A Machine Learning Approach to Simulation-Based MLE and Bayesian Estimation of Heterogeneous and Representative Agent Models

Cameron Fen*

December 6, 2023

This paper applies a simulation-based machine learning algorithm, Sequential Neural Posterior Estimation (SNPE), to the estimation of heterogeneous and representative agent macroeconomic models. Heterogeneous agent models are becoming the workhorse of modern macroeconomics but pose estimation challenges compared to their representative agent counterparts. The approach samples from the joint distribution of parameters and simulated data, then fits a conditional density estimator on the joint samples of the parameters conditional on the data to get the posterior. This algorithm has the following advantages: estimation is independent of the choice of solution method and the inner workings of the model, is faster and more accurate than current benchmark Bayesian algorithms, and simulation-based estimates are statistically efficient when well-specified. With 500 thousand iterations, the estimated posteriors are significantly higher quality than with 10 million Markov Chain Monte Carlo samples for the benchmark Smets-Wouters (2007) model. Additionally, results show improved estimation speed for Heterogeneous Agent New Keynesian (HANK) models. An empirical application demonstrates the effectiveness of this technique in estimating a heterogeneous agent model using both aggregate time series and cross-sectional micro-data.

JEL Codes: E27, E37, E47, C45, C11, C15
Keywords: Neural Networks, Bayesian Inference, Dynamic Estimation, Simulation-Based Estimators

*Cameron Fen is a PhD student at the University of Michigan, Ann Arbor, MI, 48104 (E-mail: camfen@umich.edu. Website: cameronfen.github.io.). The author thanks Siqi Liu, Xavier Bautista, Florian Gunsilius, David Childers, Eric Jang, John Leahy, and Toni Whited for helpful feedback and Alisdair McKay, Emil Lakkis, Mohammed Ait Lahcen, and Jeppe Druedahl for code. All errors are my own.

I Introduction

Macroeconomic models with heterogeneous agents allow for greater realism, but present computational challenges that make them difficult to estimate[1]. These include dealing with a high-dimensional state-space, matching both macro and micro data, and reproducing oddly shaped, multi-modal, and unidentified posteriors. In this paper, I apply a machine learning approach, Sequential Neural Posterior Estimation (SNPE), that can estimate most black box methods via simulations in a fast and accurate Maximum Likelihood Estimation (MLE) or Bayesian manner[2]. SNPE uses advances in machine learning conditional density estimation to handle high-dimensional spaces, to avoid difficult likelihood functions evaluations, and, in the Bayesian setting, to produce fast, accurate, and reproducible posteriors compared to benchmark procedures. With little modification, SNPE can estimate models solved via projection, perturbation, value function iteration, Reiter's method (Reiter, 2009), Winberry's method (Winberry, 2018), Time Iteration (Rendahl, 2017), the Sequence-Space Jacobian approach (Auclert et al., 2021), and many others.

I will discuss Bayesian estimation and, in particular, how SNPE works. Bayesian estimation uses Bayes' rule to determine the distribution of the parameters, θ, after seeing the data, X. This is the optimal updating rule which takes the likelihood weighted average of the prior to produce the posterior, $P(\theta|X)$. An issue across most Bayesian estimation is that the normalizing term, $P(X)$, that ensures the posterior sums to one, is intractable to calculate. Unlike other algorithms, SNPE is likelihood-free, so by assumption, one does not have access to the likelihood, $P(X|\theta)$, but can sample from it. SNPE proceeds by sampling from the joint distribution of the parameter space by first sampling from the prior and then from the likelihood/model induced by the prior sample. Then the algorithm uses a machine learning conditional density estimator to produce the distribution of the parameters conditional on the simulated data, $P(\theta|X)$. Conditioning the simulated data on the actual data produces the posterior.

1. For a review of heterogeneous agent solution methods with which estimation approaches must be compatible, see Reiter (2009), Winberry (2018) and Auclert et al. (2021). Other heterogeneous agent estimation techniques include Ahn et al. (2018), Liu and Plagborg-Møller (2023), and Kase, Melosi and Rottner (2022), but there is more work to be done.

2. A tutorial of the approach can be found at: `https://shorturl.at/pMQ18`

Compared to the baseline Bayesian algorithms like Metropolis-Hastings Markov Chain Monte Carlo (MCMC) and Hamiltonian Monte Carlo (HMC) (Childers et al., 2022), SNPE yields better posteriors with fewer iterations. See Section IVE. Likewise, all of the above algorithms, in addition to Sequential Monte Carlo (SMC) (Herbst and Schorfheide, 2014), require likelihood functions. With Heterogeneous Agent New Keynesian (HANK) models, the estimation algorithm allows for more accurate higher dimensional estimation with Reiter's method compared to alternatives (Ahn et al., 2018), (Bayer and Luetticke, 2020), It also improves estimation speed of models solved via Winberry's method by 13x. Theory (Chewi, 2023), and empirical results (see Section IVE) hint that SNPE is faster than MCMC especially with multi-modal distributions.

It is less natural, but one can use SNPE as an MLE estimator, which has advantages over other simulation-based methods. For a proof, see Proposition 5 in Section IIID. Compared to other simulation-based algorithms, SNPE can estimate by maximum likelihood on well specified models via simulation without much modification. Unlike competing techniques, SNPE can also simulation-base estimate structural models by adhering to the state-space structure of structural models. Competing methods require a cross-sectional approach where 200-300 time-step data is split into 200-300 data points with overlapping lags. SNPE can estimate a model using the entire time series as a single data point, which remains faithful to the true data-generating process of structural models.

In summary, compared to simulation-based models, SNPE can estimate complex and well-specified models in an efficent manner without the need of likellihood function evaluations. Compared to Bayesian algorithms, SNPE is a likelihood-free method that estimates posteriors faster and with more accuracy. For a more detailed discussion of the relative strengths and weaknesses of SNPE compared with alternatives, see Section IVI.

II Literature Review

This paper fits in the intersection of simulation-based structural estimation, Bayesian inference, and the solution and estimation of HANK models.

IIA Simulation-Based Estimation in Economics

Simulation-based estimation of models, structural or otherwise, has been a prominent approach in economics for many years. Originating from MSM (McFadden, 1989), simulation-based likelihood and inference techniques have been developed (Pakes and Pollard, 1989), (Duffie and Singleton, 1990) and improved to address the difficulty of efficient model estimation, (Gallant and Tauchen, 1996), (Lerman and Manski, 1981), (Hajivassiliou et al., 1997), (Haan and Uhlendorff, 2006), and (Gourieroux, Monfort and Renault, 1993). The closest method to SNPE is Kaji, Manresa and Pouliot (2020), which allows for efficient simulation-based estimation without modifications to the model like measurement error. Researchers have adopted these simulation-based approaches to estimate dynamic models, particularly in industrial organization (Keane and Wolpin, 1994), macroeconomics (Herbst and Schorfheide, 2015), and finance (Hennessy and Whited, 2007). As a simulation-based technique, SNPE builds upon all these methods. Advantages and disadvantages of SNPE compared to other simulation-based techniques are discussed in Section IVI, at the end of the Results Section.

IIB Bayesian Estimation

The baseline algorithm for Bayesian estimation is MCMC (Herbst and Schorfheide, 2015), along with extensions like HMC (Childers et al., 2022) and alternatives like SMC for estimation (Herbst and Schorfheide, 2014). The main simulation-based Bayesian estimation alternative compared with SNPE is Approximate Bayesian Computation (ABC) (Rubin, 1984), often used more in other fields. Furthermore, there has been some work on estimating reduced-form models with variational inference (Loaiza-Maya et al., 2022). A detailed comparison of all these models is contained in Section IVI, at the end of the Results Section.

IIC Heterogeneous Agent Models

This section will discuss improving speed and estimation approaches for Heterogeneous Agent models, which usually comes down to effective ways to deal with the heterogeneous agent distri-

bution. Some approaches advocate dimensionality reduction of the distributions space (Ahn et al., 2018) (Bayer and Luetticke, 2020). Other approaches advocate building a law of motion for the moments of the distribution (Winberry, 2018). Some approaches sidestep the use of a state-space entirely and build a linear relationship directly from the shocks to observed variables (Auclert et al., 2021). Still other approaches propose using neural networks in a projection fashion for solving (Han, Yang et al., 2021) and estimating (Kase, Melosi and Rottner, 2022) heterogeneous agent models. Additionally, we have work (Liu and Plagborg-Møller, 2023) that extends solution methods like Winberry (2018) to full information estimation of models like Khan and Thomas (2008) and Krusell and Smith (1998). More information of models solved via Reiter and Winberry, respectively, see Sections IVF and IVG. Additional details of these models can also found in Sections CF and CG in the Appendix, respectively.

III Background on SNPE and Bayesian Estimation

In recent years, Bayesian estimation has gained prominence as an approach for estimating economic models. This section sheds light on SNPE, the method proposed by this paper with advantages over the benchmark technique in economics, MCMC, which is discussed in Section AA of the Appendix. To provide a comprehensive understanding, a brief overview of Bayesian estimation is first presented. Then the SNPE algorithm is then discussed.

IIIA Bayesian Basics

Bayesian estimation attempts to find the posterior given the likelihood and prior distribution of a model using Bayes' rule:

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} \tag{1}$$

θ is the parameters I'm trying to estimate: discount rate, elasticity of substitution, capital share, etc. X is the data produced by the model: output, consumption, investment, etc.

More specifically: here $P(\theta|X)$ is the posterior, which is the objective. $P(X|\theta)$ is the likelihood and $P(\theta)$ is the prior. Approaches that perform Bayesian estimation, ranging from MCMC, variational inference, or simulation-based inference, concern techniques for calculating $P(\theta|X)$ without needing to calculate the partition function, $P(X)$. For details discussing the intuition behind MCMC, see Section AA of the Appendix.

IIIB Sequential Neural Posterior Estimation (SNPE)

At its core, SNPE is a simulation based approach for Bayesian estimation of the posterior distribution of the parameters $P(\theta|X)$. We know the prior, $P(\theta)$ and the SNPE algorithm assumes we can sample from the likelihood $P(X|\theta)$, which is a weaker assumption than MCMC. Nevertheless, even in situation where you can evaluate the likelihood, SNPE often converges faster and more accurately to the posterior. The algorithm proceeds in this way: first one draws a sample, θ, from the prior distribution, $P(\theta)$. Subsequently, simulated data, X, is generated from the model likelihood, $P(X|\theta)$. The concatenation of these two simulations results in samples from the joint distribution:

$$X, \theta \sim P(X, \theta) = P(X|\theta)P(\theta) \tag{2}$$

Note this is equivalent to sampling from the joint distribution numerator in Bayes' rule. These samples can then be used to estimate the posterior distribution, $P(\theta|X)$, through the utilization of either a conditional density estimator like a conditional mixture of Gaussians (Bishop, 1994), a normalizing flow density estimator (Mohammad-Djafari and Ayasso, 2009) or a Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). This section will discuss normalizing flow as an example of conditional density estimators for SNPE[3]. Finally, conditioning the estimated density,$P(\theta|X)$, on the real data, X', returns an estimate for the posterior: $P(\theta|X = X')$. Algorithm 1 is a cookbook description of SNPE:

3. Conditional mixture of Gaussians and GANs, are discussed in Section DA and DB, respectively, both in the Appendix

Algorithm 1: SNPE Algorithm

Input: Simulator $P(X|\theta)$, prior $P(\theta)$, data X', Flow $P_\phi(X|\theta)$, Rounds R, Samples S;

Initialize: Proposal $P^{(0)}(\theta) = p(\theta)$, data set D = {} empty to start out with; **for** $i \leftarrow 1$ **to** R **do**

  Sample $\theta^{(n)} \sim p^{(i-1)(\theta)}$ for $n = 1...S$ with Monte Carlo;

  Simulate $X^{(n)} \sim P(X|\theta^{(n)})$ for $n = 1...S$;

  Concatenate data $D = D \cup \{X^{(n)}, \theta^{(n)}\}_{n=1}^{S}$;

  while $P_\phi(\theta|X)$ *not converged* **do**

    Draw Samples $\{X^{(i)}, \theta^{(i)}\}_i^B \sim D$ from D;

    Train via Maximum Likelihood $P_\phi(\theta|X)\dfrac{P^{(i-1)}}{P(\theta)}$ on $\{X^{(i)}, \theta^{(i)}\}_i^B$;

  end

  Update Posterior/Proposal $P^{(i)}(\theta) = P_\phi(\theta|X = X')$;

end

Discussing the algorithm in words: Given a model, I can simulate X from the model, $P(X|\theta)$, – I call this a likelihood simulator. Additionally I sample parameters θ form my prior distribution. Set the proposal distribution $P^{(i=0)}(\theta)$ as the prior for round 1. Sample θ from the proposal, then sample X from the likelihood simulator conditional on θ, $P(X|\theta)$. Now you have samples from the joint distribution X, θ. One can estimate by maximum likelihood a conditional density estimator on the joint simulated data, to produce the conditional distribution $P(\theta|X)$. I will discuss how to do this for a normalizing flow conditional density estimator in Section IIIC. Then condition X in $P(\theta|X)$ on the data from the real data-generating process X'. This produces the posterior. Finally, if one wants to estimate in multiple rounds, I update the proposal distribution $P^{(i)}(\theta)$ with the current guess of the posterior, so in the next round, one samples from $P^{(i)}(\theta)$ instead of the prior and corrects for this via importance sampling. See Proposition 4 in Section IIIC. Additionally, one can find the theoretical and empirical advantages SNPE has over other approaches in Section IVI of the Results.

IIIC Normalizing Flows

I will now discuss a particular type of normalizing flow, the Neural Auto-regressive Flow (Huang et al., 2018). First I will define the building block of a normalizing flow, the bijector. Then I will derive useful properties of bijectors like invertibility, an analytical likelihood function, and a flexible "non-parametric" ability to match any continuous conditional distribution, given enough model capacity. Finally I will show how to turn the Bayesian SNPE algorithm into a simulation-based MLE estimator.

Definition: Bijector

A bijector, f_k, is a function that takes in a J dimensional vector input, z_k, whose jth element is denoted with superscript j, and produces a J dimensional vector output, z_{k+1}. Each jth element is a separate equation linking a z_k^j element to a z_{k+1}^j in the following way:

$$z_{k+1}^1 = a_k^1 * z_k^1 + b_k^1 \tag{3}$$

$$z_{k+1}^2 = a_k^2(z_k^1) * z_k^2 + b_k^2(z_k^1) \tag{4}$$

$$z_{k+1}^3 = a_k^3(z_k^1, z_k^2) * z_k^3 + b_k^3(z_k^1, z_k^2) \tag{5}$$

$$... \tag{6}$$

$$z_{k+1}^j = a_k^j(z_k^1, z_k^2 ... z_k^{j-1}) * z_k^j + b_k^j(z_k^1, z_k^2 ... z_k^{j-1}) \tag{7}$$

For example, the first element of z_k produces the first element of z_{k+1} via the relationship: $z_{k+1}^1 = a_k^1 * z_k^1 + b_k^1$. The functions $a_k^2(.)$, $b_k^2(.)$, $a_k^3(.)$, $b_k^3(.)$, and ultimately, a_k^j and b_k^j are monotonically increasing functions of all their inputs and are typically feed-forward neural networks which are explained in Section AD in the Appendix. Monotonicity can be obtained by setting all the parameters of the feed-forward neural network to be positive. For intuition, one can assume these functions are high-dimensional polynomial or spline regressions that have weights constrained to be monotonically increasing.

Proposition 1: The bijector, f_k, is an invertible function.

Proof: $f_k(z_k)$ can be constructively shown to be one to one and onto by induction over each element j of the J vector-valued equations in equations (3) to (7). See Section BA in the Appendix.

Note: This proof not only shows f_k is invertible, but shows how to obtain z_{k+1} from z_k and z_k from z_{k+1}. In particular, going from z_k to $z_{k+1} = f_k(z_k)$ involves only the right hand side of equations (3) to (7), which are functions only of z_k. Going from z_{k+1} to $f_k^{-1}(z_{k+1})$, knowing the first element of z_{k+1}, Equation (3) can be inverted:

$$z_k^1 = \frac{z_{k+1}^1 - b_k^1}{a_k^1} \tag{8}$$

Thus from z_{k+1}^1, you can obtain a z_k^1. The equation for the second element of z_{k+1}, z_{k+1}^2 is defined:

$$z_{k+1}^2 = a_k^2(z_k^1) * z_k^2 + b_k^2(z_k^1) \tag{9}$$

Since z_k^1 is already known, $a_k^2(z_k^1)$ and $b_k^2(z_k^1)$ are also known and thus one can obtain a z_k^2 from any z_{k+1}^2. By induction, if you know z_k^1, z_k^2 to z_k^{j-1} as well as z_{k+1}^j, a_k^j and b_k^j are only functions of z_k^1, z_k^2 to z_k^{j-1} and so are known constants a_j and b_j. Thus inverting the linear relationship will obtain a z_k^j from a z_{k+1}^j. The general equation then is, knowning $z_k^1, z_k^2 ... z_k^{j-1}$:

$$z_k^j = \frac{z_{k+1}^j b_k^j(z_k^1, z_k^2 ... z_k^{j-1})}{a_k^j(z_k^1, z_k^2 ... z_k^{j-1})} \tag{10}$$

Bijector to Flow to Density Estimator

In order to turn a bijector into a effective density estimator, create a composition of K (capital K) bijectors:

$$z_K = f_{K-1} \circ f_{K-2} \circ ... f_0(z_0) \tag{11}$$

Call the function $f_{K-1} \circ f_{K-2} \circ ... f_0(z_0)$ with $z_0 \sim N(\bar{0}, \bar{1})$ a normalizing flow. Then attempt to modify the parameters of each f_k in the flow such that the likelihood of every $z'_K = f_{K-1} \circ f_{K-2} \circ ... f_0(z_0)$, is maximized. In order to do this, I must define the likelihood function.

Proposition 2: A normalizing flow is a collection of bijectors, f_k, along with a base distribution, z_0, sampled from a known pdf. More rigorously, the flow is: $z'_K = f_{K-1} \circ f_{K-2} \circ ... f_0(z_0)$ s.t. $z_0 \sim N(\bar{0}, \bar{1})$ Each bijector in the normalizing flow is a valid change in measure and the log likelihood that the normalizing flow generates a sample z'_K is:

$$ln(P(z'_K)) = ln(Q(z_0)) - \sum_{k=0}^{k=K-1} ln(|det \frac{df_k(z_k)}{dz_k}|) \tag{12}$$

Proof: See Section BB in the Appendix. The formula is derived by recognizing each f_k is a bijection and then iteratively applying a change-of-variables. One can recover z_0 from z'_K by iteratively applying the inverse equation (Equation (10)) for each f_k in the flow. The distribution $Q(z_0)$ is known and could be, for example, $N(\bar{0}, \bar{1})$. So given samples from z_K, the right hand side of Equation (12) can be evaluated. Knowing this likelihood, one can maximize it with respect to the parameters in each of the $a_k^j(.)$ and $b_k^j(.)$ in each of the flows

This composition of functions, $z_K = f_{K-1} \circ f_{K-2} \circ ... f_0(z_0)$, can map samples, z_0, into samples from z_K. Furthermore, this mapping can be extremely flexible if you can modify the parameters of each a_k^j and b_k^j, using the likelihood as the loss function. Each f_k modifies the realizations of the previous output $z_k = f_{k-1}(z_{k-1})$, slowly transforming realizations $z_0 \sim Q(z_0)$ into samples distributed like $z'_K \sim P(z'_K)$, when fed through the chain of f_k's. Proposition 3 in Section IIID discusses the conditions that this map is a universal approximator of continuous distributions, converging under the Kullback-Leibler divergence. Thus, the stack of f_k's has the flexibility of a non-parametric function. In order to ensure an arbitrary conditioning structure, one should permute the order of each z_k input to f_k. Thus, different elements of each z_k can be conditioned on different other elements when fed into f_k.

Additionally, one can turn the normalizing flow into a conditional density estimator that can learn $P(\theta|X)$, the SNPE objective, from joint (θ, X) samples, by appending $X = \{X_1 ... X_N\}$to the

input of each $a_k^j(z_k^1...z_k^{j-1}, \mathbf{X_1}...\mathbf{X_N})$ and $b_k^j(z_k^1...z_k^{j-1}, \mathbf{X_1}...\mathbf{X_N})$, making each bijector, f_k, in the normalizing flow conditional on X.

