Bank of Russia

Fast Estimation of Bayesian State Space Models Using Amortized Simulation-Based Inference

WORKING PAPER SERIES

No. 104/ December 2022

Ramis Khabibullin, Sergei Seleznev

Ramis Khabibullin

Independent Researcher

Sergei Seleznev

Bank of Russia, Research and Forecasting Department

Email: seleznevsm@cbr.ru

Authors are grateful to Dmitry Gornostaev, Sergey Ivaschenko, Petr Milyutin, Alexandr Scheglov, Denis Shibitov, Vitaliy Yuferev and participants of the 15th International Conference on Computational and Financial Econometrics (CFE 2021), XXIII Yasin (April) International Academic Conference on Economic and Social Development and 2nd International Conference on Econometrics and Business Analytics (iCEBA) for their helpful comments and suggestions.

Bank of Russia Working Paper Series is anonymously refereed by members of the Bank of Russia Research Advisory Board and external reviewers.

Cover image: Shutterstock/FOTODOM

© Central Bank of the Russian Federation, 2022

Address: 12 Neglinnaya street, Moscow, 107016
Tel.: +7 495 771-91-00, +7 495 621-64-65 (fax)
Website: www.cbr.ru

All rights reserved. The views expressed in this paper are solely those of the authors and do not necessarily reflect the official position of the Bank of Russia. The Bank of Russia assumes no responsibility for the contents of the paper. Any reproduction of these materials is permitted only with the express consent of the authors.

Abstract

This paper presents a fast algorithm for estimating hidden states of Bayesian state space models. The algorithm is a variation of amortized simulation-based inference algorithms, where numerous artificial datasets are generated at the first stage, and then a flexible model is trained to predict the variables of interest. In contrast to those proposed earlier, the procedure described in this paper makes it possible to train estimators for hidden states by concentrating only on certain characteristics of the marginal posterior distributions and introducing inductive bias.

Illustrations using the examples of stochastic volatility model, nonlinear dynamic stochastic general equilibrium model and seasonal adjustment procedure with breaks in seasonality show that the algorithm has sufficient accuracy for practical use. Moreover, after pretraining, which takes several hours, finding the posterior distribution for any dataset takes from hundredths to tenths of a second.

JEL codes: C11, C15, C32, C45.

Keywords: amortized simulation-based inference, Bayesian state space models, neural networks, seasonal adjustment, stochastic volatility, SV-DSGE.

Contents

1. Introduction 5
2. Estimation procedure 6
2.1. Amortized simulation-based inference 6
2.2. From Bayesian models to state space models 7
2.3. Marginal distribution loss 8
2.4. Estimator architecture 8
3. Applications 9
3.1. Stochastic volatility model 9
3.2. Stochastic volatility DSGE model 11
3.3. Seasonal adjustment with structural breaks in seasonality 12
3.4. Computation and implementation time 14
4. Related work 15
5. Discussion 17
6. Conclusions 19
References 20
Appendix A. Informal proofs 26
A1. NPE asymptotic 26
A2. NPE for states 26
A3. NPE for marginal distribution loss 26
Appendix B. Stochastic volatility model 27
B1. Model 27
B2. Architecture and learning algorithm 27
B3. Alternative algorithm for stochastic volatility model 28
Appendix C. Stochastic volatility DSGE model 30
C1. Model 30
C2. Architecture and learning algorithm 32
C3. Alternative algorithm for stochastic volatility DSGE model 33
Appendix D. Seasonal adjustment with structural breaks in seasonality 35

1. Introduction

Bayesian state space models are widely used in applied macroeconomics. They are so widespread due to the fact that many macroeconomic and econometric models can be written in the form of state space models for subsequent estimation on real data. For example, various kinds of filters and semi-structural filters (see Hodrick and Prescott (1997), Laubach and Williams (2003)), models with stochastic volatility (see Kim, Shepard and Chib (1998), Justiniano and Primiceri (2008), Carriero, Clark and Marcellino (2016)), time-varying models (see Hamilton (1989), Primiceri (2005), Koop and Korobilis (2012)), mixed frequency models (see Chiu et al (2012) , Schorfheide and Song (2015, 2021)), dynamic factor models (see Otrok and Whiteman (1998), Stock and Watson (2011)), dynamic stochastic general equilibrium models (see Smets and Wouters (2003, 2007), Fernandez-Villaverde, Schorfheide and Rubio-Ramirez (2016)) and agent-based models (see Lux (2018), Deli Gatti and Grazzini (2020)). Bayesian parameter estimation makes it possible to mitigate the lack of long time series[1].

Despite their flexibility, in practice, estimating Bayesian state space models is a rather difficult task. Sampling algorithms are based on an iterative sampling scheme for model parameters and states and rely on steps such as Gibbs sampling (see Casella and George (1992)), Metropolis-Hastings (see Chib and Greenberg (1995)), Hamiltonian Monte Carlo (see Neal (1996)) or sequential Monte Carlo (see Del Moral, Doucet and Jasra (2006)). Even in cases where the model is linear and Gaussian or discrete with respect to states, sampling can take from tens of minutes to hours. In systems of a more general form, researchers use particle filters (see Andrieu, Doucet and Holenstein (2010), Chopin, Jacob, and Papaspiliopoulos (2012)), as a result of which estimation time only increases. Sampling algorithms are exact in the sense that they converge to the posterior distribution as the number of iterations tends to infinity, although the number of iterations required for an estimate close to the posterior distribution can be large. An alternative to them are optimization algorithms. The most common of them in the context of state space models is the variational Bayes algorithm[2] (see, Wainwright and Jordan (2008), Hoffman et al. (2013)). The variational Bayes algorithm relies on minimizing the Kullback-Leibler (or any other) divergence between the approximation and the true posterior distribution. It is often faster than sampling algorithms, but its running time is also large, especially in cases where the optimization steps cannot be written in analytical form.

[1] In this paper, we focus on time series models, but the proposed algorithm can easily be transferred to hidden space models of other types with minor modifications in the architecture.

[2] See Chapters 3 and 5 of Beal (2003) and Gunawan, Kohn, and Nott (2021) for examples of using the Bayesian variational algorithm in estimating of state space models.

In this paper, we propose a fast algorithm for estimating Bayesian state space models that is based on the principles of simulation-based inference (see Cranmer, Brehmer and Louppe (2020)). The speed of the algorithm is achieved by amortizing the task of constructing the posterior distribution of states, that is, by pretraining a model (a neural network, in our case) that predicts its posterior distribution from the data. In doing so, we focus only on states and do this for two reasons. First, in many problems it is the states, not the model parameters, that are of particular interest. For example, in the detrending problem (see Orphanides and Van Norden (2002)), the trend and cycle components, which are determined by hidden states, are of main interest. Second, simulation-based inference parameter estimation has been investigated in many other papers (see Appendix A of Lueckmann et al. (2021)) and can be easily combined with the approach discussed in this paper. The problem of constructing posterior distribution of hidden states is much more complicated due to its dimensionality and has not been studied much in the literature.

Section 2 describes the algorithm for estimating the posterior distribution of the model. Section 3 is devoted to the study of application, practical characteristics and comparison of the algorithm with commonly used alternatives. Section 4 describes related work. Section 5 discusses issues that have not been included in the paper but are important in the context of the proposed algorithm. The conclusion is presented in Section 6.

2. Estimation procedure

2.1. Amortized simulation-based inference

Our methodology for estimating state space models is based on the idea of estimating Bayesian parameters proposed by Beaumont, Zhang and Balding (2002), Blum and Francois (2010) and developed by Papamakarios and Murray (2016). The essence of the methodology is to simulate the joint distribution of parameters and data, and then predict the parameters conditional on data.

Formally, during the first step, a dataset is simulated from a model with a prior distribution $p(\theta)$ and a likelihood function $p(y|\theta)$. The i-th point consists of parameters $\theta_i \sim p(\theta)$ and observed variables $y_i \sim p(y|\theta_i)$. In the second step, an estimator is fitted to predict the distribution $p(\theta|y)$ or its characteristics, for example, by minimizing the cross-entropy between the simulated data and some parametric family of distributions $q_\varphi(\theta|y)p(y)$:

$$\varphi^* = argmin_\varphi\left(-\sum_{i=1}^{N}\left(log q_\varphi(\theta_i|y_i) + log\, p(y_i)\right)\right) \qquad (1)$$

where φ is the vector of parameters of distribution $q_\varphi(\theta|y)$, N is the number of simulations. We will omit $log\, p(y_i)$ term due to its independence from φ in the following.

The estimated distribution will tend to the posterior for any y with an infinite number of simulations and a sufficiently flexible parametric family q_φ (see Appendix A.1 for an informal proof). This actually means that the algorithm has the property of amortization, or in other words, that once estimated, the conditional distribution $q_{\varphi^*}(\theta|y)$ can be used for any data, does not require re-estimation of the model and can be calculated almost instantly.

As can be seen, this method for estimating posterior distributions (hereinafter, we will call it NPE following Papamakarios and Murray (2016)) does not require knowledge of $p(\theta)$ and $p(y|\theta)$ in an explicit form but relies only on the ability of simulating data, which is natural for most models. So, it falls into the category of *simulation-based inference* (SBI) methods (or *likelihood-free inference*).

2.2. From Bayesian models to state space models

The parameters of state space models can be estimated using the procedure presented in Section 2.1, however the goal of this paper is to estimate the hidden states. It is easy to see that if we replace the parameters with hidden states in the loss function, then the procedure described above remains valid (see Appendix A.2). Thus, in general, the algorithm for finding the posterior distribution can be written as shown below:

Algorithm 1. Simulation based state space model inference

For $i = 1, \dots, N$:

1. Simulate states and observable data:

   1.a. Draw model parameters from the prior:

   $$\theta_i \sim p(\theta)$$

   1.b. Draw states from the conditional state distribution:

   $$s_i \sim p(s|\theta_i)$$

   1.c. Draw data from the conditional data distribution:

   $$y_i \sim p(y|s_i, \theta_i)$$

2. Find the parameters of the posterior approximation of hidden states:

$$\varphi^* = argmin_\varphi\left(-\sum_{i=1}^{N} log\, q_\varphi(s_i|y_i)\right) \quad (2)$$

Despite the simplicity of Algorithm 1, there are several practical difficulties when moving from finding posterior distribution for the parameters to finding the distribution of states. First, it is

a large dimension of the hidden space. Despite the presence of various kinds of flow transformations (see Rezende and Mohamed (2015)), which are often used in SBI and in many cases recover joint distributions adequately, their application for problems of this size, complicated by amortization, is costly to compute and is associated with optimization challenges. Second, it is the large dimensionality of the data. It is nearly impossible to find summary statistics that reduce the dimensionality of the data for hidden states, in contrast to the problem of parameter estimation, where this is common practice (see, for example, SIR (T.9) and Lotka-Voltera (T.10) models in Lueckmann et al (2021)). To overcome these problems and simplify the task of training the model, we avoid modeling the dependencies between variables focusing on characteristics of marginal distributions and introduce an inductive bias for $q_\varphi(s|y)$.

