Differentiable State Space Models and Hamiltonian Monte Carlo Estimation [*]

David Childers
CMU

Jesús Fernández-Villaverde
Penn

Jesse Perla
UBC

Cameron Pfiffer
Oregon

Christopher Rackauckas
MIT & Maryland

Peifan Wu
UBC

December 9, 2021

Latest Version
Online Appendix

Abstract

We propose a methodology to take dynamic stochastic general equilibrium (DSGE) models to the data based on the combination of differentiable state space models and the Hamiltonian Monte Carlo (HMC) sampler. First, we introduce a method for differentiating perturbation solutions of DSGE models with respect to the model's parameters. The resulting output can be used for various computational tasks requiring gradients, such as building an HMC sampler to estimate first- and second-order approximations of DSGE models. The availability of derivatives also enables a general filter-free method to estimate nonlinear, non-Gaussian DSGE models by sampling the joint likelihood of parameters and latent states. We show that the gradient-based joint likelihood sampling approach is superior in efficiency and robustness to standard Metropolis-Hastings samplers by estimating a canonical real business cycle model and a medium-scale New Keynesian DSGE model.

[*] All errors are our own. Peifan thanks UBC ARC Sockeye and Compute Canada Westgrid.

1 Introduction

In this paper, we propose a methodology to take dynamic stochastic general equilibrium (DSGE) models (and related dynamic equilibrium models in other fields) to the data based on the combination of differentiable state space models and the Hamiltonian Monte Carlo (HMC) sampler. Differentiable state space models allow us to implement the HMC sampler by providing an easy method to differentiate perturbation solutions of DSGE models. The HMC has two great advantages with respect to other Markov Chain Monte Carlos (McMc). First, it draws much more efficiently from the posterior of a DSGE model than existing alternatives. Second, the HMC scales very well. Hence, we can draw from the joint distribution of parameters and latent states of the model simultaneously without having to resort to a filter to marginalize out the latent state variables of the model.

Let us unpack the many ideas in the previous paragraph. DSGE models are one of the major workhorses of modern macroeconomics. Thus, it is not a surprise that an extensive strand of literature has focused on how to take these models to the data, both from a classical or a Bayesian perspective (see the reviews in Fernández-Villaverde et al., 2016, and Fernández-Villaverde and Guerrón-Quintana, 2021).

While the Bayesian approach is particularly popular, following it, however, is not without peril. DSGE models rarely have a closed-form solution. Hence, we cannot write their moments or likelihood functions. Instead, we need to resort to numerical approximations to evaluate these empirical functions and sample from them, for instance, to find posterior distributions of parameters of interest.

Despite many years of research, some questions remain open. First, implementing the Bayesian approach usually requires an McMc sampler, the most popular of which is the Random Walk Metropolis-Hastings (RWMH) algorithm. It is well known that the RWMH algorithm (and even many of the more sophisticated improvements built on top of it) suffer from high autocorrelation across samples. That is, the draws 1,245 and 4,598 (or any other two!) of a typical simulation are much more correlated than we would like. Due to this high correlation, the effective sample size is small comparing the total samples drawn: even if we run the sampler 100,000 times, we have only the equivalent of around 1,000 samples coming from a hypothetical pure Monte Carlo (which, unfortunately, we cannot design). Furthermore, the sampling results might be sensitive to the choice of starting points, even after long runs of the sampler. These drawbacks are particularly binding when we deal with DSGE models that are richly parameterized.

Second, evaluating the likelihood function implied by the numerical solution of a DSGE model is usually done by some filter, such as the Kalman filter –when we have a first-order perturbation and the shocks that drive the dynamics of the economy are Gaussian– or the particle filter –when we deal with non-linear solutions and/or non-

Gaussian shocks (Fernández-Villaverde and Rubio-Ramírez, 2007). The filters deliver the marginal likelihood of the model with respect to its parameters by integrating over the distribution of latent states. Unfortunately, these filters can either be restrictive in their requirements (e.g., the Kalman filter) or computationally costly, non-differentiable, and difficult to tune up (e.g., the particle filter).

We tackle these two questions by implementing two complementary methods. Our first method is to apply the HMC sampler to the estimation of DSGE models. The HMC is a gradient-based sampling method that traverses the posterior efficiently and works particularly well in high-dimensional cases. While the HMC is an attractive alternative to the RWMH algorithm, its use in macroeconomics had been blocked by the need to compute the gradients of the posterior of the DSGE model, a cumbersome task.

We get around this problem by showing how to differentiate both first- and second-order perturbation solutions of DSGE models with respect to the parameters (although the core of our argument, built around the implicit function theorem, is applicable to any higher-order perturbation solutions). Essentially, we provide a local sensitivity analysis similar to Iskrev (2010), but we extend the results to second-order perturbation solutions. For example, consider a canonical medium-scale Keynesian DSGE such as Fernández-Villaverde and Guerrón-Quintana (2021) which has 14 state variables, 24 controls, and 28 parameters. The first-order perturbation solution to this is a matrix of 14×14 values for the evolution of the state, and one with 24×14 for controls. Our procedure provides the gradient of these two matrices with respect to the 28 deep parameters of the underlying model. Those gradients could be used for all sorts of purposes, such as examining how impulse response functions (IRFs) change with parameters, better calculating the loss function with simulated methods of moments, or –in our main application– helping calculate gradients of the likelihood for Bayesian samplers.[1]

Our second method is to sample both parameter and latent variables simultaneously, instead of sampling the posterior of the model parameter by marginalizing out the latent variables with a filter. In this way, we do not spend time filtering or being forced into some distributional assumptions. The idea of sampling parameters and latent states simultaneously has been around for decades (see Kim et al. 1998, for an early incarnation of this approach), but its implementation was difficult because, as soon as we have more than a few observations, we are dealing with a high-dimensional inference problem that the RWMH algorithm cannot handle even with extremely long simulations. In comparison, the scalability of the HMC means that the route of the joint likelihood becomes

[1]We specify the state space model in a discrete-time setting, and the linear equation system will be Sylvester equations. In a continuous-time setting that is widely used in HANK models, the equations have a similar form but will be Lyapunov equations. Both Sylvester equations and Lyapunov equations are linear and can be solved with standard libraries like SLICOT.

feasible. More in general, our methods mean that we can bring much larger models to the data, as long as we can find the required derivatives. Gradient-based approaches such as HMC are limited by difficult geometry and the cost of gradient calculations, not by the dimensionality of the problem.

We illustrate the two ideas above by estimating two models: a canonical real business cycle (RBC) model and the medium-scale New Keynesian model in Fernández-Villaverde and Guerrón-Quintana (2021). In our first numerical experiment, we find a first-order approximation of the RBC model and evaluate the associated (parameter) likelihood using the Kalman filter. Then, we implement the RWMH and HMC samplers. Even if we only need to sample 3 parameters, the fraction of effective draws for the RWMH is around 0.6%. In comparison, the HMC sampler gets 28% of effective draws, nearly 47 times more. This experiment shows how much more efficient the HMC sampler is.

In our second numerical experiment, we solve the RBC model using a second-order perturbation. Then, we estimate this case with the HMC sampler and the joint likelihood function of parameters and latent states. Since there are 3 parameters and 200 latent states, we sample along 203 dimensions overall. The HMC delivers a proportion of effective samples of around 4%. In comparison, the RWMH sampler on the marginal likelihood of the parameters evaluated using particle filter, a much lower dimensionality problem, can deliver only an effective sample of only 0.3%. Furthermore, the HMC sampler is more robust to the starting point of the Markov chain. After 12 minutes of execution, more than 95% of the cumulative mean of the parameters are close to the pseudo-true values in our joint likelihood approach, while the RWMH counterpart is less than 10%.

In our third and fourth numerical experiments, we repeat the first and second numerical experiments but now with the New Keynesian model instead of the RBC model. In this case, we have 22 parameters to estimate and 240 latent states. As in our first two experiments, the HMC sampler is much superior to the RWMH. For instance, while the HMC yields good McMc mixing, the RWMH with the particle filter fails to escape from the neighborhood of the starting point.

We implement all the methods in this paper through open-source modular building blocks that we hope will allow researchers to develop their own applications. First, the package `DifferentiableStateSpaceModels.jl` provides a symbolic domain-specific language for defining nonlinear models embedded in `Julia`, and to make them easy to embed in the `Julia` auto-differentiation ecosystem through `ChainRules.jl`. Second, the SciML ecosystem (through the package `DifferenceEquations.jl`) provides tools to enable differentiable simulations, Kalman filters, and joint likelihoods, These packages can expand the researchers' imagination for what is a feasible scale in the computation and estimation of DSGE models thanks to our thorough use of differentiable programming.

Differentiable Programming At the core of our paper, we have the computation of gradients of state space models. "Differentiable programming" –a term encompassing classic auto-differentiation (AD)– is an old programming paradigm, with origins going back to the late 1950s. However, the field has exploded with the recent popularity of machine learning, where it plays a key role. Classic examples of differentiable programming include calculating Jacobians for use in solving systems of nonlinear differential equations (Griewank and Walther, 2008), and the training of neural networks using first-order optimization methods (Baydin et al., 2017). Differentiable programming/AD ecosystems exist in `Julia`, `Python`, `Stan`, and other modern programming languages.

The basic idea of differentiable programming is simple and has nothing to do with numerical approximations (as the ones used in numerical derivatives). First, it is easy to code a program that substitutes analytic derivatives where appropriate –i.e., if it encounters `sin`, it should use `cos` as the derivative– and applies the chain rule when required. Second, if we think about a computer program as a sequence of mathematical functions calling each other, then a library that can analyze that program can also differentiate arbitrary functions by recursively applying the chain rule until it hits primitive gradients. Third, for the two previous steps to work in practice, we need a large library of primitive derivatives and gradients for the AD package to substitute.

The previous steps are easy to visualize when the functions to manipulate correspond to standard mathematical functions (e.g., the `log` function). This is highly valuable in itself: letting AD packages execute the chain rule is much more convenient and less error-prone than analytically working through the algebra by hand. But the surprising result is that software code one would not expect to have gradients (e.g., loops, accessing a subset of a vector, solving an entire linear system, constructors of parametric structs, Kronecker products) can be formalized with primitive gradients with respect to continuous arguments. This latter feature is what radically changes the scale of problems one can solve.

To see this, it is important to understand the difference between forward and reverse-mode AD. If one wants to calculate some function $f : \mathbb{R}^N \to \mathbb{R}$ composed of nested calls to other functions and find its gradients $\nabla f(\cdot) \in \mathbb{R}^N$ at a particular point $x^0 \in \mathbb{R}^N$, we can evaluate the chain rule forwards or backwards.[2] Intuitively, reverse-mode AD takes the x^0 and applies the functions nested inside of f until it eventually has calculated $f(x^0)$. Then, it perturbs the resulting $f(x^0)$ and applies the chain rule backward until it has calculated $\nabla f(x^0)$.[3]

[2]In the ML community, they often refer to reverse-mode AD as "back-propagation", whereas, in differential equations and control theory, it is referred to as "adjoint sensitivity".

[3]Forward-mode AD applies the chain rule forward starting from x^0 and calculates $f(x^0)$ and $\nabla f(x^0)$ at the end. forward-mode AD is equivalent to using "numerical derivatives" in its computational order, even if it is always more accurate. This is particularly attractive to compute $f : \mathbb{R} \to \mathbb{R}^N$ (where reverse-mode AD is at its worst) and to calculate sparse Jacobians and Hessians of $f : \mathbb{R}^N \to \mathbb{R}$ functions.

The critical insight that makes a huge number of applications feasible is that if one has a low dimensional output of a function, then the computational order of reverse-mode mode AD is independent of the dimensionality of the input. In particular, if one is dealing with a univariate output (e.g., a likelihood function, a moment function, a loss function), the computation of the gradient requires the same number of primitive operations for any input size. That is, the computation of the gradient uses the same number of primitive operations whether the likelihood function depends on 10 or 1,000 variables. This is why the combination of reverse-mode AD with convenient programming language support (and better hardware in GPUs, which are ideal for parallelizing many gradient calculations) has made the deep learning revolution possible.

There are many ways to implement reverse-mode AD, but the most common approach is a package for a programming language that analyzes or traces the function during execution.[4] In the case of `Python`, this functionality is the core feature of `Pytorch` and `Tensorflow`, and in `Julia`, there are many alternatives such as `Zygote.jl` (see Innes et al., 2019). The package manages the application of the chain rule, compiling both the function calculation and its gradients, and accessing libraries of primitive gradients. These primitive gradients themselves are black boxes to the AD package. For example, the AD package might have access to the primitive that the derivative of `sin` is `cos` and then the AD system implements this substitution whenever it encounters that function.[5]

Hence, differentiable programming is limited by the primitive gradients it has available and their computational complexity. Deep learning is especially amenable to differentiable programming because it uses functions with an enormous number of parameters and many levels of nesting, but where the primitive functions themselves are relatively simple (e.g., affine transformations, hyperbolic tangents, etc.). The contribution of our paper is to implement the same ideas in the perturbation solution of DSGE models, since the underlying functions we need to evaluate may be much more complicated. Even if gradients are available in principle (e.g., the generalized Schur decomposition used in DSGE solvers is composed of more primitive operations), some work is required to make them operate in practice. We show how, if primitive gradients can be written down for a much higher-level function, then we can register them with packages such as `ChainRules.jl`

[4]Some alternatives are to have that tracing/static analysis directly built into language support –such as the `Stan` language used for Bayesian estimation or to use a utility that analyzes the source separately and exports the derivatives as code, as it was common with `Fortran`, `C`, or `Matlab` source-code transformations used in scientific computing.

[5]All AD systems have a way to manage these sorts of primitive rules, but very few of them are flexible and hackable. In most cases, they need to be written in `C++` and directly built into the AD package itself. The exception to this is `Julia`, which is able to have user-configurable and expandable rules since the AD systems are often written in `Julia` themselves. See White et al. (2021) for a description of `ChainRules.jl`, which provides a package agnostic way to register primitive gradients. Also, see Schäfer et al. (2021) for how this can be utilized to have source code implemented in an AD agnostic way since there are tradeoffs that may lead to different choices among AD packages.

and seamlessly differentiate any program that uses an embedded DSGE solver in its calculation.

Literature Review The closest paper to this project is a contemporaneous work by Farkas and Tatar (2020) where the authors propose estimating linear Gaussian DSGE models with HMC methods on the marginal likelihood of parameters, using Smets and Wouters (2007) as an example. Our work nests such an application as a special case. More importantly, Farkas and Tatar (2020) use a non-customizable AD and the differentiation has to go step-by-step in their whole implementation, which means, among other things, that they cannot use the Schur decomposition method that is dominant to perturb DSGE models due to stability and speed. Instead, we offer a much more general framework to implement the HMC in perturbation solutions of DSGE models by passing the required derivatives to the downstream sampling with customized AD.

Outline The rest of this paper is organized as follows. Section 2 presents the state space representation of DSGE models and introduces the joint likelihood approach associated with this mathematical representation. Section 3 describes the HMC. Section 4 shows the gradients of solutions to DSGE models. Section 5 briefly discusses our implementation. Section 6 illustrates the estimation results on two models: a canonical RBC model and a medium-scale New Keynesian model. Section 7 concludes.

2 Likelihood Function and Evaluation

This section introduces the notation for the state space representations of a DSGE model and defines the likelihood terms we will use later in the estimation stage. In addition, we provide an AR(1) example to illustrate the difference between joint likelihood and marginal likelihood and the sampling processes for both of them.

2.1 The State space Model

Following Schmitt-Grohé and Uribe (2004), x_t is a vector of state variables that describes the economic environment, and y_t is a vector of control variables. The states are buffeted by shocks ϵ_t. Without loss of generality, we assume these shocks are independent and identically distributed. Moreover, we denote θ as the parameters to estimate. These parameters determine the preferences, technology, information sets, or monetary or fiscal policy rules for the economy. We assume that θ is time-invariant, but we can allow correlated or time-varying shocks and parameters with extra notations. See, for example, Fernández-Villaverde and Rubio-Ramírez (2007).

The first component of the state space representation of a DSGE model is a transition equation $h(\cdot)$ that characterizes the law of motion of the states:

$$x_{t+1} = h\left(x_t, \epsilon_t; \theta\right). \tag{1}$$

Associated with the transition equation, there is a policy function g that links latent state variables with the control variables:

$$y_t = g\left(x_t; \theta\right). \tag{2}$$

The second component of the state space representation is a measurement equation $q(\cdot)$ that links states and controls with observations z_t:

$$z_t = q\left(x_t, y_t; \theta\right) = q\left(x_t, g\left(x_t; \theta\right); \theta\right). \tag{3}$$

We denote $z^T = \{z_t\}_{t=1,\ldots,T}$ as the observation sequence.

2.2 Marginal vs. Joint Likelihood

We review now the concepts of the marginal and joint likelihood function associated with the previous state space representation.

Marginal Likelihood The log posterior density of θ conditional on data z^T can be decomposed as:

$$\underbrace{\ln p\left(\theta | z^T\right)}_{\text{log-posterior}} = \ln p\left(\theta, z^T\right) + C \tag{4}$$

$$= \underbrace{\ln p\left(\theta\right)}_{\text{log-prior}} + \underbrace{\ln p\left(z^T | \theta\right)}_{\text{log-likelihood}} + C \tag{5}$$

$$= \ln p\left(\theta\right) + \sum_{t=1}^{T} \ln p\left(z_t | z^{t-1}, \theta\right) + C. \tag{6}$$

Equation (5) tells us that the log posterior density is (up to a scale) the sum of log prior and the conditional density of the data. Furthermore, the second term of equation (5) is the sum of the conditional density across all T periods as shown in (6).

The sequence of $\ln p\left(z_t | z^{t-1}, \theta\right)$ can be recursively constructed through the Chapman-

Kolmogorov equation:

$$p\left(z_t|z^{t-1},\theta\right) = \int p\left(z_t, x_t|z^{t-1},\theta\right) \mathrm{d}x_t$$
$$= \int p\left(z_t|x_t,\theta\right) p\left(x_t|z^{t-1},\theta\right) \mathrm{d}x_t, \tag{7}$$

and the Bayes theorem:

$$p\left(x_t|z^t,\theta\right) = \frac{p\left(z_t|x_t,\theta\right) p\left(x_t|z^{t-1},\theta\right)}{p\left(z_t|z^{t-1},\theta\right)}, \tag{8}$$

where

$$p\left(x_t|z^{t-1},\theta\right) = \int p\left(x_t|x_{t-1},\theta\right) p\left(x_{t-1}|z^{t-1},\theta\right) \mathrm{d}x_{t-1}. \tag{9}$$

The conditional density (7) includes two terms inside the right-hand side of the integral. The first term gives us the conditional density of contemporaneous data given the latent states in the period. The second term shows how, given historical data, we can forecast the latent states in this period following (8) and (9).

In plain words: the marginal likelihood infers the states at period t from historical data. Because of the Markov structure of the state space models, the inference of states at period t only depends on the information at period $t-1$ and the observed data at period t. This is the reason we can call it a "marginal likelihood": the conditional density is a function only of parameters θ, marginalizing out all the latent states x. While equations (8) and (9) are simple in principle, they are daunting to implement in practice. In the case that the law of motion of the states $h(\cdot)$ is linear in x and ϵ, and ϵ follows a Gaussian distribution, we can filter the latent states with Kalman filter. In non-linear or non-Gaussian cases, we have to seek some approximated or particle filters.

