Research Proposal

Xiaonan Peng

December 25, 2023

Abstract

This research proposal aims to review the current methodologies for solving and estimating Dynamic Stochastic General Equilibrium (DSGE) models. We will discuss the traditional solution techniques, parameter estimation using Kalman and particle filters, and the application of Markov Chain Monte Carlo (MCMC) and Hamiltonian Monte Carlo (HMC) methods. Additionally, we will explore the potential of deep learning approaches in this context. Finally, we will discuss the code testing plan for our porject.

1 Introduction

Dynamic Stochastic General Equilibrium (DSGE) models are a central tool in macroeconomic research, providing a coherent framework for analyzing the behavior of economies over time under the influence of random shocks. These models encapsulate the dynamic interactions between multiple economic agents, grounded in microeconomic principles, and subject to temporal stochasticity. As such, DSGE models have been extensively utilized for policy analysis, forecasting, and structural examination of economic data.

The estimation of DSGE models, however, poses significant computational challenges due to the complexity inherent in their structure. Classic estimation methods have traditionally relied on Kalman and particle filters to handle latent variables and to evaluate the likelihood function, a process often constrained by high computational costs and limited by the curse of dimensionality. The Bayesian framework, with its reliance on Markov Chain Monte Carlo (MCMC) methods, has provided a way to address some of these issues. Notably, the Random Walk Metropolis-Hastings (RWMH) algorithm has been widely adopted, despite its inefficiencies in high-dimensional parameter spaces.

Recent advances [1] in computational techniques, including differentiable programming and Hamiltonian Monte Carlo (HMC) sampling, offer promising avenues to overcome these limitations. Differentiable programming allows for the automatic differentiation of model solutions, which is crucial for gradient-based optimization and sampling methods such as HMC. Such methods can traverse the posterior distribution more efficiently and are particularly advantageous for models characterized by a large number of parameters.

More recently, the advent of deep learning has opened new avenues for solving and estimating DSGE models [6]. These data-driven methods promise to handle high-dimensional and complex non-linear relationships more adeptly than traditional techniques.

This project aims to explore and reproduce the methodology proposed in seminal papers that have opened this avenue of research, such as "Differentiable State-Space Models and Hamiltonian Monte Carlo Estimation" and "Deep learning for solving dynamic economic models".

By building upon these foundational works, this research will delve into the practical application of differentiable programming in DSGE model estimation, harness the efficiency of HMC in the Bayesian estimation framework, and explore the potential of deep learning techniques to further enhance the solution and estimation of DSGE models. The ultimate goal is to develop a robust, scalable, and efficient methodology for the estimation of DSGE models that can be adopted widely within the field of macroeconomics.

This proposal outlines a comprehensive review of the methods for solving and estimating DSGE models, with a special emphasis on the RBC model. We aim to not only elucidate the traditional and contemporary techniques but also to explore the potential of deep learning in this domain, providing a bridge between macroeconomic theory and the cutting edge of computational methods.

2 Literature Review

2.1 Classical Methods to Solve DSGE Models

Dynamic stochastic general equilibrium (DSGE) models, due to their diverse formulations and uses, present a unique challenge for solution methods. Classical techniques such as those put forth by Kydland & Prescott (1982) [5] for real business cycle (RBC) models, and later by Woodford (2003) [7] and Christiano et al. (2005) [2] for New Keynesian models, have relied on log-linearization around a steady state or the application of perturbation methods to approximate the policy functions of the agents within the model. These approaches, while useful for models with mild nonlinearities and deviations from the steady state, struggle with more complex models that exhibit significant nonlinear dynamics or when evaluating models at points far from the steady state.

2.2 Estimation with Filtering

The estimation of DSGE models often requires the use of filtering techniques to deal with unobserved latent variables. The Kalman filter has been a workhorse for models that can be linearized, while the particle filter has been employed for nonlinear models with non-Gaussian shocks (Fernández-Villaverde & Rubio-Ramírez, 2007 [4]). Despite their widespread use, these filters come with limitations: the Kalman filter requires linear and Gaussian assumptions, and the particle filter can be computationally intensive and difficult to tune.

2.3 Estimation with MCMC and HMC

The Bayesian estimation of DSGE models using Markov chain Monte Carlo (MCMC) methods has advanced considerably with the advent of computational improvements. Traditional MCMC methods, such as the Random Walk Metropolis-Hastings, have been the mainstay for their simplicity and general applicability. However, the efficiency of such samplers degrades in higher dimensions, as indicated by Fernández-Villaverde (2010) [3].

The Hamiltonian Monte Carlo (HMC) method has emerged as a powerful alternative [1], exploiting gradient information to make more informed proposals, thus reducing the correlation between successive samples and improving the effective sample size, even in high-dimensional parameter spaces. The ability of HMC to efficiently explore the posterior distribution has led to its increasing popularity in DSGE model estimation.

2.4 Deep Learning for DSGE

The intersection of deep learning (DL) with DSGE model estimation is a nascent field with significant potential. Deep learning techniques, as reviewed by Goodfellow et al. (2016), have shown remarkable success in areas such as image and speech recognition, which suggests that they could be equally transformative for solving and estimating high-dimensional and complex DSGE models. By formulating economic dynamics as nonlinear regression problems, deep learning can approximate value and decision functions of dynamic models, offering a promising direction for future research in computational economics [6].