2.3. Marginal distribution loss

Moving from the joint to the marginal distributions is equivalent to subdividing the task into a set of one-dimensional tasks. In such case, the log-probability for the parametric family $q_\varphi(s|y)$ is written as:

$$log\ q_\varphi(s|y) = \sum_{t=1}^{T}\sum_{k=1}^{k} log\ q_{\varphi_{t,k}}(s^{t,k}|y) \qquad (3)$$

where t and k represent the time period and the state index in state vector. Note that for each state, the vector of parameters $\varphi_{t,k}$ is generally its own and φ consists of a set of these vectors.

We use the normal distribution with mean $m_{\varphi_{t,k}}(y)$ and standard deviation $\sigma_{\varphi_{t,k}}(y)$ for $q_{\varphi_{t,k}}(s^{t,k}|y)$. It is easy to show that for sufficiently flexible $m_{\varphi_{t,k}}$ and $\sigma_{\varphi_{t,k}}$ such an approximation exactly recovers the mean and standard deviation of the true posterior distribution (see Appendix A.3).

Moreover, the choice of normal distribution and cross-entropy as a loss function can be relaxed. Therefore, a mixture of normals or a small-scale flow-based model can replace normal distribution for marginal densities and any M-estimator (see Chapter 5 in Van der Vaart (2000)), such as quantile regression loss, can be used instead of cross-entropy.

2.4. Estimator architecture

A neural network is chosen as an estimator. It is a class of flexible models that can approximate almost any relationship in theory[3]. A natural architecture that is similar to filtering and smoothing in state space models is a Bidirectional Recurrent Neural Network (Bidirectional RNN).

[3] See Goodfellow, Bengio and Courville (2016) for an introduction to neural networks and their properties.

By sharing parameters for models with a «near» stationary data generation process[4,5], this structure allows to move away from estimating the neural network for each state separately and calculate the loss function for all states in a single pass over the data.

To make the network architecture more flexible, we use data convolution of various lengths as RNN input in addition to the raw data, and transform the output of RNN by applying a linear transformation or a fully connected neural network.

3. Applications

To illustrate the properties of the proposed method, we estimate three models: stochastic volatility model, non-linear DSGE model and seasonal adjustment model with structural breaks in seasonality.

3.1. Stochastic volatility model

The stochastic volatility model (see Kim, Shepard and Chib (1998)) is a classic example for testing various states estimation algorithms in state space models (see Tan, Bhaskaran and Nott (2020)). The model specification is exactly the same as Tan, Bhaskaran and Nott (2020) and presented in Appendix B.1. To simplify the learning process, the logarithm of absolute values of the observed data is fed into the neural network as an input (details of the network architecture and the training algorithm are presented in Appendix B.2). The model is trained on 20,000,000 generated series with a length of 800 to 1,200 observations.

First, we demonstrate the quality of the algorithm on the data sampled from the data generation process. Figure 1 shows examples of the true values of volatility logarithms and mean of posterior distribution approximation (± 2 standard deviations). The figure demonstrates that the true values are quite well estimated by the neural network. As a benchmark for future studies, we also present the negative log-likelihood (hereinafter, NLL) and mean squared error (hereinafter, MSE) on a randomly generated 1,000,000 runs in Table 1.

Unfortunately, calculation of the accuracy metrics for other algorithms, such as MCMC or stochastic variational Bayes, is difficult to compute (see the discussion on quality metrics for SBI in Lueckmann et al. (2021)). Our focus therefore lies on the comparison of the results for NYSE and

[4] We use the term *«near» stationary data generation process* to emphasize the possibility of using different state distributions for the initial time period.
[5] One can always connect several networks, if there is a shift in the data generation process.

GBPUSD[6] datasets similarly to Tan, Bhaskaran and Nott (2020). The results are compared with the adaptive MCMC algorithm based on the mixture of normals for chi-square distribution approximation, which was proposed by Kim, Shepard and Chib (1998), and the stochastic variational Gaussian approximation (hereinafter, VB) with a sparse precision matrix (both algorithms are given in Appendix B.3). As can be seen in Figure 2, although the neural network estimates are a bit noisy, they are close to the MCMC algorithm, which serves as the gold standard, as well as the variational Bayes algorithm, which is one of the fastest and most accurate approximations. It should be noted that the NYSE dataset is almost twice the maximum size of the simulation, and the neural network has never seen data of that length. Nevertheless, the trained model copes with this task.

Figure 1. True values of volatility logarithms and approximation of posterior distribution on artificial data (mean ± 2 std)

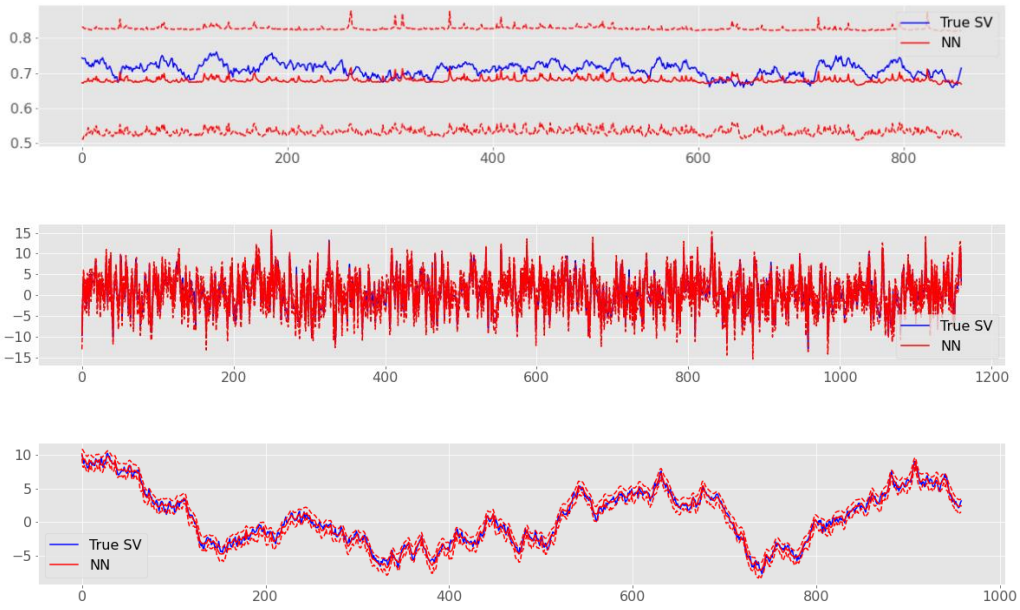
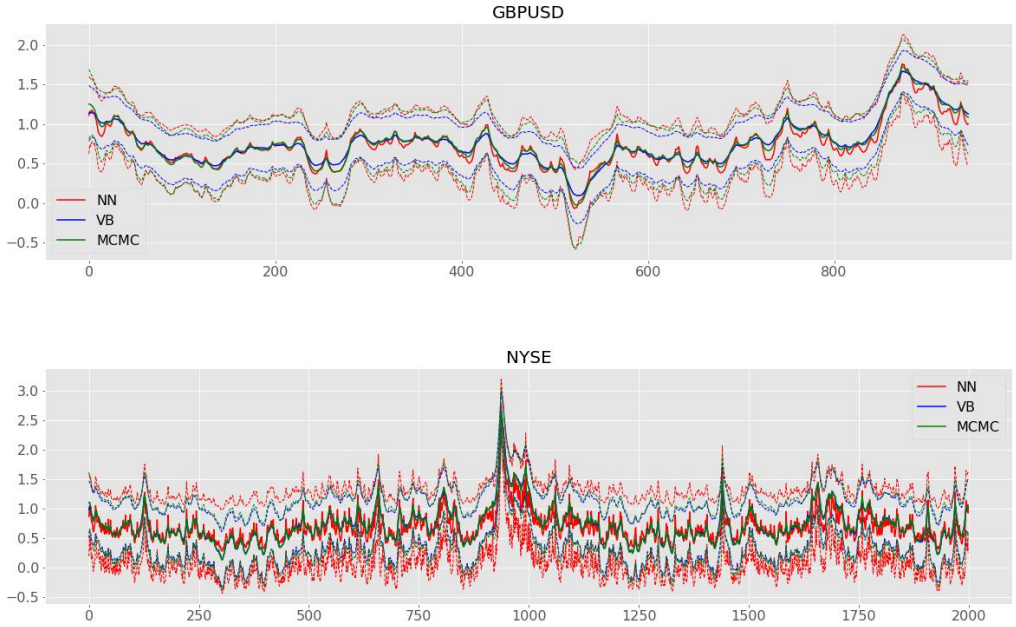


Table 1. NLL and MSE for various applications (mean ± 2 std)

	NLL	MSE
SV	$(8.17 \pm 0.21) \times 10^{-2}$	$(2.92 \pm 0{,}02) \times 10^{-1}$
SV-DSGE	$(-1.64 \pm 0{,}00) \times 10^{-1}$	$(4.77 \pm 0{,}00) \times 10^{-2}$
SA	$-3.00 \pm 0{,}01$	$(1.27 \pm 0{,}02) \times 10^{-3}$

[6] Calculating metrics, such as C2ST, are not informative for the joint distribution. Information about correlations is important for classification, but the algorithm used assumes the diagonal covariance matrix. Calculation of C2ST for marginal distributions requires training the number of classifiers equal to the number of hidden states. So, we use only visual analysis.

Figure 2. Comparison of NPE with MCMC and VB on real data for stochastic volatility model (mean ± 2 std)



3.2. Stochastic volatility DSGE model

The stochastic volatility model, while contains many hidden states, is a univariate model, both in terms of data and output[7]. The DSGE model was chosen to test the amortized SBI algorithm in multivariate context. This class of models is widely used by macroeconomists, both for practical purposes (see Linde, Smets and Wouters (2016)) and in academic research (see Walsh (2010)). Although solving non-linear DSGE models is beyond the scope of this paper, we seek to demonstrate that the proposed algorithm works well for models where filtering and likelihood estimation cannot be executed using the Kalman filter as in the case of linear models[8]. A simplified DSGE model[9] with stochastic volatility from Diebold, Schorfheide and Shin (2017) was chosen for these reasons. This model can be solved using standard algorithms for linearized models (see Blanchard and Kahn (1980), Anderson and Moore (1985), Klein (2000), Sims (2002)). Nonlinearity is introduced after the solution step as the time-varying volatility of model shocks. The neural network is estimated on

[7] The latter could potentially be an advantage, though, as information from different sources can help in training neural network parameters. Moreover, it is possible to train a univariate model for each dimension of state vector, which reduces the problem to the previous one in terms of output.