Joint Likelihood We now focus on the joint likelihood of the data z^T and the latent states $x^T = \{x_t\}_{t=1,\ldots,T}$, i.e., $\ln p\left(\theta, x^T|z^T\right)$.

As before, we start decomposing this likelihood as:

$$\begin{aligned}
\underbrace{\ln p\left(\theta, x^T | z^T\right)}_{\text{log-posterior}} &= \ln p\left(\theta, x^T, z^T\right) + C && (10)\\
&= \underbrace{\ln p\left(\theta, x^T\right)}_{\text{log-prior}} + \underbrace{\ln p\left(z^T | x^T, \theta\right)}_{\text{log-likelihood}} + C && (11)\\
&= \ln p\left(\theta\right) + \ln\left(x^T | \theta\right) + \sum_{t=1}^{T} \ln p\left(z_t | x^t, \theta\right) + C && (12)\\
&= \ln p\left(\theta\right) + \sum_{t=1}^{T} \ln p\left(z_t | x_t, \theta\right) + \sum_{t=1}^{T} \ln p\left(x_t | x_{t-1}, \theta\right) + \ln p\left(x_0 | \theta\right) + C && (13)\\
&= \ln p\left(\theta\right) + \sum_{t=1}^{T} \ln p\left(z_t | \epsilon^t, \theta\right) + \sum_{t=1}^{T} \ln p\left(\epsilon_t | \theta\right) + \ln p\left(x_0 | \theta\right) + C. && (14)
\end{aligned}$$

In this case, the log posterior is still the log prior plus the log likelihood, and we can increment the log likelihood sequentially following equation (12). Because of the Markov structure of the state space model, x_t is sufficient when we compute the likelihood, and x_t only depends on x_{t-1} given θ, as shown in equation (13).

Another way to interpret equations (10)–(13) is that we compute the log posterior taking not only the data but also the history of latent states as given, while the law of the motion of the states is pinned down by the initial state x_0, the sequence of shocks $\epsilon^T = \{\epsilon_t\}_{t=1,\dots,T}$, and the model parameters θ. Hence, the second term in (13) comes directly from the measurement function $q(\cdot)$, and the third term in (13) is just $h(\cdot)$. To bridge equations (13) and (14), we can rewrite these two terms in (13) as:

$$\begin{aligned}
\sum_{t=1}^{T} \ln p\left(z_t | x_t, \theta\right) &= \sum_{t=1}^{T} \ln p\left(z_t | \epsilon^t, \theta\right) && (15)\\
\sum_{t=1}^{T} \ln p\left(x_t | x_{t-1}, \theta\right) &= \sum_{t=1}^{T} \ln p\left(\epsilon_t | \theta\right). && (16)
\end{aligned}$$

Therefore, the joint likelihood equation can be explicitly written down given an exogenous series of shock ϵ_t. This joint likelihood computation is filter-free, as we do not have to infer the latent state series x^T or their posterior distributions at all, but rather retrieve a deterministic series of x^T from the shock series.

Consequently, instead of sampling just θ to compute a marginal likelihood $\ln p\left(\theta | z^T\right)$, we will sample both θ and ϵ^T to compute the joint likelihood $\ln p\left(\theta, x^T | z^T\right)$, which is equivalent to $\ln p\left(\theta, \epsilon^T | z^T\right)$. And the joint posterior is equal to marginal posterior when we integrate over ϵ^T,

$$p\left(\theta|z^T\right) = \int_{x^T} p\left(\theta, x^T|z^T\right) \mathrm{d}x^T = \int_{\epsilon^T} p\left(\theta, \epsilon^T|z^T\right) \mathrm{d}\epsilon^T. \tag{17}$$

2.3 An AR(1) Example with Measurement Noise

We employ a simple example to show the difference between the marginal likelihood and the joint likelihood approach. Assume that we want to estimate ρ, the persistence parameter in an AR(1) process with a measurement noise:

$$\begin{aligned} y_t &= \rho y_{t-1} + \epsilon_t \\ z_t &= y_t + u_t, \end{aligned}$$

with observations $\{z_t\}$. The shock process $\{\epsilon_t\}$ is i.i.d and following a standard Normal distribution, and the measurement noise $\{u_t\}$ is i.i.d. following a Normal distribution with variance Ω. We also denote the probability density function of standard Normal distribution as $\varphi(\cdot)$.

Marginal likelihood approach The steps to sample the marginal likelihood are:

1. Sample ρ from the prior;
2. Initiate the distribution of y_0 as $\mathcal{N}\left(0, \frac{1}{1-\rho^2}\right)$: mean $y_{0|0} = 0$ and variance $P_{0|0} = \frac{1}{1-\rho^2}$.
3. Loop over $t = 1, \ldots, T$. For each t, we apply Kalman filter to infer the posterior distribution of y_t, and accrue the likelihood L:

$$\begin{aligned} y_{t|t-1} &= \rho y_{t-1|t-1} \\ P_{t|t-1} &= \rho^2 P_{t-1|t-1} + 1 \\ y_{t|t} &= y_{t-1|t-1} + \frac{P_{t|t-1}}{P_{t|t-1} + \Omega}\left(z_t - y_{t|t-1}\right) \\ P_{t|t} &= P_{t|t-1} - \frac{P_{t|t-1}^2}{P_{t|t-1} + \Omega} \\ L &= L + \ln \varphi\left(\frac{z_t - y_{t|t-1}}{\sqrt{P_{t|t-1}}}\right). \end{aligned}$$

4. Add the log prior to the likelihood L, and sample the next ρ.

Joint likelihood approach The steps to sample the joint likelihood are:

1. Sample ρ and a vector of T elements $\{\epsilon_t\}$ from the priors.
2. Initiate $y_0 = 0$.
3. Loop over $t = 1, \dots, T$. For each t, simply compute y_t with the law of motion, and accrue the likelihood L:

$$
\begin{aligned}
y_t &= \rho y_{t-1} + \epsilon_t \\
L &= L + \ln \varphi\left(\frac{z_t - y_t}{\sqrt{\Omega}}\right).
\end{aligned}
$$

4. Add the log prior to the likelihood L, and sample the next combination of ρ and all $\{\epsilon_t\}$.

While in the end the posterior of ρ should be the same, there are some significant differences between the marginal likelihood approach and the joint likelihood approach. The marginal likelihood depends on a filter to infer the distribution of the latent states over time while only samples the parameters that govern the model; the joint likelihood approach is filter-free because the law of motion of the state variables is deterministic, at a cost that it has to draw all the parameters as well as the shocks. Hence, by using the joint likelihood, we are sampling from a high dimensional space consisting of both the parameters and the latent shocks, but we do not worry about filtering the latent states. In particular, the filtering process is a time-consuming approximation when the DSGE model is nonlinear or non-Gaussian. A typical RWMH sampling algorithm will not suffice in this case. Instead, we apply the HMC – a scalable and efficient sampler that copes with high-dimensional spaces – that we now introduce.

3 Sampling Method: the Hamiltonian Monte Carlo

This section briefly introduces the HMC, the sampling method that we utilize to estimate DSGE models (see Betancourt, 2018 for a conceptual discussion of HMC methods). To do so, we first look back at Random-Walk Metropolis-Hastings (RWMH), as RWMH is one of the most widely used MCMC methods in DSGE estimations. Given a state of a Markov chain in the parameter space θ_{i-1}, the RWMH method generates a new draw θ_i from a proposal density following a Normal distribution, and then it accepts or rejects θ_i depending on how the posterior evolves. If the posterior turns out to be higher then the proposal becomes the new state of the chain; otherwise, the state updates with a probability less than 1. In this way, the Markov chain of samples always tends to travel to areas with higher densities and reserves a possibility to escape from the local minima.

The drawback of RWMH is clear from the way it samples: it spends much time outside of the typical set – the part of the parameter space containing the relevant information to compute the expectations we care about in Bayesian analysis. To understand this statement, we need to introduce the definition of the typical set. For $\epsilon > 0$ and and I, the typical set A_ϵ^I with respect to the target posterior $p\left(\theta|z^T\right)$ is

$$A_\epsilon^I \equiv \left\{ (\theta_0, \theta_1, \ldots, \theta_I) : \left| -\frac{1}{I+1} \sum_{i=0}^{I} \ln p\left(\theta_i|z^T\right) - b\left(\theta\right) \right| \leqslant \epsilon \right\}$$

where $b\left(\theta\right) = -\int p\left(\theta_i|z^T\right) \ln p\left(\theta_i|z^T\right) \mathrm{d}\theta$ is the differential entropy of the parameters with respect to their posterior density. By a week law of large numbers, we have that $\Pr(A_\epsilon^I) > 1-\epsilon$ if I is sufficiently large. That is to say, A_ϵ^I includes "most" sequences of θ_is that are distributed according to the posterior $p\left(\theta_i|z^T\right)$.

Two properties of the typical set are crucial. First, the typical set is not the region where the posterior density is the highest; second, the typical set will be a narrower and narrower band as the number of parameters grows. These two properties indicate the reason that RWMH is not capable of sampling in high dimensional spaces, and initiating the sampling process from the mode will barely help with the sampling. The RWMH method wastes many iterations because the proposal density is blind regarding the typical set of the posterior. Therefore, most of the draws will be either far away from the typical set, or highly auto-correlated, and the typical set will not be traversed efficiently.

HMC is a solution to the problems addressed above that has gained popularity in recent years. This method improves sampling efficiency by exploiting information from the posterior's gradient and hence spends much more time in the typical set. While sampling along the gradient of the posterior would push the samples to the mode of the posterior, HMC augments the gradient information with an extra momentum force, i.e., an analogy of Hamiltonian mechanics in physics.

We augment the vector of parameters θ with an auxiliary momentum vector $\mathbf{p}$ sharing the same dimensions. Random vector $\mathbf{p}$ follows Gaussian distribution $\mathcal{N}\left(0, M\right)$. Then the Hamiltonian associated with the posterior of θ is

$$H\left(\theta, \mathbf{p}\right) = -\ln p\left(\theta\right) + \frac{1}{2}\mathbf{p}^\top M^{-1}\mathbf{p} \tag{18}$$

where $-\ln p\left(\theta\right)$, the first term of (18) is the negation of log posterior, as the analog of a "potential energy" function. The second term of (18) $\frac{1}{2}\mathbf{p}^\top M^{-1}\mathbf{p}$ acts as a "kinetic energy" term. A Markov chain that samples the combination $\{\theta, \mathbf{p}\}$ drawn from this Hamiltonian has a stationary marginal density $p\left(\theta\right)$, which provides a theoretical ground for HMC.

The way we sample from $H(\theta, \mathbf{p})$ is a Gibbs-sampling scheme proposed by Girolami and Calderhead (2011) with the following steps. First, we draw $\mathbf{p}$ from the Normal distribution specified above. Second, we run the Verlet integrator for L iterations, with step ϵ for each of these L iterations. Both ϵ and L are predetermined parameters, with ϵ controlling the step size of Hamiltonian approximation, and L determines the number of steps in each iteration. The physics analogy here is straightforward to understand. Think about a particle that moves on the manifold characterized by the log posterior of θ. Every time we get a new sample from the current position, we randomly kick the particle to an arbitrary direction. The particle moves following the Hamiltonian dynamics, and we stop the particle after $L\epsilon$ time and record the new θ. Therefore, gradient information of the posterior is necessary for HMC.

In our following sampling experiments, we use the No-U-Turn sampler (NUTS) proposed by Hoffman and Gelman (2014) which shares the same mechanisms and principles with a raw HMC method, but endogenously pick ϵ and L with sample adaptions.

4 Gradients to DSGE Solutions

This section provides the derivations of the derivatives of DSGE first- and second-order solutions with respect to the model parameters. As the downstream HMC sampler requires gradient information, we compute these values through solving equations derived from the implicit function theorem. While the first-order gradient result is documented in the literature (Iskrev, 2010), the second-order result we provide here is a novel extension.

4.1 Notations

We use the "canonical form" following Schmitt-Grohé and Uribe (2004) to characterize a variety of state space models,

$$\mathbb{E}_t \mathcal{H}\left(y', y, x', x; \theta\right) = 0, \tag{19}$$

where x are states and y are control variables, and θ are the parameters that govern the model. These notations are consistent with what we have discussed in Section 2. Since the model follows a Markov structure, we omit the time subscription and just denote x' and y' as states and control variables in the next period.

The solution to the model is of the following form,

$$y = g(x; \theta) \tag{20}$$

$$x' = h(x; \theta) + \eta \epsilon' \tag{21}$$

where functions h is the law of the motion of the states, and function g is the policy function.

We denote the deterministic steady state of the system as $\bar{x}$ and $\bar{y}$, i.e., $\bar{x}$ and $\bar{y}$ satisfies

$$\mathcal{H}\left(\bar{y}, \bar{y}, \bar{x}, \bar{x}; \theta\right) = 0 \tag{22}$$

and we approximate the solutions to g and h by perturbing around the deterministic steady states. Let $\hat{x} = x - \bar{x}$ and $\hat{y} = y - \bar{y}$ be the deviations from the steady states, and we can write the first-order and second-order solutions with the following notations.

First-order solutions:

$$\hat{y} = g_x \hat{x} \tag{23}$$
$$\hat{x}' = h_x \hat{x} + \eta \epsilon'. \tag{24}$$

Second-order solutions [6]:

$$[\hat{y}]^i = [g_x \hat{x}]^i + \frac{1}{2} \hat{x}^\top [g_{xx}]^i \hat{x} + \frac{1}{2} [g_{\sigma\sigma}]^i \tag{25}$$
$$[\hat{x}']^j = [h_x \hat{x}]^j + \frac{1}{2} \hat{x}_t^\top [h_{xx}]^j \hat{x} + \frac{1}{2} [h_{\sigma\sigma}]^j + [\eta \epsilon']^j. \tag{26}$$

We follow the DSGE solution methods proposed by Klein (2000) and Schmitt-Grohé and Uribe (2004) for first- and second-order ones. Therefore in the main text here we will skip the description of the solution algorithms. Appendix A provides a full characterization.

4.2 Solution Gradients

We derive the derivatives of the solutions with respect to the parameters θ mostly with implicit function theorem. Appendix B provides the details.

Steady States For any parameter θ, we take a total differentiation of (22) which yields

$$\mathcal{H}_x \frac{\partial \bar{x}}{\partial \theta} + \mathcal{H}_y \frac{\partial \bar{y}}{\partial \theta} + \mathcal{H}_{x'} \frac{\partial \bar{x}}{\partial \theta} + \mathcal{H}_{y'} \frac{\partial \bar{y}}{\partial \theta} + \mathcal{H}_\theta = 0,$$

where all the derivatives $\mathcal{H}$ are evaluated at steady state. This is a linear equation system when we treat $\frac{\partial \bar{x}}{\partial \theta}$ and $\frac{\partial \bar{y}}{\partial \theta}$ as unknowns.

First-order Solutions We are interested in the derivatives of the first-order solutions, g_x and h_x, with respect to the parameters θ. We denote these derivatives as $\frac{\partial g_x}{\partial \theta}$ and $\frac{\partial h_x}{\partial \theta}$. The

[6]We are using tensor notations here since g_{xx} and h_{xx} are three dimensional.

solutions g_x and h_x satisfy the following equation when evaluated at steady states,

$$\mathcal{H}_x + \mathcal{H}_y g_x + \mathcal{H}_{x'} h_x + \mathcal{H}_{y'} g_x h_x = 0, \tag{27}$$

and we differentiate (27) with respect to θ,

$$\begin{bmatrix} \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\mathcal{H}_{y}}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\mathcal{H}_{x}}{\mathrm{d}\theta} \end{bmatrix}^\top \begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix} + \begin{bmatrix} \mathcal{H}_y & \mathcal{H}_{x'} + \mathcal{H}_{y'} g_x \end{bmatrix} \begin{bmatrix} \frac{\partial g_x}{\partial \theta} \\ \frac{\partial h_x}{\partial \theta} \end{bmatrix} + \begin{bmatrix} \mathcal{H}_{y'} & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial g_x}{\partial \theta} \\ \frac{\partial h_x}{\partial \theta} \end{bmatrix} h_x = 0,$$

which is a Sylvester equation system [7] on $\frac{\partial g_x}{\partial \theta}$ and $\frac{\partial h_x}{\partial \theta}$, and we can back out these two sets of unknowns with a typical Sylvester equation solver.

Second-order Solutions The second-order solutions to the DSGE include additional four objects besides the first-order solutions: g_{xx}, h_{xx}, $g_{\sigma\sigma}$, and $h_{\sigma\sigma}$. Again, we denote these derivatives as $\frac{\partial g_{xx}}{\partial \theta}, \frac{\partial h_{xx}}{\partial \theta}, \frac{\partial g_{\sigma\sigma}}{\partial \theta}, \frac{\partial h_{\sigma\sigma}}{\partial \theta}$, respectively.

First, we derive the solutions to $\frac{\partial g_{xx}}{\partial \theta}$ and $\frac{\partial h_{xx}}{\partial \theta}$. By differentiating (27) with respect to x we get

$$\begin{bmatrix} \mathcal{H}_{y'} & 0 \end{bmatrix} \begin{bmatrix} g_{xx} \\ h_{xx} \end{bmatrix} h_x \otimes h_x + \begin{bmatrix} \mathcal{H}_y & \mathcal{H}_{y'} g_x + \mathcal{H}_{x'} \end{bmatrix} \begin{bmatrix} g_{xx} \\ h_{xx} \end{bmatrix} + C = 0 \tag{28}$$

This equation (28) is a Sylvester equation on $\begin{bmatrix} g_{xx} \\ h_{xx} \end{bmatrix}$, and we further differentiate it with respect to θ which yields another Sylvester equation on $\frac{\partial g_{xx}}{\partial \theta}$ and $\frac{\partial h_{xx}}{\partial \theta}$.

The solutions to $\frac{\partial g_{\sigma\sigma}}{\partial \theta}$ and $\frac{\partial h_{\sigma\sigma}}{\partial \theta}$ are relatively simpler. Solving (29) below yields $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$,

$$\begin{pmatrix} \mathcal{H}_{y'} + \mathcal{H}_y & \mathcal{H}_{y'} g_x + \mathcal{H}_{x'} \end{pmatrix} \begin{pmatrix} g_{\sigma\sigma} \\ h_{\sigma\sigma} \end{pmatrix} + B = 0 \tag{29}$$

and differentiating (29) with respect to the parameters θ will yield a linear equation system for $\frac{\partial g_{\sigma\sigma}}{\partial \theta}$ and $\frac{\partial h_{\sigma\sigma}}{\partial \theta}$. Matrices B and C are constants given the second-order solution to the DSGE model, and Appendix B provides a detailed construction of these two matrices.

[7] A matrix Sylvester equation has the form $AXB + CXD + E = 0$ which is a (bi-)linear equation of real matrix X. Sylvester equations are widely used in control theory, and fast solvers are available, for example SLICOT.

5 Implementation

We briefly introduce our implementation here. The whole estimation package consists of two parts. The first is a library that handles first- and second-order approximation solutions to state space models with gradient information and sensitivity analyses calculated. We implemented an open-source Julia package, `DifferentiableStateSpaceModels.jl`, that fulfilling these demands. In this package, we provide a domain-specific language interface to represent typical macro state space models with symbolic differentiation as well. The second part is utilizing a Bayesian inference package, namely `Turing.jl`, that samples the posterior with the solution and gradient information passed from upstream.

