Journal of Economic Dynamics & Control 125 (2021) 104082

Contents lists available at ScienceDirect

Journal of Economic Dynamics & Control

journal homepage: www.elsevier.com/locate/jedc

Estimation of agent-based models using Bayesian deep learning approach of BayesFlow

Takashi Shiono[a,b]

[a] Economics Research, Credit Suisse Securities (Japan) Limited, Tokyo, Japan
[b] Graduate School of Economics, The University of Tokyo, Japan

ARTICLE INFO

Article history:
Received 1 July 2020
Revised 22 January 2021
Accepted 25 January 2021
Available online 10 February 2021

Keywords:
Agent-based model
Bayesian inference
Deep learning
Parameter estimation
Neural network

ABSTRACT

This study examines the possibility of applying the novel likelihood-free Bayesian inference called BayesFlow proposed by Radev et al. (2020) for the estimation of agent-based models (ABMs). BayesFlow is a fully likelihood-free approach, which directly approximates a posterior rather than a likelihood function by learning an invertible probabilistic mapping between parameters and standard Gaussian variables, conditional on simulation data from the ABM to be estimated. BayesFlow certainly achieved superior accuracy to the benchmark method of Kernel Density Estimation-MCMC of Grazzini et al. (2017) and the more sophisticated method of Mixture Density Network-MCMC of Platt (2019), in the validation tests of recovering the ground-truth values of parameters from the simulated datasets of a standard New Keynesian ABM (NK-ABM). Furthermore, the truly empirical estimation of NK-ABM with the real data of the US economy successfully showed the desirable pattern of posterior contraction along with the increase in observation periods. This deep neural network-based method holds general applicability without any critical dependence on pre-selected design and high computational efficiency. These features are desirable when scaling the method to practical-sized ABMs, which typically have high-dimensional parameters and observation variables.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Agent-based models (ABMs) have been widely used in the field of artificial markets and related studies to investigate the microstructure of financial markets. In recent years, they have also been increasingly adopted in macroeconomic analysis as an alternative to DSGE models. According to Grazzini et al. (2017), ABMs are characterised by the following three features: 1) there are a multitude of agents that interact with each other and the environment, (2) these agents are autonomous, in the sense that there is no central coordinator such as a Walrasian auctioneer or the concealed time axis in which the central coordinator works, and (3) aggregation is performed numerically.

Thus, an ABM allows for extreme flexibility in setting the behavioural patterns of each agent in a heterogeneous manner. There is no prerequisite for rational expectation or (ex-ante) market equilibrium, as the agents' adaptive processes of learning and selection are explicitly modelled. This assumption regarding an agent's behaviour is a fundamental difference between DSGE and ABM. Recent studies of agent-based macroeconomic modelling such as Assenza et al. (2015) and Caiani et al. (2016) have succeeded in reproducing stylised macroeconomic phenomena such as the Phillips curve and the

E-mail address: tak-shiono@g.ecc.u-tokyo.ac.jp

https://doi.org/10.1016/j.jedc.2021.104082
0165-1889/© 2021 Elsevier B.V. All rights reserved.

Okun's law from the simulations of ABMs with adaptive agents. In other words, a rational agent was not a necessary condition for these observed phenomena of the macro-economy. Furthermore, their models precisely describe the relationship between the financial sector and the real economy by explicitly modelling the balance sheet for each economic agent, which are, therefore, useful in discussing the impact on the real economy from prudent policies on financial institutions. It is because of their flexibility that ABMs are increasingly being adopted for their usefulness as a complement to DSGE and will continue to be applied to various economic phenomena.

However, it has often been pointed out that an ABM has its weaknesses in the lack of empirical validation (Gallegati and Richiardi, 2009). The parameters in an ABM are usually calibrated manually to make the model's simulation outputs reproduce some particular characteristics of economic observables (e.g. fat-tail distribution or volatility clustering in stock returns). In response to this challenge, studies that statistically estimate the parameters of ABM have been vigorously explored. Lux and Zwinkels (2018) reviewed the development of empirical literature on ABMs over the past decade.

First, the prominent work of Ghonghadze and Lux (2016) estimated parameters using the Generalised Method of Moment (GMM) for a small-scale ABM, where the dynamic evolution of the agent's microstates can be described analytically using the Fokker-Planck equations. Lux (2018) applied the Sequential Monte Carlo (SMC) method to the same class of models where the likelihood is analytically tractable. Although such approaches are straightforward because they directly adopt the established procedure of GMM or SMC, they can only be applied to the ABMs whose dynamic behaviour and likelihood can be described in the closed form. Such models must be small and restrictive. In other words, the advantages of ABM such as flexibility and variety of agents' behaviour are lost.

Grazzini and Richiardi (2015) surveyed the growing literature on ABM estimation at that time and defined the framework of Simulated Minimum Distance (SMD). This is a method of determining the parameters to minimise the pre-selected distance between the actual data and the aggregate variables generated from the ABM simulation. The distance is typically defined by specific moment values (Method of Simulated Moment: MSM), or the parameter values of a meta-model such as Vector Auto-Regression estimated using the actual data and simulated data (Indirect Inference, II). This method is applicable to ABMs with intractable likelihood as long as the stationarity of simulated data by a model can be ensured.

Regarding the SMD method, Grazzini et al. (2017) pointed out that the estimator's sensitivity to an arbitrary pre-selected design is a critical weakness. When applying the SMD framework, researchers need to pre-select certain moments or meta-models, and the estimation accuracy is critically dependent on the adequacy of such pre-selection or handcrafted design. For example, if the ABM includes a parameter that only affects the mean of the simulated data and another parameter that only affects the variance, the estimation results differ greatly depending on which moments are selected as summary statistics. Moreover, the requirement of stationarity in the simulated data by ABMs limits the scope of the applicable class of models.

Addressing this arbitrariness, Kukacka and Barunik (2017) adopted the non-parametric simulated maximum likelihood (NPSML) estimation originally proposed by Kristensen and Shin (2012). It utilises non-parametric kernel smoothing over multiple i.i.d. draws of simulated time series from ABMs to construct an approximate time-by-time conditional density. This non-parametric approach has virtues[1] of avoiding critical dependency on the arbitrary pre-selections (except the lag orders of autocorrelation in conditioning variables) and dispensing with the assumption of ergodicity on ABMs. Meanwhile, it is computationally heavy (compared with the methods assuming ergodicity), because a large number of ABM simulation runs has to be implemented by every single updating step of parameter exploration for maximum likelihood. Further, the choice of optimisation method is another concern given the simulation-based nature of the approximated likelihood; heuristic methods need to be selected since gradient-based optimisations are not applicable.

Another direction of sophistication is to reduce the large computational burden by replacing costly ABM simulations with computationally light outputs from a surrogate model. Lamperti et al. (2018) proposed an approach to create a fast surrogate model using a limited number of ABM simulations and approximated the nonlinear relationship between the parameters and outputs of ABM via machine learning techniques. The actual exploration of the parameters was conducted over the surrogate model. This is a good performing and widely applicable calibration technique, but not based on the formalism of statistical inference, and requires many pre-selections critical to estimation accuracy.

While most of the above approaches are frequentist, ABM estimation via the Bayesian approach has also been explored recently. Grazzini et al. (2017) proposed three types of Bayesian inferences with the approximation of the model likelihood. First, they approximated the likelihood by smoothing the histogram of simulated data generated from an ABM with a set of candidate parameters via Kernel Density Estimation (KDE), and subsequently evaluated the probability of the real observation data (1. Non-parametric Bayesian). This method does not require pre-selection of moments or meta-models. However, as the dimension of observables in the ABM increases, the likelihood calculation with KDE becomes infeasible owing to the curse of dimensionality. The study also proposed two other computationally cheaper methods: one is a method to parameterise the steady-state distribution of the simulation data with a normal distribution (2. Parametric Bayesian), and the other is to approximate the likelihood with the 1–0 indicator function which assesses whether the relation between actual data and simulation data meets certain pre-selected criteria or not (3. Approximate Bayesian Computation). However, both the methods require the same sort of discretionary pre-selections as the SMD method. Moreover, the requirement of ergodicity reduces the flexibility of ABMs applicable to the parameter estimation.

[1] They also insist that although a larger number of simulation data has to be drawn for the kernel smoothing when dealing with higher dimensional observation variables, the NPSML is basically free from curse of dimensionality because the summation of the log-densities in the computation of total log likelihood serves as an additional smoothing device.

In terms of the advantages of Bayesian approaches with respect to frequentist approaches, Grazzini et al. (2017) insisted that 1) Bayesian methods generally do not require pre-selection of moments or meta-models, and 2) they allow the incorporation of prior information, allowing a proper statistical treatment of the uncertainty of our knowledge, and how it is updated given the available observations.

Furthermore, Platt (2020) has demonstrated that as a result of equal-footing comparisons, the Bayesian method (KDE-MCMC) of Grazzini et al. (2017) outperformed a number of sophisticated frequentist approaches, and concluded that its superiority can be attributed to a combination of the flexible likelihood approximation by KDE and the adoption of a Bayesian paradigm. The study argued that more interest should be placed on the development of Bayesian ABM estimation.

The methodological propriety of KDE-MCMC of Grazzini et al. (2017) still essentially depends on the assumption of ergodicity of ABMs, which most ABMs of practical interest are unlikely to hold. To overcome this limitation of KDE-MCMC by allowing for temporal dependencies (non-ergodicity) to be taken into account, Platt (2019) proposed a Bayesian estimation protocol that uses a neural-network-based approximation of conditional density (Mixture Density Network: MDN). It reported compelling performances in the validation tasks of recovering the ground-truth values of parameters from the simulated datasets against the KDE-MCMC of Grazzini et al. (2017). This confirmed the benefit of incorporating possible temporal dependencies in simulated time series of ABMs. One potential shortcoming of the MDN method is the huge computational burden, because it requires multiple runs of ABM simulations to train deep neural networks by every single step of a likelihood approximation.

In addition to the successes of the Bayesian methods, the discussion of a singular statistical model in the statistical learning theory may support the use of the Bayesian approach in ABM estimation. It is proved that if a stochastic model does not hold a regularity condition (i.e. if the Fisher information matrix is singular), the Cramer-Rao inequality has no meaning, and the maximum likelihood estimator is not the asymptotically best estimator (Watanabe, 2009). In this case, the Bayesian approach has more desirable properties than the maximum likelihood approach in general. It could be more beneficial to explore a Bayesian approach, because there is no guarantee that a general class of ABMs has a log-likelihood function with a positive definite Fisher information matrix. Hence, following Platt (2020), it is argued that more interests are worthwhile to be placed on Bayesian methods.