Additionally one can add nonlinearities to the equations of f_k. For example, take the equation:

$$z_{k+1}^j = a_k^j(z_k^1..z_k^{j-1}) * z_k^j + b_k^2(z_k^1...z_k^{j-1}) \tag{13}$$

One can add a nonlinearity by applying a nonlinear bijective transformation σ:

$$z_{k+1}^j = \sigma(a_k^j(z_k^1..z_k^{j-1}) * z_k^j + b_k^2(z_k^1...z_k^{j-1})) \tag{14}$$

Here σ can be the logistic function, or something like a leaky ReLU:

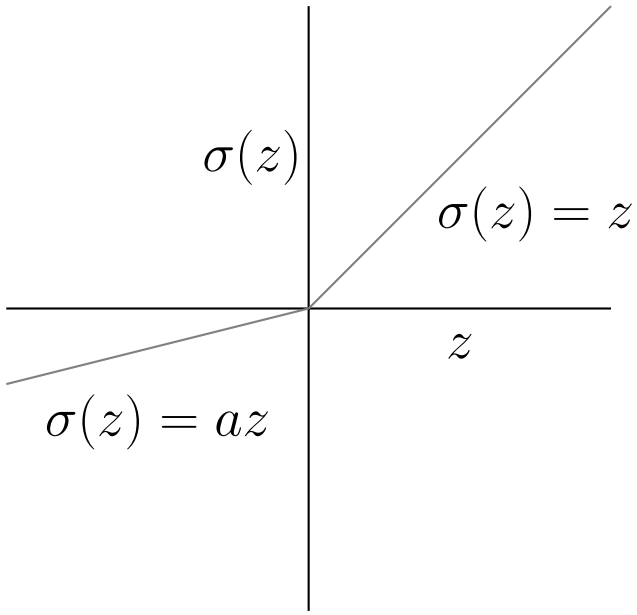

FIGURE I: LEAKY RELU NONLINEARITY FOR A NEURAL NETWORK

This allows the model to be more nonlinear compared to just modeling the transform, f_k, as linear after conditioning on other variables.

In addition, analogous to feed-forward neural networks (see Section AD in the Appendix), one can make each bijector wider, by summing together multiple inputs for each element of the input

and output vector. For example, instead of an element-to-element relationship of:

$$z_{k+1}^{j} = \sigma(a_k^j(z_k^1..z_k^{j-1}) * z_k^j + b_k^2(z_k^1...z_k^{j-1})) \tag{15}$$

one adds a third index, l (ell), where z_{k+1}^j is the sum of l individual $\sigma(a_k^j(z_k^1..z_k^{j-1}) * z_k^j + b_k^2(z_k^1...z_k^{j-1}))$. For example:

$$z_{k+1}^{j} = \sum_{l=1}^{L} \sigma(\{a_k^j(z_k^1..z_k^{j-1})\}^l * z_k^j + \{b_k^2(z_k^1...z_k^{j-1})\}^l) \tag{16}$$

Where now $\{a_k^j\}^l$ and $\{b_k^j\}^l$, are indexed over the particular bijector k, the element of each f_k bijector vector j, and finally the sum of l bijector relationships that add up to the value z_{k+1}^j.

IIID SNPE Theory

Proposition 3: A Neural Auto-regressive Normalizing Flow with the following structure for each bijector:

$$z_1^j = \sigma^{-1}(\sum_{l=1}^{L}(\{w_0^j(z_0^1...z_0^{j-1})\}^l * \sigma(\{a_0^j(z_0^1..z_0^{j-1})\}^l * z_0^j + \{b_0^2(z_0^1...z_0^{j-1})\}^l))) \tag{17}$$

where one assumes each $\{b_0^j\}^l$ is bounded between two numbers, each $\{a_0^j\}^l$ is bounded and positive, and all the $\{w_0^j\}^l$ are positive and, across the l (ell) dimension, the $\{w_0^j\}^l$ sum to 1. Finally σ is the logistic transformation and σ^{-1} is the inverse logistic. These are a set of constraints, which does not impose much in practice, but is a requirement to allow for convergence. With these requirements, a flow structured in this way can take any continuous distribution $z_0 \sim Q(z_0)$ and learn a map that converge in distribution to any other continuous distribution $z_1 \sim P(z_1)$, that has a conditioning structure expressible by the normalizing flow.

Comment: This is essentially equivalent to Equation (16) with the addition of the inverse sigma, whose only purpose is to turn each element-by-element equations: $f_k^j(z_k^j)$ – which is a logit that

ranges from 0 to 1 – to a function that has a range on the entire real line. Additionally, like other universal approximator proofs (like for feed-forward networks (Hornik, Stinchcombe and White, 1989)), the normalizing flow will only be composed with a single bijector with width (the l dimension) that can be arbitrarily large. Alternative universal approximator proofs for flows under different and/or weaker conditions also exist (Koehler, Mehta and Risteski, 2021), (Lee et al., 2021). In contrast to these proofs, deep networks[4] perform better empirically.

Proof of Proposition 3: See Appendix F and G of Huang et al. (2018) for proof. Essentially step functions can approximate any monotonic function arbitrarily well, logits can approximate step functions arbitrarily well, and logits where the parameters of a neural networks conditional on other elements and variables can approximate logits with arbitrary parameters via the universal approximation theorem of feed forward neural networks (Hornik, Stinchcombe and White, 1989). Finally, monotonic functions can map any continuous distribution from a uniform distribution via the CDF function and likewise map any distribution to a uniform distribution via the inverse CDF function. Thus monotonic functions can map arbitrary continuous distributions to one another and a flow parameterized in this way can do the same.

Comment: The assumption that the conditioning structure can be expressed by a single layer flow f_0 is problematic because some variables can't be conditioned on other variables by the flow's structure. This assumption can be entirely relaxed by adding depth to the normalizing flow (i.e. compose many $f_{k>0}$ together) with conditioning structures derived by permuting the order of z_k, differently for each k, so variables can be conditioned on different sets of variables depending on the bijector.

Proposition 4: Given the density estimator $P_\phi(\theta|X)$ and a set of conditions:

1. The number of samples of (X, θ) go to infinity and,
2. These samples are independently generated from the distribution $X, \theta \sim P(X|\theta)P'(\theta)$ given $P'(\theta)$ is a proposal distribution that has the same support as the prior $P(\theta)$ and,
3. If the true posterior is contained in a parameterization of ϕ of the conditional density

4. i.e. normalizing flows with more than one k, or feed-forward networks with more than one hidden layer

estimator, $P_\phi(\theta|X)$, then fitting the X, θ samples with the following importance sampling adjusted relationship:

$$\max_\phi \frac{P'(\theta)}{P(\theta)} * P_\phi(\theta|X) \tag{18}$$

$$s.t. \tag{19}$$

$$X, \theta \sim P(X|\theta)P'(\theta) \tag{20}$$

allows $P_\phi(\theta|X)$ to converge to the true posterior under the Kullback-Leibler divergence.

See the proof of Proposition 1 in Papamakarios and Murray (2016) and note one can directly optimize the conditional distribution of $P(\theta|X)$, by moving the substituting $P_\phi(\theta|X)$, for $P(\theta|X)$ in the final step of Equation (11) in their paper. Reproducing Equation (11), by moving the proposal-prior ratio to the left hand side gets this equation:

$$P^{(i)}(\theta|X) = P_\phi(\theta|X)\frac{P^{(i)}(\theta)}{P(\theta)} \tag{21}$$

Here $P(\theta)$ is the true prior, $P^{(i)}(\theta)$ is the proposal distribution one is sampling from and $P^{(i)}(\theta|X)$, is the conditional distribution that one would fit, if one naively fit a conditional density estimator on the joint samples. Estimating $P_\phi(\theta|X)$ directly from these samples will converge to the posterior. Additional information on importance sampling and multi-round inference is discussed in Section AB of the Appendix. QED.

Note: Regarding Propositions 3 and 4: A normalizing flow with enough parameters can produce a distribution arbitrarily close to any continuous distribution per Proposition 3. When used as a conditional density estimator in SNPE, with a large enough number of parameters, and a large enough number of (X, θ) samples, the normalizing flow can produce a distribution arbitrarily close to the true posterior under the Kullback-Leibler divergence. While they don't address

this precise problem, work by Farrell, Liang and Misra (2018) and Draxler, Schnörr and Köthe (2022) suggest convergence should be approximately quadratic and very likely polynomial. Chewi (2023) suggests MCMC, HMC and other random walk algorithms converge in sub-polynomial time when the posterior is multi-modal or flat. This provides theoretical underpinning of faster SNPE convergence.

While much of this paper focuses on SNPE for Bayesian estimation, I do spend some time discussing how to use SNPE as a frequentist simulated maximum likelihood estimator. In this proposition, I will show how to perform MLE using SNPE and uniform priors.

Proposition 5: One can estimate by maximum likelihood, on a bounded posterior space, by taking the mode of the posterior produced by performing SNPE with flat priors.

Proof: One is attempting to find the mode of the posterior – maximizing the value of θ in the posterior:

$$m = max_\theta P(\theta|X = X') \tag{22}$$

$$= max_\theta \frac{P(X'|\theta) * P(\theta)}{P(X)} \tag{23}$$

$$= max_\theta \frac{P(X'|\theta) * c}{d} \tag{24}$$

$$= max_\theta P(X'|\theta) \tag{25}$$

Going from equation (23) to equation (24), one realizes that all points inside the upper and lower bounds of the model have constant probability mass c. Likewise $P(X)$ is also constant. Thus, funding the posterior max is equivalent to finding the likelihood max. QED

IIIE Potential Tradeoffs Compared to Other Methods

SNPE has many advantages, but I will list a few of the downsides. First, SNPE is a Bayesian algorithm that can be used to do frequentest maximum likelihood estimation. Thus, if one wants a maximum likelihood estimate there is no benefit of using SNPE if one has a likelihood function

as SNPE will be slower. At the same time, while SNPE has efficiency advantages over method of moments, method of moments will also be faster. Additionally with method of moments, one can define a distance from simulations to the real data, even if the real data is not in the support of the model. As a likelihood based, SNPE will not estimate models that put no support on the data. This can be thought of as either an advantage or disadvantage. Finally, while SNPE learns to condition the distribution of the parameters on simulated data, it conditions this distribution on the entire data set at one time. This is useful in the case of macro models because there is a small, but nonzero, effect from the first time step to the last time step, so conditioning on the data all at once is more faithful to the state space data generating assumption of macro models. However, this means that if one has data with many iid samples which make the data set too high dimensional, SNPE will not work. Other techniques which aren't faithful to the state space generating process, but can estimate models using batches of small iid data points, can handle data sets with essentially arbitrary number of iid data. Models in this category include simulated or generalized method of moments and the GAN approach of (Kaji, Manresa and Pouliot, 2020). This is an advantage for the estimation of structural models, but is a disadvantage when estimating big applied-micro data sets for instance.

IV Results

This section will evaluate the performance of SNPE by examining the proper behavior of the posterior and inspecting its accuracy. To this end, seven models are estimated using the SNPE algorithm, each showcasing a unique aspect and characteristic of the algorithm. In the first model, with which I will walk through SNPE with some detail, is the Real Business Cycle (RBC) model. The second model is a corporate finance model of investment that is solved via value function iteration. The third model is the Lucas asset-pricing model solved through projection. The fourth model is a life cycle model with bequests solved via value function iteration. These four models are discussed in more detail in Section C of the Appendix and are the toy representative agent models in this paper.

The last three models are more serious and demonstrate SNPE capability even in challenging research-grade models. The fifth model is the Smets-Wouters model, which is estimated through both the MCMC and SNPE algorithms. The sixth model is the HANK model solved through Reiter's method and time iteration. The seventh model improves on full information estimation of a Winberry (2018) model by speeding up the approach taken by Liu and Plagborg-Møller (2023). This model is extended to an empirical application estimating both micro and macro moments in a full information setting. The details of the construction and implementation of these models can be found in Section CG of the Appendix.

In the empirical analysis, the accuracy of the posterior is inspected by comparing it with the calibrated parameters producing the simulated data. Through the Bernstein von Mises Theorem, the posterior should converge to the true values, although the structural models may not satisfy the assumptions of the theorem. Although the correct posterior should converge to the true parameter value as the number of data samples increases, finite sample sizes will not result in a perfect match. Hence, it is important to assess the posterior mode by examining the marginal modes and ensure that it generally matches the true value and that there is posterior mass at the true values along each marginal dimension. Importantly, given correlation across variables, the posterior mode is not the same as the modes of the posterior marginals. Nevertheless, this check provides useful information to determine the performance of SNPE in practice and how it compares with other estimation techniques, such as the MCMC algorithm. SNPE is tested using simulated data and typically 200 time steps are sampled from the simulated model, to maintain a rough correspondence to the length of real world macroeconomic data. This small number of time-steps results in some parameters being inadequately identified especially in the models with a large number of parameters, but is important for simulating realistic estimation data sets.

For the sake of illustrating the estimation benefits, all priors will be uniform over the range specified in the graphs. This most effectively demonstrates the ability of the estimation algorithm to converge on a posterior without guidance from informative priors. The following charts hopefully illustrate the flexibility and accuracy of SNPE.

IV.A RBC Model

I will walk through the RBC model using SNPE. First, as with all of my models, I use flat priors defined over an appropriate range. In this case, the parameters – α, capital share; β, discount rate; δ, depreciation rate; and ρ, productivity persistence parameter – all range from 0 to 1. These four variables are collectively called θ.

The utility function is a standard CRRA utility:

$$u(C_t) = E\sum_t \beta^t \frac{C_t^{1-\gamma}}{1-\gamma} \tag{26}$$

The production function is standard Cobb-Douglas with productivity:

$$Y_t = Z_t K_t^{\alpha} \bar{L}^{1-\alpha} \tag{27}$$

Where $\bar{L}$ is fixed labor, Z_t is productivity, and K_t is capital. Capital evolves according to a standard law of motion:

$$K_t = (1-\delta)K_{t-1} + Y_t - C_t \tag{28}$$

Productivity evolves according to a AR(1) process in logs:

$$logZ_t = \rho logZ_{t-1} + \epsilon_t \tag{29}$$

Then after sampling parameters from this four-way (α, β, δ, and ρ) uniform distribution, I sample simulated data from my RBC model, conditioned on the parameters sampled from the prior. This gives me the data series: Y, C K, and Z that I can use to match real world data. All four simulated variables combined are labeled X and the corresponding data from the true data-generating process is called X'. Depending on the exercise, X', could be data sampled from a calibrated model or real word data. The joint distribution of the sampled parameters and the simulated data is used to train a conditional density estimator that learns the probability density function $P(\theta|X)$, where

θ is the simulated parameters, and X is the simulated data. Then conditioning X on the real data, X', one learns the posterior: $P(\theta|X = X')$. This is a simplified intuitive application of the SNPE algorithm. Figure II shows the posterior distribution resulting from the application of SNPE on simulated data. More graphs and information are provided in Section CA of the Appendix.

In the below image (II), the graphs along the main diagonal indicate the marginal distributions of the posterior. The y axis on the diagonal graphs illustrates increasing probability mass and the x axis indicates the support of the posterior which ranges from the lower bound to the upper bound of the uniform prior. The off-diagonal charts are two way contour plots where the x axis corresponds to the marginal distribution at the same vertical axis and the y axis corresponds to the marginal at the same horizontal axis. The green and yellow colors on this plot indicate increasing probability mass and the blue indicates little or no probability mass.

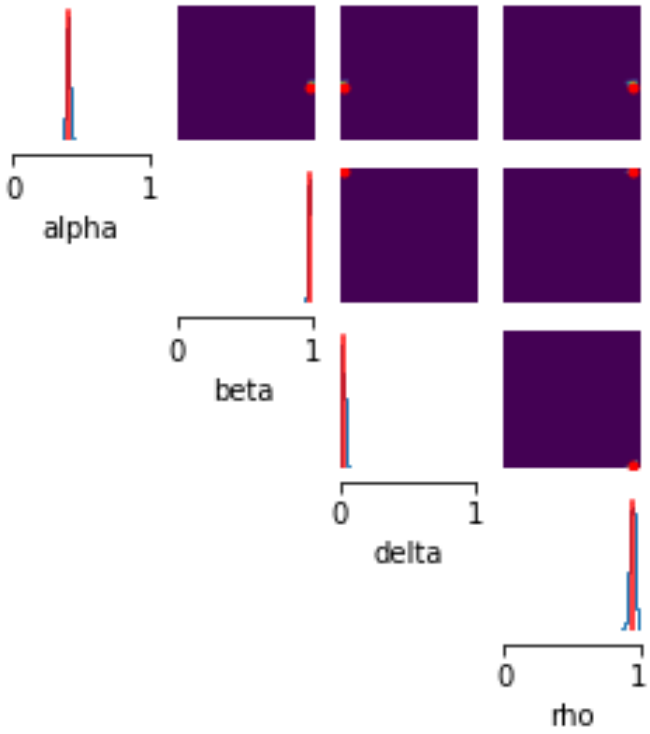

Figure II: SNPE estimation of RBC Model

As shown here, the posterior mode concentrates on the real parameter value, suggesting convergence.

IVB Investment Model

A firm is trying to control investment, K, such that the discounted series of cash flow, π, is maximized:

$$\pi(K_t, Z_t) = Z_t K_t^{\alpha} \tag{30}$$

More details of the model and data used are found in Section CB of the Appendix. Figure III is the posterior picture that comes from estimation of a partial equilibrium investment model:

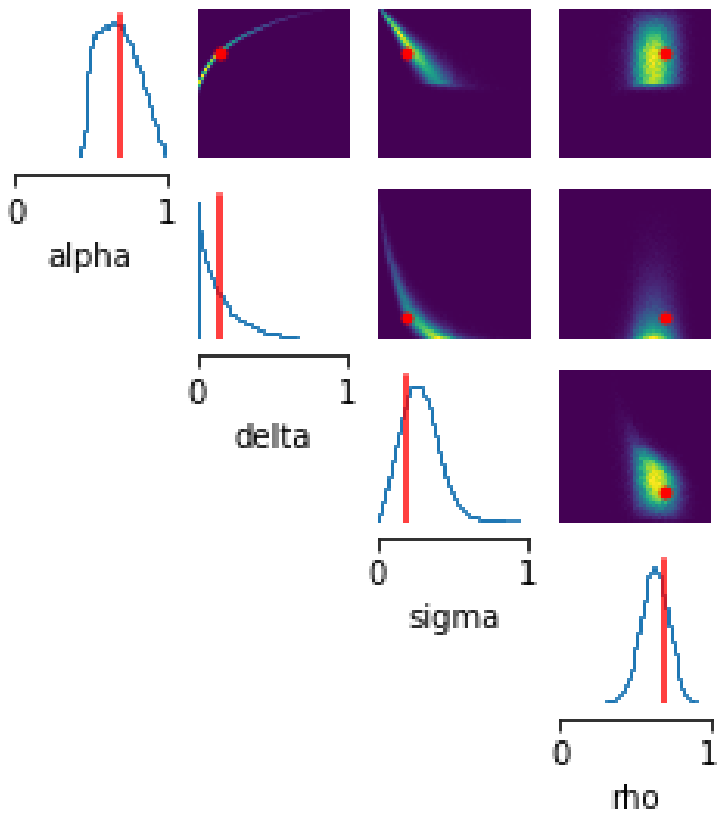

FIGURE III: SNPE ESTIMATION OF AN INVESTMENT MODEL SOLVED VIA VALUE FUNCTION ITERATION

The modes concentrate on the true parameter value, suggesting convergence.

IVC Projection - Lucas Tree Asset Pricing Solved via Projection

The Lucas tree asset pricing model, is similar to the RBC model. However, the agent can buy a number, k_t of assets that have a price P_t and produce dividends, d_t:

$$m_{t+1} = (P_{t+1} + d_{t+1})k_{t+1} \tag{31}$$

More details of the model and data used are found in Section CC of the Appendix. Figure IV is the posterior picture that comes from the Lucas tree model solved via projection.

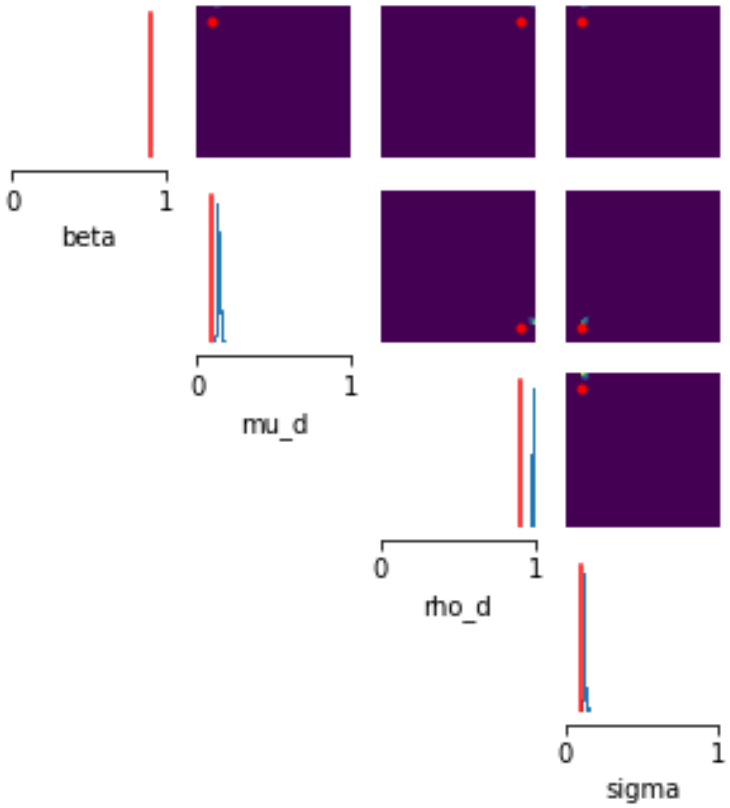

Figure IV: SNPE Estimation of Lucas Tree Model Solved via Projection

Again, the modes somewhat concentrate on the true parameter value although, in some instances, they drift from the true value, which is an error that could be fixed with additional samples.