The first part, `DifferentiableStateSpaceModels.jl`, consists of several components. First of all, we develop a domain-specific language to specify state space model variables and equations using `Symbolics.jl` (see Gowda et al., 2021). Users can also specify optional equations for steady-state evaluations or initial conditions of them. We generate the symbolic derivatives for the model. [8] The `Symbolics.jl` ecosystem allows us to define a Dynare-like domain-specific language, and then the necessary functions are differentiated symbolically and eventually exported as normal Julia functions. At that point, the connection to the previous symbolic language is severed, and these behave – and have identical performance – to handcrafted functions and gradients in Julia.[9] Second, we implement the first- and second-order solutions to the approximated DSGE model. Lastly, we compute the gradients of the model solutions to the model parameters and hook them with the AD backend. In this way, the downstream functions can call `DifferentiableStateSpaceModels.jl` and retrieve the gradient information seamlessly.

Then we take the DSGE solutions as well as the gradients to the downstream – `Turing.jl` – for estimations using the HMC methods. `Turing.jl` provides a domain-specific syntax to define probabilistic models. Users only need to define the prior distributions of the parameters to draw and call the related likelihood evaluation function from `DifferentiableStateSpaceMo` and then Julia takes care of the rest. In general, `Turing.jl` supports multiple auto-differentiation backends, and we are currently using `Zygote.jl` throughout.

[8]To get those derivatives, we have several options: (1) calculate the gradients using auto-differentiation within the function - independent of any use outside of it; (2) generate symbolic derivatives for the functions directly, bypassing AD; or (3) a mixture of symbolic derivatives and auto-differentiation. We choose (2) in this case. Internally using AD itself in its entirety or in combination with symbolic derivatives, and preferable in some cases such as heterogeneous agent models where we don't want to flatten the computational graph, but is harder to implement as it requires a mixed reverse and forward approach.

[9]This is specific to Julia as MATLAB, C++ (e.g., SymEngine), and python alternatives (e.g., SymPy) execute the internal symbolic representation each time instead.

6 Main Results

In this section, we present the estimation results of two models respectively with the methodologies described in the above sections. The first set of results are from estimating a plain-vanilla real business cycle model, and we compare efficiency and robustness against traditional canonical methods such as RWMH. The second set of results comes from estimating a medium-scale, New-Keynesian DSGE model, namely Fernández-Villaverde and Guerrón-Quintana (2021), and we show the performance of our method in practice.

6.1 The Real Business Cycle Model

We introduce now the canonical real business cycle model. The model can be characterized with equation system (30), where c, k, y, z represent consumption, capital, output, and the TFP level respectively. The model includes four equations: an intertemporal Euler equation, a resource constraint, a production function, and the law of motion of an exogenous TFP process:

$$\begin{aligned} \frac{1}{c_t} - \beta \frac{\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} + (1-\delta)}{c_{t+1}} &= 0 \\ c_t + k_{t+1} - (1-\delta) k_t - y_t &= 0 \\ y_t - e^{z_t} k_t^{\alpha} &= 0 \\ z_{t+1} - \rho z_t - \sigma \epsilon_{t+1} &= 0 \end{aligned} \tag{30}$$

For the experiments below we estimate three parameters that govern this model: α, β, and ρ. As β is close to 1, we sample $100(1/\beta - 1)$ instead. We fix the other parameters to be constant: $\delta = 0.025$ and $\sigma = 0.1$. Then we generate two sets of artificial data including the consumption c_t and investment i_t observations for the first- and the second-order estimations respectively. Both of the time series are of 200 time periods, governed by the first- and second-order approximation solutions at pseudo-true values $\alpha = 0.3$, $\beta = 0.998$, and $\rho = 0.9$. [10] The innovations of the TFP shock process, $\{\epsilon_t\}$, follow i.i.d standard Normal distributions.

Table 1: RWMH with Marginal Likelihood, RBC Model, First-order

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3024	0.0077	603.65	1.0022	0.006	2.1791	277
$100\left(\beta^{-1}-1\right)$	0.2	0.2288	0.0501	366.89	1.0035	0.003	1.3245	277
ρ	0.9	0.8956	0.0073	6328.6	1.0001	0.063	22.847	277

[10] These pseudo-true parameter values are widely used in quarterly calibrated models.

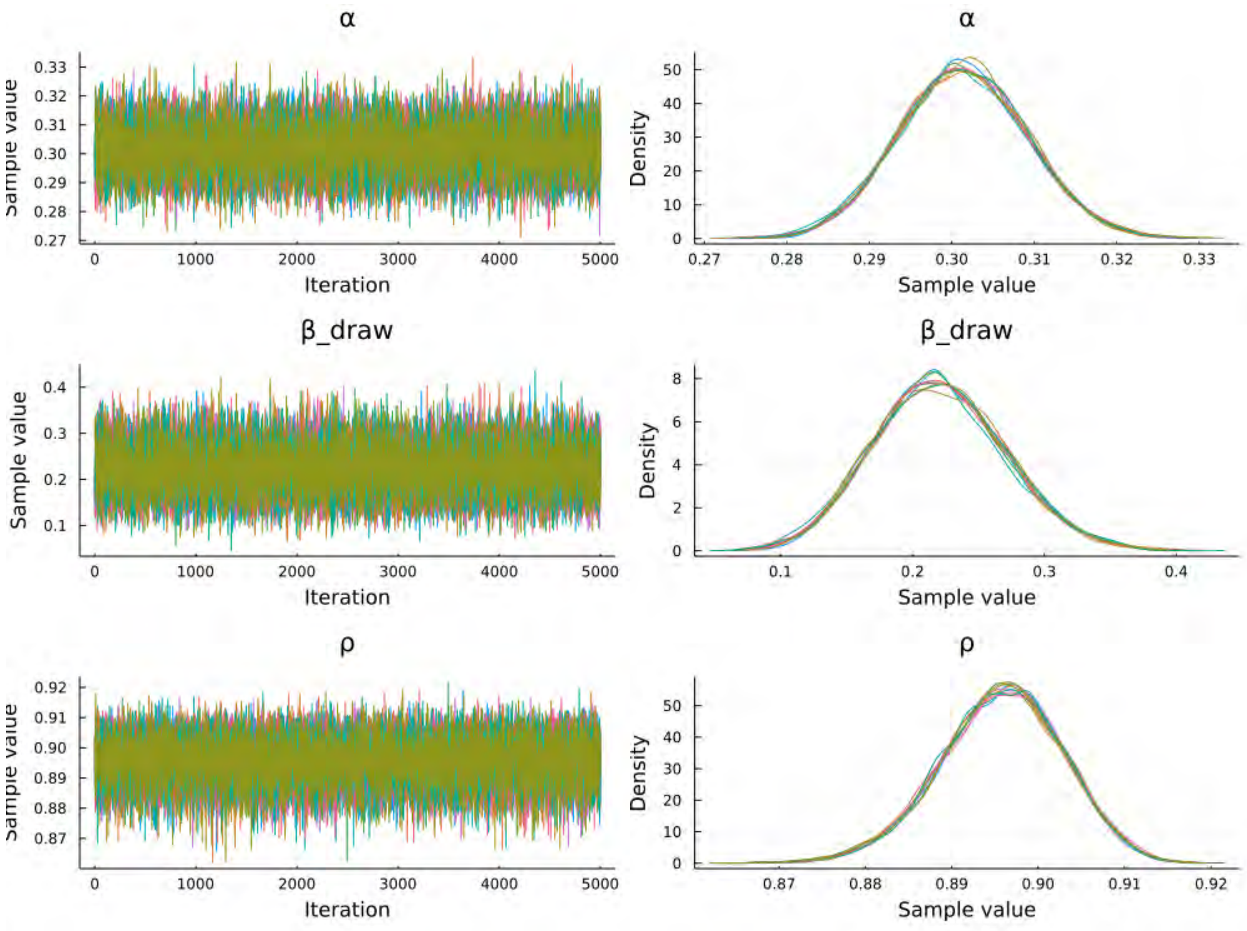


Figure 1: NUTS with Marginal Likelihood, RBC Model, First-order

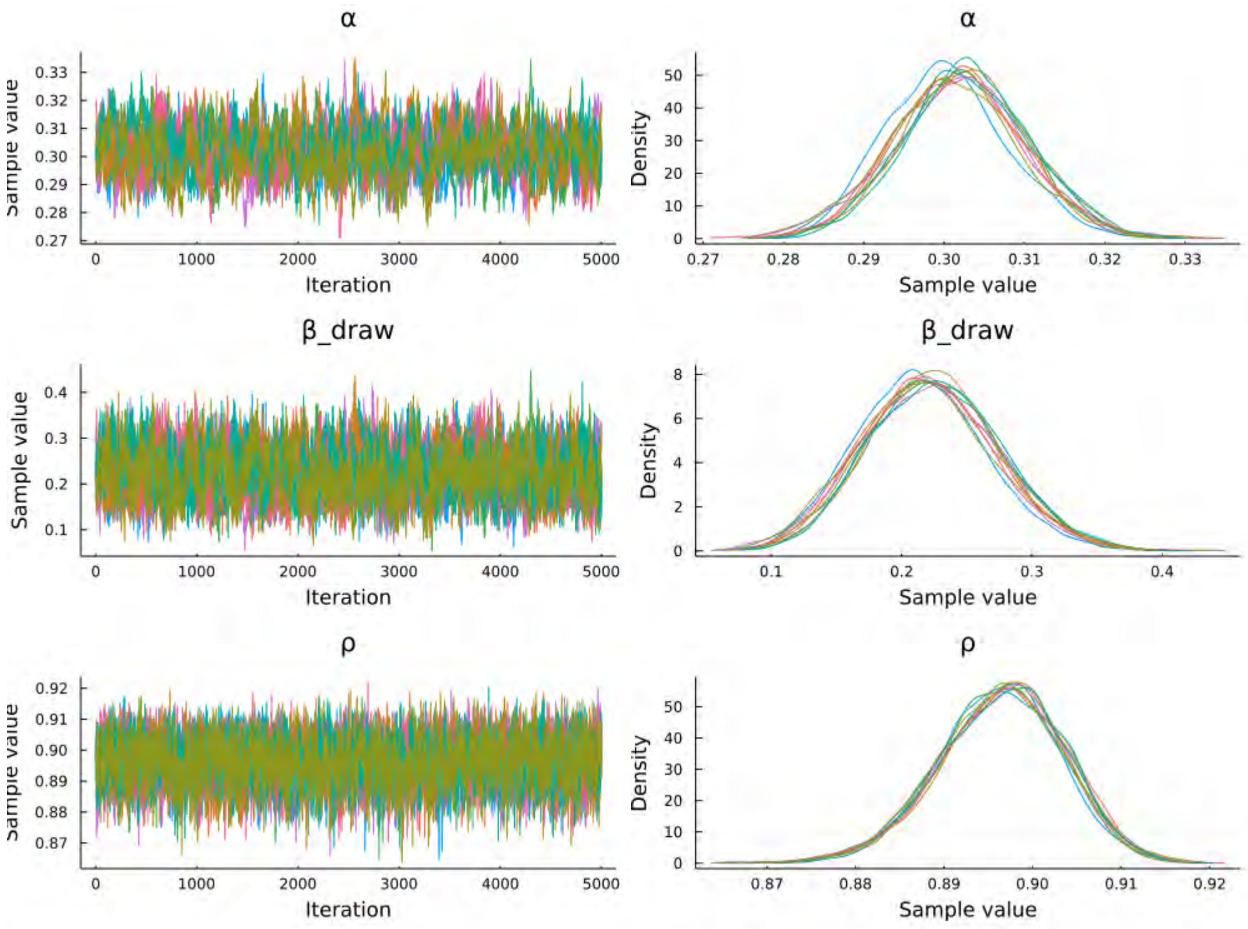


Figure 2: NUTS with Joint Likelihood, RBC Model, First-order

Table 2: NUTS with Marginal Likelihood, RBC Model, First-order

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3012	0.0079	412.09	1.0132	0.281	1.0582	389
$100\,(\beta^{-1}-1)$	0.2	0.2196	0.0523	336.05	1.0129	0.183	0.8630	389
ρ	0.9	0.8956	0.0070	864.34	0.9990	0.630	2.2196	389

Table 3: NUTS with Joint Likelihood, RBC Model, First-order

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3037	0.0091	14.61	1.0280	0.012	0.0124	1180
$100\,(\beta^{-1}-1)$	0.2	0.2344	0.0551	46.07	1.0225	0.022	0.0390	1180
ρ	0.9	0.8958	0.0075	147.60	1.0018	0.084	0.1251	1180

We examine the estimation results of first-order RBC model to begin with as Table 1 to Table 4 show. First of all, we sample the Kalman filter derived marginal likelihood on the parameters but compare the results from gradient-free RWMH sampler to the counterpart from gradient-based NUTS as in Table 1 and Table 2. With much fewer sample draws NUTS traverses the posterior more efficiently compared to RWMH, and the fraction of effective sample size (ESS%) is several magnitudes higher for NUTS. This efficiency advantage, however, is at the cost of gradient evaluation. For a simple model as first-order RBC and exact marginal likelihood evaluation from Kalman filter, a brute-force RWMH draw might perform fine speaking of the execution time, but RWMH will not suffice when the dimension of the parameters increases. Table 3 shows the joint likelihood method by drawing both the parameters and all the latent states. By comparing Table 3 with Table 2 we can see that the underlying parameters that govern the model yield a similar posterior in terms of both mean and standard deviation. The fraction of effective samples, however, drops because the joint likelihood approach draws from a much higher dimensional parameter space – $3+200=203$ overall. With Table 4 showing the related statistics, Figure 1 and Figure 2 show the trace plots and the posterior densities of the MCMC sample we draw. Each of the different lines shows one sample draw with 5000 samples, and we replicate 10 times. Without a huge number of sample draws, the NUTS sampler traverses the posterior efficiently, and this still holds when the sampling space is high dimensional in the joint likelihood case. In addition, the shapes of the posteriors are similar in these two cases.

After an initial check of the first-order RBC model which is a linear Gaussian case, we turn to second-order RBC models to illustrate the performance of the joint likelihood approach compared to RWMH with particle filter derived marginal likelihood. In a second-order RBC model, the data generating process is no longer linear Gaussian, and we apply the particle filter to filter through the latent variables in the marginal likelihood approach. Table 5 to Table 7 show the comparisons. For a relatively simple model as RBC, the performance of RWMH with particle filter is similar to that of the joint likelihood approach,

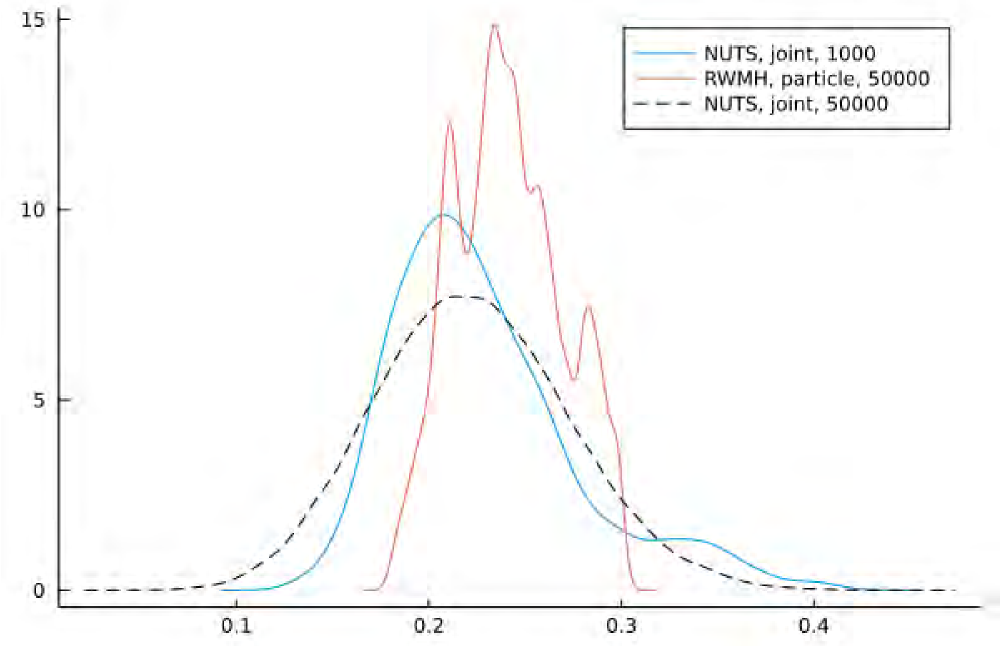


Figure 3: Posterior Density of β

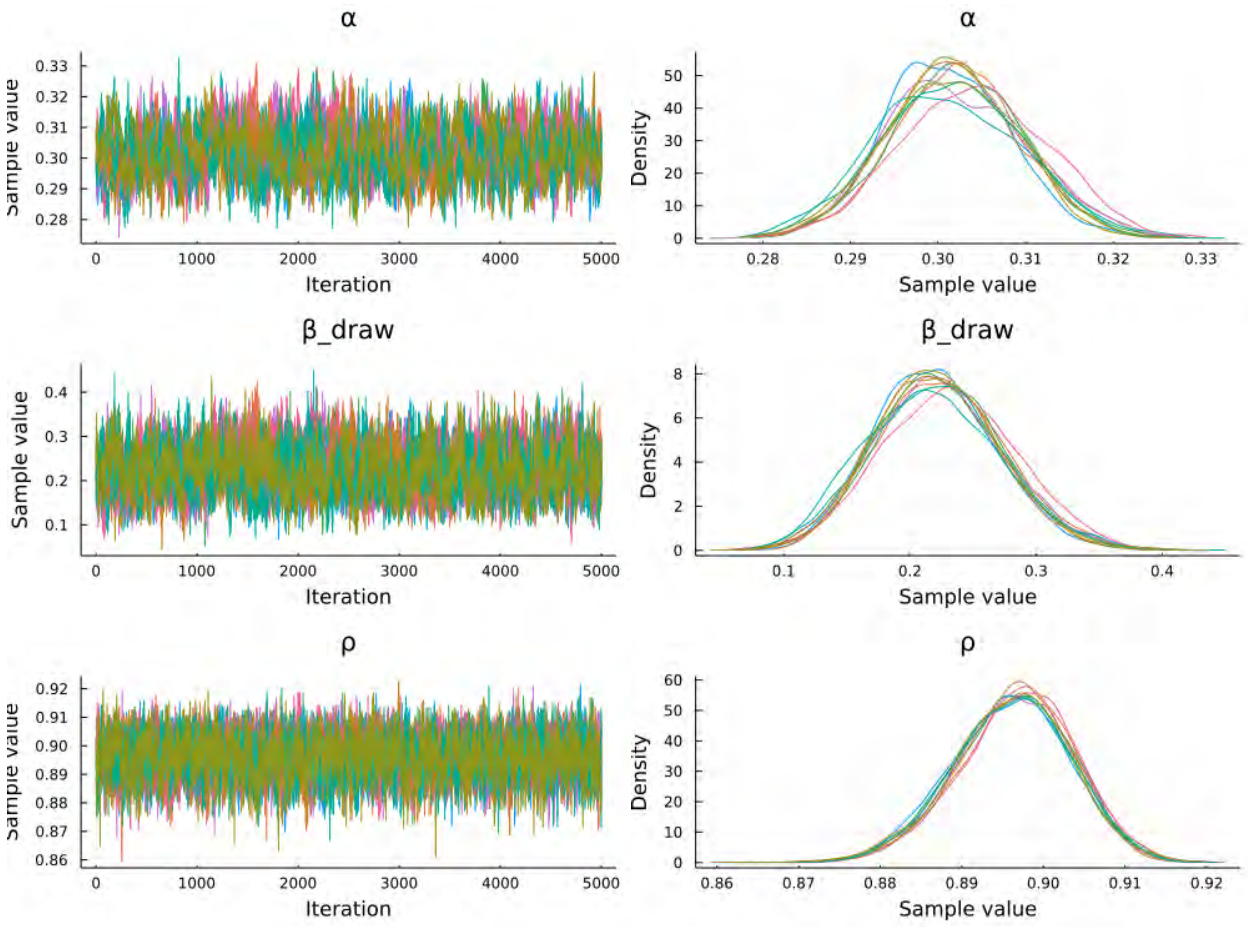


Figure 4: NUTS with Joint Likelihood, RBC Model, Second-order

Table 4: NUTS with Joint Likelihood, RBC Model, First-order, Parallel

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3019	0.0080	942.79	1.0106	0.019	0.1665	5663
$100\left(\beta^{-1}-1\right)$	0.2	0.2217	0.0509	1,513.16	1.0059	0.030	0.2672	5663
ρ	0.9	0.8964	0.0071	5,587.43	1.0023	0.111	0.9865	5663

Notes: We draw 5,000 samples across ten chains, for a total of 50,000 samples.