Against this background, the present study adopts a novel likelihood-free Bayesian inference called *BayesFlow* proposed by Radev et al. (2020) for the statistical estimation of ABMs. The core motivation to adopt this new method is its scalability, in the sense that the desirable properties shown with a small ABM can seamlessly be reproduced when applied to a larger ABM, with an efficient increase in computational burden.

Obstacles to the scalability of an estimation method typically involve the following points. First, justifying strong assumptions on simulated time series of ABMs (such as stationarity or parametric densities) becomes more difficult, because a larger ABM tends to have more complex features in its output time series. Second, the estimator's high sensitivity to arbitrary pre-selections (i.e. summary statistics, moments, meta-model, etc.) could be another hurdle to scalability, as appropriate choices are thought to become more difficult when dealing with a larger ABM. Finally, an expansion in the computational burden with an increased ABM size should not be practically prohibitive. The so-called curse of dimensionality in the non-parametric density approximation (such as KDE) is a typical example of this obstacle. As KDE requires progressively larger draws of simulational data when dealing with higher-dimensional observation variables, and a larger ABM inevitably requires more time for a single simulation run, an estimation would become practically (quantitatively) infeasible without the use of high-performance computing equipment.

BayesFlow appears better for dealing with these three obstacles to scalability. BayesFlow is a fully likelihood-free approach, which directly approximates a Bayesian posterior rather than a likelihood function. This method has general applicability, requiring only the ability to output simulation data from a mathematical forward model, and has a theoretical guarantee for sampling from the true posterior without any specific assumption on the shape of the target posterior or prior. Hence, it does not need to presume ergodicity or stationarity of simulated time series of a model. Radev et al. (2020) have showed high accuracies of BayesFlow even in the cases of potentially chaotic (the Ricker population model) and non-ergodic (the Levy-Flight model) mathematical models.

Second, in contrast to the typical likelihood-free methods such as approximate Bayesian computation, BayesFlow does not involve discretionary pre-selections or handcrafted design in the critical parts of inference. It accompanies a learnable summary network which can compress variable-length potentially large dimensional observation time series into fixed-length small dimensional summary statistics. In other words, a researcher does not need to pre-select specific moments or meta-models from specific observables as handcrafted summary statistics.

Finally, it is computationally efficient, particularly in the case that repeated inferences with different observation datasets are required. BayesFlow realises *amortised inference*, where estimation is split into a computationally intensive training phase and a much cheaper inference phase. In the training phase, BayesFlow tries to learn neural networks to output an approximate posterior that works properly for any possible observation sequence. Evaluating the trained model over a specific observation dataset (e.g. real data) is computationally cheap; therefore, the upfront training efforts amortise over multiple inferences. This amortised inference is a clear advantage with respect to the MCMC-based methods, which incur significant computational costs in collecting posterior samples, because some ABM simulations need to be run by every single step of exploring parameters. Dimensionality reduction implemented by the aforementioned summary network also contributes to reducing the computational burden of handling high-dimensional observations, in contrast to the KDE-based methods.

In the next section, I explain the basic structure of BayesFlow compared with other related methodologies. Subsequently, the procedure and results of validation are presented in Section 3, before discussing the advantages and limitations with the conclusion of the study.

2. Methods

2.1. Notation

In this section, I denote the data simulated from the ABM as $X_{1:T} \equiv (X_1, \ldots, X_T) \in \mathbb{R}^K \times, \ldots, \times \mathbb{R}^K$, where individual X_t is a vector of observable variables in a model with its dimension denoted as K. The number of observation points in a dataset is denoted as T to make it clear that simulation outputs from the ABM are usually multivariate time series. Simulated data is also expressed as $X_{1:T}(\theta)$ when it needs to be emphasised that the data are generated from the ABM with parameters θ. Actual observation data or test data (i.e. simulated pseudo-observations) are expressed with a superscript, $X^o_{1:T}$. Parameters of a forward model (i.e. the ABM) are represented as a vector $\theta \in \mathbb{R}^d$.

2.2. Agent-based model

The state of the whole system of an ABM at time t is described by the collection of all microstates of individual agent i in time period t as $S_t \equiv \{s_{i,t}\}_{i=1}^N$. The evolution or the law of transition of each agent's state is expressed as

$$s_{i,t} = \mathrm{f}_i(S_{t-1}, \xi_t, \theta), \tag{1}$$

where f_i is an agent-wise state-transition function, taking values in $\mathbb{R}^L$. $\xi_t \equiv \{\zeta_{i,t}\}_{i=1}^N$ is a vector to bundle all stochastic elements, which are supposed to be generated from the known distributions specified by a researcher according to the structure of the ABM.[2] The function f reflects the detailed modelling of an agent's learning, selection, and interaction behaviour in ABM, which are typically heterogeneous and accompany discontinuities such as *if-else* statements.

Aggregate observable variables X_t are then defined over S_t:

$$X_t = \mathrm{m}(S_t), \tag{2}$$

where a function m aggregates and transforms the collection of microstates into observable variables X_t.

Combining the state transition f_i in Eq. (1) and the observation m in Eq. (2), a simulation data generation function G of the ABM can be defined as follows:

$$X_{1:T} = G(\theta, \xi_{1:T}) \text{with } \xi_{1:T} \sim p(\xi). \tag{3}$$

This function G effectively corresponds to a single run of the ABM simulation with the parameters θ and the stochastic elements $\xi_{1:T}$.

2.3. Bayesian inference on agent-based model

In Bayesian inference, the posterior defined below contains all information about θ extractable from a series of observation variables $X_{1:T}$:

$$p(\theta|X_{1:T}) = \frac{p(X_{1:T}|\theta)p(\theta)}{\int p(X_{1:T}|\theta)p(\theta)d\theta}.$$

Even when a closed-form expression of the posterior is unobtainable, various sampling schemes from the posterior such as MCMC or SMC are applicable, as long as the likelihood of a forward model $p(X_{1:T}|\theta)$ can easily be evaluated by the actual observations $X^o_{1:T}$ together with any prior $p(\theta)$, such that

$$p\left(\theta|X^o_{1:T}\right) \propto \mathcal{L}\left(\theta; X^o_{1:T}\right) p(\theta).$$

In case of the most practical-scale ABMs, however, the likelihood function $\mathcal{L}(\theta; X^o_{1:T}) \equiv p(X^o_{1:T}|\theta)$ is generally intractable; in other words, it is not available in closed form. This is the central challenge in Bayesian inference of an ABM. The existing study by Grazzini et al. (2017) attempted to approximate the likelihood function and apply standard posterior sampling schemes such as MCMC with the approximated likelihood. They limited their scope to ergodic ABMs to ensure that the simulation time series generated from a model remains stationary around the deterministic steady-state level $g^*(\theta)$ as

$$X_{1:T} = \{g^*(\theta) + \epsilon_t\}_{t=1}^T,$$

where ϵ_t is a vector of disturbances (representing measurement errors, specification errors, etc.).

In this case, the likelihood of the observation series $X_{1:T}$ is simply a product of a time-invariant density function: $p(X_{1:T}|\theta) = \prod_t f(X_t|\theta)$.

[2] Initial microstates $S_0 = \{s_{i,0}\}_{i=1}^N$ could be included in parameters or sampled from the given distribution as stochastic elements.

One of their approaches is to approximate this density function by (1) a non-parametric way of KDE: $f(X_t|\theta) \approx \hat{f}(X_t|\theta) = KDE(X_{1:T}(\theta))$. This is simply a histogram smoothing of the simulation data generated from the ABM with a set of candidate parameters. Another approach proposed by the study is to use (2) a multivariate Gaussian density $f(X_t|\theta) \approx N(g^*(\theta), \sigma_\epsilon^2 \mathbb{I}_K)$ with additive disturbances, where $\mathbb{I}_K$ denotes an identity matrix with a dimension of subscript K, . The use of a parametric distribution can clearly reduce computational costs, particularly in the case of high-dimensional observations at the expense of expressive power (i.e. more bias to the true density), which could be prohibitive in the KDE method. In both cases, the likelihood of the parameters is calculated by evaluating the approximate density at the observed data points: $\mathcal{L}(\theta; X^o_{1:T}) = \prod_t \hat{f}(X^o_t|\theta)$. They also applied (3) approximate Bayesian computation that directly approximates a likelihood function using a 1–0 indicator function: $p(X^o_{1:T}|\theta) \approx \mathbb{I}(d[\mu(X_{1:T}(\theta)), \mu(X^o_{1:T})] < h)$, where $\mu(\cdot)$ is a summary statistic, $d[\cdot,\cdot]$ is a distance measure, and h is a threshold for distance. This approach could potentially be applicable to the wider class of ABMs with a relatively lesser computational burden than KDE, only if pre-selections or handcrafted design of an indicator function is appropriate to approximate the target likelihood. However, such a good pre-selection is difficult in practice.

Recently, Platt (2019) has introduced a sophisticated Bayesian method of an MDN. It takes temporal dependencies within the observation time series into account through the compound of approximated conditional densities, so that the ergodicity of the ABM (or stationarity of the simulated time series) needs not be assumed. Illustratively, the method fits a stochastic autoregressive model with a lag order of L to the simulated time series of the ABM, where the stochastic model is constructed as an M-th Gaussian mixture re-parameterised by neural networks:

$$\hat{f}(X_t|X_{t-L:t-1};\theta) = \sum_m \alpha_m(X_{t-L:t-1})\ N\big(X_t|\mu_m(X_{t-L:t-1}), \sigma_m^2(X_{t-L:t-1})\mathbb{I}_K\big),$$

where α_m, μ_m, and σ_m are the outputs from deep neural networks taking $X_{t-L:t-1}$ as the input. Once the above neural network model is trained by R draws of the simulated time series, the likelihood of actual observation data is evaluated by the model as

$$\mathcal{L}\big(\theta; X^o_{1:T}\big) = \prod_{t=1}^{T-L} \hat{f}\big(X^o_{t+L}|X^o_{t:t+L-1};\theta\big).$$

The MDN-MCMC method repeats this procedure of neural-net fitting and a likelihood evaluation for every single step of updating parameters using the Metropolis-Hastings algorithm.

2.4. Likelihood-free approach: BayesFlow

In this study, I adopt a novel likelihood-free Bayesian inference of *BayesFlow* proposed by Radev et al. (2020). BayesFlow appears to have three advantages: 1) general applicability, 2) automatic adjustment of summary statistics, and 3) high computational efficiency.