IVD Value Function Iteration - Life Cycle Model

The fourth model considered in this study is an 11-parameter life cycle model. Here one is maximizing utility given a low yielding deposit account and a higher yielding but illiquid "retirement" account. The plots in Figure V corresponds to the parameters of the model discussed further in

Section CD of the Appendix.

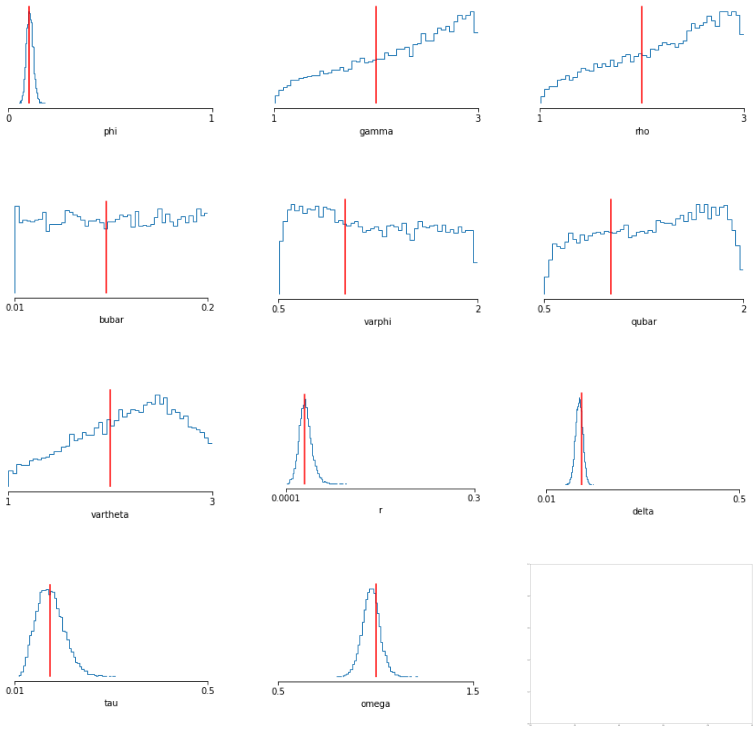

FIGURE V: SNPE ESTIMATION OF LIFE CYCLE MODEL SOLVED VIA VFI

The results of the estimation of the model parameters via value function iteration (VFI) are shown in Figure V. Although many of the parameters have converged to the true values, some of the parameters still exhibit high uncertainty. This uncertainty is due to the fact that the data was only 200 time steps long. Many of the posterior marginal distributions do concentrate around the true values, suggesting that the model has some discerning power with regard to the parameters.

Next, I will discuss the main results estimating the Smets and Wouters (2007) model, a HANK model, and one heterogeneous agent non-HANK empirical application.

IVE Smets and Wouters

The Smets-Wouters (2007) DSGE model is a representative agent new Keynesian model that describes an economy with a flexible price sector and a sector with sticky Calvo prices. The

model has eight observed variables and eight shocks that interact with the rest of the model in an auto-regressive manner. Compared to the original paper, I estimate the model on data simulated from a model with parameters calibrated on the mode of the original estimated model posterior. Estimation-wise, instead of using Smets-Wouters priors, all variables have flat priors with bounds equal to the bounds in the original paper. I then perform estimation of the simulated data.

The linearized form of the model has 36 parameters. SNPE estimated the posterior using roughly 500 thousand samples generated from the prior and likelihood. The parameters are separated into three different charts of 12 parameters each. Only marginals of the first 12 parameters are displayed to avoid visual clutter. The remaining parameter marginals are displayed in the end of Section CE of the Appendix.

Like the diagonals of the previous charts, the y axis indicates probability mass and the x axis indicates posterior support. These graphs are histograms rather than smoothed with kernel. This illustrates that the density estimators, be they conditional mixture of Gaussians or normalizing flows, generate natural looking posteriors, which MCMC is not able to generate. The variable labels are difficult to read due to the effort to provide a reasonable number of parameters in each page. Section CE of the Appendix has a list of variables in the order they are displayed.

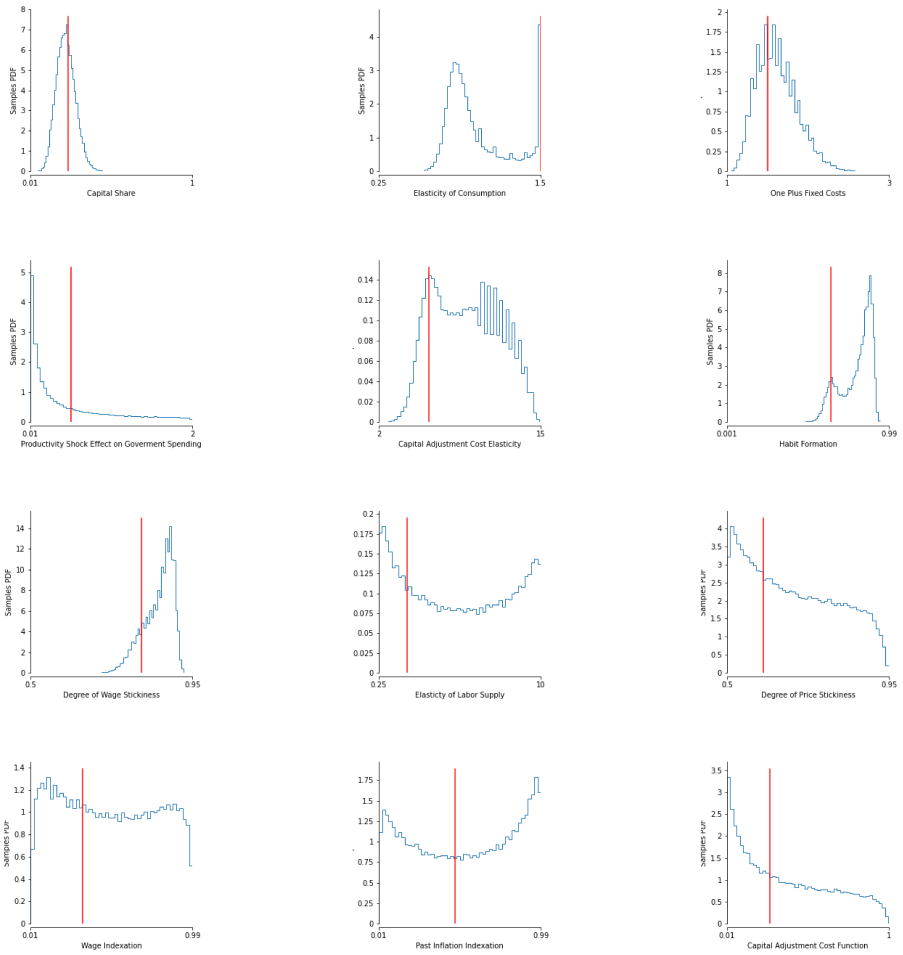

FIGURE VI: SNPE SMETS-WOUTERS POSTERIOR WITH 500 THOUSAND SAMPLES: PARAMETERS 1-12

SNPE estimation of the Smets-Wouters model is successful in estimating the various parameters. For example, the model accurately estimates the capital share but has more difficulty with the adjustments costs. The (one plus) fixed costs are estimated well. SNPE also handles irregular and otherwise difficult to identify posteriors. You can also see identification of multi-modal posterior marginals like habit formation and elasticity of consumption. Likewise, price stickiness is relatively unidentified, consistent with the disagreement in the literature.

The posterior estimated by the Smets-Wouters model with MCMC are less successful in covering

the true data generating process parameters. Both MCMC and SNPE used the Smets-Wouters Dynare (Adjemian et al., 2022) file to solve the model and much of the MCMC used the original Dynare file for estimation, with a few modifications (uniform priors, for instance). Despite this, the MCMC algorithm took roughly 30 hours to draw 10 million samples, with a 10 thousand sample burn-in, and ultimately the posterior showed limited coverage.

Below, I reproduce the (Smets and Wouters, 2007) using 10 million MCMC samples. Again only the first 12 parameters are produced to reduce clutter. The rest are displayed in the end of Section CE of the Appendix.

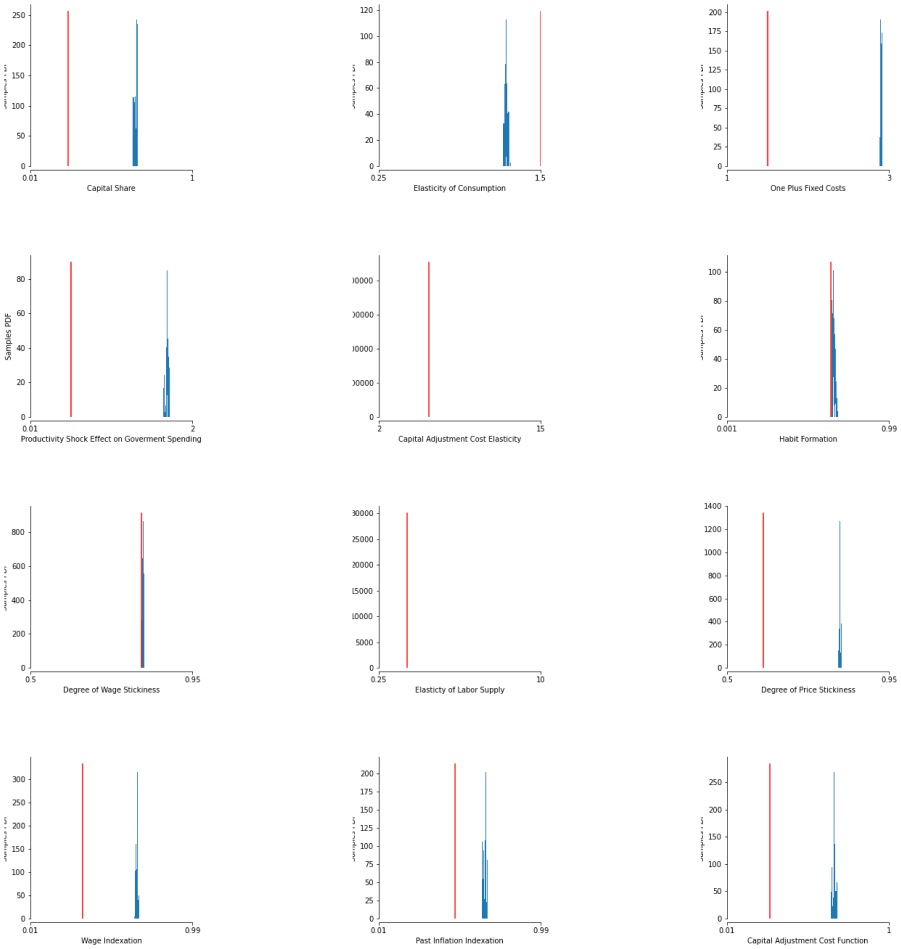

FIGURE VII: SMETS-WOUTERS MCMC POSTERIOR WITH PARAMETERS 1-12

Both algorithms, MCMC and SNPE, call the Smets and Wouters (2007) Dynare code when solving the model. The performance of the MCMC algorithm, as applied in this study, presents several challenges in terms of its ability to converge to the true data generating parameters, as can be seen in the chart. While some of the posterior marginals show good coverage of the true parameters, a substantial number of the posterior marginals exhibit a support outside of the data generating parameter value and often converge to delta functions far from the true parameters. Among many parameters, capital share and (one plus) fixed costs seem to be problematic in terms of posterior convergence. The MCMC algorithm places all its mass near .8, and 2.9, respectively, whereas the true parameters are located at .24 and 1.5. Although MCMC learns a posterior concentrated around the true value of habit formation, it repeats this error for most of the other 10 parameters, including elasticity of labor supply, price stickiness, and capital adjustment function among others. For most of these parameters, SNPE, has a mode near the true value, and the true parameter value is always in the SNPE support. Additionally, the SNPE posterior estimate has no issue reproducing multi-modal marginals, a documented problem with MCMC, illustrated in Herbst and Schorfheide (2014). The other 24 parameters are similarly problematic for MCMC, but look more reasonable when estimated via SNPE.

In the cases of uninformative priors, there is evidence of the difficulty converging under realistic computational resources for MCMC. It is widely known that MCMC, or even improvements like HMC, empirically converge slower than other machine learning alternatives (Nathoo et al., 2013). Additionally, Herbst and Schorfheide (2014) demonstrate the identification problem for MCMC in the Smets-Wouters model, while Andrews and Mikusheva (2015) highlight theoretical problems in identifying DSGE models in the MLE setting. When using MCMC-based simulated annealing to perform MLE optimization, Siarry et al. (1997) runs into issues even with a moderate number of dimensions, for example, objective functions with as few as 3-6 parameters. Additionally, Chewi (2023) points out that for multi-modal and even unidentified distributions, MCMC and related approaches like HMC and No U-Turn Sampling (Hoffman, Gelman et al., 2014) converge at a rate slower than polynomial, while work by Farrell, Liang and Misra (2018) and Draxler, Schnörr and Köthe (2022) suggest that neural networks similar to SNPE converges quadratically, or perhaps

slightly slower, but still at a polynomial rate.

The SNPE algorithm used in this study was implemented in Python, which called the Smets-Wouters solution Matlab/Dynare code. The MCMC algorithm was fully executed in native Matlab, calling the same Dynare package. This was done mirroring the Smets-Wouter original code, called with minor modifications. However, due to the difference in programming languages as the back-end Dynare code runs in C++, and the overhead of calling Matlab code in Python, the estimation process was much slower for the SNPE algorithm, with roughly 2.5 models solved per second, compared to about 100 models solved per second for the MCMC. If SNPE had the same C++ speedups as being native Dynare, one would expect SNPE would be significantly faster with roughly the same number of iterations per second as MCMC. While 10 million samples of MCMC in is faster than five hundred thousand Python iterations, MCMC encouters other compuational constraints when running for this long. For example, one issue was the amount of RAM memory required both during estimation and analysis. This suggest sampling significantly more, which MCMC estimation of this model needs, will require more involved software engineering. Nevertheless, to maintain comparison, the SNPE posterior is shown below after 150 thousand samples (as opposed to 500 thousand samples in Figure VI), which took roughly the same 30 hours, as did the 10 million samples of the MCMC. Again, only the first 12 parameters are provided to reduce clutter:

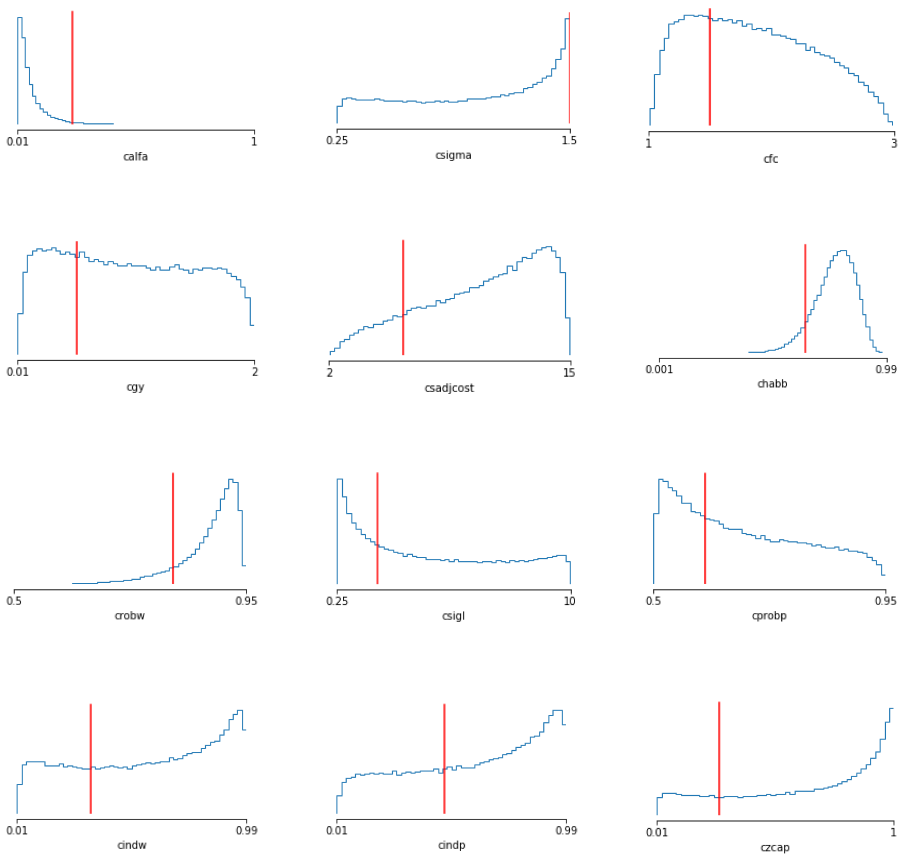

FIGURE VIII: SMETS-WOUTERS SNPE POSTERIOR WITH 150 THOUSAND SAMPLES: PARAMETERS 1-12

Although the SNPE algorithm incurs a Python time penalty, it still manages to learn a better posterior in the same amount of time as the MCMC with Dynare, as discussed above and it's not much different than the posterior obtained by running for 500 thousand samples as shown earlier. The performance of the SNPE algorithm can be further improved by optimizing the Python code through techniques such as Numba or writing the back-end algorithm in C++. However, these are beyond the scope of this paper. The results indicate that while the SNPE posterior is not perfect, based on this abbreviated sampling process, it still assigns a significant probability mass to the true data-generating process and in many cases, even concentrates around the true parameters. This is a marked improvement over the MCMC algorithm, which even after 10 million iterations, generally

only produces highly concentrated posteriors along a small range that often does not include the true data-generating parameter. Now I will move on to discuss heterogeneous agent models.

IVF HANK Model

The sixth model is the 10-parameter HANK model, with the parameters discussed in Section CF of the Appendix. Figure IX is a chart of SNPE estimating a 10-parameter HANK model solved via Reiter:

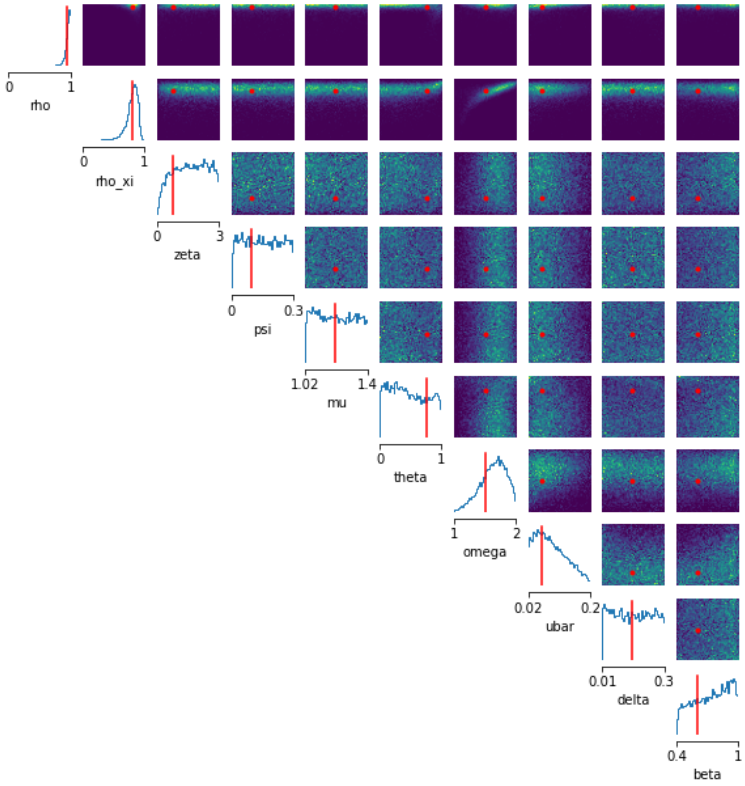

Figure IX: SNPE Estimation of the Ten Parameter HANK model

The HANK model features two observable shocks, namely the unemployment rate and the interest rate, and two observable equations. It was solved with Reiter's method (Reiter, 2009). Despite having a large latent state-space of 810 variables, the estimation of the model was carried out successfully without information losing dimensionality reduction of Ahn et al. (2018) or even

more accurate methods like Bayer and Luetticke (2020). Dimensionality reduction speeds up the estimating of models solved in this way when evaluating the likelihood, but reduces accuracy. This is because Kalman filtering is expensive. However, SNPE does not require estimating with a likelihood function and, by extension, the intermediate filtering step. Because of this, my approach is much faster, although rate of convergence as a function of the size of the state still scales similarly with or without filtering and is accurate. However, if one uses sequential Monte Carlo for nonlinear filtering (Fernández-Villaverde and Rubio-Ramírez, 2007), which is more expensive than Kalman filtering, this converges in an exponential rate (Beskos, Crisan and Jasra, 2014), significantly slowing the estimation process.