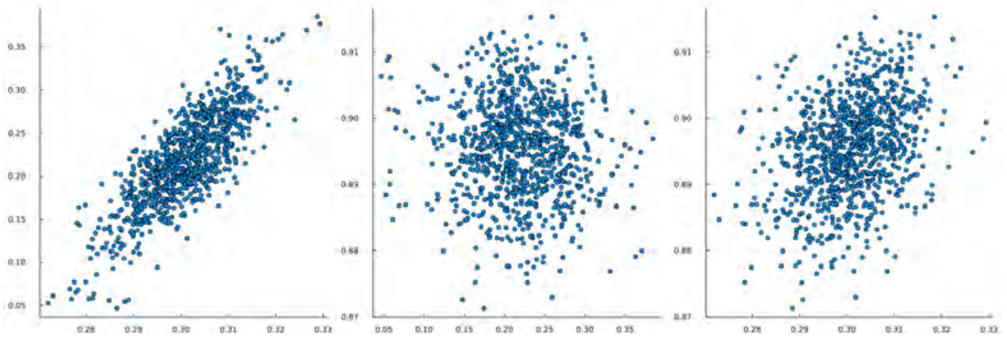
Table 5: RWMH with Marginal Likelihood on Particle Filter, RBC Model, Second-order

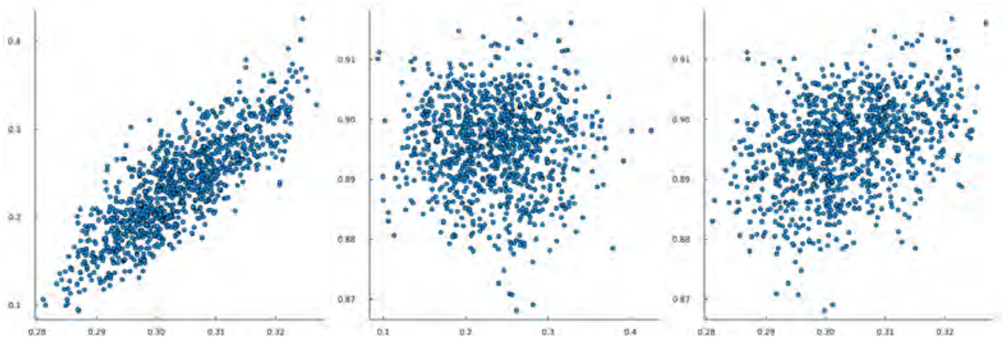
Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3037	0.0067	133.55	1.0479	0.003	0.0620	2154
$100\left(\beta^{-1}-1\right)$	0.2	0.2414	0.0281	102.90	1.1838	0.002	0.0478	2154
ρ	0.9	0.8965	0.0077	129.22	1.0012	0.002	0.0600	2154

as Table 5 and Table 6 show. Although the posterior mean and standard deviations are similar, the R-hat value is higher in the RWMH case, which means that the RWMH does not traverse the posterior well enough. To compare the efficiency between marginal likelihood with RWMH and joint likelihood with HMC, we plot the posterior densities of β in Figure 3. The black dashed line characterizes the posterior density we get after sufficiently long MCMC draws, while the blue and red line represents the density function from RWMH with particle filter and HMC with joint likelihood respectively, both running for around 5 minutes. While the RWMH yields much more samples, the density plot is less close to the true posterior compared to the HMC joint likelihood case, even though the gradient computation in HMC is costly. In addition, Figure 4 shows 10 independent MCMC draws from the HMC joint likelihood case, and the results are mixing well, suggesting the consistency of our method even it is drawing samples from a relatively high dimensional space. Table 7 shows the related statistics.

To further analyze the posterior distributions from the sampler, we layout the scatter plot of the pairwise joint distributions across the three parameters in the RBC model as in Figure 5. The three methods yield similar joint distributions of the samples, with α and β highly correlated and ρ independent.

The joint likelihood approach also provides a by-product for us. As we not only draw the parameters governing the model but also the latent state variables, we can back out the estimated latent states without running a filter and smoother in addition. Figure 6 shows the comparison between estimated posteriors and the actual shocks to generate the simulated data. The left panel and the right panel show results from first- and second-order RBC respectively. The red lines show the actual data, while the blue lines show the mean value of posterior estimation. The light blue region shows the interquartile ranges. Our estimations back out the value of original shocks accurately and the interquartile ranges look symmetric but heteroskedastic.


(a) First-order, Marginal Likelihood


(b) First-order, Joint Likelihood

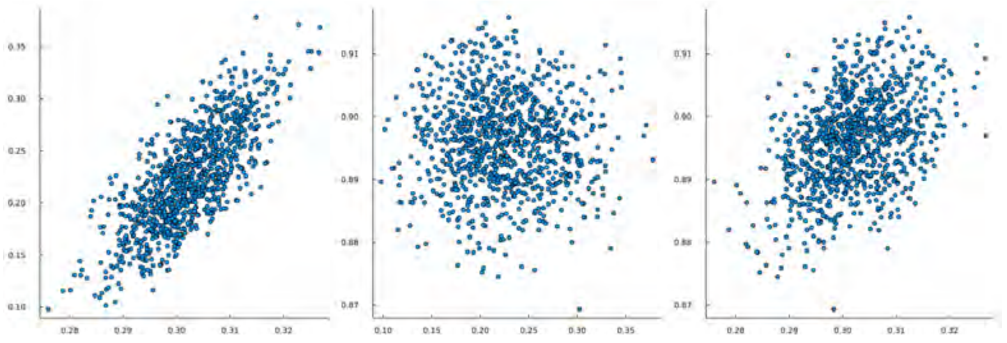

(c) Second-order, Joint Likelihood

Figure 5: Joint Distribution of Parameters
Notes: The left panel shows the correlation with α on the horizontal axis and $100(1/\beta-1)$ on the vertical axis. The middle panel shows the correlation with $100(1/\beta-1)$ on the horizontal axis and ρ on the vertical axis. The right panel shows the correlation with α on the horizontal axis and ρ on the vertical axis.

Table 6: NUTS with Joint Likelihood, RBC Model, Second-order

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3057	0.0071	39.67	1.0014	0.040	0.0160	2480
$100\left(\beta^{-1}-1\right)$	0.2	0.2455	0.0473	71.34	1.0027	0.071	0.0288	2480
ρ	0.9	0.8966	0.0072	153.86	0.9999	0.154	0.0600	2480

Table 7: NUTS with Joint Likelihood, RBC Model, Second-order, Parallel

Parameters	Pseudotrue	Post. Mean	Post. Std.	ESS	R-hat	ESS%	ESS/second	Time
α	0.3	0.3020	0.0078	984.36	1.0121	0.020	0.0497	19,810
$100\left(\beta^{-1}-1\right)$	0.2	0.2225	0.0508	1,530.21	1.0092	0.031	0.0772	19,810
ρ	0.9	0.8963	0.0072	5,021.72	1.0019	0.100	0.2535	19,810

Notes: We draw 5,000 samples across ten chains, for a total of 50,000 samples.

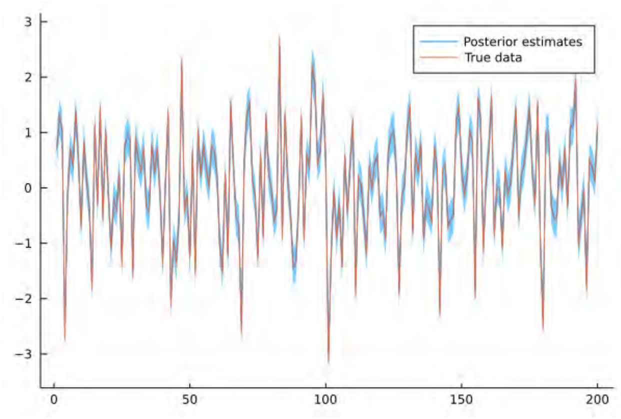
Frequentist Statistics While we carry out Bayesian estimations here, the frequentist statistics provide a sanity check of our estimation methods in general. In order to measure the frequentist statistics, we estimate the RBC model with all the three methods mentioned above with multiple replications. The pseudo-true of the underlying parameters are the same, but the generated artificial data are different due to different shock realizations. Moreover, for all the three methods above, we generate data with different lengths in time.

Table 8 shows the frequentist statistics for the estimation exercises with Kalman filter derived marginal likelihood and HMC sampler. The first two columns show the mean and standard deviation of the estimated mean and the pseudo-true value. The third and the fourth columns show the coverage probability, i.e., whether the 80% and 90% confidence interval of the estimated posterior contains the pseudo-true value. The three panels from up to down with different values in T illustrate experiments with different lengths. We can see that the coverage probabilities, on average, are more accurate as lengths of data increase. This is because lengthier data leads to more accurate identification of the underlying parameters. Moreover, the mean bias and mean squared error both decrease. Table 9 and Table 10 are the frequentist statistics for the joint likelihood approach in first- and second-order respectively. [11] We can see similar patterns compared to Table 8 which means the joint likelihood approach we propose is robust and credible to different shock realizations in general.

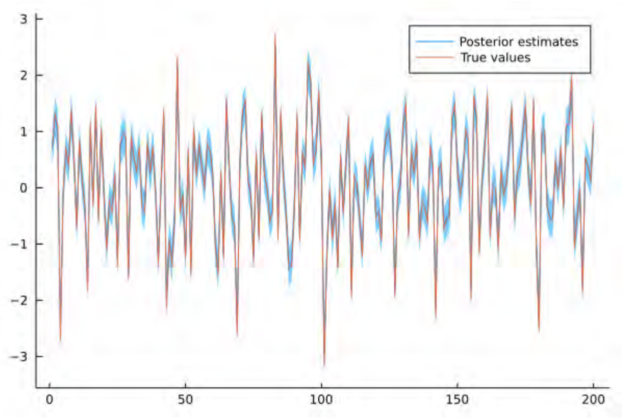
Robustness The parameter posteriors sampled based on the joint likelihood approach turn out to be more consistent than the posteriors generated from the traditional approach using marginal likelihood. We illustrate this comparison by starting the sampling process from a Cartesian product of grids on α, β, and ρ respectively. [12] Figure 9 and Fig-

[11]For the frequentist statistics of the joint likelihood exercises we provide the parameters only, not the latent state variables that we draw because they differ in each of the independent replications.

[12]We create a grid of 5 points for each of these three parameters, and the Cartesian product will be 125 starting points overall. The grid for α is $[0.25, 0.3, 0.35, 0.4, 0.45]$. The grid for $100\left(\beta^{-1}-1\right)$ is