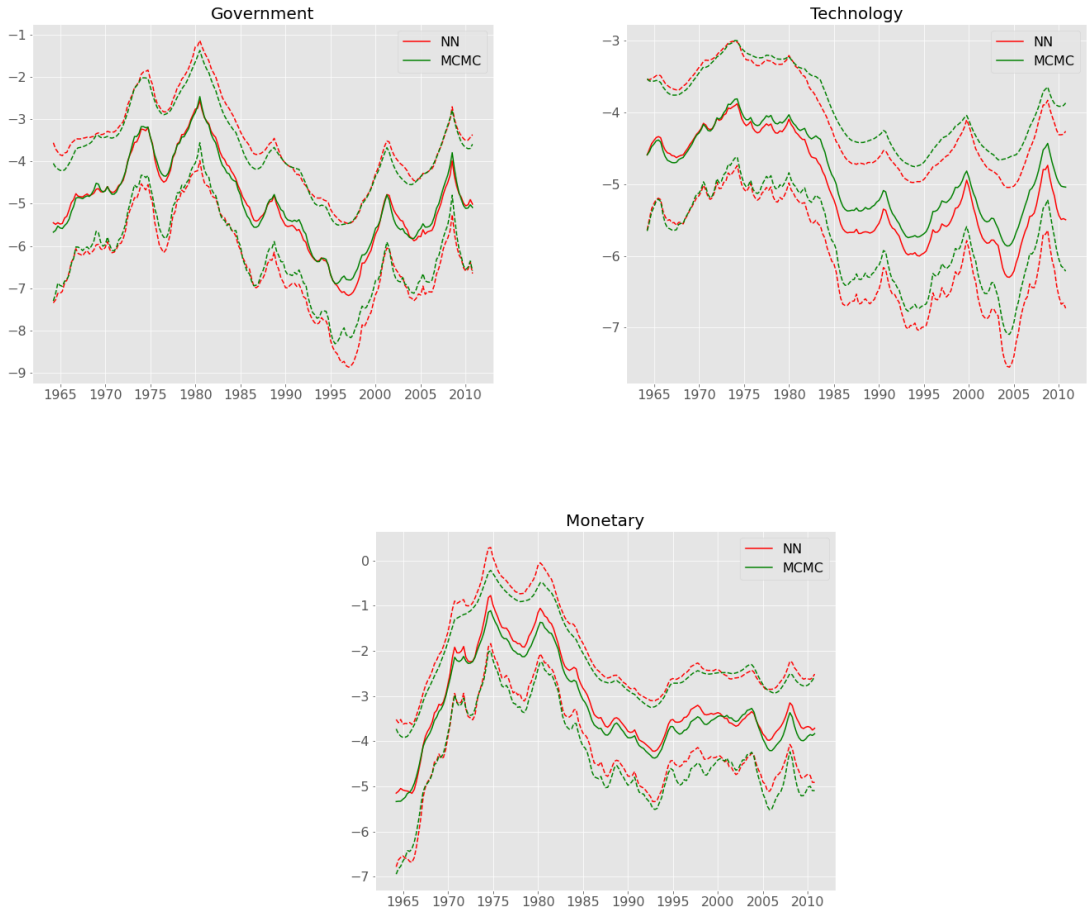
[8] Log-linearized versions of these models are often used to overcome the computational difficulties with solving and estimating non-linear models.

[9] We exclude the inflation target shock from the model and inflation expectations from the observed variables to avoid the issues of missing variables and their impact on the result. A number of additional experiments have shown that using trained values for neural network inputs in place of missing variables (see Lueckmann et al. (2017)) and introducing additional dummy variables to the RNN input can cope with this task. However, we will focus here on a simpler version of the model to separate the effect of multiple observed variables from the effect of missing data. Estimates of DSGE models using SBI will be the subject of a separate paper where we will also touch on this issue.

50,000,000 generated datasets with a length of 180 to 200 points and compared with the adaptive MCMC algorithm (model description, neural network architecture and MCMC implementation are described in Appendix C).

To illustrate the properties of NPE, we concentrated on the estimation of stochastic volatilities and present graphs and metrics for these states. However, unobservable shocks (more precisely, the logarithms of their absolute values) were also used in the estimation as a hint on the intermediate outputs of the neural network. Figure 3 shows that the trained neural network results are similar to MCMC for the US data from 1964Q2 to 2011Q1 (latest vintage in Diebold, Schorfheide and Shin (2017)). As for the previous model, Table 1 shows the NLL and MSE on 1,000,000 randomly generated datasets for future comparisons.

Figure 3. Comparison of NPE with MCMC on US data for DSGE model with stochastic volatility, stochastic volatility (mean ± 2 std)



3.3. Seasonal adjustment with structural breaks in seasonality

In addition to problems based on well-verified formulas for transition and observation equations, SBI is also suitable for those models where simulations are the primary focus. Usually,

this occurs in models where deriving equations is too cumbersome or simply impossible to compute. Special cases are tasks where it is easy to generate a lot of different data which shows the model how it should behave in various situations.

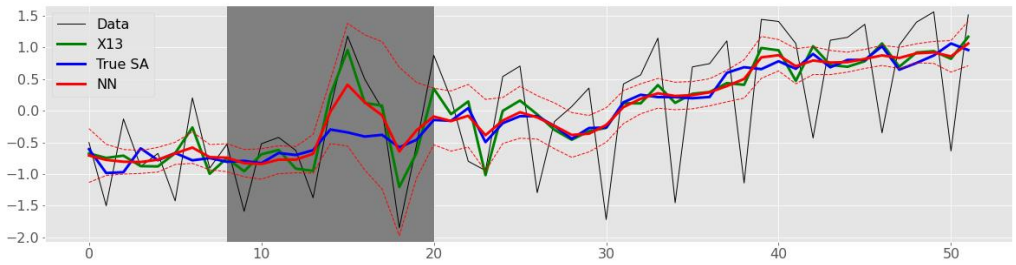
As an example, we show how a quarterly seasonal adjustment model that considers structural shifts in the seasonal component can be built. As will be shown below, a traditional X13 ARIMA-SEATS procedure (see US Census Bureau (2017)) does a poor job of this. An additional advantage is the automatically generated credible intervals.

Appendix D presents the procedure for generating artificial series with a length of 40 to 80 quarters. In fact, it consists of generating a seasonal and non-seasonal component with the probability of a shift appearing in the seasonal part. Thus, the resulting series may not contain a break.

We compare NPE not with a sampling algorithm, but with X13 for measuring the quality, in contrast to the previous two models. The purpose of this experiment is to demonstrate how one can easily generate examples of model behavior, thus specifying an implicit Bayesian model. In practice, it is usually difficult to construct a fast MCMC algorithm in such cases. However, comparison with other algorithms that solve the same practical problem is of interest from the point of view of estimating the performance of the proposed algorithm.

Figure 4 shows random examples illustrating how the proposed procedure and X13 behave on series with a shift in seasonality (the gray area shows 1.5 years around the shift). X13 does not adequately cope with the task of detecting seasonality around the quarter of shift, while NPE, on the contrary, is robust. We calculated the MSE on 100,000 randomly generated runs for X13 and on 1,000,000 runs for the neural network. The error for NPE is smaller, as can be seen from Table 2. The smaller errors in comparison to X13 are not unexpected by themselves since cross-entropy optimization should provide the estimator with the lowest MSE on this dataset. However, the gap between the errors on the series with and without shifts further emphasizes that the algorithm proposed in the paper is more accurate than the alternative widely used among macroeconomists.

Figure 4. Comparison of NPE and X13 ARIMA-SEATS on artificial series with a break in seasonality (mean ± 2 std)



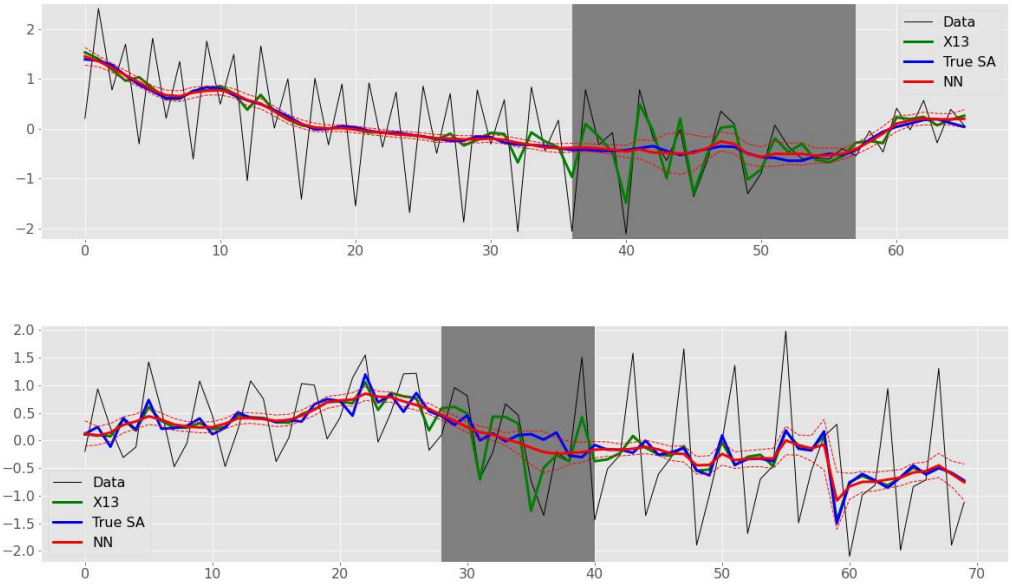


Table 2. MSE for NPE and X13-ARIMA-SEATS on artificial data

	Full sample	With shifts	Without shifts
NPE	1.3×10^{-3}	3.0×10^{-3}	1.1×10^{-3}
X13-ARIMA-SEATS	5.4×10^{-3}	25.0×10^{-3}	3.1×10^{-3}

3.4. Computation and implementation time

As noted above, NPE works almost instantly due to possessing an amortization property. Depending on the task, calculating an approximation of the posterior distribution on the CPU (Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 16GB RAM) takes tenths of a second for a pretrained model. Estimation of neural network parameters on Pytorch[10] (see Paszke et al. (2019)) using GPU (NVIDIA GeForce RTX 2070) takes about 12, 12 and 2 hours for the stochastic volatility model, DSGE model and seasonal adjustment model, respectively. Comparing the running time for amortized algorithms with alternatives that do not possess such properties comes with difficulties. On the one hand, Bayesian model estimation for a fixed dataset takes less time in our examples when MCMC or VB algorithms are used (see Table 3). This is because the amortized algorithm requires pretraining of a neural network. On the other hand, a neural network trained once can be used for various datasets (including data of different lengths) which is an advantage in the case of multiple estimations.

[10] Generation of artificial data for the model is run on the CPU and training on the GPU.

Table 3. Time of posterior computation per one dataset[11]

	NPE		VB		MCMC	Other
	CPU	GPU	CPU	GPU	CPU	CPU
SV	0.42s	0.08s	1h 27m	16m	19m	-
SV-DSGE	0.14s	0.02s	-	-	9h18m	-
SA	0.14s	0.05s	-	-	-	0.27s

The implementation of the NPE algorithm uses already available neural network libraries (coding a neural network architecture usually takes only tens of minutes since the layers of the neural network are already implemented in the respective libraries) and almost always turns out to be much easier than the implementation of MCMC and even sometimes variational estimation. The implementation of MCMC requires the derivation of the sampling algorithm and the writing of program code, which is often much more complicated than for NPE. Like NPE, stochastic VB algorithms are just as easy to implement in most cases (if automatic differentiation packages are used and there are no modules that require manual differentiation coding). The key difference is that the joint density of parameters, hidden states and data is used instead of data sampling procedure.

An important point for stochastic optimization algorithms (NPE and VB) is also the ability to almost effortlessly transfer the computation to the GPU.