3 Methodology

This research aims to advance the methodology for estimating DSGE models by leveraging recent computational techniques. We will illustrate our approach using a Real Business Cycle (RBC) model as a foundation. The RBC model, being one of the simplest forms of DSGE models, allows us to clearly demonstrate the application of classical solution methods, Kalman filter estimation, MCMC challenges, and the potential benefits of incorporating deep learning techniques.

3.1 Solution of the RBC Model

The RBC model describes an economy with a representative agent who maximizes utility over consumption and leisure, subject to a production function with stochastic shocks. The model is typically represented by a system of equations that includes a resource constraint, an Euler equation for consumption, and a law of motion for capital. The representative agent's problem can be written as:

$$\max_{\{C_t, L_t, K_{t+1}\}} E_0 \sum_{t=0}^{\infty} \beta^t U(C_t, 1 - L_t) \tag{1}$$

$$\text{s.t.} \quad C_t + I_t = Y_t = A_t F(K_t, L_t), \tag{2}$$

$$K_{t+1} = (1 - \delta) K_t + I_t, \tag{3}$$

$$\log A_{t+1} = \rho \log A_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

Where C_t is consumption, L_t is labor, K_t is capital, I_t is investment, A_t is the productivity level, U is the utility function, β is the discount factor, δ is the depreciation rate, ρ is the persistence of the productivity shock, and ϵ_{t+1} is the innovation to productivity.

To solve the RBC model, one might apply a log-linearization technique around the steady state or use perturbation methods to approximate the policy functions. These methods yield a set of linear equations that can be solved to obtain decision rules for consumption and investment as functions of the state variables.

3.2 Estimation with Kalman Filter and MCMC

For parameter estimation, the state-space form of the RBC model is used. The state-space representation includes the measurement equations, which relate observable variables to the states, and transition equations, which describe the evolution of the states over time:

$$y_t = H x_t + v_t, \quad v_t \sim \mathcal{N}(0, R), \tag{5}$$

$$x_{t+1} = F x_t + w_t, \quad w_t \sim \mathcal{N}(0, Q), \tag{6}$$

Where y_t is the vector of observables, x_t is the vector of state variables, H is the measurement matrix, F is the transition matrix, and v_t and w_t are normally distributed measurement and state disturbances with covariance matrices R and Q, respectively.

The Kalman filter provides a recursive method to infer the unobserved state variables, x_t, from the observables, y_t, and to evaluate the likelihood function of the model given the parameters. The Kalman filter, however, is most effective for linear Gaussian models. When the RBC model is nonlinear or features non-Gaussian shocks, the particle filter may be used, but at a higher computational cost.

Markov chain Monte Carlo (MCMC) methods can then be employed to sample from the posterior distribution of the model parameters. The Metropolis-Hastings algorithm is commonly used for this purpose:

$$\theta^{(i+1)} = \begin{cases} \theta^* & \text{with probability } \min\left(1, \frac{p(\theta^*|y)p(y|\theta^*)}{p(\theta^{(i)}|y)p(y|\theta^{(i)})}\right), \\ \theta^{(i)} & \text{otherwise,} \end{cases} \tag{7}$$

$$\theta^* \sim q(\theta|\theta^{(i)}), \tag{8}$$

Where θ denotes the vector of model parameters, $p(\theta|y)$ is the posterior "'latex distribution, $p(y|\theta)$ is the likelihood function, and $q(\theta|\theta^{(i)})$ is the proposal distribution.

The main difficulties with MCMC methods, particularly in high-dimensional parameter spaces, arise from their random-walk behavior and the associated high autocorrelation between samples. This often leads to a poor exploration of the posterior distribution and requires a large number of draws to achieve convergence.

The Hamiltonian Monte Carlo (HMC) method enhances traditional MCMC by utilizing gradient information to propose new states, which helps in exploring the posterior distribution more efficiently. This is particularly beneficial for high-dimensional parameter spaces, where RWMH can become inefficient.

In the context of the RBC model, HMC can be applied to sample from the posterior distribution by first rewriting the canonical MCMC as a Hamiltonian system, where the potential energy is the negative log-posterior and the kinetic energy is typically a quadratic form of auxiliary momentum variables. The joint distribution of parameters and momenta is then:

$$p(\theta, p) \propto \exp(-U(\theta) - K(p)), \tag{9}$$

where $U(\theta) = -\log p(\theta|y)$ is the potential energy and $K(p) = \frac{1}{2}p^T M^{-1} p$ is the kinetic energy with mass matrix M and momentum p. The HMC algorithm proceeds by alternately updating the momentum and the parameters using Hamiltonian dynamics, which are numerically integrated using a leapfrog scheme:

$$p(t+\epsilon) = p(t) - \frac{\epsilon}{2}\nabla_\theta U(\theta(t)), \tag{10}$$

$$\theta(t+\epsilon) = \theta(t) + \epsilon M^{-1} p(t+\epsilon), \tag{11}$$

$$p(t+\epsilon) = p(t+\epsilon) - \frac{\epsilon}{2}\nabla_\theta U(\theta(t+\epsilon)), \tag{12}$$

where ϵ is the step size. After a number of leapfrog steps, a Metropolis acceptance step is used to decide whether to accept the new state. The choice of step size ϵ and the number of leapfrog steps are crucial for the efficiency of HMC and need to be tuned for the specific model.