BayesFlow is a fully likelihood-free approach, which directly approximates a posterior function $p(\theta|X_{1:T})$, rather than a likelihood function $p(X_{1:T}|\theta)$. It does not need to presume the ergodicity of the ABM or the stationarity of the simulated time series. This likelihood-free approach only requires the ability to output simulation data from a forward model, which generates samples of observable variables by a deterministic function G of parameters θ and independent noises ξ with known distributions as follows:

$$X_t \sim p(X|\theta) \Leftrightarrow X_t = g(\theta, \xi_t) \text{with} \xi_t \sim p(\xi),$$

or T samples simultaneously as:

$$X_{1:T} \sim p(X_{1:T}|\theta) \Leftrightarrow X_{1:T} = G(\theta, \xi_{1:T}) \text{with} \xi_{1:T} \sim p(\xi). \tag{4}$$

Here, the likelihood $p(X_{1:T}|\theta)$ is only implicitly defined and is not available in closed form. Therefore, BayesFlow can be seamlessly applied to an ABM since the simulation data generation function G of the ABM in Eq. (3) meets this sampling requirement.

The goal of BayesFlow is to approximate a target posterior using a parameterised approximate posterior as accurately as possible:

$$p(\theta|X_{1:T}) \approx p_\phi(\theta|X_{1:T}).$$

BayesFlow utilises a conditional invertible neural network (cINN) to achieve this objective. cINN constitutes an invertible function $f_\phi : \mathbb{R}^d \to \mathbb{R}^d$ parameterised by a vector of parameters ϕ, for which the inverse $f_\phi^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ exists. The approximate posterior p_ϕ is then re-parameterised in terms of a cINNf_ϕ that implements a normalising flow (Rezende and Mohamed, 2015) between θ and a standard Gaussian latent variable z:

$$\theta \sim p_\phi(\theta|X_{1:T}) \Leftrightarrow \theta = f_\phi^{-1}(z; X_{1:T}) \text{with}\ z \sim N(z|0, \mathbb{I}_d).$$

The cINN is to be trained so that the outputs of its inverse f_ϕ^{-1} follow the target posterior $p(\theta|X_{1:T})$.

2.5. Learning the posterior

Thus, the training objective for cINN is to minimise the Kullback-Leibler (KL) divergence between the target and the model-induced approximate posterior for all possible series of observable variables $X_{1:T}$.

$$\begin{aligned}\hat{\phi} &= \underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_{X_{1:T}\sim p(X;T)}\Big[\mathbb{KL}\Big[p(\theta|X_{1:T})||p_\phi(\theta|X_{1:T})\Big]\Big] \\ &= \underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_{X_{1:T}\sim p(X;T)}\Big[\mathbb{E}_{\theta\sim p(\theta|X_{1:T})}\Big[\log p(\theta|X_{1:T}) - \log p_\phi(\theta|X_{1:T})\Big]\Big] \\ &= \underset{\phi}{\operatorname{argmax}}\, \mathbb{E}_{X_{1:T}\sim p(X;T)}\Big[\mathbb{E}_{\theta\sim p(\theta|X_{1:T})}\Big[\log p_\phi(\theta|X_{1:T})\Big]\Big] \\ &= \underset{\phi}{\operatorname{argmax}} \iint p(X,\theta;T)\log p_\phi(\theta|X_{1:T})dX_{1:T}d\theta\end{aligned}$$

Further, since the forward transmission of cINN outputs a standard Gaussian latent variable $f_\phi(\theta;X_{1:T}) = z$by definition, the density transformation law of the random variable enables re-parameterisation of the approximate posterior p_ϕ in terms of cINNf_ϕ as follows:

$$p_\phi(\theta|X_{1:T}) = p\big(z = f_\phi(\theta;X_{1:T})\big)\left|\det\left(\frac{\partial f_\phi(\theta;X_{1:T})}{\partial\theta}\right)\right|.$$

This is the fundamental operation of a normalising flow. Incorporating this fact, the training objective can be rewritten as

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \iint p(X,\theta;T)\Big\{\log p\big(f_\phi(\theta;X_{1:T})\big) + \log\left|\det J_{f_\phi}\right|\Big\}dX_{1:T}d\theta, \tag{5}$$

where J_{f_ϕ}stands for $\partial f_\phi(\theta;X_{1:T})/\partial\theta$ (the Jacobian of f_ϕ evaluated at θ and $X_{1:T}$).

Even without a closed-form likelihood, it is possible to generate samples from $(\theta^{(j)}, X_{1:T}^{(j)}) \sim p(X,\theta;T)$ with a forward model G and the prior $p(\theta)$ as shown in Eq. (4). Utilising the independent M sets of data-generating parameters and corresponding simulated data $\{(\theta^{(j)}, X_{1:T}^{(j)})\}_{j=1}^M$, the expectation in Eq. (5) is approximated using the Monte Carlo estimate as follows:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \frac{1}{M}\sum_{j=1}^{M}\log p\left(f_\phi\left(\theta^{(j)};X_{1:T}^{(j)}\right)\right) + \log\left|\det J_{f_\phi}^{(j)}\right|. \tag{6}$$

By taking the negative value of Eq. (6) and using the fact that $\log N(z|0,\mathbb{I}_d) \propto -\frac{1}{2}\|z\|_2^2$, the training objective for cINN becomes

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ \mathcal{L}(\phi)$$

with

$$\mathcal{L}(\phi) = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{\|f_\phi\left(\theta^{(j)};X_{1:T}^{(j)}\right)\|_2^2}{2} - \log\left|\det J_{f_\phi}^{(j)}\right|\right). \tag{7}$$

$\mathcal{L}(\phi)$ is a loss function for this posterior approximation task, which can be minimised using any stochastic gradient descent algorithm.

2.6. Summary network

When training a cINN with simulated datasets, Radev et al. (2020) recommended to use a summary network f_ψ to construct an estimate of sufficient statistics that captures all information about θ contained in $X_{1:T}$ into a fixed-size representation (i.e. a filtered vector of latent features) $\tilde{X} = f_\psi(X_{1:T})$. The advantages of using a summary network are twofold. First, BayesFlow is generalised to data with variable length T, which is particularly important for empirical applications. Furthermore, the training of cINN could be more efficient and precise with dimensionality reduction (i.e. signal extraction). Observation data tend to contain some redundancies or noises without any pre-selection or pre-processing by hand. In that sense, the introduction of a summary network effectively automates the adjustment of summary statistics or the structural design of the approximator, which obviously contributes to the scalability of the method.

BayesFlow is designed to use bidirectional long short-term memory (bi-LSTM) (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) as a summary network for time-series data. A standard LSTM network is well known to be able to deal effectively with serial data exhibiting long-memory (i.e. non-ergodic and non-stationary features) and non-linearity, such as natural language sentences or voice data. This is achieved by incorporating a 'memory block', which consists of four

elements: a self-connected core, an input gate, a forget gate, and an output gate. The core simply repeats a cycle in which the current state is expelled to become an input for the next period and, thus, 'remember' the same information in perpetuity in the absence of intervention by an input gate or a forget gate. The input, forget, and output gates receive new information from a bundle of an exogenous input vector and a previous output from the memory block; and decide whether or not to retain the information of the core (by input gate), for how long to hold the information (by forget gate), and when to pass the information outside the block (by output gate) so as to achieve an optimal fit to the training sequences. Bidirectional LSTM simply puts two independent LSTMs together. This bidirectional structure allows the networks to have both backward (from future to past) and forward (from past to future) information about the input sequence at every time step, enhancing the capturing performance on the temporal dependence structure (i.e. context) in sequences.

The original study of BayesFlow by Radev et al. (2020) has reported high accuracies of the method even in the difficult tasks to estimate parameters on potentially chaotic (the Ricker population model) and non-ergodic (the Levy-Flight model) mathematical models,[3] owing to the use of a bi-LSTM summary network. In the context of ABM estimation, this feature of BayesFlow can unleash the full flexibility of an ABM by its universal applicability (as long as target parameters are structurally identifiable from observation variables).

The parameters of the summary network were jointly learned with those of the cINN. Hence, the training objective is finalised as follows:

$$\hat{\phi}, \hat{\psi} = \underset{\phi,\psi}{\operatorname{argmax}} \, \mathbb{E}_{X_{1:T}\sim p(X;T)}\left[\mathbb{E}_{\theta\sim p(\theta|X_{1:T})}\left[\log p_\phi\left(\theta \,|\, f_\psi\left(X_{1:T}\right)\right)\right]\right]$$

$$= \underset{\phi,\psi}{\operatorname{argmin}} \, \mathcal{L}(\phi, \psi)$$

with

$$\mathcal{L}(\phi, \psi) = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{\| f_\phi\left(\theta^{(j)}; f_\psi\left(X_{1:T}^{(j)}\right)\right)\|_2^2}{2} - \log\left|\det J_{f_\phi}^{(j)}\right|\right). \tag{8}$$

2.7. Structure of invertible networks

The cINN is constructed as a chain of multiple conditional affine coupling blocks (cACBs). The idea of an ACB was originally introduced by Dinh et al. (2017), which implements an invertible nonlinear transformation: $f_{acb} : \mathbb{R}^d \to \mathbb{R}^d$ and $f_{acb}^{-1} : \mathbb{R}^d \to \mathbb{R}^d$. Each ACB consists of four separate fully connected neural networks denoted as $s_1(\cdot)$, $s_2(\cdot)$, $t_1(\cdot)$, and $t_2(\cdot)$. These internal networks need not be invertible because they are only evaluated in the forward direction during both the forward and inverse operations of an ACB. By denoting the input vector of f_{acb} as U and the output vector as V, the forward and inverse transformations of the vectors are expressed as $f_{acb}(U) = V$ and $f_{acb}^{-1}(V) = U$, respectively. This invertibility is realised by splitting the input vector into two parts $U = (U_1, U_2)$ with $U_1 = u_{1:d/2}$ and $U_2 = u_{d/2+1:d}$, and performing the following operations on the split input:

$$\begin{aligned} V_1 &= U_1 \odot \exp\left(s_1(U_2)\right) + t_1(U_2) \\ V_2 &= U_2 \odot \exp\left(s_2(U_1)\right) + t_2(U_1) \\ V &= (V_1, V_2) \end{aligned}$$

where $\odot$ represents element-wise multiplication. Here, the inverse operation is given as follows:

$$\begin{aligned} U_1 &= (V_2 - t_2(U_1)) \odot \exp\left(s_2(U_1)\right) \\ U_2 &= (V_1 - t_1(U_2)) \odot \exp\left(s_1(U_2)\right) \end{aligned}$$

This formulation ensures that the Jacobian of cINN is a strictly upper or a lower triangular matrix and, therefore, its determinant ($\det J_{f_\phi}$) is quite cheap to compute, which is an important feature when using it in a normalising flow.