4. Related work

The algorithm proposed in this paper is closely related to several directions presented in the literature. Our work is part of the literature on SBI (or likelihood-free inference) algorithms (see Crammer, Brehmer and Louppe (2020)). Until recently, Bayesian direction in this field has developed mainly as approximate Bayesian computations (hereinafter, ABC). ABC approximate the likelihood functions by introducing an auxiliary likelihood function that depends on the distance between summary statistics of the data and the corresponding statistics of the simulated data. Classic sampling algorithms are then used (see Sisson, Fan and Beaumont (2018)). The progress of machine learning algorithms, and in particular neural networks, has given rise to a whole family of algorithms that directly train posterior distributions (see Papamakarios and Murray (2016), Lueckmann et al. (2017), Greenberg, Nonnenmacher and Macke (2019), Durkan, Murray and Papamakarios (2020)),

[11] We choose the number of iterations based on convergence of parameters for MCMC (half of iterations is a burn-in period) and convergence of loss for VB.

likelihood functions (see Wood (2010), Lueckmann et al. (2019), Brehmer et al. (2020), Papamakarios, Sterratt and Murray (2019)) or likelihood function ratios (see Brehmer et al (2020), Hermans, Begy and Louppe (2020), Durkan, Murray and Papamakarios (2020)). These methods are not sensitive to tolerance hyperparameter and chosen distance between simulation data and summary statistics, contrary to ABC. However, to the best of our knowledge, apart from a few papers on probabilistic programming (see Le, Baydin and Wood (2017), Baydin et al. (2019), Munk et al. (2022)), researchers concentrate mainly on parameters, not states. Research on probabilistic programming has two key differences from the approach proposed here. First, it uses the pretrained neural network as a proposal distribution for importance sampling, rather than directly to approximate the posterior distribution. Second, a neural network in probabilistic programming takes into account the relationship between variables to achieve smaller variance for the importance sampling weights, which is a considerably more difficult task in terms of optimization[12]. Moreover, unlike this paper, implementation from scratch or modification of probabilistic programming algorithms for tasks that do not fit into the framework of standard libraries[13] is quite complicated, since it requires a deep knowledge of the addressing of random variables.

Our research is also closely related to the estimation of economic models through simulations. The simulated method of moments and its modifications (see McFadden (1989), Duffy and Singleton (1993), Gallant and Tauchen (1996)) are common in the frequentist estimation of the structural parameters of models[14]. There is a similar field of research that estimates model parameters based on minimizing various divergences between simulated and real data (see Nickl and Pötscher (2010), Kaji, Manresa and Pouliot (2022)). Gallant and McCalloch (2009)[15] proposed a Bayesian version of the simulated method of moments. A recent paper by Fen (2022) uses sequential (non-amortized) NPE for Bayesian parameter estimation. As for SBI, not many papers devoted to simulated estimation of states rather than parameters exist. The closest known to us is the paper by Deli Gatti and Grazzini (2020), where the authors estimate states (output and investment gap forecasts) on artificial data using nonparametric kernel estimation. The idea is very close to SBI and to what is proposed here, but it is computationally difficult with a large number of hidden states as the authors themselves note.

[12] It is also worth noting that probabilistic programming uses state sampling which depends on the sampled states of the previous period. This can lead to accumulation of approximation errors over long periods. Such an architecture is not quite suitable for direct approximation. However, this is not critical for subsequent resampling, especially if sequential importance sampling is used instead of importance sampling.
[13] See, for instance, PyProb.
[14] See a list of applications in Carrasco and Florens (2002).
[15] Gallant, Giacomini and Ragusa (2013) also developed a version of the Bayesian simulation method of moments based on a particle filter for models with hidden states.

Meta-learning (see Finn and Levine (2019)) is close to SBI in its mathematical formulation. Like SBI, meta-learning is based on the idea of learning from many similar tasks (see Vinyals et al. (2016)). The key differences are purpose and data. Unlike SBI, meta-learning focuses on the task of predicting rather than finding the posterior distribution. Furthermore, meta-learning usually works with real data, not simulated ones.

Many works on variational autoencoders have been devoted to the amortization of finding the distribution of hidden states (see Kingma and Welling (2019)). However, there are a number of differences from this paper. First, the data generation process is usually specified with a rather flexible model such as a neural network (see Kingma and Welling (2014)) or a Gaussian process (see Dai et al. (2016)) rather than more classical models where the hidden states have greater identifiability and interpretability. In addition, the model is usually non-Bayesian in nature[16]. Second, the loss function that is minimized is the KL divergence between the approximate posterior and posterior distributions, while in SBI it is the KL divergence between the posterior and approximate posterior distributions. The asymmetry of KL divergence leads to the fact that, with few exceptions (see Tran, Ranganath and Blei (2017)), there are not enough simulations to train variational autoencoders and one must calculate the probabilities of the data generation process. Also, in the case of diagonal approximation (as in Section 2.3), this leads to underestimation of variance (see Blei, Kucukelbir and McAuliffe (2018)). Third, real, not artificial, data are used for training variational autoencoders as for meta-learning.

5. Discussion

As has been shown in many papers (see Lueckmann et al. (2021)), amortization leads to the need for longer training of SBI algorithms than their sequential counterparts. Despite this, we use the amortized NPE algorithm for two reasons. First, sequential SBI algorithms are usually applied to the problems of small dimension (the output of neural network dimension), and their adaptation to the high-dimensional problem of finding the posterior distribution of states is not trivial and requires solving more practical issues. In particular, if one tries to focus on marginal densities, as is done in this paper, the states generated for new rounds will not look like posterior distribution due to the lack of dependence between variables. The states will be quite noisy, which worsen convergence in most

[16] Basically, neural networks are used as a data generation model, which by their nature are frequentist. Moreover, despite the fact that many applications use dropout for regularization (see Srivastava et al. (2014)), which has a Bayesian interpretation (see Kingma, Salimans and Welling (2015) and Gal and Ghahramani (2016)), the parameters are common to all data. This is ideologically different from the idea of amortizing models.

cases. Secondly, in contrast to some research on SBI, we do not set the task to finding the best algorithm under the constrained budget for the number of simulations (see Lueckmann et al. (2021)). Sequential algorithms give a significant gain for such tasks. However, the main goal of this paper is to build an algorithm that replaces long-running alternatives (as in the first two examples) or helps to estimate models where other algorithms fail (as in the third example) [17]. Amortization is a great property that helps to solve this problem if the model is frequently re-estimated.

Many issues related to the estimation of the posterior distribution of states are beyond the scope of this paper and require further research. Some of them are discussed below.

The posterior distributions estimated using the proposed algorithm, although close to the MCMC results, nevertheless differ slightly. The results are slightly noisy when estimating stochastic volatility, while in the DSGE model they are biased. This signals an opportunity for further improvements of the neural networks by increasing the flexibility of the neural network architecture, the number of simulations or by modifying the training procedure. It is well known that for a certain learning rate schedule, stochastic optimization procedures converge to one of the local optima in the asymptotics (see Chapter 5 in Kushner and Yin (2003)). The exact (even local) optimum is not achieved with a finite number of iterations[18,19]. An insufficiently flexible network and/or a small number of observations in the neighborhood of real data can lead to a situation where the model is unable to predict the posterior distribution accurately, even at the optimum[20]. Moreover, the quality of the posterior distribution approximation is likely to deteriorate with increasing problem dimensionality. Hence, one of the main tasks for the future is to study the relationship between scalability, approximation quality and neural network training time.

The mean-field Gaussian approximation considered here is usually not a problem from a practical point of view, because in most cases, researchers are interested in the first and second moments of the marginal distributions of states. Although extensions have clear theoretical solutions (M-estimators for estimating other characteristics and more flexible families of distributions), their practical implementation requires further research[21].

[17] It is implicitly assumed that a large number of model simulations can be performed in adequate time.
[18] This usually means that the learning rate does not tend to zero.
[19] Mandt, Hoffman and Blei (2017) provide intuitions about the behavior of the estimation procedure at non-zero learning rates.
[20] A good example of such type an improvement in the field of text analysis is the GPT-3 model (see Brown et al. (2020)). It has reached a fundamentally new level compared to previous models due to an order of magnitude more parameters than previously used and a huge dataset.
[21] In a number of preliminary experiments that were not included in the paper, we saw that the quantile loss also shows good results for the marginal distributions.

We have bypassed the issues of forecasting and missing variables, which are related in the sense that the forecasting problem can be thought of as a problem of constructing a posterior distribution for the missing variables on the forecasting horizon. To deal with missing variables, models can be extended by introducing additional dummy variables as one of the inputs of the neural network, showing the presence of a miss, and/or by filling in the miss with learnable parameters as done in Lueckmann et al. (2017). A similar method or alternatives based on meta-learning ideas (see Harrison, Sharma and Pavone (2020)), where only the predicted variables are used as the neural network output, can be applied to build prediction models.

NPE has both advantages and disadvantages in terms of speed as shown in Section 3.4. Therefore, the choice to use NPE or not should depend on the situation. We recommend using the NPE algorithm if frequent re-estimation of the model is expected, or if alternative algorithms are slow, or fail to do the job at all. At the same time, we also advise to verify the trained algorithm before use by comparing it with alternative ones for approximating the posterior distribution (when they are not too slow). If such verification is impossible, it is recommended to carry out at least a visual analysis on artificially generated data.

6. Conclusions

The amortized simulation-based algorithm proposed in this paper for estimating hidden states of Bayesian state space models provides an alternative to already existing algorithms in this field. In contrast to many previous papers, we consider a new approach that approximates posterior marginal distribution of states and that does not rely on probability density functions for prior distributions, transition and observation equations, but that uses only simulations of artificial data.

The NPE algorithm shows results similar to other algorithms for the stochastic volatility and DSGE models but after training, it works nearly instantly. In addition, as shown in the example with seasonal adjustment, it also performs well on tasks where the Bayesian model is not specified directly but rather through the process of simulating various situations and correct behavior in them.

References

Anderson, G. and G. Moore (1985). A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models. *Economics Letters*, 17(3): 247-252.

Andrieu, C., A. Doucet and R. Holenstein (2010). Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society Series B*, 72(3): 269-342.

Baydin, A.G., L. Shao, W. Bhimji, L. Heinrich, L.F. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, M. Ma, X. Zhao, P. Torr, V. Lee, K. Cranmer, Prabhat and F. Wood (2019). Etalumis: Bringing Probabilistic Programming to Scientific Simulators at Scale. *In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC19)*.

Beal, M.J. (2003). Variational Algorithms for Approximate Bayesian Inference. *PhD Thesis, Gatsby Computational Neuroscience Unit, University College London*.

Beaumont, M.A., W. Zhang and D.J. Balding (2002). Approximate Bayesian Computation in Population Genetics. *Genetics*, 162(4): 2025-2035.

Blanchard, O.J. and C.M. Kahn (1980). The Solution of Linear Difference Models under Rational Expectations. *Econometrica*, 48(5): 1305-1311.

Blei, D.M., A. Kucukelbir and J.D. McAuliffe (2018). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518): 859-877.

Blum, M.G.B. and O. François (2010). Non-linear Regression Models for Approximate Bayesian Computation. *Statistics and Computing*, 20: 63-73.

Brehmer, J., G. Louppe, J. Pavez, and K. Cranmer (2020). Mining Gold from Implicit Models to Improve Likelihood-Free Inference. *Proceedings of the National Academy of Sciences*, 117(10): 5242-5249.

Brown, T.B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei (2020). Language Models are Few-Shot Learners. *arXiv:2005.14165v4*.