As shown above, the approach used in the estimation process was able to identify the parameters with confidence, and the estimated posterior agrees with the data generating process. This demonstrates the feasibility of estimating HANK models with a large latent state-space via SNPE, which is computationally infeasible for likelihood approaches.

The algorithm was able to estimate the parameters ρ, ρ_ξ, ω and $\bar{u}$ with high accuracy. The mode of the distribution of these parameters was found to be either at or close to the true parameter value. However, for the other parameters, the likelihoods were found to be flat, which may be attributed to the limited impact of these parameters on the chosen observed variables or the relatively small number of time-steps in my simulated data.

An alternative Deep Learning approach for HANK estimation is Kase, Melosi and Rottner (2022). They use a Deep Learning approach to learn the relationship between the parameter values and the likelihood. Using this "extended" neural network, they can query from the likelihood using the network as a surrogate likelihood without having to solve the model. Both SNPE and the Kase, Melosi and Rottner (2022) approach use a surrogate model. However, in the case of SNPE, it learns a posterior from samples rather than learning a likelihood from likelihood queries. When the likelihood is learned, Kase, Melosi and Rottner (2022) perform MCMC by using the surrogate likelihood function to provide likelihood information. It is unclear if predicting the likelihood is more computationally efficient than density estimating from samples. However, their application uses tight priors with modes precisely at the true parameter value. Thus, even if the likelihood

was not informative, their approach would recover the true parameter values. At the same time, my approach uses uniform priors which are diffuse, and still often recover the true data-generating process at my posterior modes.

In conclusion, with SNPE, the estimation of the HANK model showed promising results, and highlights the potential of the approach used in the estimation process for models with large latent state-spaces.

IVG Winberry Heterogenous Agent Model

A seventh model is a Winberry (2018) solution method to the Krusell and Smith (1998) heterogeneous agent model with full micro-moments information as implemented in Liu and Plagborg-Møller (2023). It is discussed in more detail in Section CG of the Appendix, and further discussed in Liu and Plagborg-Møller (2023), Winberry (2018), and Krusell and Smith (1998). The consumer is standard discounted CRRA utility. Output is consumption and investment. Agents can be employed or unemployed. If employed, it works one unit of labor or else it does not work. This is denoted with an ϵ. Employment status transitions via a Markov matrix.

On the market side, wages and rental rate equal their marginal benefit. In particular, if aggregating across all capital ($\bar{k}$) and labor ($\bar{l}$):

$$w = (1-\alpha)z(\bar{k}/\bar{l})^{\alpha} \tag{32}$$

$$r = \alpha z(\bar{k}/\bar{l})^{1-\alpha} \tag{33}$$

The states are the distribution of capital and labor, denoted Γ, as well as productivity, ϵ. For each individual, they optimize via:

$$v(k,\epsilon;\Gamma,z) = max_{c,k'}\{U(c) + \beta E[v(k,\epsilon;\Gamma,z)|z,\epsilon]\} \tag{34}$$

subject to

$$c + k' = rk + wl\epsilon + (1 - \delta)k \tag{35}$$

$$\Gamma' = H(\Gamma, z, z') \tag{36}$$

$$k' \geq 0 \tag{37}$$

Each agent optimizes according to their own k and ϵ. However the prices are chosen so that when agents are optimizing, the aggregate next-period levels of k and ϵ match the levels implied by the prices. Winberry exploits the fact that if you know the individual's law of motion then you can integrate the distribution to get the aggregate prices that generate that law of motion. You can then derive a law of motion for the heterogeneous distribution. If you approximate this distribution via moments, you can derive a law of motion for the moments of the distribution. Then you can apply non-heterogeneous agent approaches to solve this model. In particular, Winberry solves the model in Dynare with perturbation.

After testing both approaches, I find that SNVI, a variation of SNPE, can handle this estimation more effectively. The SNVI is discussed further in Section DD and Algorithm 3 in the Appendix. It recovers similar posterior results as Liu and Plagborg-Møller (2023), but is around 13x faster per iteration, because I only need to sample micro and macro data, but don't need to evaluate a likelihood. I use the same number of iterations (10000) as Liu and Plagborg-Møller (2023), and this appears to converge to the same posterior distribution. Additionally, Liu and Plagborg-Møller (2023) likelihood calculation requires the micro data to be independent of the macro data conditional on the state. As a sampling technique, my approach can be applied even when this assumption does not hold. However, that is beyond the scope of the paper. Figure X shows good identification on the smaller three-parameter set:

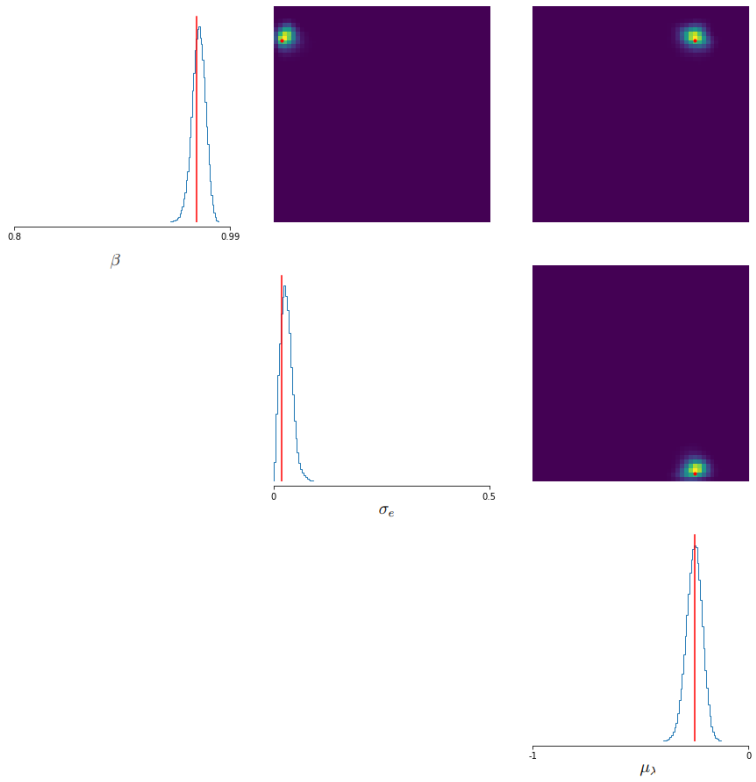

FIGURE X: SNVI ESTIMATION OF KRUSELL-SMITH MODEL

As shown above, the posterior concentrates around the true data generating parameters. Figure XI are the Liu and Plagborg-Møller (2023) results which are similar:

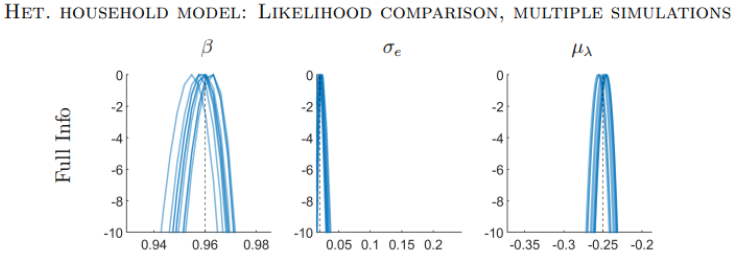

FIGURE XI: LIU AND PLAGBORG-MOLLER KRUSELL-SMITH RESULTS

Figure X shows that the SNIV estimation of the posterior concentrates almost entirely around the true parameter value, suggesting that the estimation was highly informative regarding the

parameter value and accurately reproduces the simulation parameters. The results are compared to those obtained by Liu and Plagborg-Møller (2023) (Figure XI). Both sets of results are similar and demonstrate the viability of the approach used in this study. This example illustrates how SNVI/SNPE is a fast and reliable approach to the estimation of heterogeneous agent macroeconomic models.

Both heterogeneous agent model examples demonstrate the unique competitive advantage of SNPE in estimating heterogeneous agent models. As mentioned, the computational improvement per iteration is 13x for Winberry and over 100x for Reiter due to the fact that SNPE doesn't rely on likelihood evaluation or filtering. Just as importantly, but harder to quantify, because SNPE doesn't require likelihood evaluation or filtering, it saves on programmer effort in terms of lines of code written. In particular, evaluating the likelihood function of the microdata is complicated process in terms of lines of code written (Liu and Plagborg-Møller, 2023), especially because evaluating the microdata likelihood function is a recent innovation and there aren't many, if any, libraries that automate this code writing process. In contrast, one can write SNPE estimation code in 50 to 100 lines of code as discussed in my code tutorial: `https://shorturl.at/pMQ18`.

IVH Empirical Application

Finally, using the Winberry (2018) solution method, and extending Liu and Plagborg-Møller (2023), I estimate the Krusell and Smith (1998) model on empirical data using both micro wealth percentiles, and macro variables: output, investment, consumption and wages. The posterior variables estimated are β, the discount rate; μ_l, the labor productivity heterogeneity parameter; a different measurement error for each of the 4 macro variables; one more measurement error for the micro wealth process (meas_error 5); and a lower bound on net borrowing.

The macro data comes from FRED with the following codes: GDPC1 for output; FPI for investment; PCEC for consumption; COMPRNFB for wage; finally GDPDEF for deflating nominal variables to real variables. The micro data comes from the total household net worth variable in the Survey of Income and Program Participation (SIPP). One problem with the SIPP data is that the number of people interviewed ranges from 700 people to 1700 people per year. A common way to

deal with this is to match 100 samples with a 100 random sample from the data. This throws away information. Due to the simulation advantages of SNVI, I can match percentiles of the empirical data (i.e. the first to 99th percentiles), with model generated percentiles. This provides more information and implicitly uses the entire micro data set to discipline the model. If one wanted to match percentiles with a likelihood approach, one would have to calculate 99 order statistics, which is unmanageable. However, SNPE can still estimate in this situation via simulation, but the downside is that one can no longer determine the joint relationship of wealth and employment. One can still sum employment across agents to form a macro unemployment rate. Figure XII below shows the results of the empirical exercise:

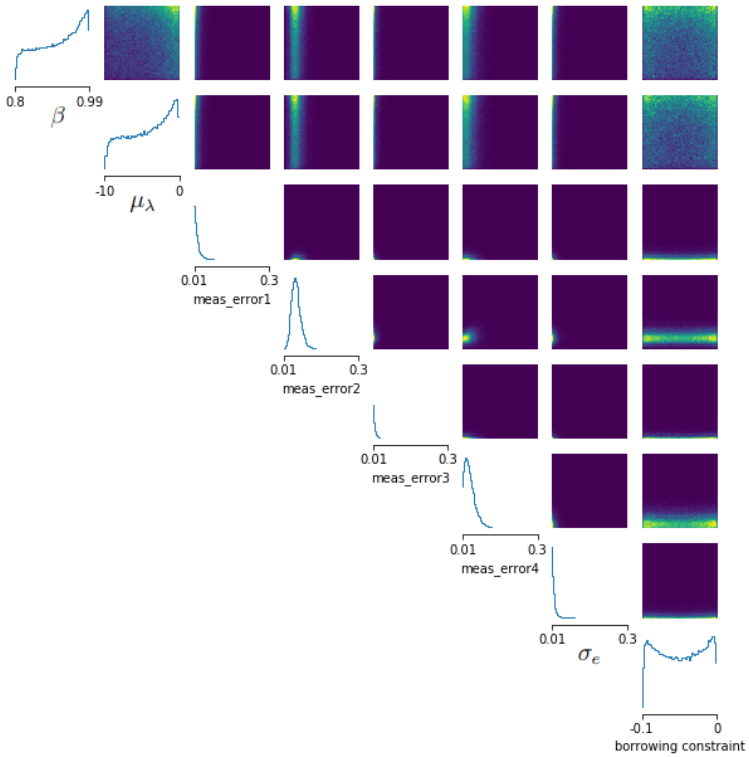

Figure XII: SNVI/SNPE Estimation of Krusell-Smith Empirical Results

The parameters of interest are somewhat diffuse, although the modes make sense. Despite practitioner's often calibrating the discount rate due to difficult with identification, the mode of the

discount rate estimated on empirical data is close to .99, which is both identified and realistic. The second term, the labor productivity parameters (μ_λ), with a mode around -.25, implies a standard deviation (heterogeneity) of the labor productivity process that Liu and Plagborg-Møller (2023) point out is in line with Piketty, Saez and Zucman (2018). The empirical exercise suggests this model has reasonable identification of the heterogeneous parameter of interest, μ_λ.

IVI Comparison of SNPE with Bayesian and Simulation-Based Alternatives

First, I will discuss SNPE compared to Bayesian alternatives for Bayesian estimation. Then, I will compare SNPE with simulation-based alternatives.

Compared to Bayesian algorithms like MCMC and HMC, SNPE doesn't require a likelihood and, as shown in Section IVE among others, often produces more accurate posteriors in medium to large models even when a likelihood can be calculated. Compared to SMC (Herbst and Schorfheide, 2014), SNPE is, again, likelihood-free and is often easier to code and utilize. Both SNPE and ABC (Rubin, 1984) are the only methods that can work as simulation-based Bayesian algorithms. However, ABC does not converge to the true posterior, unlike SNPE. Variational inference is another technique. It should theoretically be able to Bayesian estimate these models, perhaps faster than SNPE, but because it requires the use of gradients, it is difficult to apply it to structural models–especially heterogeneous models. Thus, currently the applications to structural economic models are limited, often only to reduced form models (Gefang, Koop and Poon, 2019). Additionally it requires a likelihood, which often can be relaxed at the expense of more computation (Mescheder, Nowozin and Geiger, 2017)

Compared to simulation-based algorithms like MSM (McFadden, 1989) and Indirect Inference (Smith Jr, 1993), (Gourieroux, Monfort and Renault, 1993), SNPE is efficient when the model is well specified, performing simulated maximum likelihood. Kristensen and Shin (2012) is a simulation-based maximum likelihood approach that is efficient, unlike MSM, but kernel density estimators of can only estimate distributions with a couple of dimensions. Section IVE demonstrates that SNPE can estimate 36 dimensional models. Traditional simulation-based maximum likelihood methods

(Lerman and Manski, 1981) require adding measurement errors, which changes the structure of the model. But as demonstrated in Section IVC, SNPE doesn't require measurement error. The method that is most similar to SNPE from a simulation-based perspective is Kaji, Manresa and Pouliot (2020). This is a simulation method that is efficient in well specified models, doesn't require measurement error, but does not precisely perform maximum likelihood, meaning the derived parameters will be different in misspecified models. As frequentist-first approach, this is likely to be less computationally expensive than SNPE. Ultimately, one advantage SNPE has over all the other approaches, with the potential exception of Indirect Inference, is that it can estimate a model with, for example, 200 to 300 time-steps as a single data point. This respects the state-space structure of solved structural models. See Section AC in the Appendix for more information.

In summary, SNPE can handle complex models and produces better estimates, be they posteriors or point estimates, than most competing methods.

V Conclusion

This paper introduces a novel machine learning approach, SNPE, to estimate models in a Bayesian and simulation-based frequentist framework. SNPE does not require a likelihood function and also derives posteriors faster and more accurately than baseline MCMC and other competing algorithms. The model can Bayesian estimate relatively large models, like the 36 parameter Smets and Wouters (2007) model. The algorithm has also been applied to HANK models with large latent state-spaces and has successfully produced credible posteriors estimates faster than baseline algorithms. Moreover, it can be applied to solution methods such as value function iteration and projection, which do not yield a likelihood function. This approach can be applied as broadly as MSM with improved efficiency and the ability to posterior estimate. The algorithm can take in most black box solution methods and estimate them as long as the models can produce simulated data.

The proposed approach has a range of applications beyond macroeconomics. It can be used in the fields of industrial organization, applied game theory, and econometrics. In industrial organi-

zation and game theory, the algorithm can be used to solve structural and dynamic models. In econometrics, it can be applied to estimate general latent time series models effectively, without the need for a likelihood function. Additionally, the suite of conditional density estimators can be used for density estimation in high-dimensional settings.

In conclusion, SNPE can estimate more models faster and more accurately out of the box without modifying the solution method. Additionally, it has a much broader scope of application than just macroeconomic modeling. The applications of this technique are not limited to this paper and it has the potential to be explored in greater depth in future research.

References

Adjemian, Stéphane, Houtan Bastani, Michel Juillard, Fréderic Karamé, Ferhat Mihoubi, Willi Mutschler, Johannes Pfeifer, Marco Ratto, Normann Rion, and Sébastien Villemot. 2022. "Dynare: Reference Manual Version 5." CEPREMAP Dynare Working Papers 72.

Ahn, SeHyoun, Greg Kaplan, Benjamin Moll, Thomas Winberry, and Christian Wolf. 2018. "When inequality matters for macro and macro matters for inequality." *NBER macroeconomics annual*, 32(1): 1–75.

Andrews, Isaiah, and Anna Mikusheva. 2015. "Maximum likelihood inference in weakly identified dynamic stochastic general equilibrium models." *Quantitative Economics*, 6(1): 123–152.

Auclert, Adrien, Bence Bardóczy, Matthew Rognlie, and Ludwig Straub. 2021. "Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models." *Econometrica*, 89(5): 2375–2408.

Bayer, Christian, and Ralph Luetticke. 2020. "Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation." *Quantitative Economics*, 11(4): 1253–1288.

Beskos, Alexandros, Dan Crisan, and Ajay Jasra. 2014. "On the stability of sequential Monte Carlo methods in high dimensions."

Bishop, Christopher M. 1994. "Mixture density networks."

Chan, Alan, Hugo Silva, Sungsu Lim, Tadashi Kozuno, A Rupam Mahmood, and Martha White. 2022. "Greedification operators for policy optimization: Investigating forward and reverse kl divergences." *The Journal of Machine Learning Research*, 23(1): 11474–11552.

Chewi, Sinho. 2023. "Log Concave Sampling."

Childers, David, Jesús Fernández-Villaverde, Jesse Perla, Christopher Rackauckas, and Peifan Wu. 2022. "Differentiable State-Space Models and Hamiltonian Monte Carlo Estimation." National Bureau of Economic Research.

Dixit, Avinash K, and Joseph E Stiglitz. 1977. "Monopolistic competition and optimum product diversity." *The American economic review*, 67(3): 297–308.

Draxler, Felix, Christoph Schnörr, and Ullrich Köthe. 2022. "Whitening Convergence Rate of Coupling-based Normalizing Flows." *Advances in Neural Information Processing Systems*, 35: 37241–37253.

Duffie, Darrell, and Kenneth J Singleton. 1990. "Simulated moments estimation of Markov models of asset prices."

Durkan, Conor, Iain Murray, and George Papamakarios. 2020. "On contrastive learning for likelihood-free inference." 2771–2781, PMLR.

Farrell, Max H, Tengyuan Liang, and Sanjog Misra. 2018. "Deep neural networks for estimation and inference: Application to causal effects and other semiparametric estimands." *arXiv preprint arXiv:1809.09953*.

Fernández-Villaverde, Jesús, and Juan F Rubio-Ramírez. 2007. "Estimating macroeconomic models: A likelihood approach." *The Review of Economic Studies*, 74(4): 1059–1087.

Gallant, A Ronald, and George Tauchen. 1996. "Which moments to match?" *Econometric theory*, 12(4): 657–681.

Gefang, Deborah, Gary Koop, and Aubrey Poon. 2019. "Variational Bayesian inference in large Vector Autoregressions with hierarchical shrinkage."

Glöckler, Manuel, Michael Deistler, and Jakob H Macke. 2021. "Variational methods for simulation-based inference."

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative adversarial nets." *Advances in neural information processing systems*, 27.

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

Haan, Peter, and Arne Uhlendorff. 2006. "Estimation of multinomial logit models with unobserved heterogeneity using maximum simulated likelihood." *The Stata Journal*, 6(2): 229–245.

Hajivassiliou, Vassilis A, et al. 1997. *Some practical issues in maximum simulated likelihood.* Suntory-Toyota International Centre for Economics and Related Disciplines ....

Han, Jiequn, Yucheng Yang, et al. 2021. "Deepham: A global solution method for heterogeneous agent models with aggregate shocks." *arXiv preprint arXiv:2112.14377*.

Hennessy, Christopher A, and Toni M Whited. 2007. "How costly is external financing? Evidence from a structural estimation." *The Journal of Finance*, 62(4): 1705–1745.

Herbst, Edward, and Frank Schorfheide. 2014. "Sequential Monte Carlo sampling for DSGE models." *Journal of Applied Econometrics*, 29(7): 1073–1098.

Herbst, Edward P, and Frank Schorfheide. 2015. *Bayesian estimation of DSGE models.* Princeton University Press.

Hermans, Joeri, Volodimir Begy, and Gilles Louppe. 2020. "Likelihood-free mcmc with amortized approximate ratio estimators." 4239–4248, PMLR.