(a) First-order RBC



(b) Second-order RBC

Figure 6: Inferred Shocks of RBC Model

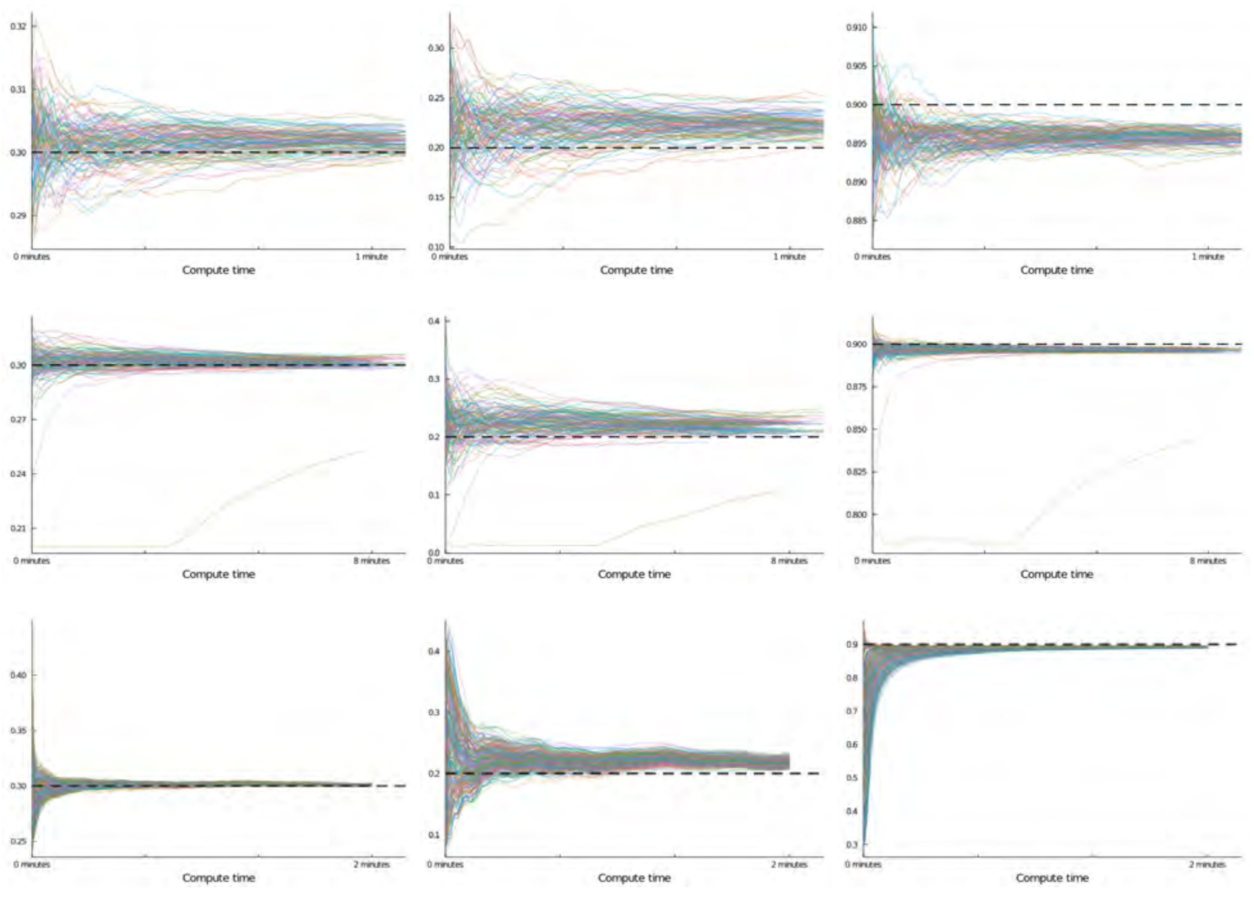


Figure 7: Robustness Comparison on First-order RBC: Cumulative Mean

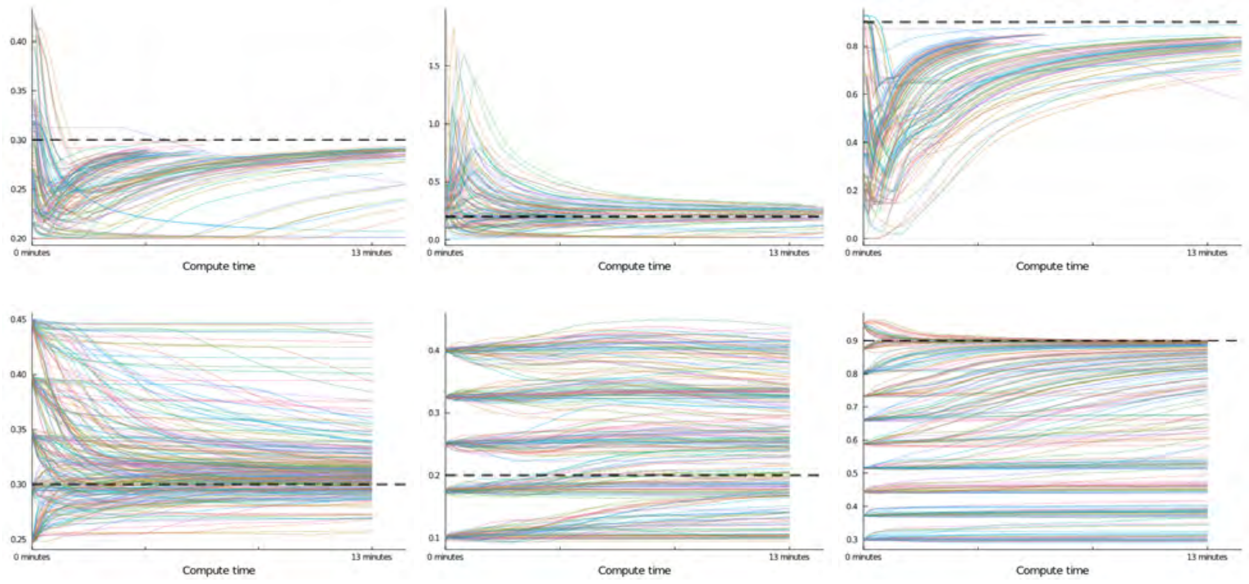


Figure 8: Robustness Comparison on Second-order RBC: Cumulative Mean

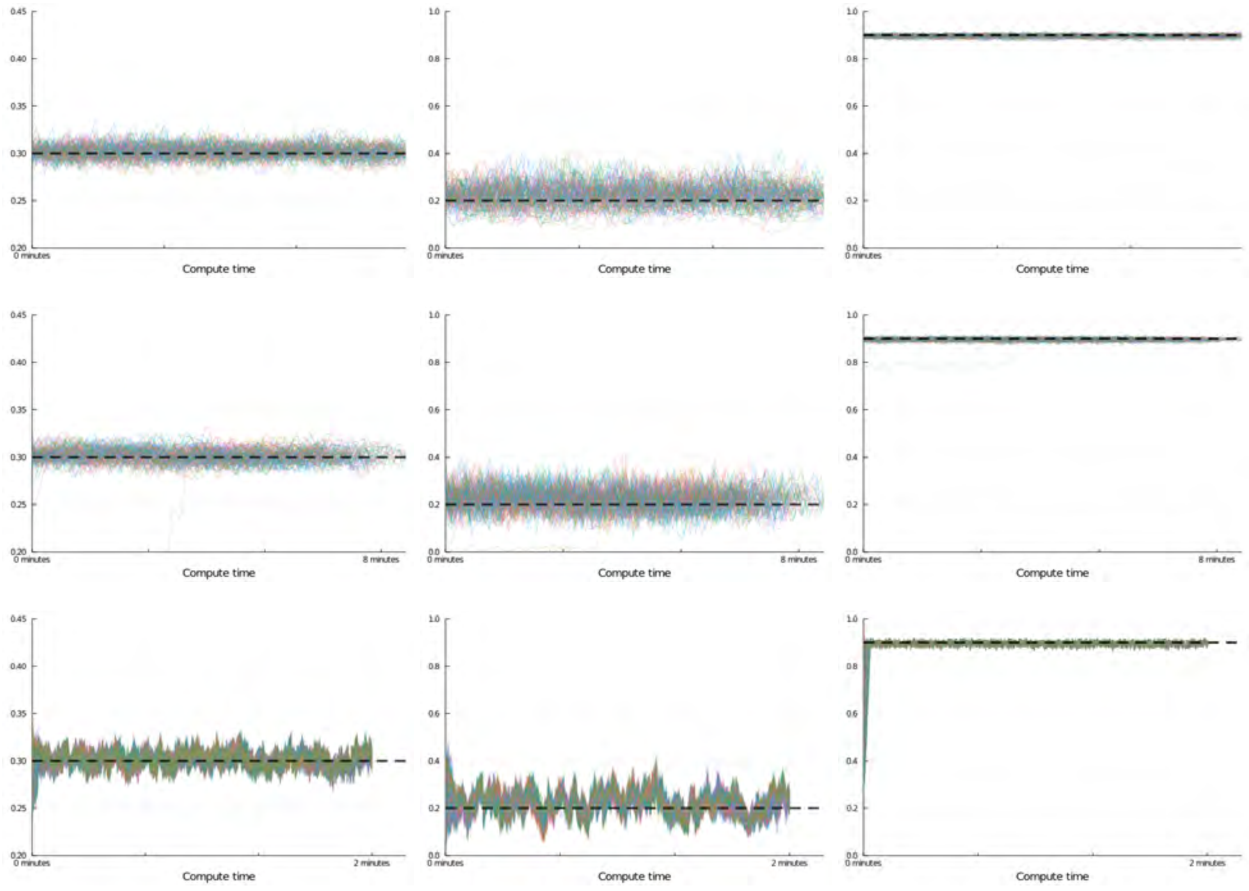


Figure 9: Robustness Comparison on First-order RBC: Trace Plot

Table 8: Frequentist Statistics – Kalman

	Parameters	Mean Bias	MSE	Cov. Prob. 80%	Cov. Prob. 90%
$T = 50$	α	-0.0028	7.25×10^{-5}	92%	98%
	$100\left(\beta^{-1} - 1\right)$	0.0221	0.0011	100%	100%
	ρ	-0.0115	0.0003	92%	98%
$T = 100$	α	-0.0029	5.35×10^{-5}	92%	95%
	$100\left(\beta^{-1} - 1\right)$	0.0062	0.0007	100%	100%
	ρ	-0.0057	4.93×10^{-5}	96%	99%
$T = 200$	α	-0.0015	3.65×10^{-5}	86%	97%
	$100\left(\beta^{-1} - 1\right)$	-0.0024	9.0×10^{-4}	99%	99%
	ρ	-0.0018	5.75×10^{-6}	100%	100%

Notes: All these statistics are generated from 100 estimation replications.

Table 9: Frequentist Statistics – 1st-order Joint

	Parameters	Mean Bias	MSE	Cov. Prob. 80%	Cov. Prob. 90%
$T = 50$	α	-0.0010	7.31×10^{-5}	94%	97%
	$100\left(\beta^{-1} - 1\right)$	0.0262	0.0015	99%	99%
	ρ	-0.0084	0.0001	98%	99%
$T = 100$	α	0.0001	4.92×10^{-5}	91%	97%
	$100\left(\beta^{-1} - 1\right)$	0.0130	0.0011	96%	100%
	ρ	-0.0033	2.58×10^{-5}	100%	100%
$T = 200$	α	-0.0022	8.84×10^{-5}	86%	93%
	$100\left(\beta^{-1} - 1\right)$	-0.0045	0.0009	92%	96%
	ρ	-0.0022	4.95×10^{-5}	99%	100%

Notes: All these statistics are generated from 100 estimation replications.

ure 10 show the trace plots, and Figure 7 and Figure 8 show the cumulative mean values. While different sampling methods share similar robustness in estimating the first-order RBC model, the results differ for the second-order case. For example, the upper panel of Figure 8 is from our joint likelihood approach with samples drawn from HMC, and the lower panel of Figure 8 is the counterparts from particle filtered marginal likelihood with RWMH, executed with Dynare. The parameters, from left to right, are α, β, and ρ respectively, with the black dashed line showing the pseudo-trues that generated the simulated data.

What we observe from Figure 8 is that even in a relatively simple problem like second-order RBC models, particle filtered marginal likelihood with RWMH takes a long time to converge to the high-density region of the posterior, while the joint likelihood approach with HMC is more consistent across different starting points. This phenomenon root in the mechanism of the sampling methods. HMC utilizes gradient information, therefore

$[0.1, 0.175, 0.25, 0.325, 0.4]$. The grid for ρ is $[0.3, 0.4625, 0.625, 0.7875, 0.95]$.

Table 10: Frequentist Statistics – 2nd-order Joint

	Parameters	Mean Bias	MSE	Cov. Prob. 80%	Cov. Prob. 90%
$T = 50$	α	-0.0026	0.0001	80%	90%
	$100\left(\beta^{-1} - 1\right)$	0.0253	0.0016	96%	98%
	ρ	-0.0108	0.0002	90%	98%
$T = 100$	α	3.58×10^{-6}	5.47×10^{-5}	84%	94%
	$100\left(\beta^{-1} - 1\right)$	0.0126	0.0012	96%	98%
	ρ	-0.0034	2.31×10^{-5}	100%	100%
$T = 200$	α	-0.0009	5.44×10^{-5}	76%	86%
	$100\left(\beta^{-1} - 1\right)$	7.20×10^{-5}	0.0013	84%	96%
	ρ	-0.0015	7.12×10^{-6}	100%	100%

Notes: All these statistics are generated from 50 estimation replications.

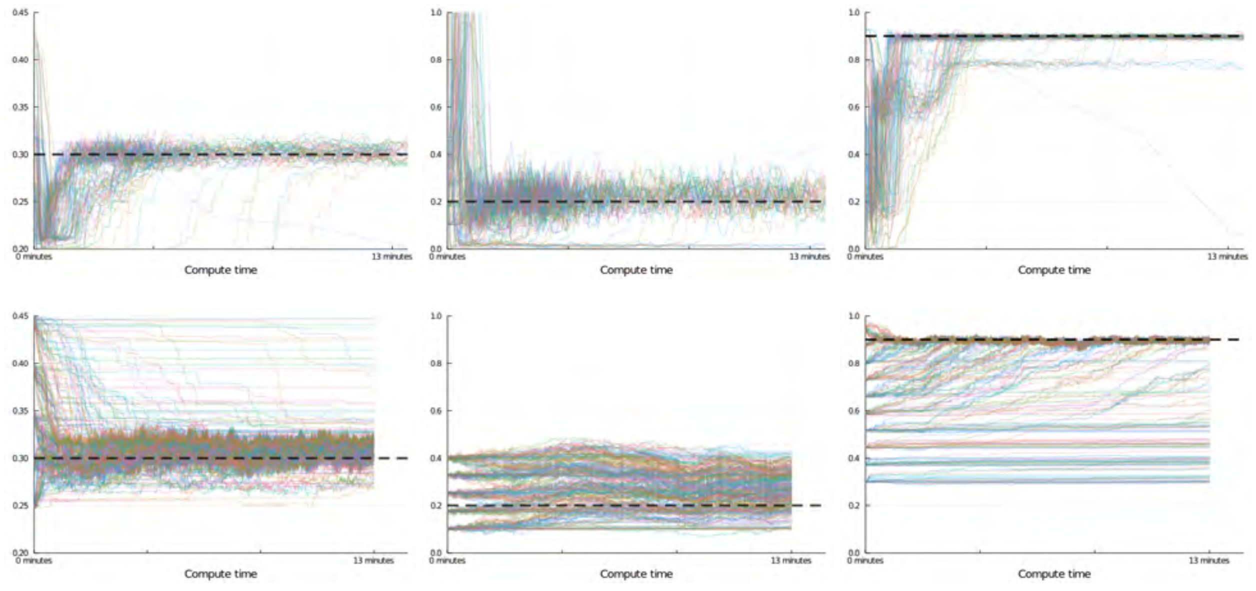


Figure 10: Robustness Comparison on Second-order RBC: Trace Plot

the sampler traverses faster and arrives at the typical set quickly, even in a high dimensional joint likelihood sample space. RWMH, on the other way around, takes gradual steps towards the typical set, because the likelihood improvement is marginal when the sampling point is distant from the pseudo-trues. We truncated the sampling time to be the same across all the experiments, and our method is more time-efficient even at the cost of gradient computations.

6.2 Estimating Fernández-Villaverde and Guerrón-Quintana (2021)

We show the estimation results of Fernández-Villaverde and Guerrón-Quintana (2021) (FVGQ), a canonical medium-scale New-Keynesian DSGE model in this section. In this model, we have a representative household that consumes, saves, supply labor, and hold nominal currencies. A final good firm produces output with a continuum of intermediate goods which are produced by monopolistic competitors facing Calvo type nominal rigidities. The representative household is the owner of all these firms. The government sets up monetary and fiscal policy, and we are considering a closed economy here. There are two unit root processes, with one governing the level of Hicks-neutral TFP and the other in investment-specific technology. [13]

There are six exogenous shocks in the model: household consumption preference, labor supply, TFP, investment efficiency, fiscal policy, and monetary policy. The first two shocks are to the preferences, and the third and fourth are two supply-side shocks. We select six time-series in real data for our estimation: inflation measured by CPI, the federal funds rate, the growth rate of real wages, the growth rate of real GDP per capita, per capita working hours, and the inverse relative price of investment with respect to the price of consumption growth. This model contains parameters to estimate and Table 11 summarizes the priors.

Figure 11 shows the estimated shocks as a by-product from the joint likelihood estimations. As we are simultaneously sampling the posterior distribution of the shock innovations along with the parameters, there is no necessity for us to back out the shock realizations (or equivalently, the posterior distributions of the latent states). If we were utilizing the marginal likelihood approach with filtering, then a smoother must be applied to back out these states. From Figure 11 we can see that we backed out the innovations to the shocks of an economic sense. During all the recessions periods, we see an unexpected drop in capital price and a temporary increase in labor supply. Moreover, we capture the Volcker period accurately as the monetary policy tightened significantly during the 1980-1982 recession.

[13]The online appendix layouts a full description and characterization of the model and data series for the estimation.

Table 11: Prior Distribution for structural parameters

Parameter	Distribution	Mean	Std
$100\,(1/\beta - 1)$	Gamma	0.25	0.1
h	Beta	0.7	0.1
κ	Normal	4	1.5
α	Normal	0.3	0.05
θ_p	Beta	0.5	0.1
χ	Beta	0.5	0.15
γ_R	Beta	0.75	0.1
γ_y	Normal	0.12	0.05
γ_Π	Normal	1.5	0.25
$100\,(\bar{\Pi} - 1)$	Gamma	0.95	0.1
$\bar{g}$	Beta	0.3	0.05
ρ_d	Beta	0.5	0.2
ρ_φ	Beta	0.5	0.2
ρ_g	Beta	0.5	0.2
σ_A	Inverse Gamma	0.1	1
σ_d	Inverse Gamma	0.1	1
σ_ϕ	Inverse Gamma	0.1	1
σ_μ	Inverse Gamma	0.1	1
σ_m	Inverse Gamma	0.1	1
σ_g	Inverse Gamma	0.1	1
Λ_μ	Gamma	0.0034	0.001
Λ_A	Gamma	0.00178	0.00075

7 Conclusion

The state space models are widely utilized in macroeconomics. The estimation of this set of models is not a trivial task, in particular, estimating nonlinear and non-Gaussian state space models has always been time-consuming and challenging. In this paper, we propose a new set of methods applying gradient-based MCMC methods with the differentiable programming paradigm to tackle this estimation problem from a novel perspective. By illustrating the sensitivity analysis to the first- and second-order approximation solutions to the state space models, we first show that the automatic differentiation tools can support relatively complicated derivative computations. Then, we provide a universal framework for nonlinear and non-Gaussian state space models. This framework samples the underlying latent state variables along with the model parameters and evaluates the joint likelihood rather than a marginal likelihood only on the parameters. We apply Hamiltonian Monte Carlo, a gradient-based scalable Bayesian sampling method, which allows efficient and robust sampling on a high dimensional space like this.

Future research along this strand can be vibrant. First of all, with the toolkit we provided that seamlessly passes gradient information of a complicated state space model

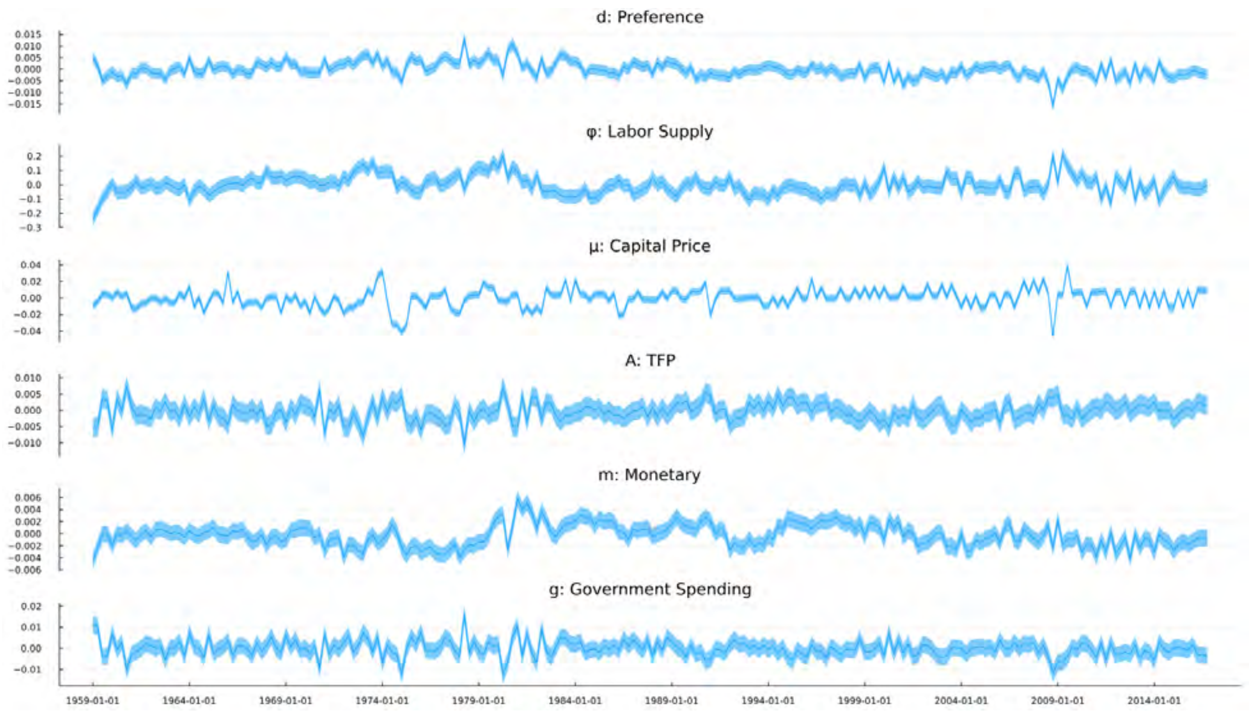


Figure 11: Inferred Shocks of Fernández-Villaverde and Guerrón-Quintana (2021)

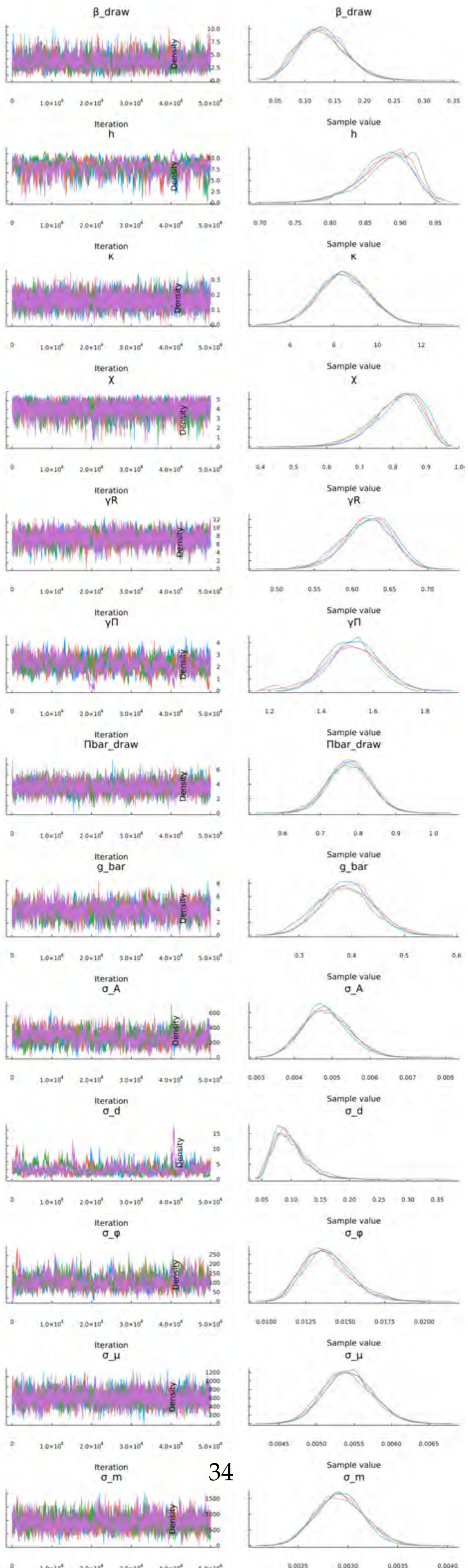
to its downstream, a variety of gradient-based estimation or sampling methods can be applied for state space model inferences. For example, we have already applied variational inference methods and gained satisfactory results in Fernández-Villaverde and Guerrón-Quintana (2021). Second, the method we proposed is scalable with dimensions of estimation, which will simplify the process of estimating heterogeneous agent models (for instance, in the spirit of Kaplan et al., 2018). Last but not the least, we showed two applications in estimating macro state space models, but the estimation method is much more general and open to any potential applications characterized by general state space models. We will definitely see widespread usage of differentiable programming in economics in the next few years.