Carrasco, M. and J.-P. Florens (2002). Simulation-Based Method of Moments and Efficiency. *Journal of Business and Economic Statistics*, 20(4): 482-492.

Carriero, A., T.E. Clark and M. Massimiliano (2019). Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors. *Journal of Econometrics*, 212(1): 137-154.

Casella, G. and E.I. George (1992). Explaining the Gibbs Sampler. *The American Statistician*, 46: 167-174.

Chib, S. and E. Greenberg (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4): 327-335.

Chiu, C.W., B. Eraker, A.T. Foerster, T.B. Kim and Hernan D. Seoane (2012). Estimating VAR's Sampled at Mixed or Irregular Spaced Frequencies: a Bayesian Approach. *Federal Reserve Bank of Kansas City RWP*, 11-11.

Chopin, N., P.E. Jacob and O. Papaspiliopoulos (2012). SMC^2: An Efficient Algorithm for Sequential Analysis of State Space Models. *Journal of the Royal Statistical Society Series B*, 75(3):397–426.

Cranmer, K., J. Brehmer and G. Louppe (2020). The Frontier of Simulation-Based Inference. *Proceedings of the National Academy of Sciences*, 117(48):30055-30062.

Dai, Z., A. Damianou, J. González and N. Lawrence (2016). Variational Auto-encoded Deep Gaussian Processes. *International Conference on Learning Representations*.

Del Moral, P., A. Doucet and A. Jasra (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B*, 68(3):411:436.

Deli Gatti, D. and J. Grazzini (2020). Rising to the Challenge: Bayesian Estimation and Forecasting Techniques for Macroeconomic Agent Based Models. *Journal of Economic Behavior and Organization*, 178:875-902.

Diebold, F.X., F. Schorfheide and M. Shin (2017). Real-time Forecast Evaluation of DSGE Models with Stochastic Volatility. *Journal of Econometrics*, 201(2):322-332.

Duffie, D. and K. Singleton (1993) Simulated Moments Estimation of Markov Models of Asset Prices. *Econometrica*, 61:929-1052.

Durbin J. and S.J. Koopman (2002). A Simple and Efficient Simulation Smoother for State Space Time Series Analysis. *Biometrika*, 89(3):603-615.

Durkan, C., I. Murray and G. Papamakarios (2020). On Contrastive Learning for Likelihood-Free Inference. *In Proceedings of the 36th International Conference on Machine Learning*.

Fen, C. (2022). Fast Simulation-Based Bayesian Estimation of Heterogeneous and Representative Agent Models using Normalizing Flow Neural Networks. *arXiv:2203.06537v1*.

Fernandez-Villaverde, J., J.F. Rubio-Ramirez and F. Schorfheide (2016). Solution and Estimation Methods for DSGE Models. *Handbook of Macroeconomics*, 2:527-724.

Finn, C. and S. Levine (2019). Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning. *The International Conference on Machine Learning*, Tutorial.

Gal, Y. and Z. Ghahramani (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Neural Information Processing Systems*.

Gallant, A.R., R. Giacomini and G. Ragusa (2013). Generalized Method of Moments with Latent Variables. *CEPR Discussion Papers*, DP9692.

Gallant, A.R. and R.E. McCulloch (2009). On the determination of general statistical models with application to asset pricing. *Journal of the American Statistical Association*, 104:117-131.

Gallant, A.R. and G. Tauchen (1996). Which moments to match? *Econometric Theory*, 12:657-681.

Goodfellow, I., Y. Bengio and A. Courville (2016). Deep Learning. *MIT Press*.

Greenberg, D., M. Nonnenmacher, and J. Macke (2019). Automatic Posterior Transformation for Likelihood-Free Inference. *In Proceedings of the 36th International Conference on Machine Learning*.

Gunawan, D., R. Kohn and D. Nott (2021). Variational Bayes Approximation of Factor Stochastic Volatility Models. *International Journal of Forecasting*, 37(4):1355-1375.

Hamilton, J, (1989). A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, 57(2):357-384.

Harrison, J., A. Sharma and M. Pavone (2020). Meta-Learning Priors for Efficient Online Bayesian Regression. *Algorithmic Foundations of Robotics XIII*, 14:318-337.

Hermans, J., V. Begy and G. Louppe (2020). Likelihood-Free MCMC with Approximate Likelihood Ratios. *In Proceedings of the 37th International Conference on Machine Learning*.

Hodrick, R. and E. Prescott (1997). Postwar U.S. Business Cycles: An Empirical Investigation. *Journal of Money, Credit and Banking*, 29(1):1-16.

Hoffman, M.D, D.M. Blei, C. Wang and J. Paisley (2013). Stochastic Variational Inference. *Journal of Machine Learning Research*, 14(1):1303-1347.

Justiniano, A. and G.E. Primiceri (2008). The Time-Varying Volatility of Macroeconomic Fluctuations. American Economic Review, 98(3):604-641.

Kaji, T., E. Manresa and G. Pouliot (2022). An Adversarial Approach to Structural Estimation. *arXiv:2007.06169v2*.

Kim, S., N. Shepard and S. Chib (1998). Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models. *The Review of Economic Studies*, 65(3):361-393.

Kingma, D.P. and J. Ba (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Kingma, D.P., T. Salimans and M. Welling (2015). Variational Dropout and the Local Reparametrization Trick. *In Advances in Neural Information Processing Systems*, 2575-2583.

Kingma, D.P. and M. Welling (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations*.

Kingma, D.P. and M. Welling (2019). An Introduction to Variational Autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307-392.

Klein, P. (2000). Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model. *Journal of Economic Dynamics and Control*, 24(10):1405-1423.

Koop, G. and D. Korobilis (2012). Forecasting Inflation Using Dynamic Model Averaging. *International Economic Review*, 53(3):867-886.

Kushner, H.J. and G.G. Yin (2003). Stochastic Approximation Algorithms and Recursive Algorithms and Applications. *Springer Science and Business Media*.

Laubach, T. and J.C. Williams (2003). Measuring the Natural Rate of Interest. *The Review of Economics and Statistics*, 85(4):1063-1070.

Le, T.A., A.G. Baydin and F. Wood (2017). Inference compilation and universal probabilistic programming. *In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Linde, J., F. Smets and R. Wouters (2016). Challenges for Central Banks' Macro Models. *Handbook of Macroeconomics*, 2:2185-2262.

Lueckmann, J.-M., G. Bassetto, T. Karaletsos and J.H. Macke (2019). Likelihood-free Inference with Emulator Networks. *In Proceedings of the 1st Symposium on Advances in Approximate Bayesian Inference*.

Lueckmann, J.-M., J. Boelts, D.S. Greenberg, P.J. Gonçalves and J.H. Macke (2021). Benchmarking Simulation-Based Inference. *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Lueckmann, J.-M., P.J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher and J.H. Macke (2017). Flexible Statistical Inference for Mechanistic Models of Neural Dynamics. *In Advances in Neural Information Processing Systems*, 30:1289-1299.

Lux, T. (2018). Estimation of Agent-Based Models Using Sequential Monte Carlo Methods. *Journal of Economic Dynamics and Control*, 91:391-408.

Mandt, S., M.D. Hoffman and D.M. Blei (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18:1-35.

McFadden, D. (1989). A Method of Simulated Moments for Estimation of Discrete Response Models Without Numerical Integration. *Econometrica*, 57(5):995-1026.

Munk, A., B. Zwartsenberg, A. Scibior, A.G. Baydin, A.L. Stewart, G. Fernlund, A. Poursartip and F. Wood (2022). Probabilistic Surrogate Networks for Simulators with Unbounded Randomness. *arXiv:1910.11950v2*.

Neal, R.M. (1996). Bayesian Learning for Neural Networks. *Springer-Verlag, Lecture Notes in Statistics*, №118.

Nickl, R. and B.M. Pötscher (2010). Efficient Simulation-Based Minimum Distance Estimation and Indirect Inference. *Mathematical Methods of Statistics*, 19:327-364.

Orphanides, A. and S. Van Norden (2002). The Unreliability of Output-gap Estimates in Real Time. *Review of Economics and Statistics*, 84(4):569-583.

Otrok, C. and C.H. Whiteman (1998). Bayesian Leading Indicators: Measuring and Predicting Economic Conditions in Iowa. *International Economic Review*, 39(4):997-1014.

Papamakarios, G. and I. Murray (2016). Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation. *In Advances in Neural Information Processing Systems*, 29:1028-1036.

Papamakarios, G., D. Sterratt and I. Murray (2019). Sequential Neural Likelihood: Fast Likelihood-Free Inference with Autoregressive Flows. *In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *In Advances in Neural Information Processing Systems*, 32: 8024-8035.

Primiceri, G.E. (2005). Time Varying Structural Vector Autoregressions and Monetary Policy. *The Review of Economic Studies*, 72(3):821-852.

Rezende, D. and S. Mohamed (2015). Variational Inference with Normalizing Flows. *In Proceedings of the 32nd International Conference on Machine Learning*.

Schorfheide, F. and D. Song (2015), Real-Time Forecasting with a Mixed-Frequency VAR, *Journal of Business and Economic Statistics*, 33(3):366-380.

Schorfheide, F. and D. Song (2021). Real-Time Forecasting with a (Standard) Mixed-Frequency VAR During a Pandemic. *NBER Working Papers*, № 29535.

Sims, C.A. (2002). Solving Linear Rational Expectations Models. *Computational Economics*, 20:1-20.

Sisson, S.A., Y. Fan, M. Beaumont (2018). Handbook of Approximate Bayesian Computation. *Chapman and Hall/CRC*.

Smets, F. and R. Wouters (2003). An Estimated Dynamic Stochastic General Equilibrium Model of the Euro Area. *Journal of the European Economic Association*, 1(5):1123-1175.

Smets, F. and R. Wouters (2007). Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach. *American Economic Review*, 97(3):586-606.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929-1958.

Stock, J.H. and M.W. Watson (2011). Dynamic Factor Models. *In Clements M.J. and D.F. Hendry Oxford Handbook on Economic Forecasting*.

Tan, L., A. Bhaskaran and D. Nott (2020). Conditionally Structured Variational Gaussian Approximation with Importance Weights. *Statistics and Computing*, 30(5).

Tan, L. and D. Nott (2018). Gaussian Variational Approximation with Sparse Precision Matrix. *Statistics and Computing*, 28(2):259-275.

Tran, D., R. Ranganath and D.M. Blei (2017). Hierarchical Implicit Models and Likelihood-Free Variational Inference. *Neural Information Processing Systems*.

US Census Bureau (2017). X-13 ARIMA-SEATS Reference Manual. *US Census Bureau*.

Van der Vaart, A.W. (2000). Asymptotic Statistics. *Cambridge University Press*.

Vinyals, O., C. Blundell, T. Lillicrap, K. Kavukcuoglu and D. Wierstra (2016). Matching Networks for One Shot Learning. *In Advances in Neural Information Processing Systems*.

Wainwright, M.J. and M. Jordan (2008). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1-305.

Walsh C.E. (2010). Monetary Theory and Policy. *MIT Press*, 3rd edition.

Wood, S.N. (2010). Statistical Inference for Noisy Nonlinear Ecological Dynamic Dystems. Nature, 466(7310):1102-1104.