Hoffman, Matthew D, Andrew Gelman, et al. 2014. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, 15(1): 1593–1623.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer feedforward networks are universal approximators." *Neural networks*, 2(5): 359–366.

Huang, Chin-Wei, David Krueger, Alexandre Lacoste, and Aaron Courville. 2018. "Neural autoregressive flows." 2078–2087, PMLR.

Kaji, Tetsuya, Elena Manresa, and Guillaume Pouliot. 2020. "An adversarial approach to structural estimation." *arXiv preprint arXiv:2007.06169*.

Kase, Hanno, Leonardo Melosi, and Matthias Rottner. 2022. "Estimating nonlinear heterogeneous agents models with neural networks."

Keane, Michael P, and Kenneth I Wolpin. 1994. "The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte Carlo evidence." *the Review of economics and statistics*, 648–672.

Khan, Aubhik, and Julia K Thomas. 2008. "Idiosyncratic shocks and the role of nonconvexities in plant and aggregate investment dynamics." *Econometrica*, 76(2): 395–436.

Koehler, Frederic, Viraj Mehta, and Andrej Risteski. 2021. "Representational aspects of depth and conditioning in normalizing flows." 5628–5636, PMLR.

Kristensen, Dennis, and Yongseok Shin. 2012. "Estimation of dynamic models with nonparametric simulated maximum likelihood." *Journal of Econometrics*, 167(1): 76–94.

Krusell, Per, and Anthony A Smith, Jr. 1998. "Income and wealth heterogeneity in the macroeconomy." *Journal of political Economy*, 106(5): 867–896.

Lee, Holden, Chirag Pabbaraju, Anish Prasad Sevekari, and Andrej Risteski. 2021. "Universal approximation using well-conditioned normalizing flows." *Advances in Neural Information Processing Systems*, 34: 12700–12711.

Lerman, Steven, and Charles Manski. 1981. "On the use of simulated frequencies to approximate choice probabilities." *Structural analysis of discrete data with econometric applications*, 10: 305–319.

Liu, Laura, and Mikkel Plagborg-Møller. 2023. "Full-information estimation of heterogeneous agent models using macro and micro data." *Quantitative Economics*, 14(1): 1–35.

Loaiza-Maya, Rubén, Michael Stanley Smith, David J Nott, and Peter J Danaher. 2022. "Fast and accurate variational inference for models with many latent variables." *Journal of Econometrics*, 230(2): 339–362.

McFadden, Daniel. 1989. "A method of simulated moments for estimation of discrete response models without numerical integration." *Econometrica: Journal of the Econometric Society*, 995–1026.

Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. 2017. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks." 2391–2400, PMLR.

Mohammad-Djafari, Ali, and Hacheme Ayasso. 2009. "Variational Bayes and mean field approximations for Markov field unsupervised estimation." 1–6, IEEE.

Nathoo, FS, ML Lesperance, AB Lawson, and CB Dean. 2013. "Comparing variational Bayes with Markov chain Monte Carlo for Bayesian computation in neuroimaging." *Statistical methods in medical research*, 22(4): 398–423.

Pakes, Ariel, and David Pollard. 1989. "Simulation and the asymptotics of optimization estimators." *Econometrica: Journal of the Econometric Society*, 1027–1057.

Papamakarios, George, and Iain Murray. 2016. "Fast ε-free inference of simulation models with bayesian conditional density estimation." *Advances in neural information processing systems*, 29.

Piketty, Thomas, Emmanuel Saez, and Gabriel Zucman. 2018. "Vol. 133 May 2018 Issue 2." *The Quarterly Journal of Economics*, 553: 609.

Reiter, Michael. 2009. "Solving heterogeneous-agent models by projection and perturbation." *Journal of Economic Dynamics and Control*, 33(3): 649–665.

Rendahl, Pontus. 2017. "Linear time iteration." IHS Economics Series.

Rubin, Donald B. 1984. "Bayesianly justifiable and relevant frequency calculations for the applied statistician." *The Annals of Statistics*, 1151–1172.

Siarry, Patrick, Gérard Berthiau, François Durdin, and Jacques Haussy. 1997. "Enhanced simulated annealing for globally minimizing functions of many-continuous variables." *ACM Transactions on Mathematical Software (TOMS)*, 23(2): 209–228.

Smets, Frank, and Rafael Wouters. 2007. "Shocks and frictions in US business cycles: A Bayesian DSGE approach." *American economic review*, 97(3): 586–606.

Smith Jr, Anthony A. 1993. "Estimating nonlinear time-series models using simulated vector autoregressions." *Journal of Applied Econometrics*, 8(S1): S63–S84.

Wainwright, Martin J, and Michael Irwin Jordan. 2008. *Graphical models, exponential families, and variational inference.* Now Publishers Inc.

Winberry, Thomas. 2018. "A method for solving and estimating heterogeneous agent macro models." *Quantitative Economics*, 9(3): 1123–1151.

Appendix

A Algorithmic Details: MCMC and SNPE

First, I will discuss a high level introduction of how MCMC woks. Then I will discuss multi-round SNPE inference, which speeds up SNPE and related algorithms. Finally, I will spend a little time introducing and discussing the ability of SNPE to adhere to the data generating process of a structural model where the entire time-series is a single data point generated by the state-space model which is the reduced form of any solved structural model.

AA MCMC

MCMC is the benchmark technique for Bayesian inference in macroeconomics (Herbst and Schorfheide, 2015). In MCMC, one knows both $P(X|\theta)$ and $P(\theta)$. Thus the numerator of the posterior, $P(X|\theta)P(\theta)$, is known. However,

$$P(X) = \int_{-\infty}^{\infty} P(X|\theta)P(\theta)d\theta \tag{38}$$

is usually intractable to calculate, as you have to integrate through a high dimensional continuous space, so one doesn't know the normalizing constant to ensure the numerator sums to one. MCMC gets around the intractable partition function by evaluating the likelihood ratios of $P(X|\theta)P(\theta)$ between different parameter values. For example, given two points θ and θ', since $P(X)$ is a constant, the relative ratio of the posterior likelihood of these two points is known, since $P(X)$ cancels out.

$$\frac{P(X|\theta')P(\theta')}{P(X|\theta)P(\theta)} \tag{39}$$

Since one knows the relative probabilities, one can design a random walk so that a walker visits θ points in proportion to their relative probabilities. One way to do that is if a walker can choose to

compare any point in the posterior to the point the walker is currently at, the walker will move to the second point with the likelihood according to the ratio of the probabilities. For example, if the second point is two- thirds as likely to be in the posterior, which would produce a likelihood ratio $\frac{2/3}{1}$, the walker will move with probability two-thirds to the second point and with probability one-third staying at the same point and generating another sample. If the other point has, for example, 50 percent more probability mass: $\frac{1.5}{1}$, the walker will move to the second point for certain. To adjust for the fact the ratio, $\frac{1.5}{1}$, should move with probability 1.5 – higher than one – the understanding is that if the walker were randomly sampled to move back, the move back would be with probability with $\frac{1}{1.5} = \frac{2/3}{1}$, and with probability of 1/3 to stay at the original point and generate another sample. Figure XIII shows the random walk algorithm exploring a Gaussian distribution including some samples that were rejected due to low probability of occurring.

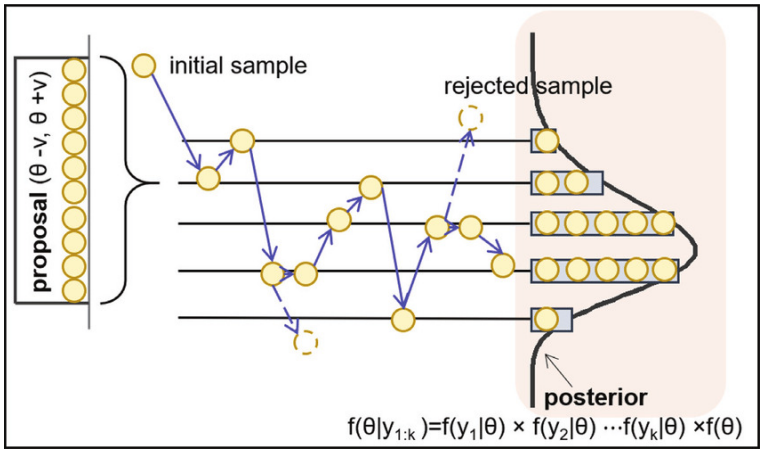

Figure XIII: Random Walk MCMC

In this way, the walker moves across the space of the posterior with relative frequency according to $P(X|\theta)P(\theta)$ ratio between points, which ultimately means that one is sampling from $P(\theta|X)$ without calculating the normalizing partition function. In practice, MCMC is a little more complicated, as often the parameter space is unbounded, and one can't sample from the entire space with equal probability. Thus, one uses a proposal distribution that might have a Gaussian probability to sample MCMC points that are nearer with higher probability. Because of this unequal sampling,

one need to reweigh the proposed point, using intuition similar to importance sampling, to take into account the fact that nearer points are more likely to be selected than further points. This allows MCMC to sample, in the long run, from the entire space according to their relative probabilities. In particular, call $P(X|\theta)P(\theta)$ the unnormalized posterior for a point θ, and $P(X|\theta')P(\theta')$ the unnormalized posterior for a point θ'. Assume a proposal distribution, $f(\theta|\theta')$, which is the probability of visiting θ from θ', and $f(\theta'|\theta)$ which is the probability of visiting θ' from θ. Then the expression that reflects the relative likelihood ratios and samples from the posterior will have a probability of moving to point θ' of:

$$min\{\frac{\frac{P(X|\theta')P(\theta')}{f(\theta'|\theta)}}{\frac{P(X|\theta)P(\theta)}{f(\theta|\theta')}}, 1\} \tag{40}$$

Since often the f distribution is assumed to be symmetric, the f terms drop out leaving the ratios of the unnormalized probabilities used to decide when to walk. Unsurprisingly, this approaches is quite slow as one has to visit the whole space sequentially with higher probability points being visited many more times than lower probability points. Alternatives like variational inference have been proposed which speed up the inference at the cost of some bias (Wainwright and Jordan, 2008), but it is beyond the scope of this paper. SNPE does not require knowing the likelihood, something that may be intractable to calculate depending on the solution algorithm for macroeconomic models.

AB Multi-round SNPE Inference

Beyond this explanation of how SNPE works, there are, in practice, some addition steps to improve the estimation speed of the algorithm. For example, convergence is quicker if one uses multi-round inference to sample simulated data X in the neighborhood of the real data. Instead of sampling from the prior, θ is sampled from the best approximation of the posterior, leading to more relevant samples of X when it is sampled from the model conditioned on θ. However, this process requires importance sampling to correct for the shift in distribution as θ is no longer being drawn from the prior. The first round entails sampling from the prior, while later rounds involve

sampling from the updated estimate of the posterior and applying importance sampling to account for the distribution shift. This can be mathematically expressed as:

$$P'(\theta|X) = P(\theta|X)\frac{P'(\theta)}{P(\theta)} \tag{41}$$

Here $P'(\theta)$ is the proposed distribution that one is sampling from, $P(\theta)$ is the true prior, and $P'(\theta|X)$ is the estimated conditional distribution obtained by sampling from the proposal distribution instead of the prior. The use of a flexible machine learning density estimator, such as a normalizing flow or a conditional mixture of Gaussians (discussed in Section IIIC of the Results and Section DA of the Appendix respectively), allows for the unbiased estimation of the posterior. This density estimator will be called $P_\phi(\theta|X)$, where the ϕ indicates that this estimator has parameters and is attempting to approximate the conditional distribution $P(\theta|X)$ by modifying ϕ. When one samples from $P'(\theta)$ instead of the prior $P(\theta)$, the distribution to which one's conditional density estimator converges is $P'(\theta|X)$ from ((41)). If one wants a density estimator to approximate $P(\theta|X)$, $P_\phi(\theta|X)$ will converge to $P(\theta|X)$ after adjusting the sample likelihood by the factor $\frac{P'(\theta)}{P(\theta)}$. See Appendix A of Papamakarios and Murray (2016) for a proof and more information.

AC State-Space Assumption

Unlike most other simulation approaches like Kaji, Manresa and Pouliot (2020), Kristensen and Shin (2012), McFadden (1989), and Gourieroux, Monfort and Renault (1993), SNPE respects the state-space generating structure of macro models, treating the entire time-series as a single data point. A solved structural model has a state-space reduced-form structure. This means the data set should be treated as a single observation with hundreds of time-steps. However, most other simulation approaches attempt to estimate structural models one time-step at a time in conjunction with a small number of auto-regressive lags, much like how a vector auto-regression is estimated, contrary to the state space assumption behind the data in a structural model.

AD Feed-Forward Neural Networks

At its core, feed-forward neural networks are compositions of vector-valued linear equations with non-linearities between each composition. For example, a neural network with a single hidden layer could be constructed:

$$Y = W_2 * \sigma(W_1 * X + b_1) + b_2 \tag{42}$$

Here, X is some vector-valued input with dimensionality n by assumption. W_1 is a k by n matrix that turns an n-dimensional input into a k-dimensional basis for the second layer. Likewise, b_1 is a k-dimensional intercept term. W_2 turns the k-dimensional basis vector from the first linear regression into the m dimensional output vector and b_2 is the m-dimensional intercept term. Since W_1, W_2, b_1, b_2, are all linear operations, arbitrary compositions of them will still be linear. However, if you add a non-linearity term, σ, between the compositions, with loose regularity conditions, a single hidden layer neural network as shown above can approximate any continuous function (Hornik, Stinchcombe and White, 1989), as the k-dimensional basis expands to infinity. The non-linearity can be almost any function. The one I use is the Leaky ReLU, which is shown as the σ function applied to the first vector equation, $W_1 * X + b_1$, below in Figure XIV:

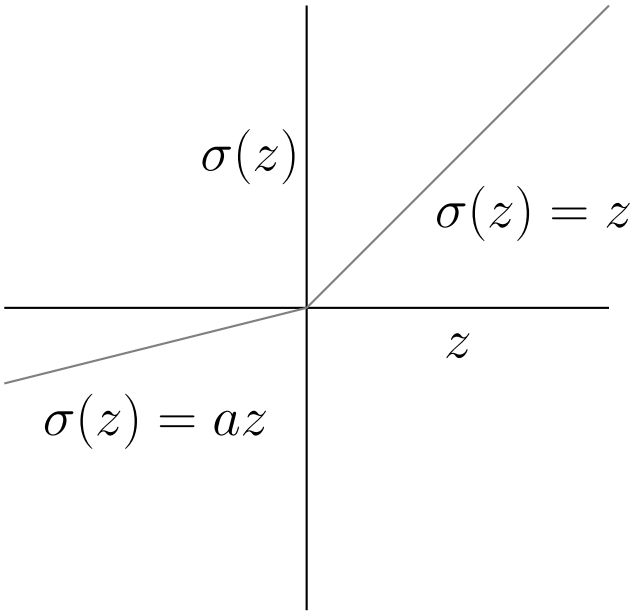

Figure XIV: Leaky ReLU as a Neural Network Nonlinearity

This transforms the output of $W_1 * X + b_1$, whose values span the x axis, by the kinked nonlinearity, to produce the y axis output in order to make the neural network more expressive. Other non-linearities include Logits, hyperbolic tangents, ReLUs, and ELUs, which are modifications of the Leaky ReLU. Additionally, one can make deeper neural networks by composing more vector linear regressions with non-linearities, parameterized by W_3 and b_3, for example.

Here is a graphical representation of layers in a neural network stacked one on top of another. Usually feed-forward neural networks have more than one hidden layer:

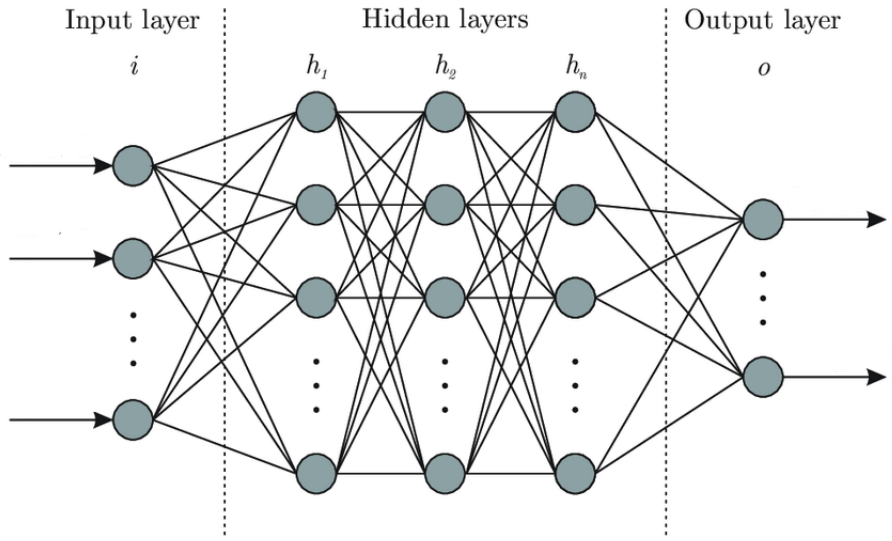

FIGURE XV: FEED FORWARD NEURAL NETWORK

In this case, there are three hidden layer in combination with the input vector and the output vector. These layers can be arbitrarily wide, with width and depth associated with increased flexibility, but also increased parameter count with the risk of over-fitting. A feed-forward neural network can be monotonically increasing in all its elements if every parameter value is positive (Huang et al., 2018).

B PROOFS AND PROPERTIES OF NORMALIZING FLOWS

BA Invertibility of a Bijector

Proof: In order to prove invertiblity, one must show the function is 1-1 and onto:

1-1: For each $z_{k+1} \in Z_{k+1}$ there is at most one $z_k \in Z_k$ with $f(z_k) = z_{k+1}$.

Assume by contradiction that you have two different vectors, z_k and z_k^* that produce the same $z_{k+1} = f(z_k) = f(z_k^*)$. Then there must be at least one element in which z_k differs from z_k^*. Call the jth element of z_k^j and z_k^{j*} the first such element that is different. Then we have the jth

relationship, for z_{k+1}^j using z_k^j and z_k^{j*}:

$$z_{k+1}^j = a_k^j(z_k^1, z_k^2 ... z_k^{j-1}) * z_k^j + b_k^3(z_k^1, z_k^2 ... z_k^{j-1}) \tag{43}$$

$$z_{k+1}^j = a_k^j(z_k^{1*}, z_k^{2*} ... z_k^{(j-1)*}) * z_k^{j*} + b_k^3(z_k^{1*}, z_k^{2*} ... z_k^{(j-1)*}) \tag{44}$$

Because z_k^j is the element that is the first such difference, then

$$z_k^1, z_k^2 ... z_k^{j-1} = z_k^{1*}, z_k^{2*} ... z_k^{(j-1)*} \tag{45}$$

where the above equation states that every element of z_k up to element $j-1$ is equal to the $j-1$ elements of z_k^*. Since this is true then $a_k^j(z_k^1, z_k^2 ... z_k^{j-1})$ an $b_k^j(z_k^1, z_k^2 ... z_k^{j-1})$ are the same as $a_k^j(z_k^{1*}, z_k^{2*} ... z_k^{(j-1)*})$ and $b_k^j(z_k^{1*}, z_k^{2*} ... z_k^{(j-1)*})$. Call the value these functions equal a and b

By assumption:

$$a * z_k^j + b = z_{k+1}^j = a * z_k^{j*} + b \tag{46}$$

$$and\ z_k^j \neq z_k^{j*} \tag{47}$$

Since a_k^j is a neural network function with all positive parameters, a cannot be zero. In that case, the linear relationship is a bijection, thus if z_{k+1}^j is the same, z_k^{*j} and z_k^j must be the same. This proves that z_k is the same as z_k^*, as they cannot have a first such element that is different. This proves the function is 1-1.