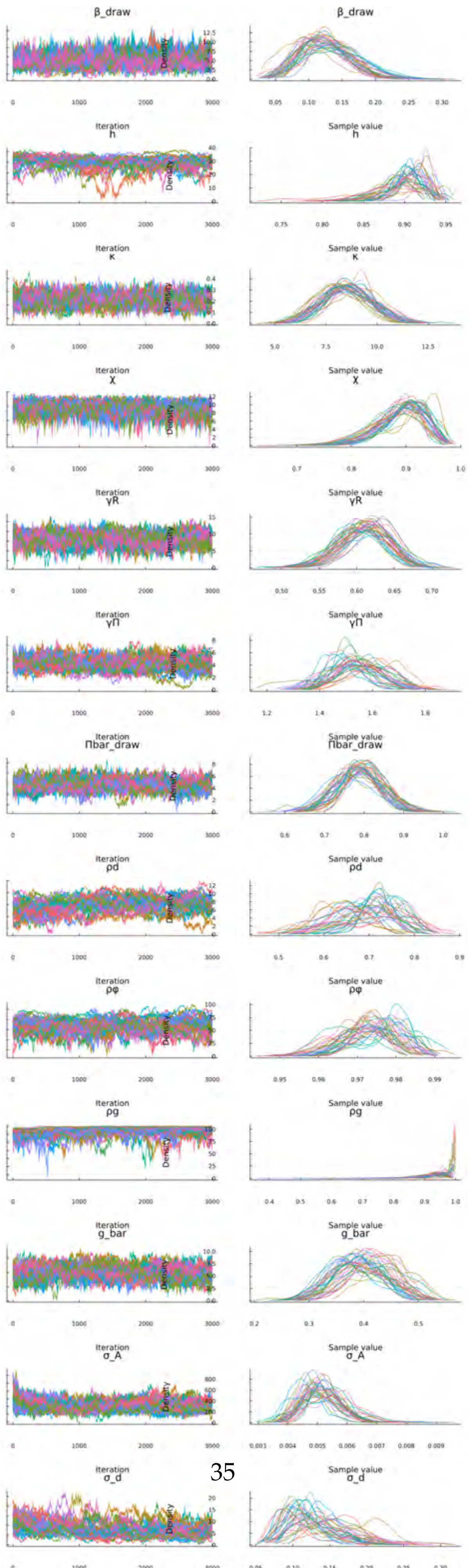
Table 12: Estimation with marginal likelihood approach, first-order FVGQ model

parameters	mean	std	ess	rhat
$\bar{g}$	0.515899	0.0625615	368.376	1.00583
h	0.868682	0.0667568	230.275	1.00219
Λ_A	0.00116548	0.000244789	513.093	1.00026
Λ_μ	0.00577486	0.000391655	832.384	1.00015
$100\,(\bar{\Pi}-1)$	0.934494	0.0878034	1039.63	0.999392
α	0.292393	0.0217562	385.006	1.00235
$100\,(1/\beta-1)$	0.144239	0.0468432	1350.98	0.999566
γ_R	0.615112	0.0339691	760.218	0.999657
γ_Π	1.21828	0.165471	73.9837	1.0039
θ_p	0.802364	0.0403749	157.453	0.999114
κ	7.73702	1.38322	246.787	0.999187
ρ_d	0.807469	0.0818411	215.386	1.00744
ρ_g	0.951315	0.0603435	144.366	1.00425
ρ_φ	0.978821	0.00825107	195.694	0.999149
σ_A	0.00826692	0.00174755	581.122	1.0034
σ_d	0.124462	0.066943	257.565	1.00278
σ_g	0.0071145	0.00135781	128.841	1.01598
σ_m	0.00290885	0.000237841	1008.79	0.999811
σ_μ	0.00544747	0.000311396	1105.1	1.0003
σ_ϕ	0.0139098	0.00219333	744.511	1.00049
χ	0.738719	0.111171	199.604	1.00709

Table 13: Estimation with joint likelihood, first-order FVGQ model

parameters	mean	std	ess	rhat
$\bar{g}$	0.396885	0.0547821	217.875	1.01205
h	0.89308	0.0468923	107.447	1.05759
Λ_A	0.00152115	0.000239741	214.941	1.01329
Λ_μ	0.00586824	0.000397612	209.578	1.01284
$100\,(\bar{\Pi}-1)$	0.784925	0.0585665	288.996	1.01055
α	0.255145	0.020716	171.773	1.0168
$100\,(1/\beta-1)$	0.131962	0.0428444	366.458	1.01815
γ_R	0.623479	0.0329517	287.944	1.01374
γ_Π	1.56887	0.117258	119.791	1.15572
θ_p	0.767652	0.0321262	158.629	1.06112
κ	8.4237	1.23334	363.534	1.02184
ρ_d	0.707503	0.0964784	97.075	1.04763
ρ_g	0.958456	0.0302879	148.99	1.05296
ρ_φ	0.976573	0.00834117	117.409	1.10795
σ_A	0.00519128	0.000744698	164.511	1.09215
σ_d	0.101411	0.0439758	103.577	1.02903
σ_g	0.00663444	0.000965019	142.625	1.06336
σ_m	0.00289127	0.000252272	372.254	1.00369
σ_μ	0.00546604	0.000344543	421.534	1.00982
σ_ϕ	0.0140017	0.00155692	167.677	1.02045
χ	0.752136	0.121688	187.561	1.01484


β_draw
h
κ
χ
γR
γΠ
Πbar_draw
g_bar
σ_A
σ_d
σ_φ
σ_μ
σ_m
Iteration
Sample value
Density



β_draw
h
κ
χ
γR
γΠ
Πbar_draw
ρd
ρφ
ρg
g_bar
σ_A
σ_d
Iteration
Sample value
Density

References

BAYDIN, A. G., B. A. PEARLMUTTER, A. A. RADUL, AND J. M. SISKIND (2017): "Automatic Differentiation in Machine Learning: A Survey," *J. Mach. Learn. Res.*, 18, 5595–5637.

BETANCOURT, M. (2018): "A Conceptual Introduction to Hamiltonian Monte Carlo," .

FARKAS, M. AND B. TATAR (2020): "Bayesian estimation of DSGE models with Hamiltonian Monte Carlo," Tech. rep., IMFS Working Paper Series.

FERNÁNDEZ-VILLAVERDE, J. AND P. A. GUERRÓN-QUINTANA (2021): "Estimating DSGE models: Recent advances and future challenges," *Annual Review of Economics*, 13.

FERNÁNDEZ-VILLAVERDE, J. AND J. F. RUBIO-RAMÍREZ (2007): "Estimating macroeconomic models: A likelihood approach," *Review of Economic Studies*, 74, 1059–1087.

FERNÁNDEZ-VILLAVERDE, J., J. F. RUBIO-RAMÍREZ, AND F. SCHORFHEIDE (2016): "Solution and estimation methods for DSGE models," in *Handbook of macroeconomics*, Elsevier, vol. 2, 527–724.

GIROLAMI, M. AND B. CALDERHEAD (2011): "Riemann manifold langevin and hamiltonian monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73, 123–214.

GOWDA, S., Y. MA, A. CHELI, M. GWOZDZ, V. B. SHAH, A. EDELMAN, AND C. RACKAUCKAS (2021): "High-performance symbolic-numerics via multiple dispatch," *arXiv preprint arXiv:2105.03949*.

GRIEWANK, A. AND A. WALTHER (2008): *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM.

HOFFMAN, M. D. AND A. GELMAN (2014): "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, 15, 1593–1623.

INNES, M., A. EDELMAN, K. FISCHER, C. RACKAUCKAS, E. SABA, V. B. SHAH, AND W. TEBBUTT (2019): "A differentiable programming system to bridge machine learning and scientific computing," *arXiv preprint arXiv:1907.07587*.

ISKREV, N. (2010): "Local identification in DSGE models," *Journal of Monetary Economics*, 57, 189–202.

KAPLAN, G., B. MOLL, AND G. L. VIOLANTE (2018): "Monetary policy according to HANK," *American Economic Review*, 108, 697–743.

KIM, S., N. SHEPHARD, AND S. CHIB (1998): "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models," *Review of Economic Studies*, 65, 361–393.

KLEIN, P. (2000): "Using the generalized Schur form to solve a multivariate linear rational expectations model," *Journal of economic dynamics and control*, 24, 1405–1423.

SCHÄFER, F., M. TAREK, L. WHITE, AND C. RACKAUCKAS (2021): "AbstractDifferentiation. jl: Backend-Agnostic Differentiable Programming in Julia," *arXiv preprint arXiv:2109.12449*.

SCHMITT-GROHÉ, S. AND M. URIBE (2004): "Solving dynamic general equilibrium models using a second-order approximation to the policy function," *Journal of Economic Dynamics and Control*, 28, 755–775.

SMETS, F. AND R. WOUTERS (2007): "Shocks and frictions in US business cycles: A Bayesian DSGE approach," *American Economic Review*, 97, 586–606.

WHITE, L., M. ZGUBIC, M. ABBOTT, J. REVELS, A. ARSLAN, S. AXEN, S. SCHAUB, N. ROBINSON, Y. MA, G. DHINGRA, WILLTEBBUTT, N. HEIM, A. D. W. ROSEMBERG, D. WIDMANN, N. SCHMITZ, C. RACKAUCKAS, R. HEINTZMANN, FRANKSCHAE, K. FISCHER, A. ROBSON, MATTBRZEZINSKI, A. ZHABINSKI, M. BESANÇON, P. VERTECHI, S. GOWDA, A. FITZGIBBON, C. LUCIBELLO, C. VOGT, D. GANDHI, AND F. CHORNEY (2021): "JuliaDiff/ChainRules.jl: v1.14.0," .

Appendix A Perturbation Solution and Notation

In this section, we define the perturbation solution to the DSGE model, both in first- and second-order, with notations and implementation details.

A.1 Tensor Notations

We use tensors for convenience when referring to high dimensional objects. For Jacobian matrices, $\left[f_y\right]^i_\alpha = \frac{\partial f^i}{\partial y^\alpha}$ is the (i, α) element of the derivative of f with respect to y. The dimension of the Jacobian matrix will be $m \times n$ where m is the dimension of f and n is the dimension of y. $i = 1...m$ and $\alpha = 1...n$.

An example of tensor contraction notation is below,

$$\left[f_y\right]^i_\alpha \left[g_x\right]^\alpha_j = \sum_{\alpha=1}^{n} \frac{\partial f^i}{\partial y^\alpha} \frac{\partial g^\alpha}{\partial x^j}$$

For Hessian matrices, $\left[\mathcal{H}_{xy}\right]^i_{\alpha\gamma}$ is row i, column α, page γ of 3-dimensional matrix. If m, n, k are the dimensions for $\mathcal{H}, x, y$ separately then $i = 1, \ldots, m$, $\alpha = 1, \ldots, n$, $\gamma = 1, \ldots, k$.

A.2 Definitions

Dimensions of related vectors and matrices:

- x: $n_x \times 1$
- y: $n_y \times 1$
- ϵ: $n_\epsilon \times 1$
- η: $n_x \times n_\epsilon$
- Σ: $n_\epsilon \times n_\epsilon$

Denote $\bar{x}$, $\bar{y}$ as the deterministic steady state that satisfies $\mathcal{H}\left(\bar{y}, \bar{y}, \bar{x}, \bar{x}\right) = 0$ when muting the shock processes.

- All vectors are in columns.
- n_x: the number of state variables; n_y: the number of control variables; n_θ: the number of deep parameters; n_ϵ: the number of exogenous shocks.
- There are $n = n_x + n_y$ equations in the system. $\mathcal{H} : \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}^n$.

– $x' = h(x) + \eta\epsilon$: h is the law of the motion of the states, ϵ is the exogenous shocks, i.i.d., with a variance-covariance (VCV) matrix Σ, η is the loadings of the shocks; $y = g(x)$: g is the policy function.

A.3 First-order Solution

We need to solve g_x and h_x. The dimensions of g_x and h_x are $n_y \times n_x$ and $n_x \times n_x$, respectively. A first-order Taylor expansion of (19) yields the following equation

$$\mathcal{H}_{y'}y' + \mathcal{H}_y y + \mathcal{H}_{x'}x' + \mathcal{H}_x x = 0,$$

where $\mathcal{H}_{x;ij} = \frac{\partial \mathcal{H}_i}{\partial x_j}$, $i = 1, \ldots, n$, $j = 1, \ldots n_x$;$\mathcal{H}_{y;ij} = \frac{\partial \mathcal{H}_i}{\partial y_j}$, $i = 1, \ldots, n$, $j = 1, \ldots n_y$. All of these derivatives are evaluated at the deterministic steady state. Put the equation above in another form

$$\begin{bmatrix} \mathcal{H}_{x'} & \mathcal{H}_{y'} \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} \mathcal{H}_x & \mathcal{H}_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0. \tag{A.1}$$

Then, following Klein (2000), we apply generalized Schur decomposition to matrices $\begin{bmatrix} \mathcal{H}_{x'} & \mathcal{H}_{y'} \end{bmatrix}$ and $\begin{bmatrix} \mathcal{H}_x & \mathcal{H}_y \end{bmatrix}$ from (A.1), and follow Blanchard-Kahn condition to reorder so that $\forall i, \left|\frac{S_{22,i}}{T_{22,i}}\right| < 1$. Therefore, it results

$$\begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix} \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0. \tag{A.2}$$

The dimension of T_{22} or S_{22} should be n_y as in Blanchard-Kahn condition to ensure that the converging solution exists and is unique. Therefore,

$$g_x = -Z_{22}^{-1} Z_{21} \tag{A.3}$$

$$h_x = -\left(Z_{11} + Z_{12} g_x\right)^{-1} \left(S_{11}\right)^{-1} T_{11} \left(Z_{11} + Z_{12} g_x\right) \tag{A.4}$$

A.4 Second-order Solution

We need to solve $g_{xx}, h_{xx}, g_{\sigma\sigma}, h_{\sigma\sigma}$. The dimensions of g_{xx} and h_{xx} are $n_y \times n_x \times n_x$ and $n_x \times n_x \times n_x$. The dimensions of $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$ are $n_y \times 1$ and $n_x \times 1$.

Second-order perturbation of (19) yields

$$
\begin{aligned}
&\left(\left[\mathcal{H}_{y'y'}\right]^{i}_{\alpha\gamma}[g_x]^{\gamma}_{\delta}[h_x]^{\delta}_{k}+\left[\mathcal{H}_{y'y}\right]^{i}_{\alpha\gamma}[g_x]^{\gamma}_{k}+\left[\mathcal{H}_{y'x'}\right]^{i}_{\alpha\delta}[h_x]^{\delta}_{k}+\left[\mathcal{H}_{y'x}\right]^{i}_{\alpha k}\right)[g_x]^{\alpha}_{\beta}[h_x]^{\beta}_{j}\\
&\qquad+\left[\mathcal{H}_{y'}\right]^{i}_{\alpha}[g_{xx}]^{\alpha}_{\beta\delta}[h_x]^{\delta}_{k}[h_x]^{\beta}_{j}+\left[\mathcal{H}_{y'}\right]^{i}_{\alpha}[g_x]^{\alpha}_{\beta}[h_{xx}]^{\beta}_{jk}\\
&+\left(\left[\mathcal{H}_{yy'}\right]^{i}_{\alpha\gamma}[g_x]^{\gamma}_{\delta}[h_x]^{\delta}_{k}+[\mathcal{H}_{yy}]^{i}_{\alpha\gamma}[g_x]^{\gamma}_{k}+\left[\mathcal{H}_{yx'}\right]^{i}_{\alpha\delta}[h_x]^{\delta}_{k}+[\mathcal{H}_{yx}]^{i}_{\alpha k}\right)[g_x]^{\alpha}_{j}\\
&\qquad+[\mathcal{H}_{y}]^{i}_{\alpha}[g_{xx}]^{\alpha}_{jk}\\
&+\left(\left[\mathcal{H}_{x'y'}\right]^{i}_{\beta\gamma}[g_x]^{\gamma}_{\delta}[h_x]^{\delta}_{k}+\left[\mathcal{H}_{x'y}\right]^{i}_{\beta\gamma}[g_x]^{\gamma}_{k}+[\mathcal{H}_{x'x'}]^{i}_{\beta\delta}[h_x]^{\delta}_{k}+[\mathcal{H}_{x'x}]^{i}_{\beta k}\right)[h_x]^{\beta}_{j}\\
&\qquad+[\mathcal{H}_{x'}]^{i}_{\beta}[h_{xx}]^{\beta}_{jk}\\
&+\left[\mathcal{H}_{xy'}\right]^{i}_{j\gamma}[g_x]^{\gamma}_{\delta}[h_x]^{\delta}_{k}+[\mathcal{H}_{xy}]^{i}_{j\gamma}[g_x]^{\gamma}_{k}+[\mathcal{H}_{xx'}]^{i}_{j\delta}[h_x]^{\delta}_{k}+[\mathcal{H}_{xx}]^{i}_{jk}=0
\end{aligned}
$$

where $i=1,\dots,n$, $\alpha,\gamma=1,\dots,n_y$, $\beta,\delta,j,k=1,\dots,n_x$. If we squeeze the 2nd and 3rd dimension of g_{xx} and h_{xx} matrices (hence the dimensions are $n_y\times n_x^2$ and $n_x\times n_x^2$), then the above equation can be written as

$$
\begin{bmatrix}\mathcal{H}_{y'} & 0\end{bmatrix}\begin{bmatrix}g_{xx}\\ h_{xx}\end{bmatrix}h_x\otimes h_x+\begin{bmatrix}\mathcal{H}_{y} & \mathcal{H}_{y'}g_x+\mathcal{H}_{x'}\end{bmatrix}\begin{bmatrix}g_{xx}\\ h_{xx}\end{bmatrix}+C=0, \tag{A.5}
$$

where C is a $(n_x+n_y)*(n_x^2)$ matrix. We compute each row of C below,

$$
C_{i,\cdot}=\operatorname{vec}\left(\begin{bmatrix}g_xh_x\\ g_x\\ h_x\\ I\end{bmatrix}^{T}\begin{bmatrix}\mathcal{H}_{y'y'} & \mathcal{H}_{y'y} & \mathcal{H}_{y'x'} & \mathcal{H}_{y'x}\\ \mathcal{H}_{yy'} & \mathcal{H}_{yy} & \mathcal{H}_{yx'} & \mathcal{H}_{yx}\\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'y} & \mathcal{H}_{x'x'} & \mathcal{H}_{x'x}\\ \mathcal{H}_{xy'} & \mathcal{H}_{xy} & \mathcal{H}_{xx'} & \mathcal{H}_{xx}\end{bmatrix}_{i}\begin{bmatrix}g_xh_x\\ g_x\\ h_x\\ I\end{bmatrix}\right). \tag{A.6}
$$

Notice that the matrix in the middle of (A.6) is the Hessian of $\mathcal{H}_i$.