Appendix A. Informal proofs

A1. NPE asymptotic

Given a flexible parametric family $q_\varphi(\theta|y)$ and an infinite number of datasets, problem (1) becomes equivalent to the following problem:

$$\begin{aligned} q^* &= argmin_q E_{p(\theta,y)}(-\log q(\theta|y) - \log p(y)) \\ &= argmin_q \int\int (-\log q(\theta|y) - \log p(y)) p(\theta,y) d\theta dy \\ &= argmin_q \int \left(-\int \log q(\theta|y)\, p(\theta|y) d\theta\right) p(y) dy \end{aligned}$$

Note that the optimization problem is split into a set of separate cross-entropy minimizations for each individual dataset $-\int \log q(\theta|y)\, p(\theta|y) d\theta$. The cross-entropy minimum is reached when the distributions coincide, which means that

$$q^* = p(\theta|y)$$

A2. NPE for states

The only thing we need to prove for the validity of Algorithm 1 is that the joint distribution of sampled states and data is the marginal distribution of the data generation process. We can then use the results of Appendix A.1 by redefining s as θ.

The sample θ_i, s_i, y_i from Algorithm 1 is identical to the sample from the data generation process, i.e. $\theta_i, s_i, y_i \sim p(\theta, s, y)$. Integrating over θ we obtain that $s_i, y_i \sim p(s, y)$, where $p(s, y)$ is the marginal distribution of the data generation process.

A3. NPE for marginal distribution loss

Consider an M-estimator loss function $m(x, s, y)$, where y is the set of observed variables, s is the set of hidden states, and x is the set of posterior distribution characteristics estimated by the M-estimator. Let's assume that $f_\varphi(y)$ is a parametric family of functions that maps dataset to the characteristics of the posterior distribution. With a flexible function f and the number of simulations tending to infinity, we obtain

$$f^* = argmin_f E_{p(s,y)} m(f(y), s, y) = argmin_f \int \left(\int m(f(y), s, y) p(s|y) ds\right) p(y) dy$$

The problem splits into a set of minimizations for individual datasets $\int m(f(y), s, y) p(s|y) ds$ and $f^*(y)$ coincides with the M-estimator asymptotic value for each y similarly to Appendix A.1. Thus, a set of classical M-estimators can be used to estimate the characteristics of the posterior distribution.

Means and standard deviations converge to their true values when an independent normal distribution is used as an approximation for the posterior distribution. So, if the function f is sufficiently flexible and the number of simulations tends to infinity, the mean and standard deviations converge to their true values.

Appendix B. Stochastic volatility model

B1. Model

Prior:

$$\alpha \sim N\left(0, \sqrt{10}\right), \qquad \kappa \sim N\left(0, \sqrt{10}\right), \qquad \psi \sim N\left(0, \sqrt{10}\right)$$

$$\sigma = \log(1 + e^{\alpha}), \qquad \rho = \frac{1}{1 + e^{-\psi}}$$

Transition equation:

$$SV_t \sim N\left(\frac{\kappa}{2}(1-\rho) + \rho SV_{t-1}, \frac{\sigma}{2}\right), \qquad t = 2, \dots, T$$

$$SV_1 \sim N\left(\frac{\kappa}{2}, \frac{\sigma}{2\sqrt{1-\rho^2}}\right)$$

Observation equation:

$$y_t \sim N(0, e^{SV_t})$$

B2. Architecture and learning algorithm

Algorithm B1. Pretraining stochastic volatility model (B =100, N_{sim} = 200 000, T_{lb} = 800, T_{ub} = 1200, $c = 10^{-30}$**)**

For $n = 1, \dots, N_{sim}$:

1. Draw T_n from uniform discrete distribution $T \sim U(T_{lb}, T_{ub})$.
2. Simulate n^{th} batch:

   For $b = 1, \dots, B$:

   2.a. Draw $\sigma^b, \kappa^b, \rho^b$ from prior.

   2.b. Draw $SV^b = \left\{SV_1^b, \dots, SV_{T_n}^b\right\}$ conditioned on $\sigma^b, \kappa^b, \rho^b$.

2.c. Draw $\tilde{y}^b = \{log(c + |y_1^b|), \dots, log(c + |y_{T_n}^b|)\}$ conditioned on SV^b.

$SV_n^{batch} = \{\{SV^1, \tilde{y}^1\}, \dots, \{SV^B, \tilde{y}^B\}\}$

3. Compute per state loss using architecture illustrated in Figure B1:

$$L_n = -\frac{1}{BT_n}\sum_{b=1}^{B}\sum_{t=1}^{T_n} log\, p\left(SV_t^b | m_t(\tilde{y}^b, \varphi), \sigma_t(\tilde{y}^b, \varphi)\right)$$

4. Make an optimization step with respect to φ using ADAM algorithm.

ADAM (Kingma and Ba (2014)) is applied with standard settings except for the learning rate:

$$\varepsilon_n = \begin{cases} 10^{-3}, if\ n < 3 \times 10^4 \\ 10^{-4}, if\ 3 \times 10^4 \leq n < 10^5 \\ 10^{-5}, if\ n \geq 10^5 \end{cases}$$

B3. Alternative algorithm for stochastic volatility model

The algorithms used for comparison are the adaptive MCMC and VB algorithms. The MCMC algorithm is based on an approximation of the logarithm of the square of a Gaussian random variable as a mixture of 7 normal distributions (see Kim, Chib and Shepard (1998)). For this purpose, the observation equation is rewritten as:

$$log\, y_t^2 = 2SV_t + \sum_{k=1}^{7} z_t e_t^k$$

$$e_t^k \sim N(\mu_k, \sigma_k)$$

$$p(z_t = k) = \omega_k$$

where $\mu_k, \sigma_k, \omega_k$ are constants defined in Kim, Chib and Shepard (1998). Algorithm B2 describes the complete procedure.

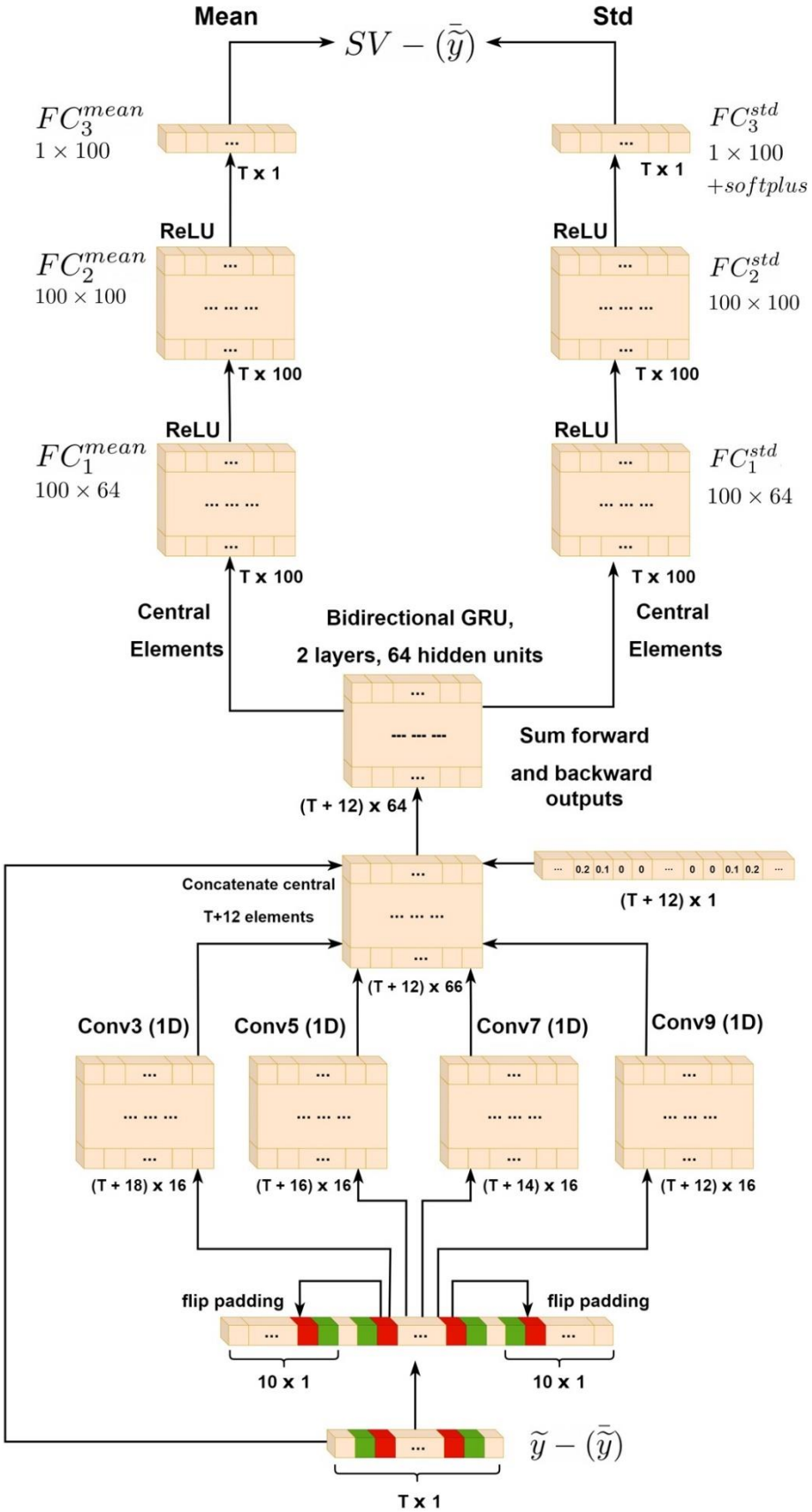
Algorithm B2. Adaptive MCMC algorithm for stochastic volatility model ($N_{sim} = 2000, c = 1.5, \Sigma_0 = 0.1I$**)**

For $n = 1, \dots, N_{sim}$:

1. For $t = 1, \dots, T$ draw discrete approximation to chi-squared distribution:

$$z_t^n \sim p(z_t | SV_t^{n-1}, y_t)$$

Figure B1. Neural network architecture for calculating mean and standard deviation in the stochastic volatility model



2. Draw parameters $\theta = \{\alpha, \kappa, \psi\}$ using Random Walk Metropolis-Hastings algorithm with adaptive proposal (see Roberts and Rosenthal (2009)):

$$q(\theta'|\theta_{n-1}) = 0.95N\left(\theta^{n-1}, \frac{c^2}{3}\Sigma_{n-1}\right) + 0.05N\left(\theta^{n-1}, \frac{0.01^2}{3}I\right)$$

and acceptance rate:

$$ar_n = \min\left(\frac{p(y_1, \dots, y_T|z_1^n, \dots, z_T^n, \theta')p(\theta')}{p(y_1, \dots, y_T|z_1^n, \dots, z_T^n, \theta^{n-1})p(\theta^{n-1})}, 1\right)$$

3. Draw stochastic volatility:

$$SV_1^n, \dots, SV_T^n \sim p(SV_1, \dots, SV_T|z_1^n, \dots, z_T^n, y_1, \dots, y_T, \theta^n)$$

Note that after introducing the variables $z_1, \dots, z_T$, steps 2 and 3 of Algorithm B2 can be implemented via standard Kalman filter and Kalman sampler procedures (see Durbin and Koopman (2002)).