Subsequently, the ACB is augmented to take the summary statistics $\tilde{X}$ as a conditioning input, so as to switch the pattern of bidirectional transformations along with the values of observations $X_{1:T}$ as follows:

$$\begin{aligned} V_1 &= U_1 \odot \exp\left(s_1\left(U_2, \tilde{X}\right)\right) + t_1\left(U_2, \tilde{X}\right) \\ V_2 &= U_2 \odot \exp\left(s_2\left(U_1, \tilde{X}\right)\right) + t_2\left(U_1, \tilde{X}\right). \end{aligned}$$

This structure is a cACB. BayesFlow stacks multiple cACBs to make the entire neural network architecture (i.e. cINN) expressive enough to implement a potentially complex mapping between the d- dimensional vector of parameters θ and the same dimensional vector of unit Gaussian variables z. Eventually, the entire cINN is expressed as a function $z = f_\phi(\theta; \hat{X})$, together with the inverse operation $\theta = f_\phi^{-1}(z; \hat{X})$.

[3] They tested accuracy of recovering the ground-truth parameter values from out-of-sample simulation data.

2.8. Amortised inference

For most Bayesian inference algorithms, the entire estimation process must be repeated from scratch when dealing with different observation sequences (e.g. $X^{(i)}_{1:T}$ and $X^{(j)}_{1:T}$ with $i \neq j$). In contrast, BayesFlow realises *amortised inference*, where estimation is split into a computationally expensive training phase and a much cheaper inference phase. In the training phase, BayesFlow tries to learn neural networks to output an approximate posterior $\widehat{p_\phi}(\theta|X_{1:T})$ that works well for any possible observation sequence $X_{1:T}$. cINN is trained up front so that its inverse operation outputs samples from an approximate posterior given observations: $f^{-1}_\phi(z|X^o_{1:T}) = \hat{\theta} \sim p_\phi(\theta|X^o_{1:T})$ with $z \sim N(0, \mathbb{I}_d)$. Hence, evaluating the trained model over a specific observation dataset $X^o_{1:T}$ is computationally very challenging; therefore, the upfront training efforts amortise over multiple inferences.

Putting it all together, a whole procedure of Bayesian inference with BayesFlow is summarised as Algorithm 1.

Algorithm 1
Bayesian inference with BayesFlow

Algorithm 1.

Training (with online simulation data generations)
repeat
  Sample sequence length of observations: $T \sim U(T_{min}, T_{max})$
  Sample a batch of parameters from prior: $\{\theta^{(j)}\}_{j=1}^{M} \sim p(\theta)$
  Simulate M data sets size T via the data generation function Eq.(4): $\left\{X^{(j)}_{1:T} = G\left(\theta^{(j)}, \xi_{1:T}\right)\right\}_{j=1}^{M}$
  Pass $\left\{X^{(j)}_{1:T}\right\}_{j=1}^{M}$ into summary network to obtain summary statistics: $\left\{\tilde{X}^{(j)} = f_\psi\left(X^{(j)}_{1:T}\right)\right\}_{j=1}^{M}$
  Pass $\left\{\left(\theta^{(j)}, X^{(j)}_{1:T}\right)\right\}_{j=1}^{M}$ into cINN to obtain $\left\{z^{(j)} = f_\phi\left(\theta^{(j)}; \tilde{X}^{(j)}\right)\right\}_{j=1}^{M}$
  Compute loss according to Eq.(8)
  Update neural network parameters ϕ, ψ via backpropagation
until convergence to $\hat{\phi}, \hat{\psi}$

Inference (given observed or test data X^o)
Compute summary of the data $\tilde{X}^o = f_{\hat{\psi}}(X^o_{1:T})$
for $l = 1, \ldots, L$ **do**
  Sample $z^{(l)} \sim N(0, \mathbb{I}_d)$
  Compute inverse $\theta^{(l)} = f^{-1}_\phi\left(z^{(l)}; \tilde{X}^o\right)$
end
Return $\{\theta^{(l)}\}_{l=1}^{L}$ as a sample from $p(\theta|X^o)$

3. Experiments

3.1. Training

All networks were implemented in Python using the *PyTorch* library and trained on a single-GPU machine equipped with an NVIDIA(R) GTX1050Ti graphics card. The stochastic gradient descent was implemented by the Adam optimiser with the default setting of *the PyTorch* package (learning rate of 0.001). Following the original study of BayesFlow (Radev et al., 2020), an *online learning* approach is adopted, where data are simulated from ABMs on demand. The generalisation loss of Eq. (8) (i.e. the KL-divergence between a target posterior and a model-induced approximate posterior) is expected to decrease with increasing update steps with i.i.d. draws of a fresh pair of parameter vectors and simulation data. Together with the fact that deep neural networks are universal function approximators (Sonoda and Murata,2017; Barron, 1993; Cybenko, 1989), this availability of sufficiently large i.i.d. training data supports the convergence of an approximate posterior to a target posterior.

In this study, I performed a total of 20 000 online update steps in the training with each step using a new pair of parameters and simulated time series from the ABM. Meanwhile, if one simulation incurs a high computational cost, a researcher can opt to perform *an offline learning* approach, in which a fixed number of ABM simulations (i.e. training data generations) according to the computational budget are conducted ex-ante and, subsequently, the widely parallelised batch learning will be performed on the GPU.[4] In any case, the converged networks can repeatedly be used to perform amortised inference over different observation datasets.

[4] One technique to further enhance computational efficiency is to launch multiple processes for ABM data generations on the CPU, together with the main process for network training by the GPU. The number of SGD iterations on the same batch of simulation data should be adjusted for the total time of the SGD iterations to match the time for a simulation-run and saving it on disk.

As for the hyper-parameters, I opt to use a default BayesFlow with 5 ABCs, and a summary vector of size 32 obtained through 3-layer bidirectional LSTM without extensive tune-up.

3.2. Performance validation settings

To evaluate the performance of applying BayesFlow to ABM estimation, I opt to calculate the following two simple metrics defined between the ground-truth parameter vectors $\{\theta^{(l)}\}_{l=1}^{L}$ which generate the test simulation datasets (i.e. pseudo-observations) $\{X^{o}_{1:T}(\theta^{(l)})\}_{l=1}^{L}$ and the estimated parameter vectors $\{\hat{\theta}^{(l)}\}_{l=1}^{L}$ reproduced from the test datasets. The test was conducted over 100 different sets of ground-truth values of parameters: $L = 100$.

(1) Normalised Rooted Mean Squared Error: $NRMSE = \sqrt{\sum_{l=1}^{L} \frac{(\theta^{(l)} - \hat{\theta}^{(l)})^2}{\theta_{max} - \theta_{min}}}$

(2) Coefficient of Determination: $R^2 = 1 - \sum_{l=1}^{L} \frac{(\theta^{(l)} - \hat{\theta}^{(l)})^2}{(\theta^{(l)} - \bar{\theta}^{(l)})^2}$

The competing benchmark for validation is the non-parametric KDE with MCMC (a random walk Metropolis-Hastings algorithm) proposed by Grazzini et al. (2017), in which 4 000 iterations are conducted for each estimation procedure. This KDE-MCMC method is reported to have outperformed a number of sophisticated frequentist approaches as a result of equal-footing comparison tests in Platt (2020). For reference purposes, two other methods (explained in Section 2.3) of Parametric-Gaussian-MCMC and MDN-MCMC of Platt (2019) were also applied to the same validation exercise.

The former method is a sort of pseudo-likelihood estimation, and more restrictive than the KDE-MCMC method, because it assumes that each value of the output time series is independently generated from an identical Gaussian distribution after an ergodic ABM reaches a statistical equilibrium. This method naturally incurs bias when applied to an ABM with a non-Gaussian density. The latter method of MDN-MCMC is more flexible than the KDE-MCMC method as it allows for temporal dependencies in output time series of the ABM (i.e. dispensing ergodicity assumption).

In the actual experiments, I opt to perform Bayesian inference on the standard NK-ABM with 8 parameters proposed by Grauwe (2012) and estimated in Grazzini et al. (2017), which is as explained below.

3.3. Validation experiment by a macroeconomic ABM with 8 parameters

3.3.1. Model

The validation experiment takes a standard NK-ABM with bounded rational agents, where heterogeneous agents use simple heuristics to formulate their expectations over future inflation and the output gap of the macro-economy. The aggregate behaviour of the model is described by the following three standard NK equations:

$$\begin{aligned} \text{Aggregate } \textit{Demand}: \ & y_t = a_1 E_t y_{t+1} + (1-a_1) y_{t-1} + a_2 (r_t - E_t \pi_{t+1}) + \epsilon^y_t, \\ \text{Aggregate Supply}: \ & \pi_t = b_1 E_t \pi_{t+1} + (1-b_1)\pi_{t-1} + b_2 y_t + \epsilon^{\pi}_t, \\ \text{Taylor Rule}: \ & r_t = c_1(\pi_t - \pi^*) + c_2 y_t + c_3 r_{t-1} + \epsilon^r_t, \end{aligned} \tag{9}$$

where y is the output gap, π is the inflation, and r is the nominal interest rate. For all $k \in [y, \pi, r]$, ϵ^k is the structural shock generated from $N(0, \sigma_k^2)$. A zero inflation target is assumed, $\pi^* = 0$, and E_t represents the expectation formulation by the agents.

Each agent selects a forecasting rule that has marked the highest performance in their past experience. Grauwe (2012) considered two forecasting rules: 1) a fundamentalist rule where agents believe that the output gap and inflation will revert to its equilibrium values of 0 and π^*, respectively, in the next period, and 2) an adaptive rule where agents believe that the output gap and inflation in the next period will be the same as in the previous period:

$$\begin{aligned} \text{Fundamentalist}: \ & E_t y_{t+1} = 0, \ \text{and} E_t \pi_{t+1} = \pi^*, \\ \text{Adaptive}: \ & E_t y_{t+1} = y_{t-1}, \ \text{and} E_t \pi_{t+1} = \pi_{t-1}. \end{aligned}$$

The actual choice by each agent i in period t is made at random with the probability of selecting the adaptive rule $p_{i,t,z}$ proportionate to the past performances as follows:

$$p_{i,t,z} = \frac{\exp\left(\gamma L^F_{i,t,z}\right)}{\exp\left(\gamma L^F_{i,t,z} + \gamma L^A_{i,t,z}\right)} \text{for } z \in [y, \pi],$$

where $L^F_{i,t,z}$ and $L^A_{i,t,z}$ are the losses defined as the mean squared forecasting errors over the past 10 periods. The parameter γ measures the intensity of the agent's choice. When $\gamma = 0$, the choice of forecast rule is purely stochastic (a fifty-fifty chance).

