] Ayush Chopra, Alexander Rodríguez, Jayakumar Subramanian, Arnau Quera-Bofarull, Balaji Krishnamurthy, B. Aditya Prakash, and Ramesh Raskar. 2023. Differentiable Agent-Based Epidemiology. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1848–1857.

[10] Rama Cont. 2007. Volatility clustering in financial markets: empirical facts and agent-based models. *Long memory in economics* (2007), 289–309.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HkpbnH9lx

[12] Joel Dyer. 2022. *Likelihood-free Bayesian inference for dynamic, stochastic simulators in the social sciences*. Ph. D. Dissertation. University of Oxford.

[13] Joel Dyer, Patrick Cannon, J Doyne Farmer, and Sebastian Schmon. 2022. Black-box Bayesian inference for economic agent-based models. *arXiv preprint arXiv:2202.00625* (2022).

[14] Joel Dyer, Patrick Cannon, J Doyne Farmer, and Sebastian M Schmon. 2022. Calibrating Agent-based Models to Microdata with Graph Neural Networks. In *ICML 2022 Workshop AI for Agent-Based Modelling*.

[15] Joel Dyer, Patrick Cannon, and Sebastian M Schmon. 2023. Approximate Bayesian Computation with Path Signatures. arXiv:2106.12555 [stat.ME]

[16] Reiner Franke. 2009. Applying the method of simulated moments to estimate a small agent-based asset pricing model. *Journal of Empirical Finance* 16, 5 (2009), 804–815.

[17] Christian Gourieroux, Alain Monfort, and Eric Renault. 1993. Indirect inference. *Journal of applied econometrics* 8, S1 (1993), S85–S118.

[18] Jakob Grazzini, Matteo G. Richiardi, and Mike Tsionas. 2017. Bayesian estimation of agent-based models. *Journal of Economic Dynamics and Control* 77 (2017), 26–47. https://doi.org/10.1016/j.jedc.2017.01.014

[19] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[20] Geoffrey Grimmett and David Stirzaker. 2001. *Probability and random processes*. Oxford university press.

[21] Stanislao Gualdi, Marco Tarzia, Francesco Zamponi, and Jean-Philippe Bouchaud. 2015. Tipping points in macroeconomic agent-based models. *Journal of Economic Dynamics and Control* 50 (2015), 29–61.

[22] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

[24] Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. 2022. An optimization-centric view on Bayes' rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research* 23, 132 (2022), 1–109.

[25] Ignacio N. Lobato and Carlos Velasco. 2000. Long Memory in Stock-Market Trading Volume. *Journal of Business & Economic Statistics* 18, 4 (2000), 410–427. http://www.jstor.org/stable/1392223

[26] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[27] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. 2020. Monte Carlo Gradient Estimation in Machine Learning. *The Journal of Machine Learning Research* 21, 1 (2020), 5183–5244.

[28] Corrado Monti, Marco Pangallo, Gianmarco De Francisci Morales, and Francesco Bonchi. 2023. On learning agent-based models from data. *Scientific Reports* 13, 1 (2023), 9268.

[29] Mijung Park, Wittawat Jitkrittum, and Dino Sejdinovic. 2016. K2-ABC: Approximate Bayesian Computation with Kernel Embeddings. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 51)*, Arthur Gretton and Christian C. Robert (Eds.). PMLR, Cadiz, Spain, 398–407. https://proceedings.mlr.press/v51/park16.html

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

[31] Donovan Platt. 2020. A comparison of economic agent-based model calibration methods. *Journal of Economic Dynamics and Control* 113 (2020), 103859.

[32] Donovan Platt. 2021. Bayesian Estimation of Economic Simulation Models using Neural Networks. *Computational Economics* (2021), 1–52.

[33] Felix Prenzel, Rama Cont, Mihai Cucuringu, and Jonathan Kochems. 2022. Dynamic Calibration of Order Flow Models with Generative Adversarial Networks. In *Proceedings of the Third ACM International Conference on AI in Finance* (New York, NY, USA) *(ICAIF '22)*. Association for Computing Machinery, New York, NY, USA, 446–453. https://doi.org/10.1145/3533271.3561777

[34] Arnau Quera-Bofarull, Ayush Chopra, Anisoara Calinescu, Michael Wooldridge, and Joel Dyer. 2023. Bayesian calibration of differentiable agent-based models. *ICLR Workshop on AI for Agent-based Modelling* (2023).

[35] Arnau Quera-Bofarull, Joel Dyer, Anisoara Calinescu, J. Doyne Farmer, and Michael Wooldridge. 2023. BlackBIRDS: Black-Box Inference foR Differentiable Simulators. *Journal of Open Source Software* 8, 89 (2023), 5776. https://doi.org/10.21105/joss.05776

[36] Arnau Quera-Bofarull, Joel Dyer, Anisoara Calinescu, and Michael Wooldridge. 2023. Some challenges of calibrating differentiable agent-based models. *ICML Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators* (2023).

[37] Danilo Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1530–1538. https://proceedings.mlr.press/v37/rezende15.html

[38] Victor Storchan, Svitlana Vyetrenko, and Tucker Balch. 2021. Learning who is in the market from time series: market participant discovery through adversarial calibration of multi-agent simulators. *arXiv preprint arXiv:2108.00664* (2021).

[39] Esteban G Tabak and Cristina V Turner. 2013. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics* 66, 2 (2013), 145–164.

[40] Esteban G Tabak and Eric Vanden-Eijnden. 2010. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences* 8, 1 (2010), 217–233.

[41] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

Received July 24, 2023; accepted September 28, 2023