(A.5) is a Sylvester equation for $\begin{bmatrix}g_{xx}\\ h_{xx}\end{bmatrix}$. We can use some existing solver for the generalized Sylvester equation; or, another way to exploit the radius of h_x is to use a doubling method. [14]

After we solve g_{xx} and h_{xx}, we can write down the equations for $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$,

[14]Rewrite the above equation as $AXF\otimes F+BX+C=0$ where

$$
\begin{aligned}
A&=\begin{bmatrix}\mathcal{H}_{y'} & 0\end{bmatrix}\\
F&=h_x\\
B&=\begin{bmatrix}\mathcal{H}_{y} & \mathcal{H}_{y'}g_x+\mathcal{H}_{x'}\end{bmatrix}
\end{aligned}
$$

$$\begin{aligned}\left[\mathcal{H}_{y'}\right]^i_\alpha [g_x]^\alpha_\beta [h_{\sigma\sigma}]^\beta + \left[\mathcal{H}_{y'}\right]^i_\alpha [g_{\sigma\sigma}]^\alpha + [\mathcal{H}_y]^i_\alpha [g_{\sigma\sigma}]^\alpha + [\mathcal{H}_{x'}]^i_\beta [h_{\sigma\sigma}]^\beta \\ + \left[\mathcal{H}_{y'y'}\right]^i_{\alpha\gamma} [g_x]^\gamma_\delta [\eta]^\delta_\xi [g_x]^\alpha_\beta [\eta]^\beta_\phi [\Sigma]^\phi_\xi \\ + \left[\mathcal{H}_{y'x'}\right]^i_{\alpha\delta} [\eta]^\delta_\xi [g_x]^\alpha_\beta [\eta]^\beta_\phi [\Sigma]^\phi_\xi \\ + \left[\mathcal{H}_{y'}\right]^i_\alpha [g_{xx}]^\alpha_{\beta\delta} [\eta]^\delta_\xi [\eta]^\beta_\phi [\Sigma]^\phi_\xi \\ + \left[\mathcal{H}_{x'y'}\right]^i_{\beta\gamma} [g_x]^\gamma_\delta [\eta]^\delta_\xi [\eta]^\beta_\phi [\Sigma]^\phi_\xi \\ + [\mathcal{H}_{x'x'}]^i_{\beta\delta} [\eta]^\delta_\xi [\eta]^\beta_\phi [\Sigma]^\phi_\xi = 0,\end{aligned}$$

where $i = 1, \ldots, n$, $\alpha, \gamma = 1, \ldots, n_y$, $\beta, \delta = 1, \ldots, n_x$, $\phi, \xi = 1, \ldots, n_\epsilon$. This yields a linear equation for $\begin{bmatrix} g_{\sigma\sigma} \\ h_{\sigma\sigma} \end{bmatrix}$. We reorganize the above equation and we get

$$\begin{pmatrix} \mathcal{H}_{y'} + \mathcal{H}_y & \mathcal{H}_{y'} g_x + \mathcal{H}_{x'} \end{pmatrix} \begin{pmatrix} g_{\sigma\sigma} \\ h_{\sigma\sigma} \end{pmatrix} + B = 0. \tag{A.7}$$

Define function sum $(A) = \sum_{i,j} A_{ij}$, and the notation for element-wise product (Hadamard product) as $\odot$, then B follows

$$B^i = \text{sum}\left(\left[\begin{bmatrix} g_x \\ I \end{bmatrix}^T \begin{bmatrix} \mathcal{H}_{y'y'} & \mathcal{H}_{y'x'} \\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'x'} \end{bmatrix}^i \begin{bmatrix} g_x \\ I \end{bmatrix} + \left[\mathcal{H}_{y'}\right]^i g_{xx}\right] \odot [\eta\Sigma\eta']\right)$$

for all $i = 1, \ldots, n$.

Here $\left[\mathcal{H}_{y'}\right]^i g_{xx}$ is a tensor shrink. That is to say, $\left\{\left[\mathcal{H}_{y'}\right]^i g_{xx}\right\}_{\beta\delta} = \sum_{\alpha=1}^{n_y} \left[\mathcal{H}_{y'}\right]^i_\alpha [g_{xx}]^\alpha_{\beta\delta}$. To implement this we can first squeeze g_{xx} into a $n_y \times (n_x^2)$ matrix, and then reshape the matrix product $\left[\mathcal{H}_{y'}\right]^i g_{xx}$ into a $n_x \times n_x$ matrix.

Then, we apply an iterative doubling method to solve out X for this specific form of Sylvester equation,

$$\begin{aligned} X_1 &= -B^{-1}C \\ X_{n+1} &= X_n + A_n X_n F_n \otimes F_n \\ A_{n+1} &= A_n A_n \\ F_{n+1} &= F_n F_n \end{aligned}$$

Appendix B Perturbation Solution Derivatives

This section derives the derivatives of deterministic steady states, solutions in first- and second-order, with respect to the model parameters.

B.1 Derivative of the Steady States

The model parameters θ in general will change steady state values $\bar{x}, \bar{y}$. We are interested in $\frac{\partial \bar{x}}{\partial \theta}$ and $\frac{\partial \bar{y}}{\partial \theta}$.

We take the derivatives of (19) with respect to θ,

$$F_\theta\left(x;\theta\right) = \mathcal{H}_x \frac{\partial x}{\partial \theta} + \mathcal{H}_y \frac{\partial y}{\partial \theta} + \mathcal{H}_{x'} \frac{\partial x'}{\partial \theta} + \mathcal{H}_{y'} \frac{\partial y'}{\partial \theta} + \mathcal{H}_\theta = 0,$$

where $\mathcal{H}_{\theta;ij} = \frac{\partial \mathcal{H}_i}{\partial \theta_j}, i = 1, \ldots, n, j = 1, \ldots n_\theta$. We evaluate this function at the deterministic steady states $x' = x = \bar{x}, y' = y = \bar{y}$,

$$\begin{bmatrix} \mathcal{H}_y + \mathcal{H}_{y'} & \mathcal{H}_x + \mathcal{H}_{x'} \end{bmatrix} \begin{bmatrix} \frac{\partial \bar{y}}{\partial \theta} \\ \frac{\partial \bar{x}}{\partial \theta} \end{bmatrix} + \mathcal{H}_\theta = 0, \tag{B.1}$$

(B.1) is a linear equation system when we treat $\frac{\partial \bar{x}}{\partial \theta}$ and $\frac{\partial \bar{y}}{\partial \theta}$ as unknowns.

B.2 First-order Results

We are interested in $\frac{\partial g_x}{\partial \theta}$ and $\frac{\partial h_x}{\partial \theta}$.

From (19), the first-order model solution satisfies

$$F_x\left(x;\theta\right) = \mathcal{H}_x + \mathcal{H}_y g_x + \mathcal{H}_{x'} h_x + \mathcal{H}_{y'} g_x h_x = 0. \tag{B.2}$$

For each element θ_i of θ, we take derivative of (B.2),

$$F_{x,\theta_i}\left(x;\theta\right) = \frac{\mathrm{d}\mathcal{H}_x}{\mathrm{d}\theta_i} + \frac{\mathrm{d}\mathcal{H}_y}{\mathrm{d}\theta_i} g_x + \mathcal{H}_y \frac{\partial g_x}{\partial \theta_i} + \frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta_i} h_x + \mathcal{H}_{x'} \frac{\partial h_x}{\partial \theta_i} + \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} g_x h_x + \mathcal{H}_{y'} \frac{\partial g_x}{\partial \theta_i} h_x + \mathcal{H}_{y'} g_x \frac{\partial h_x}{\partial \theta_i} = 0,$$

and we evaluate the system at the steady state $x' = x = \bar{x}, y' = y = \bar{y}$. Notice that $\mathcal{H}$ is

$\mathcal{H}(y', y, x', x; \theta)$, and chain rule applies here as we compute the total derivative of $\mathcal{H}$:

$$\begin{aligned}
\left[\frac{\mathrm{d}\mathcal{H}_x}{\mathrm{d}\theta_i}\right]^{\alpha}_{\beta} &= \left[\mathcal{H}_{xy'} + \mathcal{H}_{xy}\right]^{\alpha}_{\beta\xi}\left[\frac{\partial\bar{y}}{\partial\theta_i}\right]^{\xi} + \left[\mathcal{H}_{xx'} + \mathcal{H}_{xx}\right]^{\alpha}_{\beta\delta}\left[\frac{\partial\bar{x}}{\partial\theta_i}\right]^{\delta} + \left[\frac{\partial\mathcal{H}_x}{\partial\theta_i}\right]^{\alpha}_{\beta} \\
\left[\frac{\mathrm{d}\mathcal{H}_y}{\mathrm{d}\theta_i}\right]^{\alpha}_{\gamma} &= \left[\mathcal{H}_{yy'} + \mathcal{H}_{yy}\right]^{\alpha}_{\gamma\xi}\left[\frac{\partial\bar{y}}{\partial\theta_i}\right]^{\xi} + \left[\mathcal{H}_{yx'} + \mathcal{H}_{yx}\right]^{\alpha}_{\gamma\delta}\left[\frac{\partial\bar{x}}{\partial\theta_i}\right]^{\delta} + \left[\frac{\partial\mathcal{H}_y}{\partial\theta_i}\right]^{\alpha}_{\gamma} \\
\left[\frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta_i}\right]^{\alpha}_{\beta} &= \left[\mathcal{H}_{x'y'} + \mathcal{H}_{x'y}\right]^{\alpha}_{\beta\xi}\left[\frac{\partial\bar{y}}{\partial\theta_i}\right]^{\xi} + \left[\mathcal{H}_{x'x'} + \mathcal{H}_{x'x}\right]^{\alpha}_{\beta\delta}\left[\frac{\partial\bar{x}}{\partial\theta_i}\right]^{\delta} + \left[\frac{\partial\mathcal{H}_{x'}}{\partial\theta_i}\right]^{\alpha}_{\beta} \\
\left[\frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i}\right]^{\alpha}_{\gamma} &= \left[\mathcal{H}_{y'y'} + \mathcal{H}_{y'y}\right]^{\alpha}_{\gamma\xi}\left[\frac{\partial\bar{y}}{\partial\theta_i}\right]^{\xi} + \left[\mathcal{H}_{y'x'} + \mathcal{H}_{y'x}\right]^{\alpha}_{\gamma\delta}\left[\frac{\partial\bar{x}}{\partial\theta_i}\right]^{\delta} + \left[\frac{\partial\mathcal{H}_{y'}}{\partial\theta_i}\right]^{\alpha}_{\gamma}, \qquad \text{(B.3)}
\end{aligned}$$

where $\alpha = 1, \ldots, n$, $\gamma, \xi = 1, \ldots, n_y$, $\beta, \delta = 1, \ldots, n_x$. We can apply $\frac{\partial\bar{x}}{\partial\theta}$ and $\frac{\partial\bar{y}}{\partial\theta}$, the result in the last subsection here.

Stack the unknowns as $\begin{bmatrix} \frac{\partial g_x}{\partial\theta_i} \\ \frac{\partial h_x}{\partial\theta_i} \end{bmatrix}$, therefore for each element θ_i of θ we will solve a Sylvester equation,

$$\begin{bmatrix} \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} \\ \frac{\mathrm{d}\mathcal{H}_{y}}{\mathrm{d}\theta_i} \\ \frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta_i} \\ \frac{\mathrm{d}\mathcal{H}_{x}}{\mathrm{d}\theta_i} \end{bmatrix}^T \begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix} + \begin{bmatrix} \mathcal{H}_y & \mathcal{H}_{x'} + \mathcal{H}_{y'} g_x \end{bmatrix} \begin{bmatrix} \frac{\partial g_x}{\partial\theta_i} \\ \frac{\partial h_x}{\partial\theta_i} \end{bmatrix} + \begin{bmatrix} \mathcal{H}_{y'} & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial g_x}{\partial\theta_i} \\ \frac{\partial h_x}{\partial\theta_i} \end{bmatrix} h_x = 0.$$

B.3 Second-order Results

We are interested in $\frac{\partial g_{xx}}{\partial\theta}, \frac{\partial h_{xx}}{\partial\theta}, \frac{\partial g_{\sigma\sigma}}{\partial\theta}, \frac{\partial h_{\sigma\sigma}}{\partial\theta}$.

We differentiate the Sylvester equation (A.5) here (which yields a Sylvester equation itself): for each element θ_i of θ,

$$\begin{gathered}
AXB + DX + C = 0 \\
\Downarrow \\
A\frac{\partial X}{\partial\theta_i}B + D\frac{\partial X}{\partial\theta_i} + \left[\frac{\partial A}{\partial\theta_i}XB + AX\frac{\partial B}{\partial\theta_i} + \frac{\partial D}{\partial\theta_i}X + \frac{\partial C}{\partial\theta_i}\right] = 0. \qquad \text{(B.4)}
\end{gathered}$$

This equation has a similar form to (B.4), and can be solved with a similar algorithm. Notice that these coefficients A, B, D don't change across different θ_i, hence we solve them in parallel.

We derive the variables above in (B.4) by applying chain rules,

$$\frac{\partial A}{\partial \theta_i} = \left[\begin{array}{cc} \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} & 0 \end{array} \right] \tag{B.5}$$

$$\frac{\partial B}{\partial \theta_i} = \frac{\partial h_x}{\partial \theta_i} \otimes h_x + h_x \otimes \frac{\partial h_x}{\partial \theta_i} \tag{B.6}$$

$$\frac{\partial D}{\partial \theta_i} = \left[\begin{array}{cc} \frac{\mathrm{d}\mathcal{H}_{y}}{\mathrm{d}\theta_i} & \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} g_x + \mathcal{H}_{y'} \frac{\partial g_x}{\partial \theta_i} + \frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta_i} \end{array} \right] \tag{B.7}$$

$$\begin{aligned}
\frac{\partial C_{j,\cdot}}{\partial \theta_i} = \operatorname{vec} & \left(\begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} h_x + g_x \frac{\partial h_x}{\partial \theta_i} \\ \frac{\partial g_x}{\partial \theta_i} \\ \frac{\partial h_x}{\partial \theta_i} \\ 0 \end{bmatrix}^T \Psi_j \begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix} \right) \\
+ \operatorname{vec} & \left(\begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix}^T \Psi_j \begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} h_x + g_x \frac{\partial h_x}{\partial \theta_i} \\ \frac{\partial g_x}{\partial \theta_i} \\ \frac{\partial h_x}{\partial \theta_i} \\ 0 \end{bmatrix} \right) \\
+ \operatorname{vec} & \left(\begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix}^T \frac{\mathrm{d}\Psi_j}{\mathrm{d}\theta_i} \begin{bmatrix} g_x h_x \\ g_x \\ h_x \\ I \end{bmatrix} \right)
\end{aligned} \tag{B.8}$$

where $j = 1, \ldots, n$ iterating on each equation of $\mathcal{H}$.

We compute the derivatives with the form $\frac{\mathrm{d}[\mathcal{H}_{ab}]_j}{\mathrm{d}\theta_i}$ in the last line of Equation (B.8) through total derivative which requires third-order derivatives of the original equation system,

$$\frac{\mathrm{d}\left[\mathcal{H}_{ab}\right]_j}{\mathrm{d}\theta_i} = \left\{ \left[\mathcal{H}_{aby'} + \mathcal{H}_{aby}\right]_{\xi} \left[\frac{\partial \bar{y}}{\partial \theta_i}\right]^{\xi} + \left[\mathcal{H}_{abx'} + \mathcal{H}_{abx}\right]_{\delta} \left[\frac{\partial \bar{x}}{\partial \theta_i}\right]^{\delta} \right\}_j + \frac{\partial \left[\mathcal{H}_{ab}\right]_j}{\partial \theta_i} \tag{B.9}$$

where $a, b \in [y', y, x', x]$, $\xi = 1, \ldots, n_y$, $\delta = 1, \ldots, n_x$. The combination, therefore, can be expressed as

$$\begin{aligned}\frac{\mathrm{d}\Psi_j}{\mathrm{d}\theta_i} = & \left[\frac{\partial \Psi_j}{\partial y'} + \frac{\partial \Psi_j}{\partial y}\right]_{\xi} \left[\frac{\partial \bar{y}}{\partial \theta_i}\right]^{\xi} \\ & + \left[\frac{\partial \Psi_j}{\partial x'} + \frac{\partial \Psi_j}{\partial x}\right]_{\delta} \left[\frac{\partial \bar{x}}{\partial \theta_i}\right]^{\delta} \\ & + \frac{\partial \Psi_j}{\partial \theta_i}\end{aligned} \tag{B.10}$$

For the equations with $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$ we know it satisfies Equation (A.7),

$$A \begin{bmatrix} g_{\sigma\sigma} \\ h_{\sigma\sigma} \end{bmatrix} + B = 0, \tag{B.11}$$

therefore for each θ_i in θ we take the differentiation on both sides of the equation,

$$\frac{\partial A}{\partial \theta_i} \begin{bmatrix} g_{\sigma\sigma} \\ h_{\sigma\sigma} \end{bmatrix} + A \begin{bmatrix} \frac{\partial g_{\sigma\sigma}}{\partial \theta_i} \\ \frac{\partial h_{\sigma\sigma}}{\partial \theta_i} \end{bmatrix} + \frac{\partial B}{\partial \theta_i} = 0. \tag{B.12}$$

Equation (B.12) is a linear equation in $\begin{bmatrix} \frac{\partial g_{\sigma\sigma}}{\partial \theta_i} \\ \frac{\partial h_{\sigma\sigma}}{\partial \theta_i} \end{bmatrix}$. We enumerate $j = 1, \ldots, n$ for the index of the equations.

$$\frac{\partial A}{\partial \theta_i} = \begin{bmatrix} \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} + \frac{\mathrm{d}\mathcal{H}_{y}}{\mathrm{d}\theta_i} & \frac{\mathrm{d}\mathcal{H}_{y'}}{\mathrm{d}\theta_i} g_x + \mathcal{H}_{y'} \frac{\partial g_x}{\partial \theta_i} + \frac{\mathrm{d}\mathcal{H}_{x'}}{\mathrm{d}\theta_i} \end{bmatrix} \tag{B.13}$$

$$
\begin{aligned}
\frac{\partial B^j}{\partial \theta_i} &= \text{sum}\left(\left[\begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} \\ 0 \end{bmatrix}^T \begin{bmatrix} \mathcal{H}_{y'y'} & \mathcal{H}_{y'x'} \\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'x'} \end{bmatrix}^j \begin{bmatrix} g_x \\ I \end{bmatrix}\right] \odot [\eta \Sigma \eta']\right) \\
&+ \text{sum}\left(\left[\begin{bmatrix} g_x \\ I \end{bmatrix}^T \begin{bmatrix} \mathcal{H}_{y'y'} & \mathcal{H}_{y'x'} \\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'x'} \end{bmatrix}^j \begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} \\ 0 \end{bmatrix}\right] \odot [\eta \Sigma \eta']\right) \\
&+ \text{sum}\left(\left[\begin{bmatrix} g_x \\ I \end{bmatrix}^T \frac{\mathrm{d}\begin{bmatrix} \mathcal{H}_{y'y'} & \mathcal{H}_{y'x'} \\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'x'} \end{bmatrix}^j}{\mathrm{d}\theta_i} \begin{bmatrix} g_x \\ I \end{bmatrix}\right] \odot [\eta \Sigma \eta']\right) \\
&+ \text{sum}\left(\left[\frac{\mathrm{d}\left[\mathcal{H}_{y'}\right]^j}{\mathrm{d}\theta_i} g_{xx} + \left[\mathcal{H}_{y'}\right]^j \frac{\partial g_{xx}}{\partial \theta_i}\right] \odot [\eta \Sigma \eta']\right) \\
&+ \text{sum}\left(\left[\begin{bmatrix} g_x \\ I \end{bmatrix}^T \begin{bmatrix} \mathcal{H}_{y'y'} & \mathcal{H}_{y'x'} \\ \mathcal{H}_{x'y'} & \mathcal{H}_{x'x'} \end{bmatrix}^j \begin{bmatrix} g_x \\ I \end{bmatrix} + \left[\mathcal{H}_{y'}\right]^j g_{xx}\right] \odot \left[\eta \frac{\partial \Sigma}{\partial \theta_i} \eta'\right]\right) \qquad \text{(B.14)}
\end{aligned}
$$

or, for $\frac{\partial B^j}{\partial \theta_i}$, if we use the same matrix for $\mathcal{H}_{..}$ as in Equation (B.8),

$$
\begin{aligned}
\frac{\partial B^j}{\partial \theta_i} &= \text{sum}\left(\left(\begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \Psi_j \begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix} + \begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix}^T \Psi_j \begin{bmatrix} \frac{\partial g_x}{\partial \theta_i} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix}^T \frac{\mathrm{d}\Psi_j}{\mathrm{d}\theta_i} \begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix}\right.\right. \\
&\left.\left. + \frac{\mathrm{d}\left[\mathcal{H}_{y'}\right]^j}{\mathrm{d}\theta_i} g_{xx} + \left[\mathcal{H}_{y'}\right]^j \frac{\partial g_{xx}}{\partial \theta_i}\right) \odot [\eta \Sigma \eta']\right) \\
&+ \text{sum}\left(\left(\begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix}^T \Psi_j \begin{bmatrix} g_x \\ 0 \\ I \\ 0 \end{bmatrix} + \left[\mathcal{H}_{y'}\right]^j g_{xx}\right) \odot \left[\eta \frac{\partial \Sigma}{\partial \theta_i} \eta'\right]\right) \qquad \text{(B.15)}
\end{aligned}
$$

since $\Sigma = \Gamma\Gamma'$, we have $\frac{\partial \Sigma}{\partial \theta_i} = \frac{\partial \Gamma}{\partial \theta_i}\Gamma' + \Gamma\left(\frac{\partial \Gamma}{\partial \theta_i}\right)'$.