This model is known to exhibit two unique dynamics (Grauwe, 2012). The first is the intermittent effectiveness of the central bank's inflation targeting, where inflation remains close to the target level in one regime while fluctuating widely in the other. The second feature is an endogenous business cycle characterised by unpredictable waves of optimism and pessimism, which are responsible for the non-normality in the distribution of the output gap, despite that the system equations are driven by normally distributed structural shocks. From the perspective of ABM estimation, the non-stationarity

Table 1
Default values and priors of parameters in the New Keynesian ABM. $\boldsymbol{a}_2$ and $\boldsymbol{\gamma}$ are transformed to match the supports of those prior distributions to their theoretically assumed domains.

Parameter	Default	Prior
π^*	0.0	*fixed*
a_1	0.5	*U(0,3)*
τ=-1/a_2	3.0	*Γ(2,0.5)*
b_1	0.5	*U(0,1)*
b_2	0.05	*U(0,1)*
c_1	2.0	*fixed*
c_2	0.5	*fixed*
c_3	0.5	*fixed*
γ/10	0.5	*B(2,2)*
σ_y	0.5	*U(0,2)*
σ_π	0.5	*U(0,2)*
σ_r	0.5	*U(0,2)*

Table 2
Performance results on a macroeconomic ABM with 8 parameters.

	Parameter	BayesFlow	KDE-MCMC	Gaussian	MDN-MCMC
NRMSE	a_1	**0.487**	0.576	0.504	0.498
	τ=-1/a_2	**0.116**	0.876	0.626	0.345
	b_1	0.322	0.329	**0.302**	0.327
	b_2	**0.086**	0.151	0.296	0.175
	γ/10	**0.208**	0.253	0.248	0.259
	σ_y	**0.077**	0.198	0.418	0.207
	σ_π	**0.075**	0.171	0.394	0.149
	σ_r	**0.067**	0.274	0.392	0.154
R2	a_1	**0.807**	0.691	0.723	0.743
	τ=-1/a_2	**0.992**	0.924	0.790	0.935
	b_1	0.685	0.695	**0.746**	0.697
	b_2	**0.979**	0.945	0.763	0.920
	γ/10	**0.913**	0.771	0.814	0.802
	σ_y	**0.994**	0.953	0.846	0.939
	σ_π	**0.994**	0.960	0.851	0.967
	σ_r	**0.994**	0.936	0.732	0.961

and non-normality in observable variables require a sufficiently flexible method to approximate their likelihood or posterior. The parametric-Gaussian approach appears unsuitable for this type of AMBs, while the pre-selection of adequate summary statistics or a distance function also appears difficult.

The NK-ABM has 8 parameters to be estimated ($a_1,\ a_2,\ b_1,\ b_2,\ \gamma, \sigma_y, \sigma_\pi, \sigma_r$), the priors of which are listed in Table 1, together with the values of other calibrated parameters. Following Grazzini et al. (2017), the parameters of the Taylor rule ($c_1,\ c_2,\ c_3$) are fixed.

3.3.2. Estimation methods

In the validation procedure, I tried to recover the ground-truth values of the parameters from the simulated time series of this ABM for each separate trial of $l = 1, \ldots, 100$ (i.e. the 100 different parameter sets were tested). Subsequently, NRMSE and R2 between the ground-truth and estimated values of the parameters were calculated for the proposed method of 1) BayesFlow, and the benchmark method of 2) KDE-MCMC, together with 3) Parametric-Gaussian-MCMC and 4) MDN-MCMC for additional references. The length of the simulated time series (pseudo-observations) used in the estimations was set to 500.

3.3.3. Results of parameter recovery tests

The results of the estimations are presented in Table 2 and Fig. 1. BayesFlow succeeded in recovering the ground-truth values of $a_2,\ b_2, \sigma_y, \sigma_\pi$, and $\sigma_r$very precisely, while the value of γ was also well recovered with some fluctuation. In contrast, it was difficult to reconstruct the ground-truth values of a_1 and $b_1$were difficult to be reconstructed from the pseudo-observation datasets, which suggests the existence of an identification issue on these parameters when only observing the macroeconomic variables ($y,\ \pi,\ r$). The endogenous switching in the pattern of expectation formulation among the agents makes it difficult to identify the time-invariant shares between forward-looking and backward-looking terms in the equations of output and inflation (Eq. (9)).

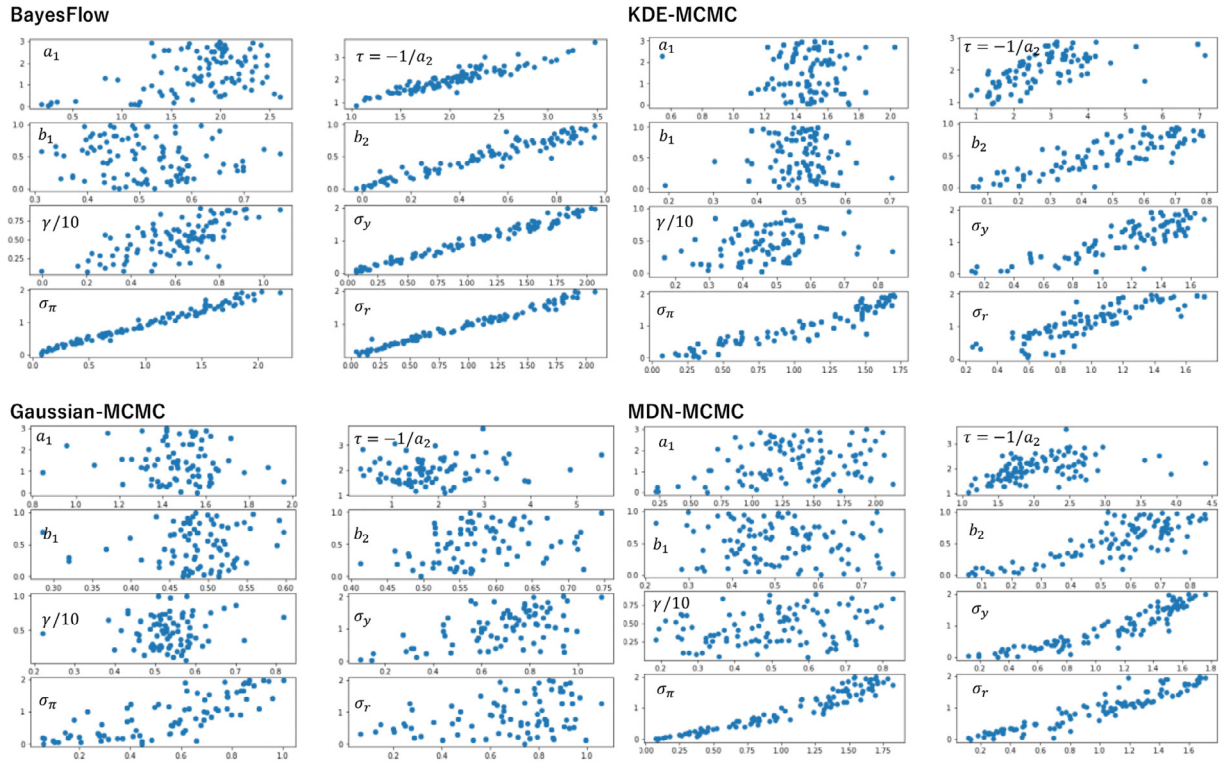

Fig. 1. Parameter recovery plots (based on the 100 sets of parameters) where x-axis takes value of the estimated parameter and y-axis is the ground-truth parameter.

On the contrary, the results of KDE-MCMC broadly showed inferior accuracies to BayesFlow in terms of both NRMSE and R2. In addition to a_1 and b_1, it also failed to recover the ground-truth values of a_2 and γ. The values of the other parameters $(b_2, \sigma_y, \sigma_\pi, \sigma_r)$ were relatively well reproduced, although the errors were significant relative to BayesFlow. These estimation errors could be reduced by increasing the length or draws of simulation time series. However, such operations would further incur additional computational costs. In other words, under the same computational budget, BayesFlow is far more efficient than KDE-MCMC.

Meanwhile, the results of parametric-Gaussian-MCMC were inferior to both BayesFlow and KDE-MCMC, as expected. This method relies on the restrictive assumption that an ergodic ABM conforms to a stationary Gaussian distribution in the steady state. Therefore, the inferior performance to the KDE method suggests that this assumption of parametric density incurs a larger approximation bias.

Finally, MDN-MCMC outperformed KDE-MCMC, which is consistent with the results of Platt (2019), and confirms the benefit of allowing temporal dependencies in the simulated time series of NK-ABM. However, it was clearly less accurate than BayesFlow. While both BayesFlow and MDN-MCMC allow non-stationarity (temporal dependencies) and equip a sufficiently expressive approximator (i.e. deep neural networks) for a posterior or a likelihood, one of the qualitative differences between the two methods is the lag length of temporal dependencies. The MDN method requires the pre-selection of a fixed lag length in conditional densities, which affects the approximation of likelihood. However, since the structure of time dependency in simulated time series could vary over parameter values of the ABM, an appropriate lag length in the density approximation would also change over the parameter values. It is difficult to deal with the pre-selected fixed lag of MDN-MCMC. In contrast, BayesFlow makes it possible to endogenously adjust the structure of temporal dependencies in response to various time series with the functioning of bidirectional LSTM summary networks trained by many different pairs of ABM parameters and resulting time series. Moreover, the quantitative difference in computational efficiency is significant between BayesFlow and MDN-MCMC, which is described below. Feasible precision of the MDN method would practically be bounded by the computational budget spent by the researcher.

In summary, the achieved precision in the parameter recovery tests on NK-ABM ordered 1) BayesFlow, 2) MDN-MCMC, 3) KDE-MCMC, and 4) Gaussian-MCMC.

3.3.4. Computational burden

In terms of computational burden, as I performed 100 separate estimations (100 different sets of ground-truth parameters) to calculate the validation metrics, the difference in computational costs became large between the amortised in-

Table 3
Summary of computational costs by each method for the 100 sets of parameter recovery test. Based on a machine equipped with Intel Core i7 CPU 2.80GHz and a single GPU of NVIDIA(R) GTX1050Ti.