The VB estimator uses a Gaussian approximation algorithm where the precision matrix is sparse (see Tan and Nott (2018)). 20,000 iterations of the ADAM algorithm with a learning rate of 0.001 and a batch size of 100 are used for training.

Appendix C. Stochastic volatility DSGE model

C1. Model

The DSGE model, similar to Diebold, Schorfheide and Shin (2017) is considered, but with several modifications. First, we remove inflation expectations from the observed variables and exclude the shock of inflation target from the model to avoid dealing with omitted variables. Second, prior distributions are slightly changed for processes associated with stochastic volatility to make the simulations more realistic.

Prior:

$$\tau \sim N(1.5, 0.36), \quad \nu_l \sim G(2, 0.75), \quad \iota \sim B(0.5, 0.15), \quad \zeta \sim B(0.5, 0.1), \quad \psi_1 \sim N(1.5, 0.25),$$

$$\psi_2 \sim N(0.12, 0.05), \quad -400\, log\, \beta \sim G(1, 0.4), \quad 400\, log\, \pi_* \sim G(2.48, 0.4),$$

$$100\, log\, \gamma \sim N(0.4, 0.1), \quad \rho_R \sim B(0.5, 0.2), \quad \rho_g \sim B(0.5, 0.2), \quad \varphi_z \sim U(-1, 1),$$

$$(100\sigma_R)^2 \sim IG(0.1, 2), \quad \left(10\sigma_g\right)^2 \sim IG(0.1, 2), \quad (10\sigma_z)^2 \sim IG(0.1, 2), \quad \left(0.2\sigma_g^{SV}\right)^2 \sim IG(0.05, 2),$$

$$(0.2\sigma_z^{SV})^2 \sim IG(0.05, 2), \quad (0.2\sigma_R^{SV})^2 \sim IG(0.05, 2), \quad \rho_g^{SV} \sim N(0.9, 0.07), \quad \rho_z^{SV} \sim N(0.9, 0.07),$$

$$\rho_R^{SV} \sim N(0.9, 0.07)$$

where $N(a,b), B(a,b), G(a,b)$ are normal, beta and gamma distributions with mean a and standard deviation b, $U(a,b)$ is a uniform distribution with upper and lower bounds a and b, $IG(a,b)$ is an inverse gamma distribution with probability density $p(x) \sim x^{-b-1} e^{-\frac{ba^2}{2x}}$.

Transition equations:

The transition equations are given in the form:

$$s_t \sim N\big(A(\theta)s_{t-1}, B(\theta)diag(e^{SV_t})B^T(\theta)\big), \qquad t = 2, \dots, T$$

$$SV_t = \{SV_t^g, SV_t^z, SV_t^R\}$$

$$s_1 \sim N\big(0, P(\theta)\big)$$

$$SV_t^i \sim N\big(\rho_i^{SV} SV_{t-1}^i, \sigma_i^{SV}\big), \qquad t = 2, \dots, T, \;\; i \in \{g, z, R\}$$

$$SV_1^i \sim \left(0, \frac{\sigma_i^{SV}}{\sqrt{1 - \left(\rho_i^{SV}\right)^2}}\right), \qquad i \in \{g, z, R\}$$

where $\theta = \{\tau, \nu_l, \iota, \zeta, \psi_1, \psi_2, \beta, \pi_*, \gamma, \rho_R, \rho_g, \varphi_z, \sigma_R, \sigma_g, \sigma_z\}$, $s_t = \{y_t, c_t, g_t, \pi_t, R_t, z_t, dy_t\}$, $P(\theta)$ is the solution of equation:

$$P(\theta) = A(\theta)P(\theta)A^T(\theta) + B(\theta)B^T(\theta)$$

and $A(\theta)$ and $B(\theta)$ is a stable solution[22] of the following linear system of stochastic discrete equations:

$$c_t = E_t(c_{t+1} + z_{t+1}) - \frac{1}{\tau}(R_t - E_t\pi_{t+1})$$

$$\pi_t = \frac{\iota}{(1+\beta\iota)}\pi_{t-1} + \frac{\beta}{(1+\beta\iota)}E_t\pi_{t+1} + \frac{(1-\zeta\beta)(1-\zeta)}{(1+\beta\iota)\zeta}(c_t + \nu_l y_t)$$

$$y_t = c_t + g_t$$

$$R_t = \rho_R R_{t-1} + (1-\rho_R)\big(\psi_1\pi_t + \psi_2(y_t - y_{t-1} + z_t)\big) + \sigma_R e_t^R$$

$$z_t = -\varphi_z z_{t-1} + \sigma_z e_t^z$$

$$g_t = \rho_g g_{t-1} + \sigma_g e_t^g$$

[22] Parameters for which there are many stable solutions or there is no stable solution are excluded.

$$dy_t = y_t - y_{t-1} + z_t$$

where $y_t, c_t, g_t, \pi_t, R_t, z_t, dy_t$ are variables that correspond to deviations from the steady state of output, consumption, exogenous process responsible for the share of government consumption, inflation, interest rate, exogenous technological process and GDP growth, e_t^R, e_t^z, e_t^g are monetary policy, technology and government consumption shocks.

Observation equations:

The observation equations have the form:

$$obs_t = \begin{bmatrix} dy_t^{obs} \\ \pi_t^{obs} \\ R_t^{obs} \end{bmatrix} = \begin{bmatrix} 100\, log\, \gamma \\ 100\, log\, \pi_* \\ 100(log\, \gamma + log\, \pi_* - log\, \beta) \end{bmatrix} + \begin{bmatrix} 100\, dy_t \\ 100\, \pi_t \\ 100\, R_t \end{bmatrix}$$

where dy_t^{obs} is quarterly real GDP growth, π_t^{obs} is quarterly price growth and R_t^{obs} is interest rate in quarterly terms.

C2. Architecture and learning algorithm

Algorithm C1. Pretraining stochastic volatility DSGE model ($N_{presim} = 1\,000\,000$, $B = 100$, $N_{sim} = 500\,000$, $T_{lb} = 180$, $T_{ub} = 200$, $w = 1$, $c = 10^{-30}$**)**

Set $n_{presim} = 0, \theta = \{\}, A = \{\}, B = \{\}$.

While $n_{presim} < N_{presim}$:

1. Draw $\theta_{n_{presim}}$ from prior.
2. Solve DSGE model[23].
3. If solution is stable[24] append $\theta_{n_{presim}}, A_{n_{presim}}, B_{n_{presim}}$ in θ, A, B and increment n_{presim} by 1.

For $n = 1, \dots, N_{sim}$:

1. Draw T_n from uniform discrete distribution $T \sim U(T_{lb}, T_{ub})$.
2. Simulate n^{th} batch:

   For $b = 1, \dots, B$:

   2.a. Draw $\{\theta^b, A^b, B^b\}$ uniformly from $\{\theta, A, B\}$ and $\sigma_g^{SV,b}, \sigma_z^{SV,b}, \sigma_R^{SV,b}, \rho_g^{SV,b}$,

[23] Anderson and Moore (1985) algorithm is applied.
[24] In our case, there is almost no non-unique stable or unstable solutions. See Lueckmann et al. (2017) as a one of examples how to deal with situations where certain regions of the parameter space are implausible.

$\rho_z^{SV,b}, \rho_R^{SV,b}$ from prior.

2.b. Draw $SV^b = \{SV_1^b, \dots, SV_{T_n}^b\}$, $s^b = \{s_1^b, \dots, s_{T_n}^b\}$ and $\tilde{e}^b = \Big\{\{log(c + |e_1^{R,b}|), log(c + |e_1^{z,b}|), log(c + |e_1^{g,b}|)\}, \dots, \Big\{\{log(c + |e_{T_n}^{R,b}|), log(c + |e_{T_n}^{z,b}|), log(c + |e_{T_n}^{g,b}|)\}\Big\}\Big\}$ conditioned on draw from 2a.

2.c. Draw $obs^b = \{obs_1^b, \dots, obs_{T_n}^b\}$ conditioned on SV^b, s^b and θ^b.

$SV_n^{batch} = \{\{SV^1, obs^1\}, \dots, \{SV^B, obs^B\}\}$

3. Compute loss using architecture illustrated in Figure C1:

$$L_n = -\frac{1}{3BT_n}\sum_{b=1}^{B}\sum_{t=1}^{T_n}\sum_{i\in\{g,z,R\}}\Big(log\, p\left(SV_t^{i,b}|m_{t,i}^{SV}(obs^b,\varphi), \sigma_{t,i}^{SV}(obs^b,\varphi)\right) + w\, log\, p\left(\tilde{e}_t^{i,b}|m_{t,i}^{e}(obs^b,\varphi), \sigma_{t,i}^{e}(obs^b,\varphi)\right)\Big)$$

4. Make an optimization step with respect to φ using the ADAM algorithm.

The learning rate, ε_n, for the ADAM algorithm has the following schedule:

$$\varepsilon_n = \begin{cases} 10^{-3}, if\ n < 3\times10^4 \\ 10^{-4}, if\ 3\times10^4 \le n < 10^5 \\ 10^{-5}, if\ 10^5 \le n < 2\times10^5 \\ 10^{-6}, if\ 2\times10^5 \le n < 3.5\times10^5 \\ 3\times10^{-7}, if\ n \ge 3.5\times10^5 \end{cases}$$