Onto: For each $z_{k+1} \in Z_{k+1}$ there is at least one $z_k \in Z_k$ with $f(z_k) = z_{k+1}$.
Knowing the first element of z_{k+1}, the relationship:

$$z_{k+1}^1 = a_k^1 * z_k^1 + b_k^1 \tag{48}$$

This is a linear equation and can be inverted:

$$z_k^1 = \frac{z_k^1 - b_k^1}{a_k^1} \tag{49}$$

Thus from z_{k+1}^1, you can obtain a z_k^1. The equation for the second element of z_{k+1}, z_{k+1}^2 is defined:

$$z_{k+1}^2 = a_k^2(z_k^1) * z_k^2 + b_k^2(z_k^1) \tag{50}$$

Since z_k^1 is already known, $a_k^2(z_k^1)$ and $b_k^2(z_k^1)$ are also known and thus one can obtain a z_k^2 from any z_{k+1}^2. By induction, if you know z_k^1, z_k^2 to z_k^{j-1} as well as z_{k+1}^j, a_k^j and b_k^j are only functions of z_k^1, z_k^2 to z_k^{j-1} and so are known constants a_j and b_j. Thus inverting the linear relationship will obtain a z_k^j from a z_{k+1}^j. QED

BB Deriving the Likelihood Function of a Flow

Proof: First, since each f_k is a bijector, $z_k \sim Q(z_k)$ and $z_{k+1} \sim f_k(z_k)$ and $z_{k+1} \sim P(z_{k+1})$ and $z_k \sim f_k^{-1}(z_{k+1})$ are both absolutely continuous with respect to one another. Thus, f an f^{-1} are both valid change in measure. By the Radon-Nikodym theorem, the likelihood of $P(z_{k+1}) = f_k(z_k)$ s.t. $z_k \sim Q(z_k)$ is:

$$P(z_{k+1} = f_k(z_k)) = Q(z_k = f_i^{-1}(z_{k+1}))|det\frac{df_k^{-1}(z_{k+1})}{dz_{k+1}})| = Q(z_k)|det\frac{df_k(z_k)}{dz_k}|^{-1} \tag{51}$$

Applying the Radon-Nikodym theorem to each f_k in the composition of f_k's that make up the normalizing flow and applying the natural log gives an analytical formula for the log likelihood:

$$ln(P(z_K)) = ln(Q(z_0)) - \sum_{k=0}^{k=K-1} ln(|det\frac{df_k(z_k)}{dz_k}|) \tag{52}$$

Keep in mind, one can get from z_{k+1} to z_k via f_k^{-1} from the onto section of Proposition 1, thus one calculate z_0 from z_K' by inverting f_k^{-1} for every k. QED

C Economic Models

SNPE and other baseline estimation algorithms are applied to a suite of economic models. This section will discuss the economic details of the models being estimated.

CA SNPE Estimation of a Real Business Cycle (RBC) Model

The results of estimating the RBC model using the SNPE algorithm are presented and discussed. The RBC Model is standard in macroeconomics. This version has four parameters – α, β, δ, and ρ – with a fixed Constant Relative Risk Aversion (CRRA) elasticity of 2. The first 100 iterations are dropped to attain a steady-state behavior. This RBC model comes from Alisdair McKay's note on heterogeneous agent models: `https://alisdairmckay.com/Notes/HetAgents/index.html`.

Additional results of the posterior are displayed below in Figure XVI, where the red bar represents the true parameter values and the blue line represents the posterior as estimated by the simulation-based inference approach after 11 thousand iterations:

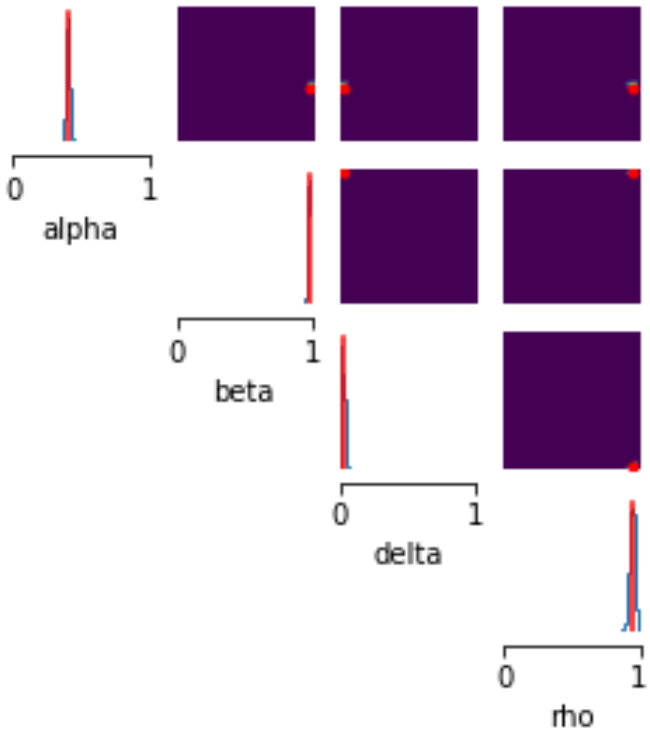

Figure XVI: SNPE: Simulated Data RBC

The main diagonal density plots in the figure shows the posterior marginals as blue curves. The off-diagonal heat maps show two-way densities, with the color indicating the probability mass. As you can see, the posterior concentrates on the true data-generating parameter, which is indicated by the red lines.

The approximate true posterior is obtained by running the MCMC for 2 million iterations as show below:

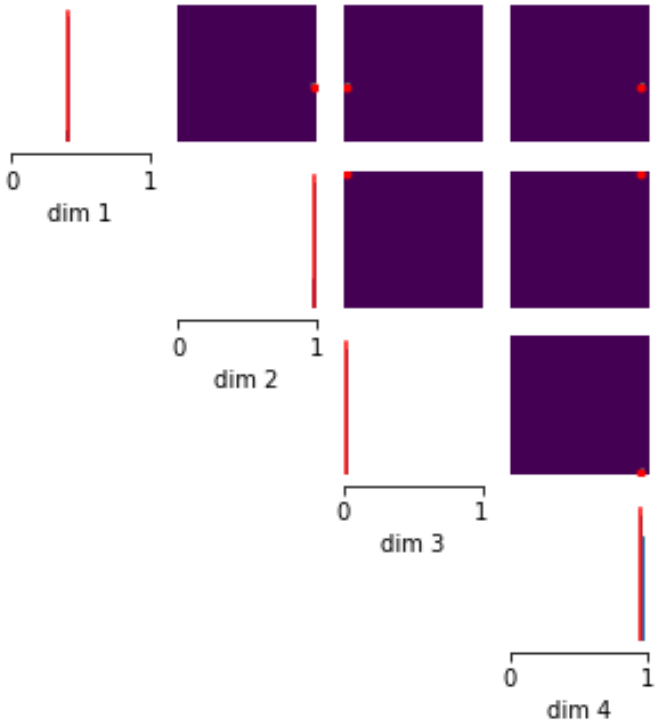

FIGURE XVII: MCMC: SIMULATED DATA RBC

As can be seen, SNPE converges to the MCMC ground truth solution, with significantly less iterations.

Figure XVIII below shows the same RBC model estimated on real data where there is no available ground truth. The parameters derived from the real data are not necessarily realistic, which is expected as the RBC model is too simple. The estimation approach is uncertain about identification as indicated by the spread of the posterior. The unrealistic values include the discount rate, β, and the auto-regressive parameter on the productivity process, ρ. The depreciation rate, δ, is the only parameter that is relatively reasonable. Below is the posterior estimated on real data:

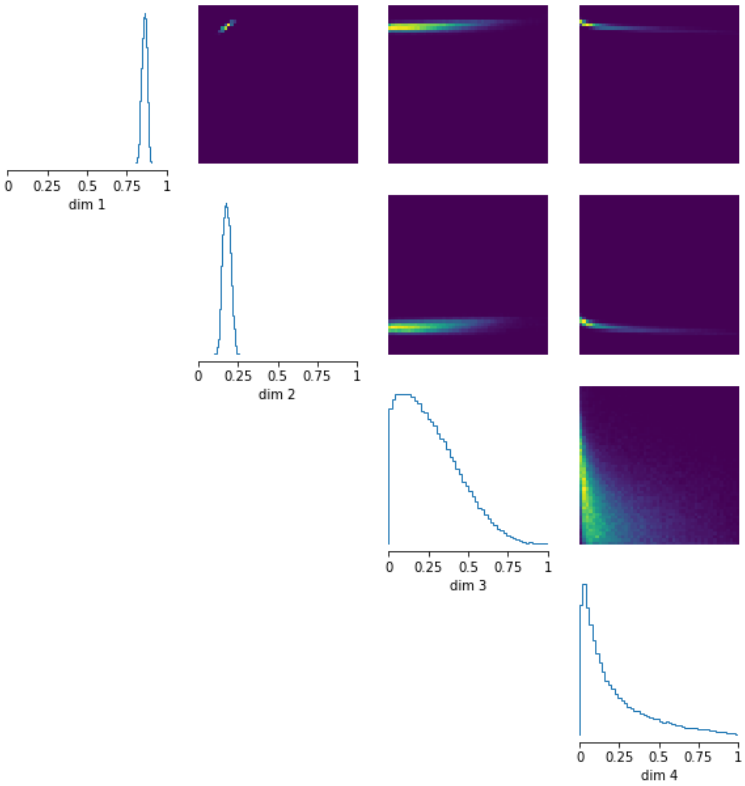

FIGURE XVIII: SNPE ESTIMATING RBC WITH REAL DATA

In conclusion, the results indicate that the SNPE algorithm can estimate the RBC model to a good degree of accuracy. The RBC model's limitations in real-world applications are also highlighted, as it fails to estimate some parameters that are not in line with theoretical predictions. This highlights the need for more complex models to better reflect real-world economic scenarios, which will be illustrated in the next models.

CB SNPE Estimation of a Investment Model

In this section, I am estimating a partial equilibrium model of firm investment by utilizing value function iteration. While this is not quite a macro model, as a structural approach the techniques used are the same. The model is specified with four parameters – α, δ, σ, and ρ. α is the capital

share in the production function. δ is depreciation rate of capital. σ is the standard deviation of log productivity AR process. ρ is the slope of the log productivity AR process. The interest rate was set at 5 percent and the discount factor was calculated as 1/(1+r).

This model is a straightforward infinite horizon partial equilibrium model with risk neutral managers choosing investment to maximize the present value of a stream of firm cash flows and comes from solution code written by Emil Lakkis and Toni Whited.

Utility/cash flows are give by this formula:

$$U(K_t, I_t, Z_t) = \sum_t \beta^t * (\pi(K_t, Z_t) - I_t) \tag{53}$$

Where K_t, Z_t, and I_t are the same as in the RBC model – capital, productivity and investment respectively. π is the profit function.

The capital law of motion is like the RBC model:

$$K_t = (1 - \delta)K_{t-1} + I_t \tag{54}$$

Productivity evolves according to a AR(1) process in logs:

$$logZ_t = \rho logZ_{t-1} + \epsilon_t \tag{55}$$

Cash flow is defined as:

$$\pi(K_t, Z_t) = Z_t K_t^{\alpha} \tag{56}$$

This model shares similarities with the RBC model, with the main difference being the utility function and the value function solution method. This estimation approach matches capital levels generated by a parameterized model. To estimate the model parameters, two different algorithms were applied – a simulation-based inference approach using SNPE, and SNVI. SNVI is discussed further in the Section DD of the Appendix. Both algorithms are estimating the same posterior,

thus it's reassuring that the results are similar. The results are shown Figure XIX (SNPE) and Figure XX (SNVI):

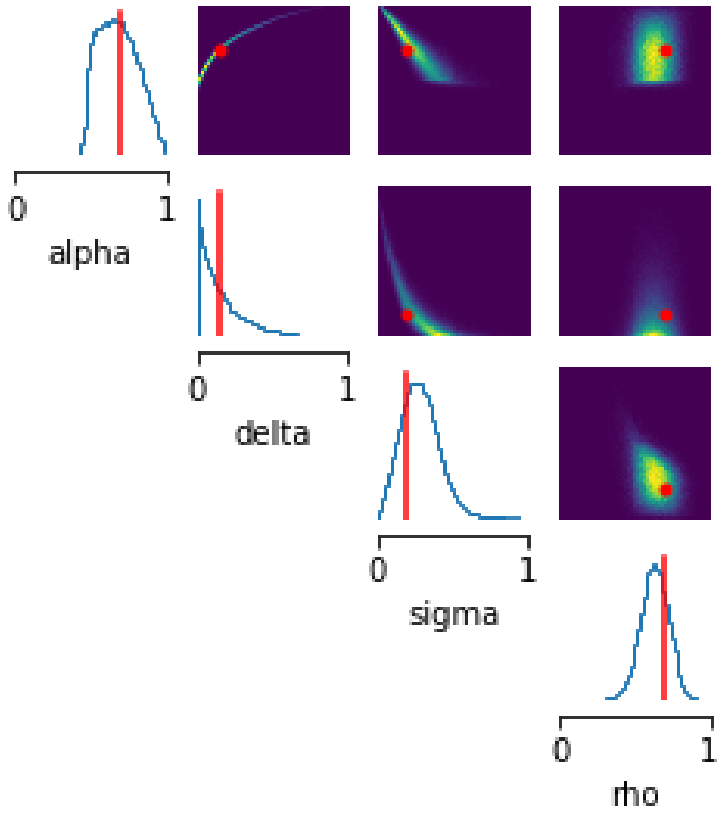

Figure XIX: Value Function Iteration Estimated with SNPE

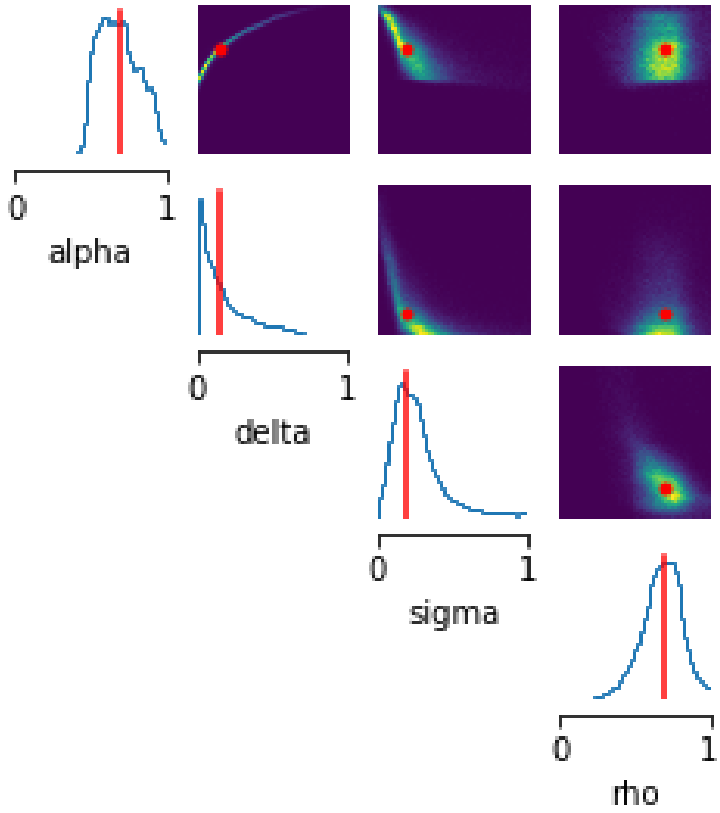

Figure XX: Value Function Iteration Estimated with SNVI

From the visual inspection of the charts XIX and XX, it can be observed that the posteriors obtained from the two estimation approaches are similar, suggesting that the same distribution was learned. Additionally, the mode of the distribution appears to be close to the actual parameter values, which indicates that the model is capable of accurately recovering the true parameters.

CC SNPE Estimation of a Lucas Asset Pricing Projection Model

In the following section, we demonstrate the estimation procedure on a Lucas Asset Pricing model, which is solved via projection without measurement error. Following Mohammed Ait Lahren's code (`https://notes.quantecon.org/submission/5f44808ad24fdb001162a53b`), each

agent maximizes CRRA utility:

$$max_i E_t[\beta^t \frac{c_t^{1-\gamma} - 1}{1-\gamma} \tag{57}$$

with γ set to 3. The law of motion is:

$$k_{t+1} = (1 + d_t/P_t)k_t - c_t/P_t \tag{58}$$

The state of the system is defined as dividends received plus price of the amount of assets owned:

$$m_{t+1} = (P_{t+1} + d_{t+1})k_{t+1} \tag{59}$$

Putting these three equations together one gets the pricing kernel:

$$p_t = \beta^t E_t[(\frac{c_{t+1}}{c_t})^{-\gamma}(d_{t+1} + p_{t+1})] \tag{60}$$

Dividends evolve under an $AR(1)$ process:

$$d_{t+1} = \mu_d + \rho_d * d_t + \epsilon \tag{61}$$

where ϵ is a shock with standard deviation σ. When behaving optimally, consumption also equals the dividends received. This model matches the expected return and risk free rate at a variety of dividend levels, which demonstrates the ability of this model to not only estimate on models solved via projection, but also to work on cross sectional problems as well.

The purpose of this estimation is to showcase the capability of the SNPE approach to estimate models solved via projection. The results are show in Figure XXI:

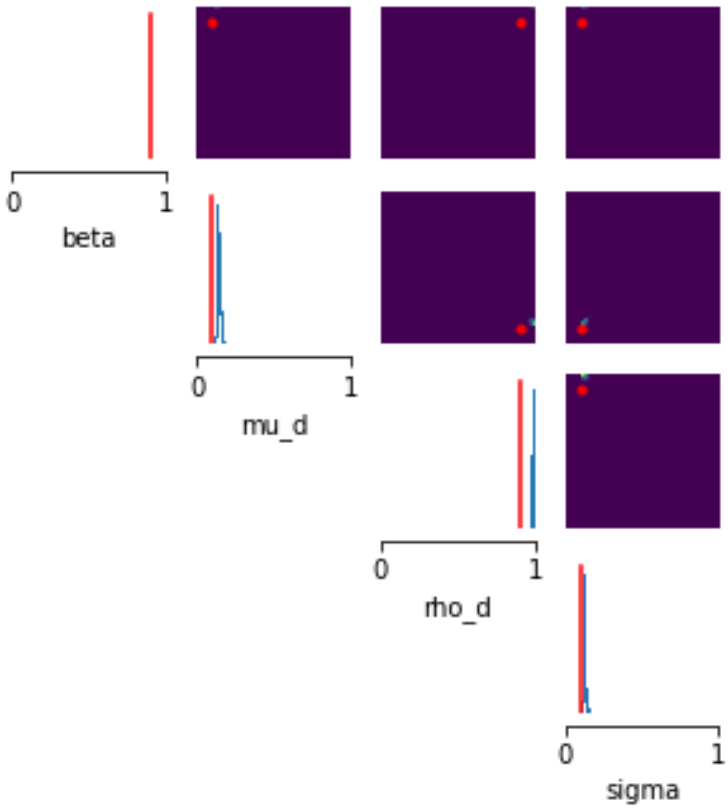

FIGURE XXI: SNPE ESTIMATION OF LUCAS TREE PROJECTION MODEL

This estimation procedure took under 10 hours to complete on an 8-core Intel i7 machine, highlighting the computational efficiency of this approach. This result also demonstrates that models solved via projection can be effectively estimated without measurement error using the SNPE framework. The posterior accuracy is pretty good, but it seems like the posterior marginals have a mode that is slightly shifted and while close to the true parameter value, do not put much support on the true value along the μ_d and ρ_d dimensions.

CD SNPE Estimation of a Value Function Iteration Solved Life Cycle Model

This section shows an estimate of a life cycle model with bequests over time. The model illustrates the benefit of the SNPE approach as it cannot be estimated by perturbation because of the kink at the intersection of the consumption versus adjusting illiquid savings. Thus one should use simulation approaches to Bayesian estimate this model, which is solved via value function iteration.

This model comes from Jeppe Druedahl's open source model example: `https://github.com/pkofod/vfi/blob/master/Fast%20VFI.ipynb`. The utility function is:

$$u(b_t, c_t) = \frac{[\phi(b_t + \underline{b})^{1-\gamma} + (1-\phi)c_t^{1-\gamma}]^{\frac{1-\rho}{1-\gamma}}}{1-\rho} \tag{62}$$

Here b_t is bequest or non-liquid savings. c_t is consumption. b, ϕ, γ, ρ are all parameters. The bequest utility function (end of life utility) is:

$$\nu(a_t, b_t) = \varphi \frac{(a_t + b_t + \underline{q})^{1-\vartheta}}{1-\vartheta} \tag{63}$$

Here a_t is liquid savings and q, θ, φ are all parameters. The value function is give by:

$$v_t(m_t, n_t, l_t) = \max\{v_t^{keep}(m_t, n_t, l_t), v_t^{adj}(m_t, n_t, l_t)\} \tag{64}$$

$$\text{s.t.} \tag{65}$$

$$x_t = m_t + (1-\tau)n_t \tag{66}$$

Here m_t is beginning of period liquid savings (analogous to end of period a_t) and n_t is beginning of period illiquid savings (analogous to end of period b_t). x_t is the pooled value of total liquid and

illiquid savings. l_t is labor income. The post-decision value function is:

$$w_t(a_t, b_t, l_t) = \nu(a_t, b_t), t = T \tag{67}$$

$$w_t(a_t, b_t, l_t) = \mathbb{E}_t \left[\max\{v_t^{keep}(m_{t+1}, n_{t+1}, l_{t+1}), v_t^{adj}(x_{t+1}, l_{t+1})\} \right], t < T \tag{68}$$

$$\text{s.t.} \tag{69}$$

$$l_{t+1} \sim F(l_t) \tag{70}$$

$$m_{t+1} = (1 + r_a)a_t + \omega l_{t+1} \tag{71}$$

$$n_{t+1} = (1 - \delta)b_t \tag{72}$$

$$x_{t+1} = m_{t+1} + (1 - \tau)n_{t+1} \tag{73}$$

Here labor income comes from a distribution $F(l_t)$. Likewise r_a, ω, δ, τ are all parameters. The keep value function is:

$$v_t^{keep}(m_t, n_t, l_t) = \max_{c_t \in [0, m_t]} u(n_t, c_t) + \beta w_t(a_t, b_t, l_t) \tag{74}$$

$$\text{s.t.} \tag{75}$$

$$a_t = m_t - c_t \tag{76}$$

$$b_t = n_t \tag{77}$$

The keep value function is the value function at a point t if the agent chooses to consume and not withdraw or deposit funds into the illiquid asset, b_t. The adjust value function is:

$$v_t^{adj}(x_t, l_t) = \max_{b_t \in [0, x_t]} v_t^{keep}(m_t, n_t, l_t) \tag{78}$$

$$\text{s.t.} \tag{79}$$

$$m_t = x_t - b_t \tag{80}$$

$$n_t = b_t \tag{81}$$

The adjust value function is the value of adjusting the illiquid asset. Notice the agent is unable

to consume this period. This cannot be effectively modeled with a approach like perturbation, because of its kinked decision rule. In contrast, SNPE can obtain Bayesian posterior distribution. This model matches the liquid asset, illiquid asset and consumption to synthetic data generated by a model. The chart is Figure V in Section IVD.