		BayesFlow	KDE-MCMC	Gaussian	MDN-MCMC
Total computational time	(s)	30,131s	1,200,130s	840,130s	6,600,130s
	(h)	8h	333h	233h	1,833h
Formula		Btr × (Sd+Upd) + Ts × Sd+Ai	Ts × MCitr × (R × Sd+Lik+Upd) + Ts × Sd	Ts × MCitr × (R × Sd+Lik+Upd) + Ts × Sd	Ts × MCitr × (R × Sd+Lik+Upd) + Ts × Sd
# of total simulation runs		20,100	400,100	400,100	4,000,100
Variables in formula					
Seconds for a single simulation run	(Sd)	1.3s	1.3s	1.3s	1.3s
Seconds for a likelihood approx.	(Lik)	-	1.2s	0.3s	3.0s
Seconds for an update of parameters	(Upd)	0.2s [SGD]	0.5s [M-H]	0.5s [M-H]	0.5s [M-H]
Seconds for an amortized inference	(Ai)	0.8s	-	-	-
# of test parameter sets	(Ts)	100	100	100	100
# of MCMC iterations	(MCitr)	-	4,000	4,000	4,000
# of BayesFlow training steps	(Btr)	20,000	-	-	-
# of Simulation runs for a likelihood approx.	(MCitr)	-	1	1	10

ference of BayesFlow and the other case-by-case methods. As for the benchmark method of KDE-MCMC, a single simulation run of NK-ABM with 500 observation periods (where the first 500 periods were discarded as burn-in) took 1.3s. The likelihood construction and the Metropolis-Hastings update of candidate parameter values cost additional 1.7s per one occurrence. Subsequently, as I conducted 4 000 iterations in the Metropolis-Hastings algorithm for each estimation, the total computational time for the 100 separate estimations by KDE-MCMC amounted to massive 333 h (1 120 000 = 100 × 4 000 × (1.3s+1.7s) + 100 × 1.3s). I inevitably parallelised the CPU core processes over the 100 separate estimations.

On the contrary, BayesFlow took a much shorter time of 8 h (30131s) in total. While the training phase with 20 000 online training steps cost 30 000s as one step took 1.5s (=1.3s [training data generation] + 0.2s [Stochastic Gradient Decent]), the amortised inference with the 100 separate test datasets only took 131 (1.3s × 100 [test data generation] + 0.8s [amortised inference]). Furthermore, the training of BayesFlow reached the convergence with much fewer steps than 20 000, meaning that the computational time could be further shortened. Meanwhile, it looks unrealistic to cut MCMC iterations to less than 4 000 to obtain the accepted samples of more than 1 000.

Regarding the experiment by MDN-MCMC, it was computationally unfeasible to inherit the same setting of hyper-parameters as Platt (2019), owing to the longer computational time for a single simulation run of NK-ABM, and the requirement of 100 iterations of estimations under limited computational resources. Therefore, I had to set the number of simulation runs R per a single likelihood approximation as 10 rather than 100, and the length of simulated time series T as 500 rather than 1000 in the original settings,[5] both of which enable to cut the time for a likelihood approximation. Still, the resulting total computational time for the 100 separate estimations was tremendous 1,833 h (6600130 =100 × 4 000 × (13s+3.5s)+1.3s × 100). If the original hyper-parameters were adopted, the total computational time would become prohibitively huge: 17,278 h (62 200 130 = 100 × 4 000 × (150s+5.5s) + 1.3s × 100). As such, it might be better to regard the validation scores of MDN-MCMC above just for reference, showing how the method performs under a limited computational budget. In other words, it is at least possible to say that the superior precision of BayesFlow to MDN-MCMC in this test setting provided evidence of the higher computational efficiency (cost-performance) of BayesFlow, which is practically important for scalability. Table 3 summarises information about the computational costs by each method.

3.3.5. *Robustness to cyclical equilibrium*

As mentioned above, NK-ABM of Grauwe (2012) exhibits a unique dynamics of an endogenous business cycle, which is a sort of cyclical equilibrium of the system and is responsible for the non-ergodicity (i.e. temporal dependence) and non-normality in the output time series.

This subsection attempts to ensure that BayesFlow can effectively cope with the potential difficulties of a unique dynamics in the NK-ABM time series. Specifically, the accuracies in the ground-truth recovery tests are compared between the case of true parameters representing a cyclical equilibrium and the other case of parameters corresponding to a stable equilibrium.

Grauwe (2012) described that a choice intensity parameter γ governs the type of equilibria in NK-ABM. When $\gamma = 0$, the switching mechanism of each agent's forecast rule is purely stochastic and time series fluctuations are driven only by exogenous stochastic factors, which in that sense corresponds to a stable equilibrium (i.e. the ergodic assumption holds). As γ increases, a deterministic endogenous cycle starts to emerge and affect the fluctuation of the observable variables. An illustrative metric on the dominance of the endogenous cycle is the correlation between the output gap and the percentage

[5] Other hyper-parameters of MDN training are inherited from Platt (2019): 3 for lag length of conditional density, 3 for number of hidden layers of MDN, 16 for number of diagonal distribution to mix, 32 for dimension of a hidden layer, and 512 for training batch size.

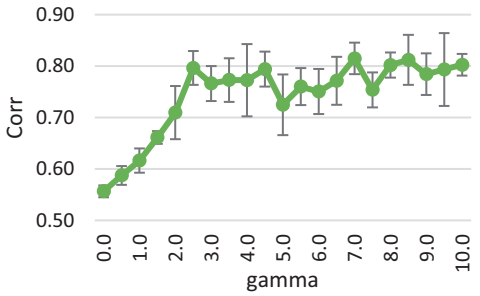

Fig. 2. The correlation between the output gap and the percentage of agents with a positive forecast. Error band stands for 1-standard deviation over 100 simulation-runs.

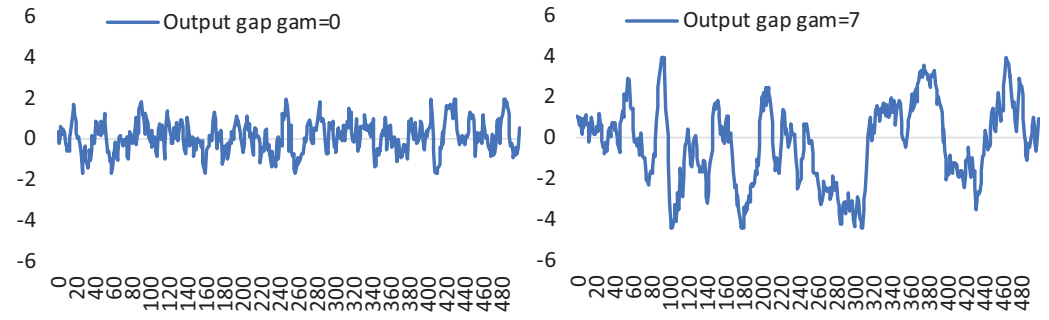

Fig. 3. Simulated time series of output gap by NK-ABM with the different values of γ.

Table 4
Posterior mean of absolute estimation error on each parameter by method and value of γ.

True values		Posterior Mean of Absolute Error			
		BayesFlow		KDE-MCMC	
		γ=0	γ=7	γ=0	γ=7
a_1	0.5	± 0.06	± 0.19	± 0.52	± 0.89
τ=-1/a_2	3.0	± 0.27	± 0.19	± 2.91	± 1.63
b_1	0.50	± 0.15	± 0.25	± 0.49	± 0.23
b_2	0.05	± 0.07	± 0.01	± 0.07	± 0.19
γ/10	0. or 0.7	± 0.10	± 0.16	± 0.03	± 0.20
σ_y	0.5	± 0.02	± 0.03	± 0.49	± 0.93
σ_π	0.5	± 0.02	± 0.02	± 0.50	± 0.64
σ_r	0.5	± 0.04	± 0.02	± 0.49	± 0.27

of agents with a positive forecast (the so-called animal spirit). Fig. 2 shows that the correlation increases quickly as γ increases from 0 to 2.5, and reaches saturation beyond this level.

Based on this pattern, I consider the parameter value $\gamma = 0$ representing a stable equilibrium, and $\gamma = 7$ as an example of cyclical equilibrium. The simulated time series for both values are depicted in Fig. 3.[6]

As a result of the experiments shown in Table 4 and Fig. 4, BayesFlow successfully recovered most of the ground-truth parameter values from both the pseudo-observation data ($\gamma = 0$ and $\gamma = 7$). As mentioned already, it was difficult to re-construct the ground-truth values of a_1 and b_1 from the simulated dataset, owing to the identification issue rooted in the endogenous switching of a forecast rule. When all the agents took an adaptive rule, the expectation variables degenerated to lagged variables. This does not occur when $\gamma = 0$, leading to the superior accuracy in the estimation of a_1 and b_1 in the stable equilibrium. In terms of the other identifiable parameters, the accuracy does not deteriorate when dealing with the data of endogenous cycle ($\gamma = 7$). This confirms the robustness against the different types of equilibria within the same ABM.

Finally, when compared to the results of the same tests implemented by KDE-MCMC, BayesFlow recorded higher accuracies on identifiable parameters in both the cases of γ, and the superiority tended to be clearer when $\gamma = 7$. This suggests

[6] Values of the other parameters are reported in Table 4.