Here for $\frac{\mathrm{d}\left[\mathcal{H}_{y'}\right]^j}{\mathrm{d}\theta_i} g_{xx}$ and $\left[\mathcal{H}_{y'}\right]^j \frac{\partial g_{xx}}{\partial \theta_i}$ we first squeeze the second and the third dimensions of g_{xx} matrix, and then reshape the result coming from matrix multiplication.

Appendix C Sequential Solution and Derivative

Given the previous sections Appendices A and B this section derives the $\{\hat{x}_t, \hat{y}_t\}_{t=1}^T$ and $\frac{d\hat{x}_t}{d\theta}, \frac{d\hat{y}_t}{d\theta}, \frac{d\hat{x}_t}{dx_0}, \frac{d\hat{y}_t}{dx_0}$, and $\frac{d\hat{x}_t}{d\epsilon}, \frac{d\hat{y}_t}{d\epsilon}$ for all $t = 1, \dots T$. These are conditional on a particular ϵ, θ, and $\hat{x}_0$.

C.1 Solution

The system could be either linear or non-linear. For non-linear systems we need to apply pruning by shutting down the higher-order effect of the shocks. This is to preserve the monotonicity of likelihood in shocks.

First-order (linear) ones:

$$\hat{y}_t = g_x \hat{x}_t \tag{C.1}$$
$$\hat{x}_{t+1} = h_x \hat{x}_t + \eta \epsilon_{t+1} \tag{C.2}$$

Second-order (non-linear) ones ($i = 1, \dots, n_y$ and $j = 1, \dots, n_x$):

$$\hat{x}^f_{t+1} = h_x \hat{x}^f_t + \eta \epsilon_{t+1} \tag{C.3}$$
$$[\hat{y}_t]^i = [g_x \hat{x}_t]^i + \frac{1}{2} \left(\hat{x}^f_t\right)^T [g_{xx}]^i \hat{x}^f_t + \frac{1}{2} [g_{\sigma\sigma}]^i \tag{C.4}$$
$$[\hat{x}_{t+1}]^j = [h_x \hat{x}_t]^j + \frac{1}{2} \left(\hat{x}^f_t\right)^T [h_{xx}]^j \hat{x}^f_t + \frac{1}{2} [h_{\sigma\sigma}]^j + [\eta \epsilon_{t+1}]^j \tag{C.5}$$
$$\hat{x}^f_0 = \hat{x}_0 \tag{C.6}$$

C.2 Derivatives

Parameters We derive $\frac{\partial \hat{x}_t}{\partial \theta_i}, \frac{\partial \hat{y}_t}{\partial \theta_i}$.

$$\begin{aligned}
\frac{\partial \hat{x}_t}{\partial \theta_i} &= \frac{\partial \left[h\left(\hat{x}_{t-1}; \theta\right) + \eta \epsilon_t\right]}{\partial \theta_i} \\
&= h_1\left(\hat{x}_{t-1}; \theta\right) \frac{\partial \hat{x}_{t-1}}{\partial \theta_i} + h_{2,i}\left(\hat{x}_{t-1}; \theta\right) \\
&= h_1\left(\hat{x}_{t-1}; \theta\right) h_1\left(\hat{x}_{t-2}; \theta\right) \frac{\partial \hat{x}_{t-2}}{\partial \theta_i} + h_1\left(\hat{x}_{t-1}; \theta\right) h_{2,i}\left(\hat{x}_{t-2}; \theta\right) + h_{2,i}\left(\hat{x}_{t-1}; \theta\right) \\
&\vdots \\
&= \sum_{j=0}^{t-1} \left[\prod_{k=j+1}^{t-1} h_1\left(\hat{x}_k; \theta\right)\right] h_{2,i}\left(\hat{x}_j; \theta\right)
\end{aligned} \tag{C.7}$$

$$\begin{aligned}\frac{\partial \hat{y}_t}{\partial \theta_i} &= \frac{\partial \left[g\left(\hat{x}_t;\theta\right)\right]}{\partial \theta_i} \\ &= g_1\left(\hat{x}_t;\theta\right)\frac{\partial \hat{x}_t}{\partial \theta_i} + g_{2,i}\left(\hat{x}_t;\theta\right) \qquad \text{(C.8)}\end{aligned}$$

In first-order (linear) case, we use the following recursion for $\frac{\partial \hat{x}_t}{\partial \theta_i}, \frac{\partial \hat{y}_t}{\partial \theta_i}$,

$$\frac{\partial \hat{x}_{t+1}}{\partial \theta_i} = h_x \frac{\partial \hat{x}_t}{\partial \theta_i} + \frac{\partial h_x}{\partial \theta_i}\hat{x}_t \qquad \text{(C.9)}$$

$$\frac{\partial \hat{y}_t}{\partial \theta_i} = g_x \frac{\partial \hat{x}_t}{\partial \theta_i} + \frac{\partial g_x}{\partial \theta_i}\hat{x}_t \qquad \text{(C.10)}$$

$$\frac{\partial \hat{x}_0}{\partial \theta_i} = 0 \qquad \text{(C.11)}$$

In second-order (nonlinear) case with pruning, we use the following recursion,

$$\frac{\partial \hat{x}^f_{t+1}}{\partial \theta_i} = h_x \frac{\partial \hat{x}^f_t}{\partial \theta_i} + \frac{\partial h_x}{\partial \theta_i}\hat{x}^f_t \qquad \text{(C.12)}$$

$$\frac{\partial \hat{x}^f_0}{\partial \theta_i} = 0 \qquad \text{(C.13)}$$

$$\begin{aligned}\left[\frac{\partial \hat{x}_{t+1}}{\partial \theta_i}\right]^k &= \left[h_x \frac{\partial \hat{x}_t}{\partial \theta_i}\right]^k + \left[\frac{\partial h_x}{\partial \theta_i}\hat{x}_t\right]^k \\ &+ \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \theta_i}\right)^T \left[h_{xx}\right]^k \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T \left[\frac{\partial h_{xx}}{\partial \theta_i}\right]^k \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T \left[h_{xx}\right]^k \frac{\partial \hat{x}^f_t}{\partial \theta_i} + \frac{1}{2}\left[\frac{\partial h_{\sigma\sigma}}{\partial \theta_i}\right]^k \end{aligned} \qquad \text{(C.14)}$$

$$\begin{aligned}\left[\frac{\partial \hat{y}_t}{\partial \theta_i}\right]^j &= \left[g_x \frac{\partial \hat{x}_t}{\partial \theta_i}\right]^j + \left[\frac{\partial g_x}{\partial \theta_i}\hat{x}_t\right]^j \\ &+ \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \theta_i}\right)^T \left[g_{xx}\right]^j \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T \left[\frac{\partial g_{xx}}{\partial \theta_i}\right]^j \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T \left[g_{xx}\right]^j \frac{\partial \hat{x}^f_t}{\partial \theta_i} + \frac{1}{2}\left[\frac{\partial g_{\sigma\sigma}}{\partial \theta_i}\right]^j \end{aligned} \qquad \text{(C.15)}$$

$$\left[\frac{\partial \hat{x}_0}{\partial \theta_i}\right]^k = 0 \qquad \text{(C.16)}$$

where $k = 1, \ldots, n_x$, $j = 1, \ldots, n_y$.

Shocks From the chain rule we have

$$\begin{aligned}\frac{\partial \hat{x}_t}{\partial \epsilon_i} &= \frac{\partial h\left(\hat{x}_{t-1}\right)}{\partial \hat{x}_{t-1}} \frac{\partial \hat{x}_{t-1}}{\partial \epsilon_i} \\ &= \left(\prod_{j=i}^{t-1} \frac{\partial h\left(\hat{x}_j\right)}{\partial \hat{x}_j}\right) \eta \qquad \text{(C.17)} \\ \frac{\partial \hat{y}_t}{\partial \epsilon_i} &= \frac{\partial g\left(\hat{x}_t\right)}{\partial \hat{x}_t} \frac{\partial \hat{x}_t}{\partial \epsilon_i} \\ &= \frac{\partial g\left(\hat{x}_t\right)}{\partial \hat{x}_t} \left(\prod_{j=i}^{t-1} \frac{\partial h\left(\hat{x}_j\right)}{\partial \hat{x}_j}\right) \eta \qquad \text{(C.18)}\end{aligned}$$

In first-order (linear) case, we use the following recursions for $\frac{\partial \hat{x}_t}{\partial \epsilon_i}$ and $\frac{\partial \hat{y}_t}{\partial \epsilon_i}$ where $t \geqslant i$,

$$\frac{\partial \hat{x}_{t+1}}{\partial \epsilon_i} = h_x \frac{\partial \hat{x}_t}{\partial \epsilon_i} \qquad \text{(C.19)}$$

$$\frac{\partial \hat{y}_t}{\partial \epsilon_i} = g_x \frac{\partial \hat{x}_t}{\partial \epsilon_i} \qquad \text{(C.20)}$$

$$\frac{\partial \hat{x}_i}{\partial \epsilon_i} = \eta \qquad \text{(C.21)}$$

For second-order (non-linear) pruning case, we use the following recursions for $\frac{\partial \hat{x}_t}{\partial \epsilon_i}$ and $\frac{\partial \hat{y}_t}{\partial \epsilon_i}$ where $t \geqslant i$,

$$\frac{\partial \hat{x}^f_{t+1}}{\partial \epsilon_i} = h_x \frac{\partial \hat{x}^f_t}{\partial \epsilon_i} \qquad \text{(C.22)}$$

$$\left[\frac{\partial \hat{y}_t}{\partial \epsilon_i}\right]^j = \left[g_x \frac{\partial \hat{x}_t}{\partial \epsilon_i}\right]^j + \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \epsilon_i}\right)^T [g_{xx}]^j \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T [g_{xx}]^j \frac{\partial \hat{x}^f_t}{\partial \epsilon_i} \qquad \text{(C.23)}$$

$$\left[\frac{\partial \hat{x}_{t+1}}{\partial \epsilon_i}\right]^k = \left[h_x \frac{\partial \hat{x}_t}{\partial \epsilon_i}\right]^k + \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \epsilon_i}\right)^T [h_{xx}]^k \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T [h_{xx}]^k \frac{\partial \hat{x}^f_t}{\partial \epsilon_i} \qquad \text{(C.24)}$$

$$\frac{\partial \hat{x}^f_i}{\partial \epsilon_i} = \eta \qquad \text{(C.25)}$$

$$\frac{\partial \hat{x}_i}{\partial \epsilon_i} = \eta \qquad \text{(C.26)}$$

Initial Condition And apply chain rule we have

$$\frac{\partial \hat{x}_t}{\partial \hat{x}_0} = \frac{\partial \left[h\left(\hat{x}_{t-1};\theta\right) + \eta\epsilon_t\right]}{\partial \hat{x}_0} = \frac{\partial h\left(\hat{x}_{t-1};\theta\right)}{\partial \hat{x}_{t-1}} \frac{\partial \hat{x}_{t-1}}{\partial \hat{x}_0} \tag{C.27}$$

$$\frac{\partial \hat{y}_t}{\partial \hat{x}_0} = \frac{\partial g\left(\hat{x}_t;\theta\right)}{\partial \hat{x}_0} = \frac{\partial g\left(\hat{x}_t;\theta\right)}{\partial \hat{x}_t} \frac{\partial \hat{x}_t}{\partial \hat{x}_0} \tag{C.28}$$

where for linear case,

$$\frac{\partial \hat{y}_t}{\partial \hat{x}_0} = g_x \frac{\partial \hat{x}_t}{\partial \hat{x}_0} \tag{C.29}$$

$$\frac{\partial \hat{x}_{t+1}}{\partial \hat{x}_0} = h_x \frac{\partial \hat{x}_t}{\partial \hat{x}_0} \tag{C.30}$$

$$\frac{\partial \hat{x}_0}{\partial \hat{x}_0} = I \tag{C.31}$$

and for second-order case,

$$\frac{\partial \hat{x}^f_{t+1}}{\partial \hat{x}_0} = h_x \frac{\partial \hat{x}^f_t}{\partial \hat{x}_0} \tag{C.32}$$

$$\frac{\partial \hat{x}^f_0}{\partial \hat{x}_0} = I \tag{C.33}$$

$$\left[\frac{\partial \hat{x}_{t+1}}{\partial \hat{x}_0}\right]^k = \left[h_x \frac{\partial \hat{x}_t}{\partial \hat{x}_0}\right]^k + \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \hat{x}_0}\right)^T [h_{xx}]^k \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T [h_{xx}]^k \frac{\partial \hat{x}^f_t}{\partial \hat{x}_0} \tag{C.34}$$

$$\left[\frac{\partial \hat{y}_t}{\partial \hat{x}_0}\right]^j = \left[g_x \frac{\partial \hat{x}_t}{\partial \hat{x}_0}\right]^j + \frac{1}{2}\left(\frac{\partial \hat{x}^f_t}{\partial \hat{x}_0}\right)^T [g_{xx}]^j \hat{x}^f_t + \frac{1}{2}\left(\hat{x}^f_t\right)^T [g_{xx}]^j \frac{\partial \hat{x}^f_t}{\partial \hat{x}_0} \tag{C.35}$$

$$\frac{\partial \hat{x}_0}{\partial \hat{x}_0} = I \tag{C.36}$$

Appendix D Kalman filter and its derivatives

For linear Gaussian state space models, we can evaluate the series of posterior distributions of the latent states, and hence the marginal likelihood by Kalman filter. We compute the derivatives associated with this whole process as well.

Kalman filter To run Kalman filter, in addition to the initial condition $\hat{x}_0$ we need the prior covariance matrix of it, P_0. A natural choice of P_0 is the solution to the Lyapunov equation

$$h_x P_0 h_x' - P_0 + \eta \Sigma \eta' = 0 \tag{D.1}$$

For notation simplicity, we define $G = Q \begin{bmatrix} g_x \\ I \end{bmatrix}$. Let the constant loadings of steady state values be H. Then the formula to update with time period t data follows the recursion,

$$\begin{aligned} x_{t|t-1} &= h_x x_{t-1} && \text{(D.2)} \\ P_{t|t-1} &= h_x P_{t-1} h_x' + \eta \Sigma \eta' && \text{(D.3)} \\ z_t &= G x_{t|t-1} + H \bar{u} && \text{(D.4)} \\ V_t &= G P_{t|t-1} G' + \Omega && \text{(D.5)} \\ z_t &\sim \mathcal{N}\left(z_t, V_t\right) && \text{(D.6)} \\ x_t &= x_{t|t-1} + P_{t|t-1} G' V_t^{-1} \left(z_t - z_t\right) && \text{(D.7)} \\ P_t &= P_{t|t-1} - P_{t|t-1}' G' V_t^{-1} G P_{t|t-1} && \text{(D.8)} \end{aligned}$$

Derivatives By differentiating the Lyapunov equation we compute $\frac{\partial P_0}{\partial \theta_i}$ from another Lyapunov equation,

$$h_x \frac{\partial P_0}{\partial \theta_i} h_x' - \frac{\partial P_0}{\partial \theta_i} + \left(\frac{\partial h_x}{\partial \theta_i} P_0 h_x' + h_x P_0 \frac{\partial h_x'}{\partial \theta_i} + \eta \frac{\partial \Sigma}{\partial \theta_i} \eta' \right) = 0$$

We also track the derivatives of each of the quantities above, with respect to the parameters that we are interested in, θ.

$$\frac{\partial x_{t|t-1}}{\partial \theta_i} = \frac{\partial h_x}{\partial \theta_i} x_{t-1} + h_x \frac{\partial x_{t-1}}{\partial \theta_i} \tag{D.9}$$

$$\frac{\partial P_{t|t-1}}{\partial \theta_i} = \frac{\partial h_x}{\partial \theta_i} P_{t-1} h_x' + h_x \frac{\partial P_{t-1}}{\partial \theta_i} h_x + h_x P_{t-1} \left(\frac{\partial h_x}{\partial \theta_i} \right)^T + \eta \frac{\partial \Sigma}{\partial \theta_i} \eta' \tag{D.10}$$

$$\frac{\partial z_t}{\partial \theta_i} = \frac{\partial G}{\partial \theta_i} x_{t|t-1} + G \frac{\partial x_{t|t-1}}{\partial \theta_i} + H \frac{\partial \bar{u}}{\partial \theta_i} \tag{D.11}$$

$$\frac{\partial V_t}{\partial \theta_i} = \frac{\partial G}{\partial \theta_i} P_{t|t-1} G' + G \frac{\partial P_{t|t-1}}{\partial \theta_i} G' + G P_{t|t-1} \left(\frac{\partial G}{\partial \theta_i} \right)^T \tag{D.12}$$

$$\tilde{z}_t \sim \mathcal{N}\left(z_t, V_t\right)$$

$$\begin{aligned} \frac{\partial x_t}{\partial \theta_i} &= \frac{\partial x_{t|t-1}}{\partial \theta_i} + \frac{\partial P_{t|t-1}}{\partial \theta_i} G' V_t^{-1} \left(\tilde{z}_t - z_t\right) + P_{t|t-1} \left(\frac{\partial G}{\partial \theta_i} \right)^T V_t^{-1} \left(\tilde{z}_t - z_t\right) \\ &\quad - P_{t|t-1} G' V_t^{-1} \frac{\partial V_t}{\partial \theta_i} V_t^{-1} \left(\tilde{z}_t - z_t\right) - P_{t|t-1} G' V_t^{-1} \frac{\partial z_t}{\partial \theta_i} \end{aligned} \tag{D.13}$$

$$\begin{aligned} \frac{\partial P_t}{\partial \theta_i} &= \frac{\partial P_{t|t-1}}{\partial \theta_i} - \frac{\partial P_{t|t-1}}{\partial \theta_i}' G' V_t^{-1} G P_{t|t-1} - P_{t|t-1}' \left(\frac{\partial G}{\partial \theta_i} \right)^T V_t^{-1} G P_{t|t-1} \\ &\quad + P_{t|t-1}' G' V_t^{-1} \frac{\partial V_t}{\partial \theta_i} V_t^{-1} G P_{t|t-1} - P_{t|t-1}' G' V_t^{-1} \frac{\partial G}{\partial \theta_i} P_{t|t-1} - P_{t|t-1}' G' V_t^{-1} G \frac{\partial P_{t|t-1}}{\partial \theta_i} \end{aligned} \tag{D.14}$$