C3. Alternative algorithm for stochastic volatility DSGE model

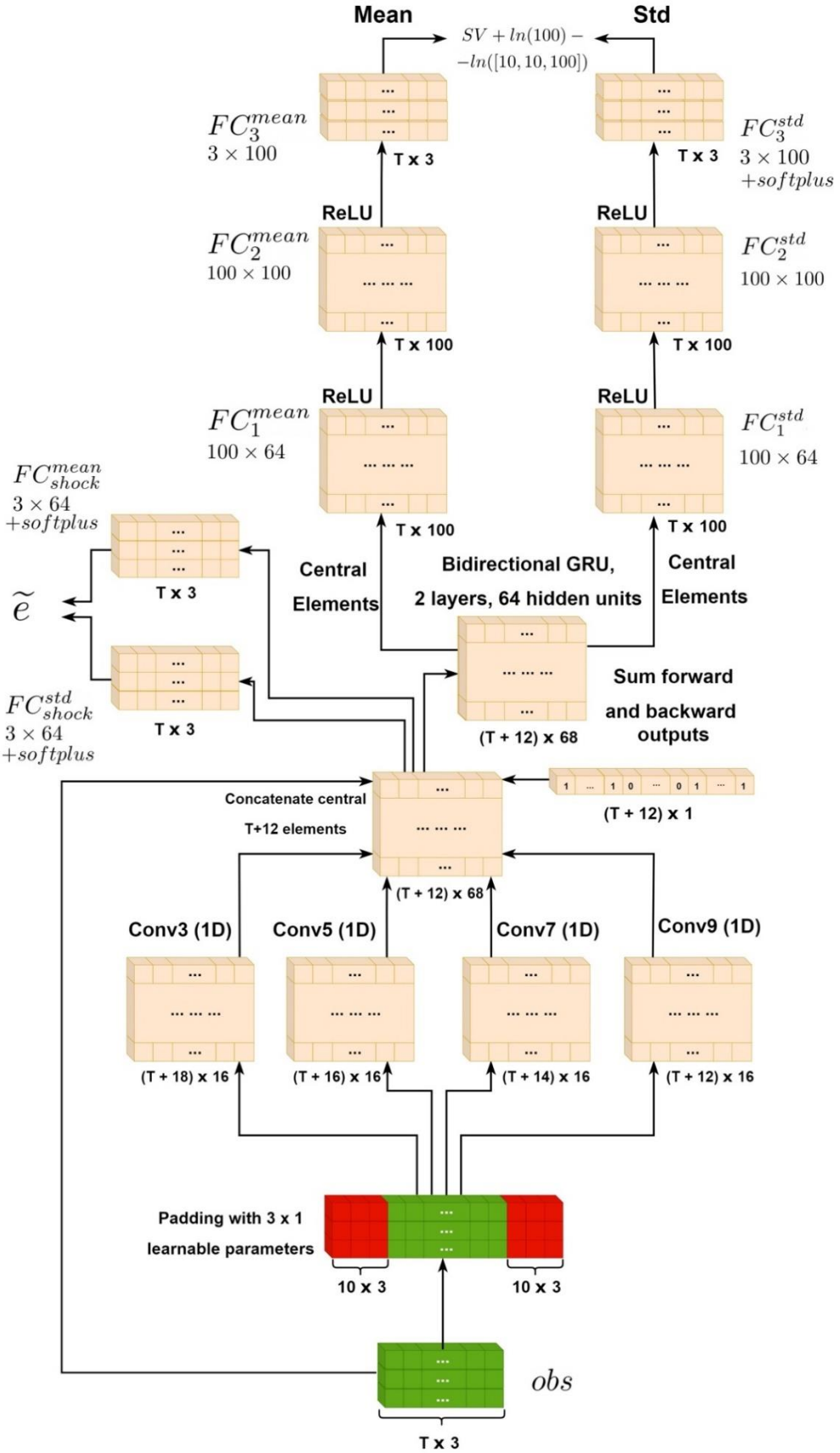
Algorithm C1 is compared with an adaptive MCMC algorithm similar to that proposed by Justiniano and Primiceri (2008) and Diebold, Schorfheide and Shin (2017). The key difference is the replacement of Gibbs sampling step for sampling parameters of stochastic volatilities by Random Walk Metropolis-Hastings step (with marginalized states).

Algorithm C2. Adaptive MCMC algorithm for stochastic volatility DSGE model **($N_{sim} = 100\,000, c = 1.5, \Sigma_0^{\theta} = 0.1I, \Sigma_0^{\theta^{SV}} = 0.1I$)**

For $n = 1, \dots, N_{sim}$:

1. Draw parameters θ using Random Walk Metropolis-Hastings algorithm with adaptive proposal:

Figure C1. Neural network architecture for calculating mean and standard deviation in the SV-DSGE model



$$q(\theta'|\theta_{n-1}) = 0.95N\left(\theta^{n-1}, \frac{c^2}{15}\Sigma_{n-1}^{\theta}\right) + 0.05N\left(\theta^{n-1}, \frac{0.01^2}{15}I\right)$$

and acceptance rate:

$$ar_n = \min\left(\frac{p(obs_1, \dots, obs_T|SV_1^{n-1}, \dots, SV_T^{n-1}, \theta')p(\theta')}{p(obs_1, \dots, obs_T|SV_1^{n-1}, \dots, SV_T^{n-1}, \theta^{n-1})p(\theta^{n-1})}, 1\right)$$

2. Draw errors $e_1, \dots, e_T$:

$$e_1^n, \dots, e_T^n \sim p(e_1, \dots, e_T|obs_1, \dots, obs_T, SV_1^{n-1}, \dots, SV_T^{n-1}, \theta^n)$$

3. For $t = 1, \dots, T$ draw discrete approximation to chi-squared distribution:

$$z_t^{i,n} \sim p\left(z_t^i|SV_1^{n-1}, \dots, SV_T^{n-1}, e_1^n, \dots, e_T^n\right), \quad i \in \{g, z, R\}$$

4. Draw parameters $\theta^{SV} = \left\{\sigma_g^{SV}, \sigma_z^{SV}, \sigma_R^{SV}, \rho_g^{SV}, \rho_z^{SV}, \rho_R^{SV}\right\}$ using Random Walk Metropolis-Hastings algorithm with adaptive proposal:

$$q(\theta^{SV\prime}|\theta_{n-1}^{SV}) = 0.95N\left(\theta^{SV,n-1}, \frac{c^2}{6}\Sigma_{n-1}^{\theta^{SV}}\right) + 0.05N\left(\theta^{SV,n-1}, \frac{0.01^2}{6}I\right)$$

and acceptance rate:

$$ar_n = \min\left(\frac{p(e_1^n, \dots, e_T^n|z_1^n, \dots, z_T^n, \theta^{SV\prime})p(\theta^{SV\prime})}{p(e_1^n, \dots, e_T^n|z_1^n, \dots, z_T^n, \theta^{SV,n-1})p(\theta^{SV,n-1})}, 1\right)$$

5. Draw stochastic volatility:

$$SV_1^n, \dots, SV_T^n \sim p(SV_1, \dots, SV_T|z_1^n, \dots, z_T^n, e_1, \dots, e_T, \theta^{SV,n})$$

Appendix D. Seasonal adjustment with structural breaks in seasonality

The data generation process is not directly specified here, unlike in previous models. Instead, we describe the data generation procedure:

Algorithm D1. Data generator with breaks in seasonality ($T_{lb} = 40$, $T_{ub} = 80$, $T^{period} = 4, B = 100$**)**

1. Draw T_n from uniform discrete distribution $T \sim U(T_{lb}, T_{ub})$.
2. Simulate batch components:

   For $b = 1, \dots, B$:

   2.a. Generate non-seasonal component NS^b:

$$e_t^b \sim Student(0, 1, 3 + |\eta_t|), \qquad t = -199, \dots T_n$$

$$\eta_t \sim N(0,3), \qquad t = -199, \dots T_n$$

$$e_t^{shift,b} \sim N(0,20)Bernouli(0.01), \qquad t = -196, \dots T_n$$

$$\rho_{AR,1}^b, \rho_{AR,2}^b, \rho_{AR,3}^b, \rho_{MA,1}^b, \rho_{MA,2}^b, \rho_{MA,3}^b \sim Bernouli(0.5)U[-0.5,0.98]$$

$$\varepsilon_t = e_t^b + \left(\rho_{MA,1}^b + \rho_{MA,2}^b + \rho_{MA,3}^b\right)e_{t-1}^b + \left(\rho_{MA,1}^b\rho_{MA,2}^b + \rho_{MA,2}^b\rho_{MA,3}^b + \rho_{MA,1}^b\rho_{MA,3}^b\right)e_{t-2}^b + \rho_{MA,1}^b\rho_{MA,2}^b\rho_{MA,3}^b e_{t-3}^b + e_t^{shift,b}, \qquad t = -196, \dots T_n$$

$$x_t^b = \left(\rho_{AR,1}^b + \rho_{AR,2}^b + \rho_{AR,3}^b\right)x_{t-1}^b - \left(\rho_{AR,1}^b\rho_{AR,2}^b + \rho_{AR,2}^b\rho_{AR,3}^b + \rho_{AR,1}^b\rho_{AR,3}^b\right)x_{t-2}^b + \rho_{AR,1}^b\rho_{AR,2}^b\rho_{AR,3}^b x_{t-3}^b + \varepsilon_t, \qquad t = -196, \dots T_n$$

$$x_{-199}^b, x_{-198}^b, x_{-197}^b = 0$$

$$c^b \sim N(0,0.005), scale^b \sim N\left(0, \frac{0.007}{std(x^b)}\right)$$

$$NS^{*b} = \left\{c^b + scale^b x_{-199}, \dots, c^b + scale^b x_{T_n}\right\}$$

$$I^{integrated,b} \sim Bernouli(0.5)$$

$$NS^b = I^{integrated,b}cumsum(NS^{*b}) + \left(1 - I^{integrated,b}\right)NS^{*b}$$

2.b. Generate seasonal component S^b:

$$\sigma^b \sim N\left(0, \frac{0.2}{\sqrt{40}}\right)$$

$$I_t^{shift,b} \sim Bernouli(0.01), \qquad t = -199, \dots T_n$$

$$z_t^b = I_{t-3}^{shift,b} I_{t-2}^{shift,b} I_{t-1}^{shift,b} I_t^{shift,b}, \qquad t = -196, \dots T_n$$

$$e_t^{S,b} \sim N(0,1), \qquad t = -196, \dots T_n$$

$$s_{-199}^b, s_{-198}^b, s_{-197}^b \sim N(0,1)$$

$$s_t^b = -\left(1 - z_t^b\right)\left(s_{t-1}^b + s_{t-2}^b + s_{t-3}^b + \sigma^b e_t^{S,b}\right) + z_t^b e_t^{S,b}$$

$$scale^{S,b} \sim N\left(0, 3std(NS^{*b})\right)$$

$$S^b = \left\{scale^{S,b} s_{-199}^b, \dots, scale^{S,b} s_{T_n}^b\right\}$$

3. Create batch of size $2B$:

   For $b = 1, \dots, B$:

$$y^{*b} = S^b + NS^b$$

$$y^b = \left\{\frac{y_1^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{y_T^b - mean(y^{*b})}{std(y^{*b})}\right\}$$

$$y^{B+b} = \left\{\frac{y_T^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{y_1^b - mean(y^{*b})}{std(y^{*b})}\right\}$$

$$sa^b = \left\{\frac{NS_1^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{NS_T^b - mean(y^{*b})}{std(y^{*b})}\right\}$$

$$sa^{B+b} = \left\{\frac{NS_T^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{NS_1^b - mean(y^{*b})}{std(y^{*b})}\right\}$$

$$y = \{y^1, \dots, y^{2B}\}, \qquad sa = \{sa^1, \dots, sa^{2B}\}$$

The neural network estimation algorithm approximating the mean and standard deviation of the posterior distribution is similar to those described for other models.

Algorithm D2. Pretraining seasonal adjustment with structural breaks in seasonality ($N_{sim} = 100\ 000$**)**

For $n = 1, \dots, N_{sim}$:

1. Simulate n^{th} batch using Algorithm D1:

$$sa_n^{batch} = \{\{sa^1, y^1\}, \dots, \{sa^{2B}, y^{2B}\}\}$$

2. Compute per state loss using architecture illustrated in Figure D1:

$$L_n = -\frac{1}{2BT_n}\sum_{b=1}^{2B}\sum_{t=1}^{T_n} log\, p\left(sa_t^b | m_t(y^b, \varphi), \sigma_t(y^b, \varphi)\right)$$

3. Make an optimization step with respect to φ using the ADAM algorithm.

The schedule for the ADAM algorithm is defined as:

$$\varepsilon_n = \begin{cases} 10^{-3}, if\ n < 1.5 \times 10^4 \\ 10^{-4}, if\ 1.5 \times 10^4 \le n < 5 \times 10^4 \\ 10^{-5}, if\ n \ge 5 \times 10^4 \end{cases}$$

Figure D1. Neural network architecture for calculating mean and standard deviation in the SA model