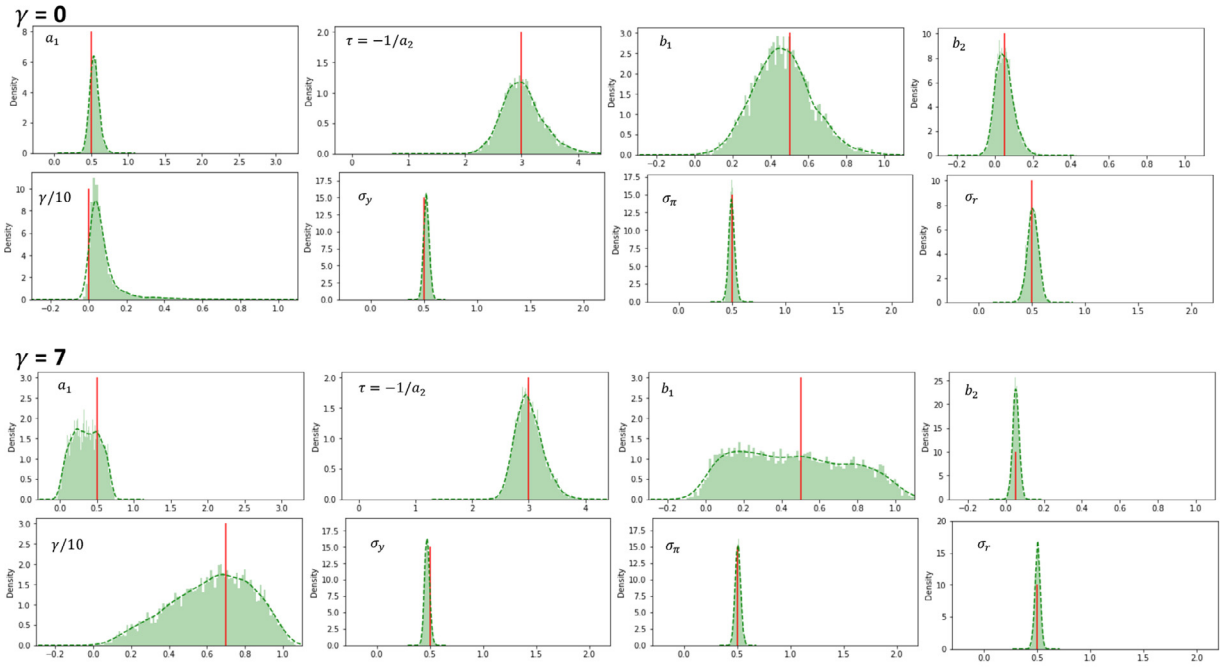

Fig. 4. Posterior distributions generated by BayesFlow from the simulated time series of NK-ABM with $\boldsymbol{\gamma} = 0$ or $\boldsymbol{\gamma} = 7$. The vertical line represents a ground truth value.

that the ergodicity assumption and the resulting ignorance of the temporal dependency of the KDE method incurred a larger bias in the likelihood approximation.

3.4. Estimation of the macroeconomic ABM with real data

I subsequently estimated NK-ABM with real observation data, and compared the posteriors obtained with BayesFlow to those obtained with KDE-MCMC. The main purpose of this experiment was to verify that BayesFlow works properly for observations generated from an unknown Data Generation Process (DGP), where the ABM to be estimated is potentially misspecified to the true DGP.

The parameter recovery test in the previous section used the pseudo-observation data newly simulated from the same ABM as the one used in the training (i.e. NK-ABM). In that sense, the statistical model to be estimated was correctly specified, which is, however, not always the practical case (particularly when real data are used for estimation). If BayesFlow is very vulnerable to such misspecification (due to a kind of over-fitting to the DGP of the ABM used in training), a posterior obtained with BayesFlow is expected to fail in shrinking properly from a prior, as it poorly regards the input real data that does not have much available information about the ABM parameters to be estimated. Therefore, it is worth checking whether a posterior obtained by BayesFlow shows appropriate contractions when using actual economic data. In addition, an estimation of NK-ABM with the same real data by the benchmark method of KDE-MCMC was also conducted for the purpose of comparing results.

I used data on real GDP, GDP deflator, and Fed Fund Rate for the US during 3Q1954 and 3Q2020 ($T = 265$).[7] The output gap is obtained by applying an HP filter to the log GDP series, while the quarterly inflation rate and interest rate are demeaned. The priors are the same as those in Section 3.3.

The estimation results shown in Fig. 5 clearly demonstrate the desired shrinkage in the posteriors of most parameters constructed by BayesFlow. Furthermore, Fig. 6 illustrates the improvements in estimation (i.e. variance shrinkage) with an increase in the observation periods. This feature of *posterior contraction* confirms that BayesFlow certainly works when applied with real data generated from a never-experienced DGP.

Yet again, the exceptions are a_1 and b_1 (constant shares of forward-looking variables in the demand and supply equations, respectively, Eq. 9), showing somewhat contracted but still volatile distributions. The parameter recovery test in the previous section showed that the values of these two parameters are structurally difficult to be identified from the three observation variables (y, π, r).

[7] The GDPC1, GDPDEF, and FFR series in the FRB St. Louis FRED database are used.

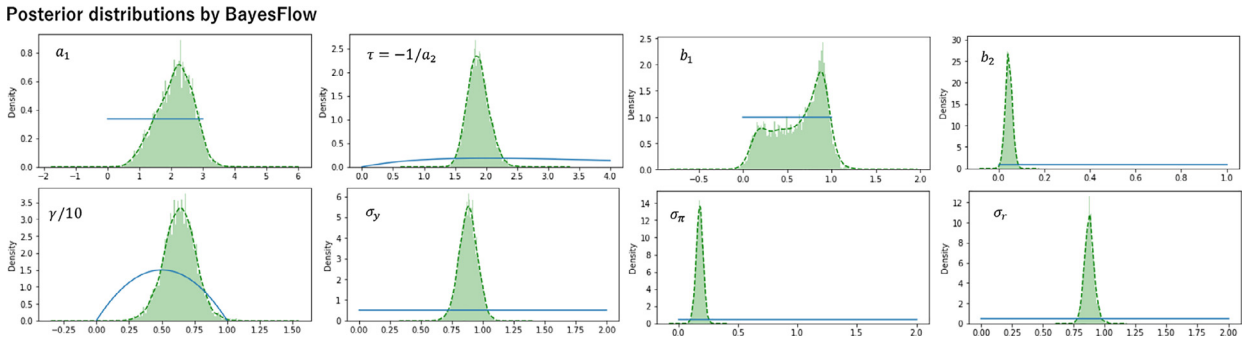

Fig. 5. Posterior distributions of NK-ABM parameters generated by BayesFlow on US economic data. The bold solid line is the prior of each parameter.

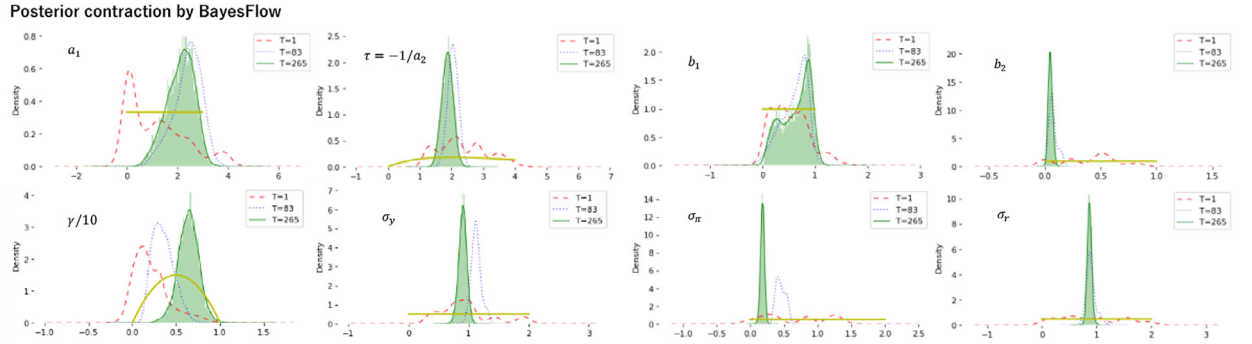

Fig. 6. Visualisations of posterior contraction along with increasing observation length. The T=1 observation period is just 3Q2020, the T=83 is from 1Q2000 to 3Q2020, and the T=265 (full sample) is from 3Q1954 to 3Q2020. The bold solid line is the prior of each parameter.

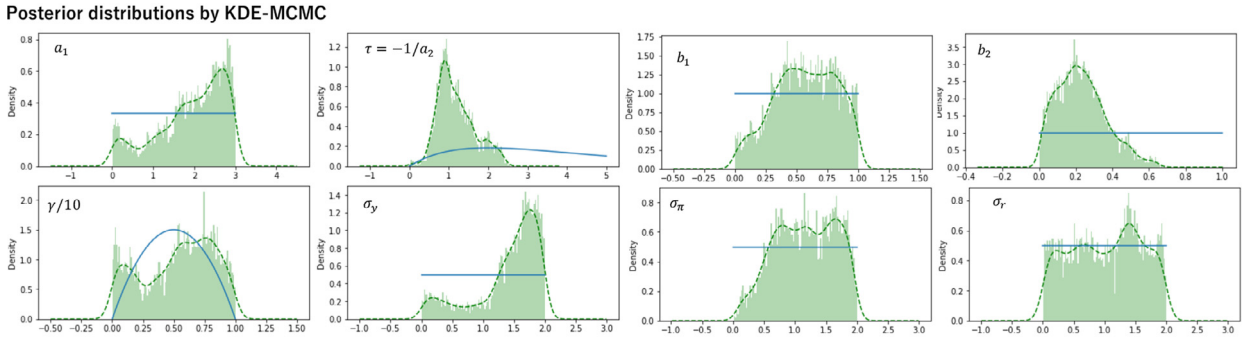

Fig. 7. Posterior distributions of NK-ABM parameters generated by KDE-MCMC on US economic data. The bold solid line is the prior of each parameter.

When compared with the result of KDE-MCMC (Fig. 7), BayesFlow constitutes clearly tighter posterior distributions, while the general patterns of posterior means are similar between the two methods (Fig. 8). This point corroborates the outperformance of BayesFlow to the benchmark method of KDE-MCMC with regard to estimation precision.

Computational efficiency is another advantage of BayesFlow. It only requires 1.3s to produce 5 000 posterior samples as the networks (cINN) once trained on NK-ABM can be reused for a new inference with a different dataset. In contrast, KDE-MCMC actually costs 72 000s (=20 000 × (1.3s[a simulation run] + 2.3s[a likelihood evaluation and M-H update])) to generate the same number of accepted samples.

4. Discussion and concluding remarks

In this study, I explore how the novel likelihood-free Bayesian inference of BayesFlow proposed by Radev et al. (2020) can be applied for the estimation of an ABM.