The implementation of HMC in the estimation of DSGE models with Python and TensorFlow will involve using automatic differentiation to compute the gradients required for the leapfrog integration, thus allowing for a more efficient exploration of the parameter space compared to traditional MCMC methods.

3.3 Incorporating Deep Learning into DSGE Estimation

Deep learning (DL) offers a novel approach to estimating DSGE models by approximating the policy functions directly from data. This is done by formulating the problem as a nonlinear regression:

$$\min_{\theta \in \Theta} E\left[\left(f(X_t, \theta) - Y_t \right)^2 \right], \tag{13}$$

where $f(X_t, \theta)$ represents the decision rules or policy functions parameterized by deep neural networks, X_t are state variables, Y_t are control variables, and Θ is the parameter space for the neural network.

The deep learning approach can be particularly beneficial for solving models that are highly nonlinear or for which the policy functions cannot be characterized analytically. It circumvents the need for linearization or perturbation methods, which may not be accurate in regions far from the steady state. The deep learning framework also allows for the integration of the estimation of shocks and latent variables, which can be challenging for traditional estimation methods:

$$\min_{\theta \in \Theta} E\left[\left(g(X_t, \epsilon_t, \theta) - Y_t \right)^2 \right], \tag{14}$$

where $g(X_t, \epsilon_t, \theta)$ includes the shock terms ϵ_t as part of the input to the neural network, alongside the state variables X_t.

By leveraging stochastic gradient descent and backpropagation, the deep learning model can be trained to minimize the discrepancy between the predicted and actual values of the control variables, thereby learning the implied policy functions and shock processes of the DSGE model.

4 Code Testing Plan

A robust code testing plan is crucial for ensuring the accuracy and stability of the numerical methods and machine learning algorithms used in this research. The following subsections describe the testing strategies that will be implemented throughout the development of the Python and TensorFlow codebase for solving DSGE models, implementing MCMC and HMC, and applying deep learning techniques.

4.1 Unit Testing

Unit tests will be written for each individual component of the code to ensure that each function and class behaves as expected in isolation. These tests will cover a range of scenarios, including standard use cases, boundary conditions, and edge cases.

- **DSGE Model Components**: Test the implementation of the utility functions, production functions, and shock processes.
- **Numerical Solvers**: Test the perturbation methods, filtering algorithms, and optimization routines for accuracy and convergence.
- **MCMC and HMC Algorithms**: Test the sampling algorithms to ensure they correctly implement the algorithms and produce samples with the expected properties.
- **Deep Learning Models**: Test the neural network architectures, loss functions, and training procedures to verify correct implementation and learning behavior.

4.2 Integration Testing

Once unit tests are passing, integration tests will be conducted to ensure that the various components of the codebase work together harmoniously.

- **Model Estimation Pipeline**: Test the end-to-end process from model specification to parameter estimation to validate that all components integrate correctly.
- **Data Flow**: Test the flow of data through the system, ensuring that inputs and outputs are correctly processed across different modules.

4.3 System Testing

System tests will be performed to evaluate the performance of the entire codebase, replicating the conditions under which the system will run in production.

- **Performance Testing**: Assess the efficiency and speed of the computation, particularly for large-scale DSGE models and deep learning training processes.
- **Stress Testing**: Determine the robustness of the system under high loads, such as when performing large numbers of MCMC/HMC iterations or training large neural networks.

5 References

References

[1] David Childers, Jesús Fernández-Villaverde, Jesse Perla, Christopher Rackauckas, and Peifan Wu. Differentiable state-space models and hamiltonian monte carlo estimation. Technical report, National Bureau of Economic Research, 2022.

[2] Lawrence J Christiano, Martin Eichenbaum, and Charles L Evans. Nominal rigidities and the dynamic effects of a shock to monetary policy. *Journal of political Economy*, 113(1):1–45, 2005.

[3] Jesús Fernández-Villaverde. The econometrics of dsge models. *SERIEs*, 1(1-2):3–49, 2010.

[4] Jesús Fernández-Villaverde, Juan F Rubio-Ramírez, Thomas J Sargent, and Mark W Watson. Abcs (and ds) of understanding vars. *American economic review*, 97(3):1021–1026, 2007.

[5] Finn E Kydland and Edward C Prescott. Time to build and aggregate fluctuations. *Econometrica: Journal of the Econometric Society*, pages 1345–1370, 1982.

[6] Lilia Maliar, Serguei Maliar, and Pablo Winant. Deep learning for solving dynamic economic models. *Journal of Monetary Economics*, 122:76–101, 2021.

[7] Michael Woodford. Optimal interest-rate smoothing. *The Review of Economic Studies*, 70(4):861–886, 2003.