While most practical-scale ABMs do not have an analytically tractable (i.e. closed-form) likelihood function, the existing literature has developed ways to circumvent this problem. Among the leading estimation methods, Platt (2020) has demonstrated that the Bayesian approach of KDE-MCMC proposed by Grazzini et al. (2017) outperformed a number of sophisticated frequentist approaches as a result of equal-footing comparisons. This approach assumes the ergodicity and resulting station-

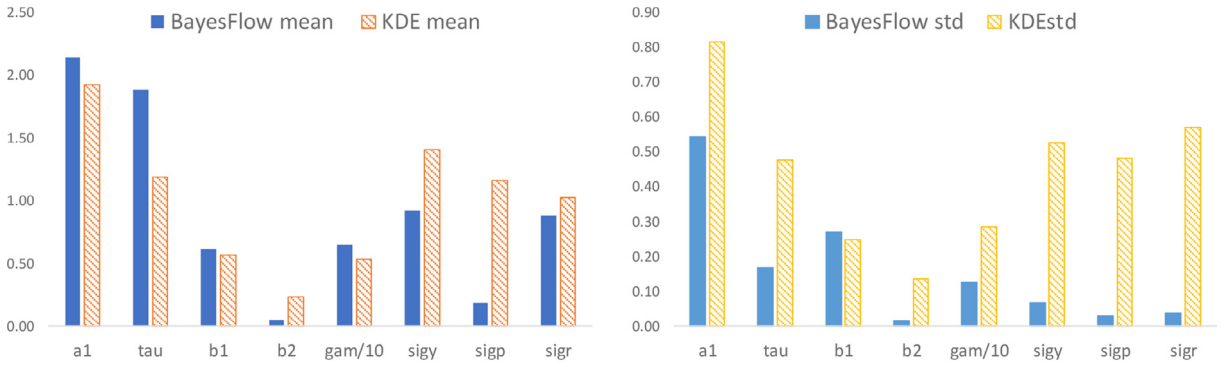

Fig. 8. Posterior means and standard deviations of posterior distributions of NK-ABM by BayesFlow and KDE-MCMC on US economic data.

arity of output time series of ABM, and uses KDE to approximate the stationary density of the simulated time series, where parameter values are explored by MCMC (Metropolis-Hastings algorithm) with the likelihood of actual observation data being evaluated by the approximated density.

However, the ergodicity assumption (i.e. no consideration of temporal dependencies in time series) and progressive computational costs when dealing with higher-dimensional observables could be potential shortcomings of this method. Motivated to overcome the lack of temporal dependencies, Platt (2019) has proposed a new Bayesian estimation protocol that uses a neural-network-based approximation of conditional density (MDN), and reported compelling performances against KDE-MCMC. Still, one remaining shortcoming of the MDN-MCMC method is its huge computational burden. This is because it requires multiple runs of ABM simulation to train deep neural networks by every single step of a likelihood approximation.

In that sense, the current challenge in ABM estimation appears to be the scalability of methodology to large-sized (high-dimensional observations or parameters) ABMs. In other words, the desirable properties of an estimation method proved on a small ABM should be seamlessly reproduced on a larger ABM, with an efficient increase in computational burden.

Obstacles to scalability typically involve the following points: 1) too restrictive assumptions on simulated time series of ABM, 2) critical dependence of estimation accuracy on arbitrary pre-selection, and 3) progressive computational burden to the size (i.e. dimension of observable variables or parameters) of the ABM.

The core motivation of introducing BayesFlow into the context of ABM estimation is to simultaneously cope with these three obstacles to the scalability. BayesFlow is a fully likelihood-free approach, which directly approximates a posterior rather than a likelihood function by learning an invertible probabilistic mapping that implements a normalising flow between parameters and standard Gaussian variables conditioned by data from simulations. This method holds general applicability, only requiring the ability to output simulation data from a mathematical forward model and has a theoretical guarantee for sampling from the true posterior without any specific assumption on the shape of the target posterior or prior. Hence, it does not need to presume the ergodicity or stationarity of simulated time series from a model.

Second, BayesFlow does not involve discretionary handcrafted design in the critical parts of inference. It accompanies a learnable summary network which can compress variable lengths of potentially large dimensional inputs into fixed-length and small-dimensional summary statistics. While the hyper-parameters of neural networks need to be specified, the selection does not critically affect the shape of the approximate posterior as long as the realised expressive-power is sufficient to make the learning of cINN and summary network well converged.

Finally, it is computationally efficient, particularly in the case where repeated inferences with different observation datasets are required (e.g. multi-country applications). BayesFlow realises amortised inference, where estimation is split into a computationally expensive training phase and a much cheaper inference phase. In the training phase, BayesFlow tries to learn a model to output an approximate posterior that works well for any possible observation sequence. Evaluating the trained model over a specific observation dataset is computationally cheap and, therefore, the upfront training efforts amortise over multiple inferences. This amortised inference is a clear advantage with respect to the MCMC-based methods, which incur significant computational costs to collect posterior samples as ABM simulations need to be repeated with every single step of updating candidate parameters. Dimensionality reduction implemented by the aforementioned summary network also contributes to reducing the computational burden of handling high-dimensional observations, in contrast to the KDE-based methods. In addition, parallel computations are applicable to both generating simulation dataset and training with parallelised mini-batches. Practically, this computational efficiency could be the most impressive advantage of BayesFlow; this is because feasible precision of any simulation-based estimation method tends to be effectively bounded by the computational budget.

As a result of the experiments, BayesFlow certainly achieved superior accuracies to the benchmark method of KDE-MCMC and the reference method of MDN-MCMC in the validation task of recovering the ground-truth values of parameters from the simulated (pseudo-observation) datasets of a standard NK-ABM with 8 parameters. The method did not involve any extensive search of the hyper-parameters or handcrafted pre-selections of summary statistics, and took a significantly shorter computational time than the compared MCMC approaches.

Furthermore, the truly empirical estimation of NK-ABM using the real observation data of the US economy is carried out to check the robustness of BayesFlow against the data generated from the unknown DGP. The results successfully showed the desired property of posterior contraction with increasing observation periods. While the means of the obtained posteriors exhibited a similar pattern with those constructed by KDE-MCMC, the variances were significantly tighter in the case of BayesFlow. Furthermore, as BayesFlow networks once trained on a particular ABM can be reused, the marginal computational time for the real data estimation was just 1.3 s, compared to 72 000 s taken by another estimation by KDE-MCMC from scratch. These results provide evidence of the net improvement of BayesFlow against the benchmark method of KDE-MCMC.

These advantages suggest that BayesFlow is a promising method for estimating not only ABMs but also other types of complex macroeconomic models with intractable likelihood, such as nonlinear DSGEs or heterogeneous agent models, which tend to require numerical approximation to derive its dynamic system.

Notwithstanding these advantages, the limitations of the method should also be mentioned. First, the Monte Carlo error remains in the training objective by approximating the mathematical expected values with a finite sum of the samples, even though potentially unlimited samples can be used in the online training scheme. Second, hyper-parameters of neural networks might require slight fine-tuning to unleash the full potential of BayesFlow, although the default settings are sufficient to achieve excellent performance.

References

Assenza, T., Delli Gatti, D., Grazzini, J., 2015. Emergent dynamics of a macroeconomic agent based model with capital and credit. J. Econ. Dyn. Control 50, 5–28. doi:10.1016/j.jedc.2014.07.001.

Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. Inf. Theory 39 (3), 930–945. doi:10.1109/18.256500.

Caiani, A., Godin, A., Caverzasi, E., Gallegati, M., Kinsella, S., Stiglitz, J.E., 2016. Agent based-stock flow consistent macroeconomics: towards a benchmark model. J. Econ. Dyn. Control 69, 375–408. doi:10.1016/j.jedc.2016.06.001.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. 2 (4), 303–314. doi:10.1007/BF02551274.

Dinh, L., Sohl-Dickstein, J., Bengio, S., 2017. Density estimation using real NVP. In: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings.

Gallegati, M., Richiardi, M.G., 2009. Agent based models in economics and complexity. In: Encyclopedia of Complexity and Systems Science. Springer, New York, pp. 200–224. doi:10.1007/978-0-387-30440-3_14.

Ghonghadze, J., Lux, T., 2016. Bringing an elementary agent-based model to the data: estimation via GMM and an application to forecasting of asset price volatility. J. Empirical Finance 37, 1–19. doi:10.1016/j.jempfin.2016.02.002.

De Grauwe, P., 2012. Booms and busts: new Keynesian and behavioural explanations. In: What's Right with Macroeconomics? Edward Elgar Publishing, pp. 149–180. doi:10.4337/9781781007402.00016.

Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. 18 (5–6), 602–610. doi:10.1016/j.neunet.2005.06.042.

Grazzini, J., Richiardi, M., 2015. Estimation of ergodic agent-based models by simulated minimum distance. J. Econ. Dyn. Control 51, 148–165. doi:10.1016/j.jedc.2014.10.006.

Grazzini, J., Richiardi, M.G., Tsionas, M., 2017. Bayesian estimation of agent-based models. J. Econ. Dyn. Control 77, 26–47. doi:10.1016/j.jedc.2017.01.014.

Kristensen, D., Shin, Y., 2012. Estimation of dynamic models with nonparametric simulated maximum likelihood. J. Econometrics 167 (1), 76–94. doi:10.1016/j.jeconom.2011.09.042.

Kukacka, J., Barunik, J., 2017. Estimation of financial agent-based models with simulated maximum likelihood. J. Econ. Dyn. Control 85, 21–45. doi:10.1016/j.jedc.2017.09.006.

Lamperti, F., Roventini, A., Sani, A., 2018. Agent-based model calibration using machine learning surrogates. J. Econ. Dyn. Control 90, 366–389. doi:10.1016/j.jedc.2018.03.011.

Lux, T., 2018. Estimation of agent-based models using sequential Monte Carlo methods. J. Econ. Dyn. Control 91, 391–408. doi:10.1016/j.jedc.2018.01.021.

Lux, T., Zwinkels, R.C.J., 2018. Empirical validation of agent-based models. Handb. Comput. Econ. 4, 437–488. doi:10.1016/bs.hescom.2018.02.003.

Platt, D., 2019. Bayesian Estimation of Economic Simulation Models Using Neural Networks http://arxiv.org/abs/1906.04522.

Platt, D., 2020. A comparison of economic agent-based model calibration methods. J. Econ. Dyn. Control 113, 103859. doi:10.1016/j.jedc.2020.103859.

Radev, S.T., Mertens, U.K., Voss, A., Ardizzone, L., Köthe, U., 2020. BayesFlow: Learning Complex Stochastic Models with Invertible Neural Networks http://arxiv.org/abs/2003.06281.

Rezende, D.J., Mohamed, S., 2015. Variational inference with normalizing flows. In: 32nd International Conference on Machine Learning, 2. ICML, pp. 1530–1538.

Schuster, M., Paliwal, K.K., 1997. Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45 (11), 2673–2681. doi:10.1109/78.650093.

Sonoda, S., Murata, N., 2017. Neural network with unbounded activation functions is universal approximator. Appl. Comput. Harmon. Anal. 43 (2), 233–268. doi:10.1016/j.acha.2015.12.005.

Watanabe, S., 2009. Algebraic Geometry and Statistical Learning Theory. Cambridge Univesity Press Algebraic Geometry and Statistical Learning Theory doi:10.1017/cbo9780511800474.