CE SNPE Estimation of a Smets-Wouters 2007 Model

This section will mainly illustrate the equations and the parameters so one can cross reference parameters shown in the diagram with the correct equation in the DSGE model. This model comes from the Smets and Wouters (2007) paper. More details on the economics of the model can be found in the paper. This model can be estimated both with MCMC and SNPE. Thus, this model illustrates posterior accuracy even when using an uninformative prior, in contrast with MCMC, which goes hand in hand with estimation speed when using SNPE vs MCMC.

Here is a reference table for all the Smets-Wouter abbreviations and the actual variable name of posterior marginal distributions, written in order of appearance contained later on in the charts of this section. The bounds and the true calibrated values are also listed for ease of use, although they can be also recovered in any of the graphs by looking at the x axis and the red line in the respective marginal distribution plot that corresponds to that parameter. The first twelve variables are first pictured in Figure VI of Section IVE in the Results. Then modifications are presented in later charts in that section.

Variable Abbrev	Variable Name	Calibrated Value	Lower Bound	Upper Bound
calfa	Capital Share	.24	.01	1.0
csigma	Elasticity of Consumption	1.5	.25	1.5
cfc	One Plus Fixed Costs	1.5	1.0	3.0
cgy	Productivity Shock Effect on Government Spending	.51	.01	2.0
csadjcost	Capital Adjustment Cost Elasticity	6.0144	2.0	15
chabb	Habit Formation	.6361	.001	.99
cprobw	Degree of Wage Stickiness	.8087	.5	.95
csigl	Elasticity of Labor Supply	1.9423	.25	10
cprobp	Degree of Price Stickiness	.6	.5	.95
cindw	Wage Indexation	.3243	.01	.99
cindp	Past Inflation Indexation	.47	.01	.99
czcap	Capital Adjustment Cost Function	.2696	.01	1.0
crpi	Taylor Rule Inflation Coefficient	1.488	1.0	3.0
crr	Taylor Rule Degree of Interest Rate Smoothing	.8762	.5	.975
cry	Taylor Rule Output Gap Coefficient	.0593	.001	.5
crdy	Taylor Rule Output Growth Gap Coefficient	.2347	.001	.5
crhoa	AR Term on Productivity Shocks	.9977	.01	.9999
crhob	AR Term on Risk Premium Shocks	.5799	.01	.9999
crhog	AR Term on Government Spending Shocks	.9957	.01	.9999
crhoqs	AR Term in Investment Shocks	.7165	.01	.9999
crhoms	AR Term in the Taylor Rule Shock	.015	.01	.9999
crhopinf	AR Term in Inflation Rate Shock	.015	.01	.9999
crhow	AR Term in Wage Shock	.015	.001	.9999
cmap	Price Markup Shock to Inflation Moving Average	.015	.01	.9999
cmaw	Price Markup Shock to Wage Moving Average	.015	.01	.9999
constepinf	Trend Inflation Growth Rate	.7	.1	2.0
constebeta	Constant Inverse Discount Rate	.7420	.01	2.0
constelab	Steady State Hours Worked	0.0	-10	10
ctrend	Common GDP, Consumption, Investment, and Wages Trend	.3982	.1	.8
ea	Stdev of Productivity Shock	.4618	.01	3.0
eb	Stdev of Risk Premium Shocks	1.8513	.025	5.0
eg	Stdev of Government Spending Shock	.6090	.01	3.0
eqs	Stdev of Investment Shock	.6017	.01	3.0
em	Stdev of Taylor Rule Shock	.2397	.01	3.0
epinf	Stdev of Inflation Shock	.1455	.01	3.0
ew	Stdev of Wage Shock	.2089	.01	3.0

First, a brief discussion of the overview of the model. The model is a Representative Agent New Keynesian model with stickiness in both wages and prices. The model has a flexible price set

of equations which determines potential GDP for monetary policy and a sticky price system with a price markup equation that determines the true economy. I will only be discussing the sticky price system of equations with an implicit understanding that there is a mirroring flexible price system. Shocks all have auto-regressive components to more accurately match data. All equations are linearized.

The aggregate resource constraint is:

$$y_t = ccy * c_t + ciy * i_t + \epsilon_t^g + 1 * crkky * z_t \tag{82}$$

The Euler equation is given by:

$$\begin{aligned} c_t =& (chabb/cgamma)/(1 + chabb/cgamma) * c_{t-1} + (1/(1 + chabb/cgamma)) * c_{t+1} + \\ & ((csigma - 1) * cwhlc/(csigma * (1 + chabb/cgamma))) * (l_t - l_{t+1}) - \\ & (1 - chabb/cgamma)/(csigma * (1 + chabb/cgamma)) * (r_t - \pi_{t+1}) + \epsilon_t^b \end{aligned} \tag{83}$$

The investment Euler equations are:

(84)

$$i_t = (1/(1+cbetabar*cgamma))*(i_{t-1}+cbetabar*cgamma*i_{t+1}+(1/(cgamma^2*csadjcost))*q_t)+\epsilon_t^i$$

The arbitrage equation for the value of the capital stock is:

$$\begin{aligned} q_t =& -r_t + \pi_{t+1} + (1/((1 - chabb/cgamma)/(csigma * (1 + chabb/cgamma)))) * \epsilon_t^b + \\ & (crk/(crk + (1 - ctou))) * r_{t+1}^k + ((1 - ctou)/(crk + (1 - ctou))) * q_{t+1} \end{aligned} \tag{85}$$

The linearized production function is:

$$y_t = cfc * (calfa * k_t + (1 - calfa) * l_t + \epsilon_t^a) \tag{86}$$

Capital utilization:

$$k_t^s = k_{t-1} + z_t \tag{87}$$

Capital utilization is some function of the interest rate for capital:

$$z_t = (1/(czcap/(1 - czcap))) * r_t^k \tag{88}$$

Capital's law of motion is:

$$k_t = (1 - cikbar) * k_{t-1} + cikbar * i_t + cikbar * cgamma^2 * csadjcost * \epsilon_t^i \tag{89}$$

The markup equation is:

$$\mu_t^p = calfa * k_t^s + (1 - calfa) * w_t - \epsilon_t^a \tag{90}$$

This equation does not have a flexible price counterpart. Note that the observed variable equations also will not have a flexible price counterpart.

The New Keynesian Phillips curve:

(91)

$$\begin{aligned}\pi_t =&(1/(1 + cbetabar * cgamma * cindp)) * (cbetabar * cgamma * \pi_{t+1} + cindp * \pi_{t-1} + \\ &((1 - cprobp) * (1 - cbetabar * cgamma * cprobp)/cprobp)/((cfc - 1) * curvp + 1) * (\mu_t^p)) + \\ &\epsilon_t^p\end{aligned}$$

The rental rate of capital equation:

$$r_t^k = w_t + l_t - k_t \tag{92}$$

The path of wages is:

(93)

$$w_t = (1/(1 + cbetabar * cgamma)) * w_{t-1} + (cbetabar * cgamma/(1 + cbetabar * cgamma)) * w_{t+1} +$$
$$(cindw/(1 + cbetabar * cgamma)) * \pi_{t-1} - (1 + cbetabar * cgamma * cindw)/(1 + cbetabar * cgamma) * \pi +$$
$$(cbetabar * cgamma)/(1 + cbetabar * cgamma) * \pi_{t+1} +$$
$$(1 - cprobw) * (1 - cbetabar * cgamma * cprobw)/((1 + cbetabar * cgamma) * cprobw) *$$
$$(1/((clandaw - 1) * curvw + 1)) * (csigl * l_t + (1/(1 - chabb/cgamma)) * c_t -$$
$$((chabb/cgamma)/(1 - chabb/cgamma)) * c_{t-1} - w_t) + \epsilon_t^w$$

The Taylor rule followed by central banks is:

(94) $r_t = crpi * (1 - crr) * \pi_t + cry * (1 - crr) * (y_t - y_t^p) + crdy * (y_t - y_t^p - y_{t-1} + y_{t-1}^p) + crr * r_{t-1} + \epsilon_t^r$

The observed variables equations are:

(95) $dy_t = y_t - y_{t-1} + ctrend$

(96) $dc_t = c_t - c_{t-1} + ctrend$

(97) $di_t = i_t - i_{t-1} + ctrend$

(98) $dw_t = w_t - w_{t-1} + ctrend$

(99) $\pi_t^o = \pi_t + constepinf$

(100) $r_t^o = r_t + conster$

(101) $l_t^o = l_t + constelab$

The autoregressive equations for shocks are:

$$\epsilon_t^a = crhoa * \epsilon_{t-1}^a + ea \tag{102}$$

$$\epsilon_t^b = crhob * \epsilon_{t-1}^b + eb \tag{103}$$

$$\epsilon_t^g = crhog * \epsilon_{t-1}^g + eg + cgy * ea \tag{104}$$

$$\epsilon_t^i = crhoqs * \epsilon_{t-1}^i + eqs \tag{105}$$

$$\epsilon_t^r = crhoms * \epsilon_{t-1}^r + em \tag{106}$$

$$\epsilon_t^p = crhopinf * \epsilon_{t-1}^p + epinfma - cmap * epinfma(-1) \tag{107}$$

$$epinfma = epinf \tag{108}$$

$$\epsilon_t^w = crhow * \epsilon_{t-1}^w + ewma - cmaw * ewma(-1) \tag{109}$$

$$ewma = ew \tag{110}$$

Some of the higher level parameters in the above equations are also related to the parameters in my diagram by the following equations:

(111) $$cpie = 1 + constepinf/100$$

(112) $$cgamma = 1 + ctrend/100$$

(113) $$cbeta = 1/(1 + constebeta/100)$$

(114) $$clandap = cfc$$

(115) $$cbetabar = cbeta * cgamma^{(-csigma)}$$

(116) $$cr = cpie/(cbeta * cgamma^{(-csigma)})$$

(117) $$crk = (cbeta^{(-1)}) * (cgamma^{csigma}) - (1 - ctou)$$

(118) $$cw = \frac{(calfa^{calfa} * (1 - calfa)^{(1-calfa)}}{(clandap * crk^{calfa}))^{(1/(1-calfa))}}$$

(119) $$cikbar = (1 - (1 - ctou)/cgamma)$$

(120) $$cik = (1 - (1 - ctou)/cgamma) * cgamma$$

(121) $$clk = ((1 - calfa)/calfa) * (crk/cw)$$

(122) $$cky = cfc * (clk)^{(calfa-1)}$$

(123) $$ciy = cik * cky$$

(124) $$ccy = 1 - cg - cik * cky$$

(125) $$crkky = crk * cky$$

(126) $$cwhlc = (1/clandaw) * (1 - calfa)/calfa * crk * cky/ccy$$

(127) $$cwly = 1 - crk * cky$$

(128) $$conster = (cr - 1) * 100$$

The seven observed equations – dy_t, dc_t, di_t, dw_t, π_t^o, r_t^o, and l_t^o – are matched using data simulated from a model with true parameters equal to the Smets and Wouters (2007) mean parameters, with slight modifications as some parameters were calibrated to 0. The MCMC approach uses the Smets and Wouters (2007) Dynare code with small modifications like the use of a uniform prior.

Further results are shown in Section IVE in the body of the paper.

Here are the 36 parameters estimated by SNPE after 500 thousand iterations:

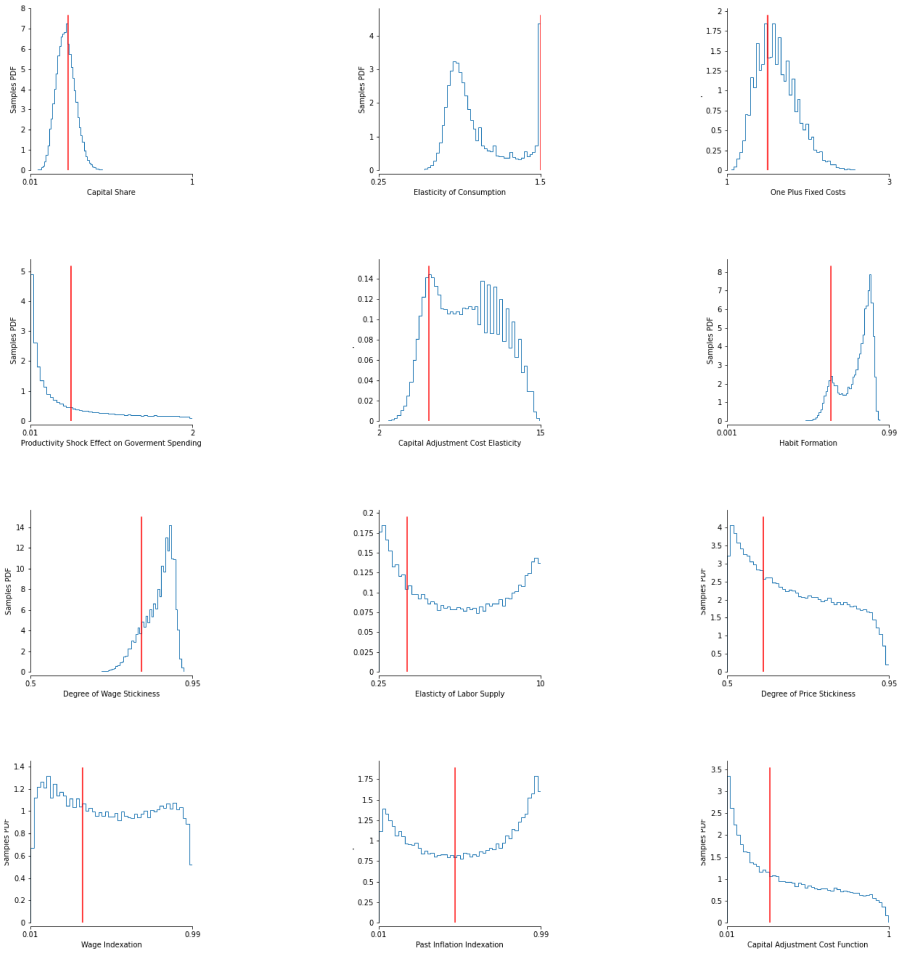

FIGURE XXII: SMETS-WOUTERS SNPE POSTERIOR WITH PARAMETERS 1-12

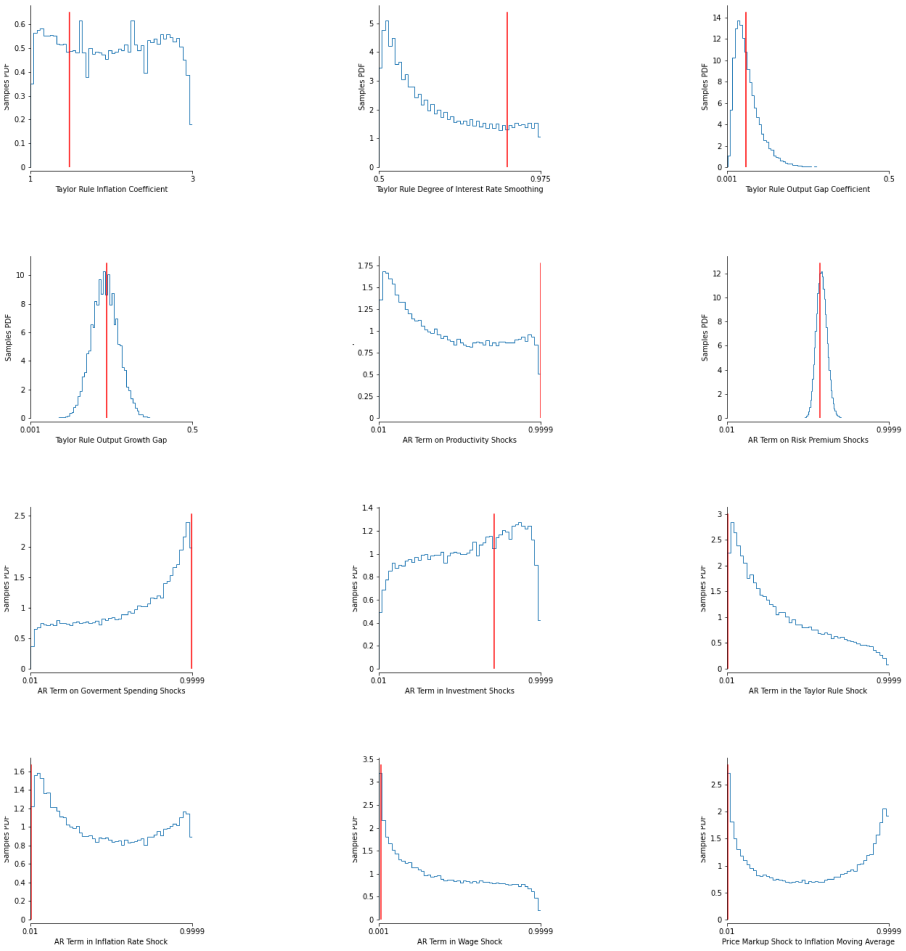

FIGURE XXIII: SMETS-WOUTERS SNPE POSTERIOR WITH PARAMETERS 13-24

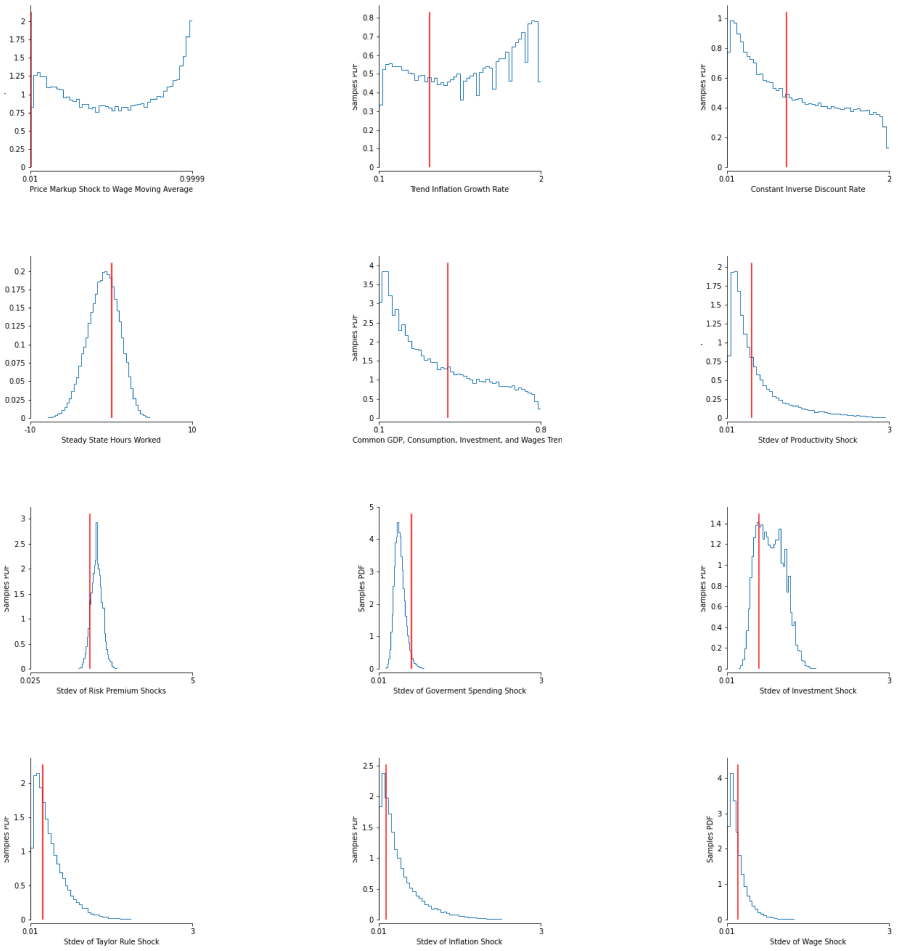

FIGURE XXIV: SMETS-WOUTERS SNPE POSTERIOR WITH PARAMETERS 25-36

For comparison, here are the 36 parameters estimated via MCMC:

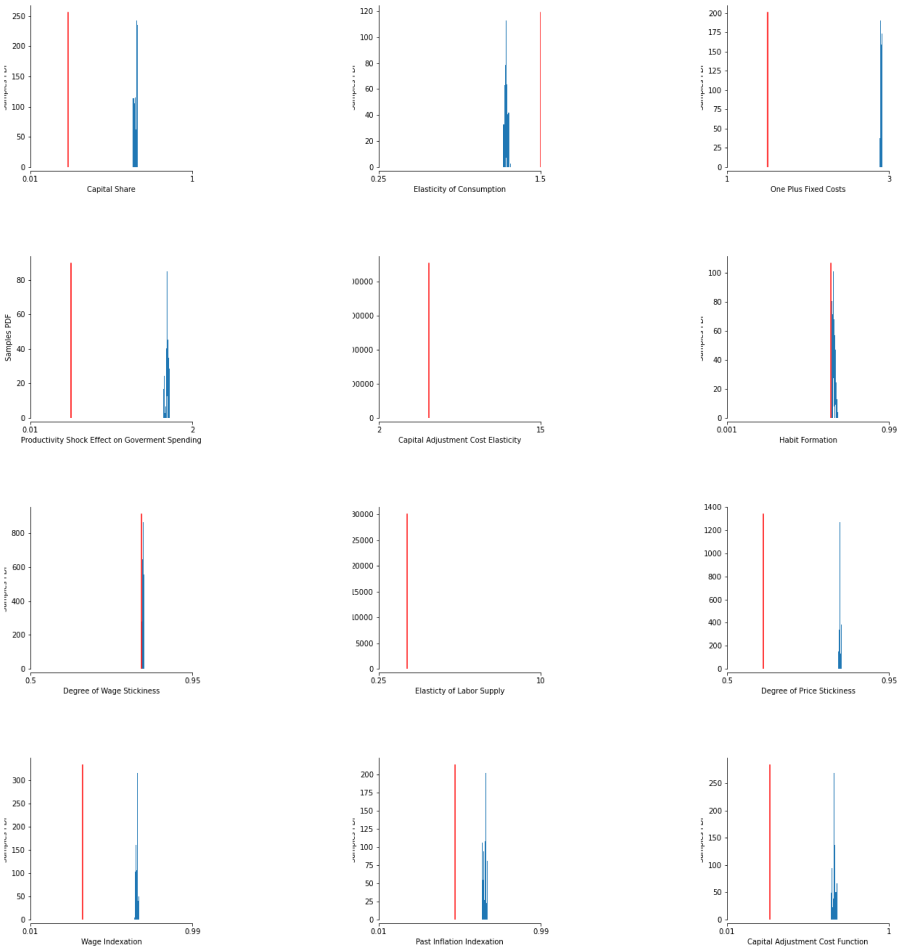

FIGURE XXV: SMETS-WOUTERS MCMC POSTERIOR WITH PARAMETERS 1-12

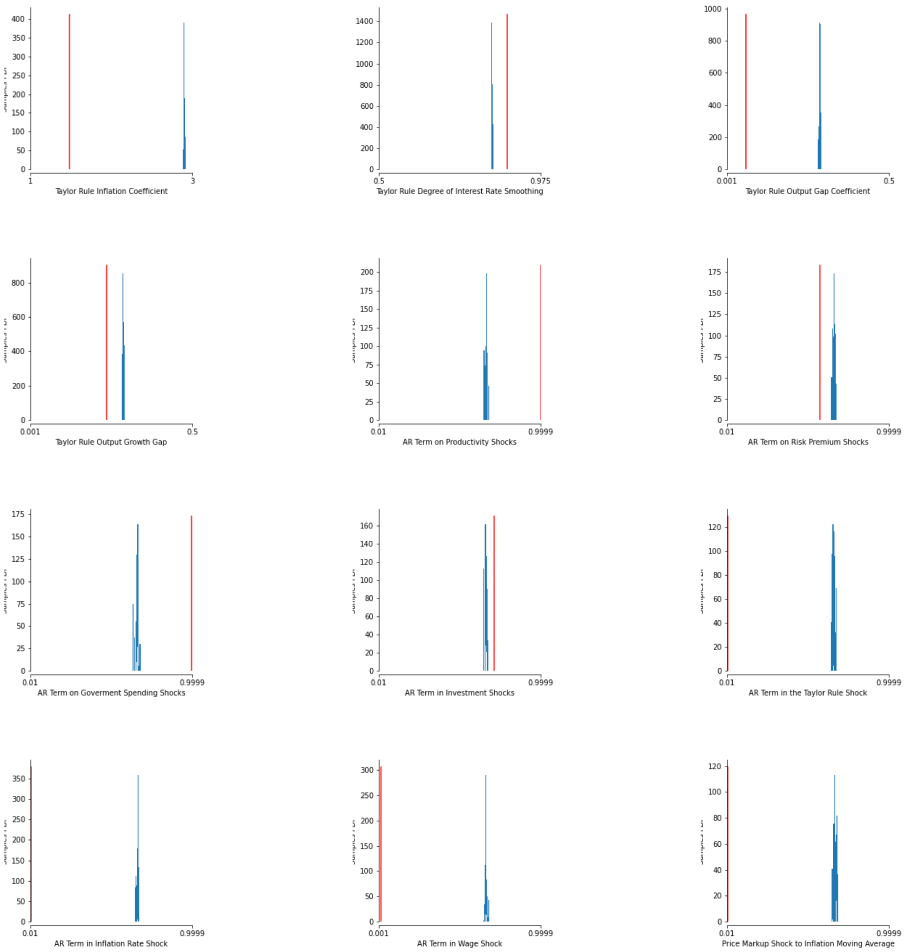

FIGURE XXVI: SMETS-WOUTERS MCMC POSTERIOR WITH PARAMETERS 13-24

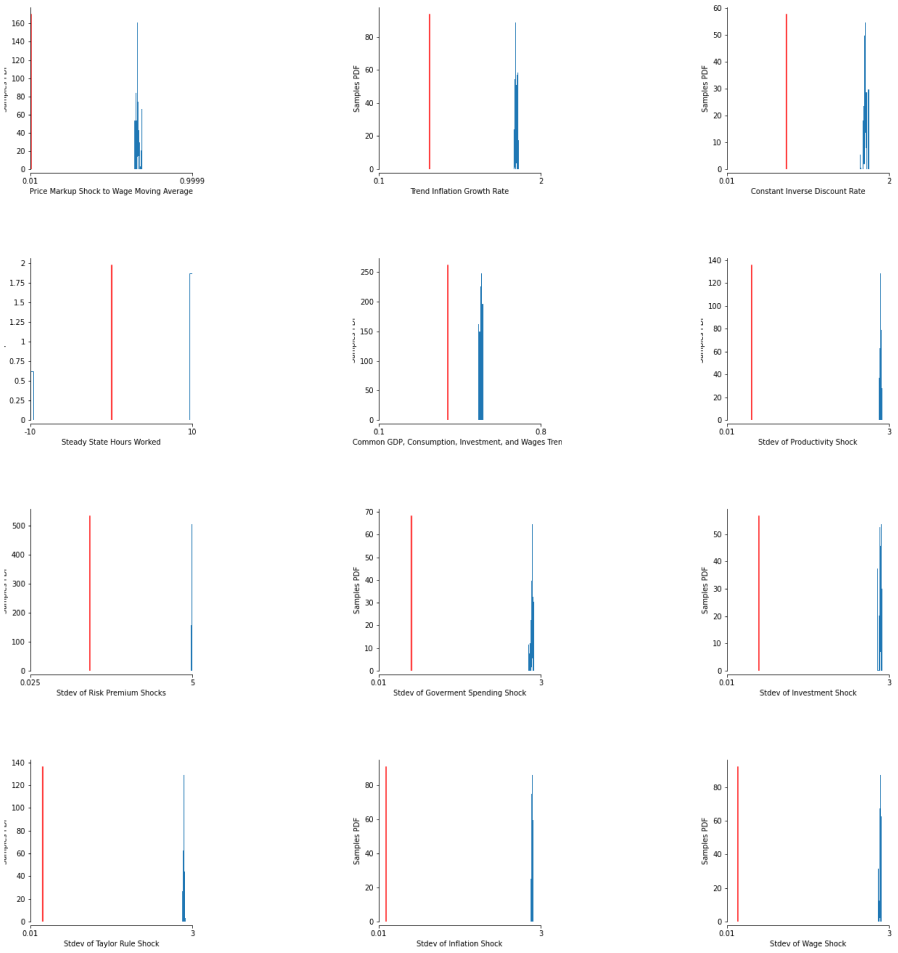

FIGURE XXVII: SMETS-WOUTERS MCMC POSTERIOR WITH PARAMETERS 25-36

Finally, here is the posterior estimated by SNPE over only 150 thousand iterations to match the computational time to estimate 10 million MCMC samples:

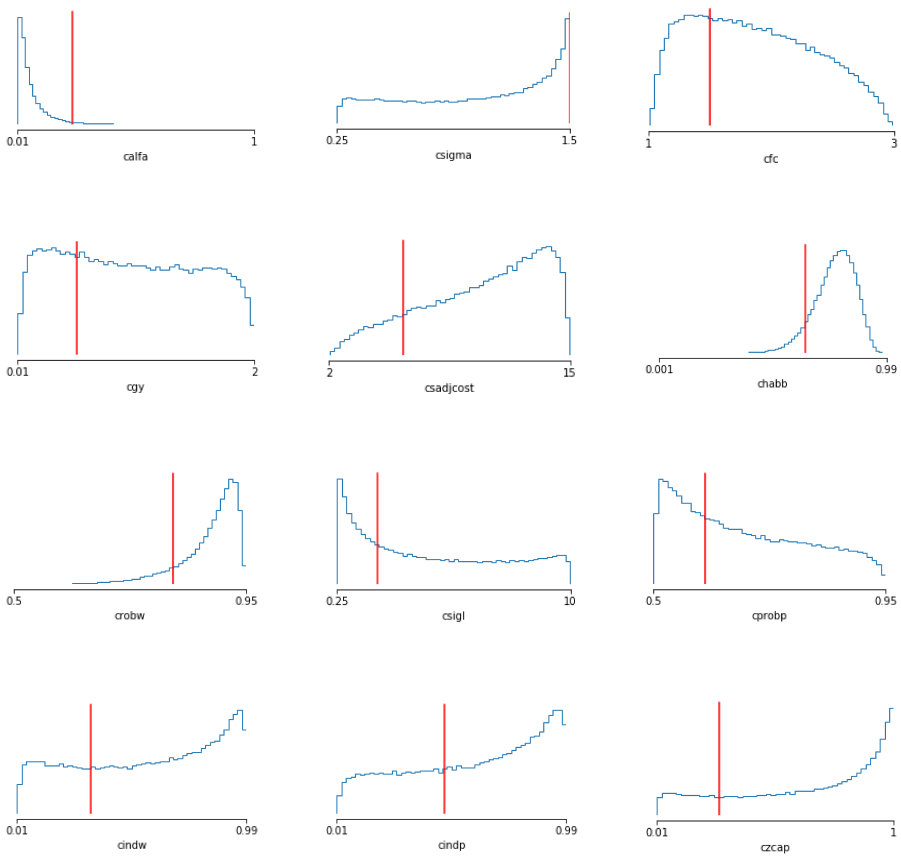

Figure XXVIII: Smets-Wouters SNPE Posterior with 150 Thousand Samples: Parameters 1-12

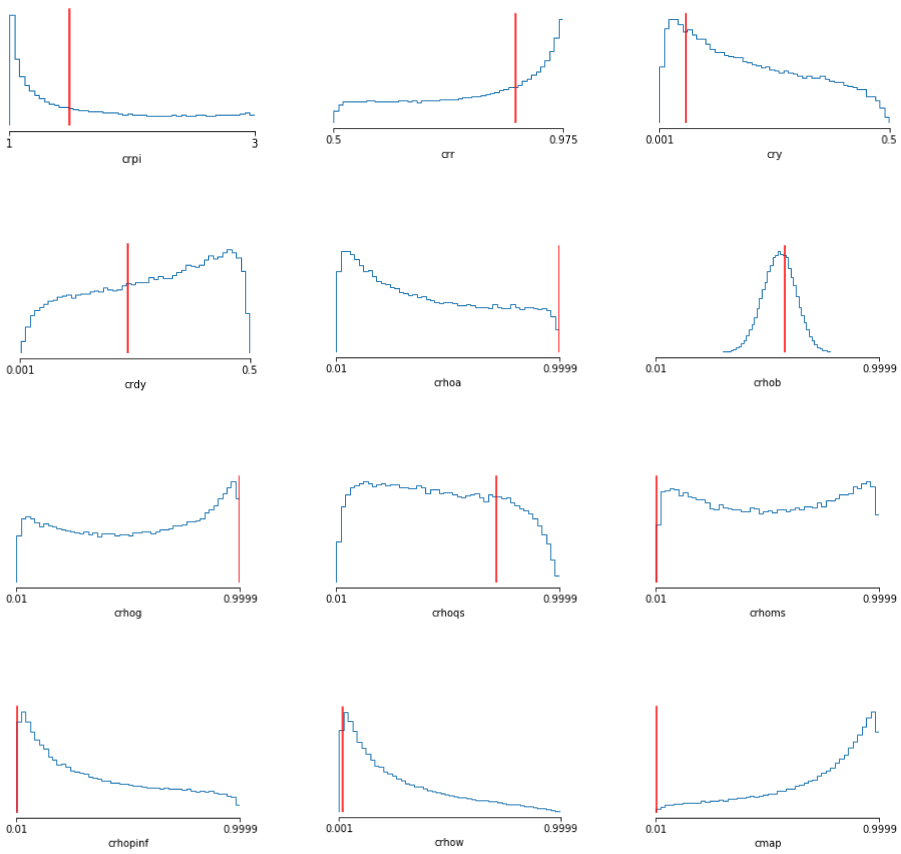

Figure XXIX: Smets-Wouters SNPE Posterior with 150 Thousand Samples: Parameters 13-24

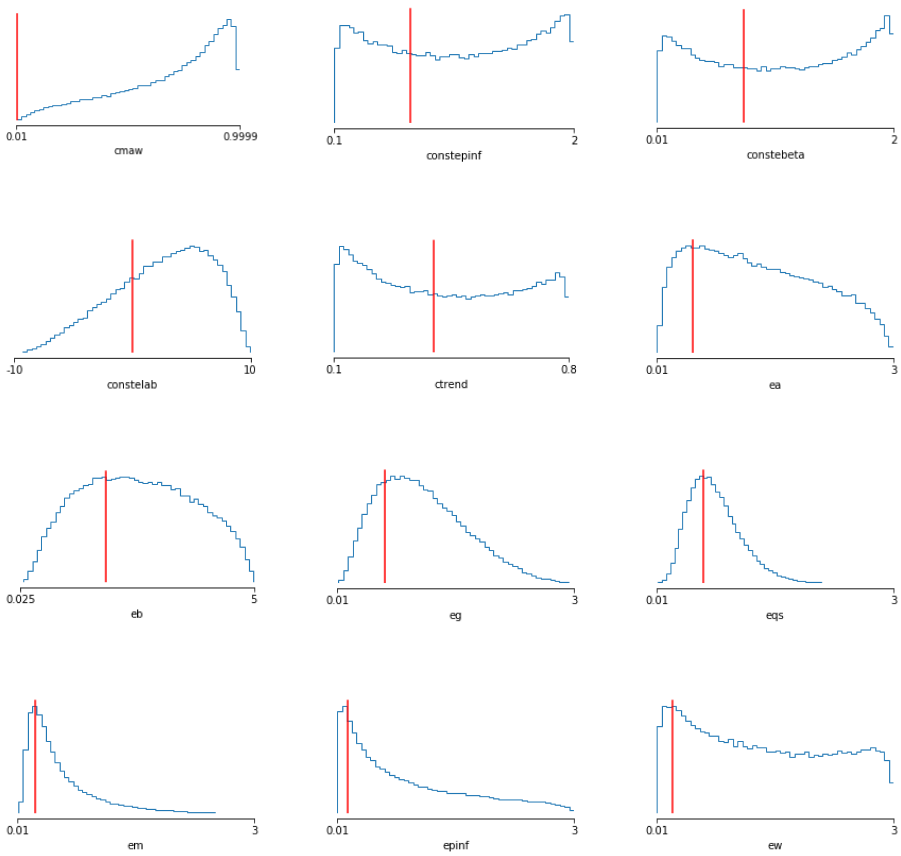

FIGURE XXX: SMETS-WOUTERS SNPE POSTERIOR WITH 150 THOUSAND SAMPLES: 25-36

This shows that SNPE produces better posteriors in the same amount of time, even when disadvantaged by choice of programming language, still producing cleaner posteriors which place support on the true data-generating process.

CF SNPE Estimation of a HANK Reiter Model

Two different HANK models are estimated. The one discussed in this section is solved via Reiter's method. This provides an example of estimation of models pushing the frontier in economics. The benefits of this approach allows one to estimate heterogeneous agent models in a likelihood manner that is fast enough to avoid resorting to dimensionality reduction. The HANK model also

comes from Alisdair McKay's tutorial, which adds a heterogeneous distribution of wealth as a state variable for households combined with a relatively standard New Keynesian model.

Preferences are still CRRA, like the RBC model. The firm side is now New Keynesian with intermediate goods produced via:

$$y_{j,t} = Z_t N_{j,t} \tag{129}$$

Here Z_t is aggregate productivity and N_t is labor employed by firm j producing variety $y_{j,t}$. The final good is produced via the Dixit-Stiglitz aggregator Dixit and Stiglitz (1977):

$$Y_t = \left(\int_0^1 y_{j,t}^{\frac{\epsilon-1}{\epsilon}}\right)^{\frac{\epsilon}{\epsilon-1}} \tag{130}$$

Here ϵ is the elasticity of substitution between intermediates. Output is defined as:

$$Y_t = A_t \int_0^1 n_{j,t} dj = A_t N_t \tag{131}$$

Where $n_{j,t}$ is the labor for the individual intermediate goods. Additionally from the accounting identity, output is also, $Y_t = C_t + \psi M_t H_t$ where the second term is hiring costs.

Government and bonds are defined by the following nominal and real budget constraint equations:

$$\tau P_t(w_t + d_t)(1 - u_t) + P_t B = (1 + i_{t-1}) P_{t-1} B + P_t b u_t \tag{132}$$

$$\tau(w_t + d_t)(1 - u_t) + B = R_t B + b u_t \tag{133}$$

Labor market is a standard McCall Search model:

$$N_t = (1-\delta)N_{t-1} + H_t \tag{134}$$

$$1 - u_t = (1-\delta)(1-u_{t-1}) + H_t \tag{135}$$

$$M_t = \frac{H_t}{u_{t-1} + \delta N_{t-1}} \tag{136}$$

$$M_t = \frac{1 - u_t - (1-\delta)(1-u_{t-1})}{u_{t-1} + \delta(1-u_{t-1})} \tag{137}$$

The household maximizes the CRRA utility like the RBC model. The households budget constraint in real money is:

$$a_t + C_t = R_t * a_{t-1} + earnings \tag{138}$$

Here a_t is savings and R_t is the interest rate as determined by the bond market clearing condition. Earnings are given by:

$$\text{earnings} = \begin{cases} (1-\tau_t)\,(w_t + d_t) & \text{if employed} \\ b & \text{if unemployed,} \end{cases} \tag{139}$$

Here w_t is wages, d_t is dividend income and b is unemployment benefits.

The stochastic matrix governing employment/unemployment transitions is given by:

$$\begin{pmatrix} 1 - M_t & \delta \\ M_t & 1 - \delta \end{pmatrix} \tag{140}$$

Again, M_t is governed by the labor market equations above.

The firm solves the Dixit-Stiglitz cost problem:

$$y_{j,t} = \left(\frac{p_{j,t}}{P_t}\right)^{-\varepsilon} Y_t \tag{141}$$

with price index P_t given by

$$P_t^{1-\varepsilon} = \int_0^1 p_{j,t}^{1-\varepsilon} dj. \tag{142}$$

The Calvo pricing adjustment parameter θ implies that the intermediate firm chooses price p^* to maximize:

$$\mathbb{E}_t \sum_{s=t}^{\infty} \theta^{s-t} R_{t,s}^{-1} \left[\frac{p_t^*}{P_s} y_{j,s} - w_s n_{j,s} - \psi M_s h_{j,s} \right] \tag{143}$$

Here $n_{j,t}$ denotes employment at firm j in time t. $h_{j,t}$ denotes hiring and $R_{t,s}^{-1}$ denotes the real interest rate discounted s periods back to time t. This model is solved using Reiter's method in combination with time iteration. This model matches unemployment and interest rate to data generated by a parameterized model.

CG SNPE Estimation of a Winberry Approach to Krusell-Smith

The second heterogeneous agent model is Krusell and Smith (1998) solved with Winberry (2018), which is presented in the results and also the basis for my empirical application. The consumer is standard:

$$E[\sum_t = 0^{\infty} \beta^t U(c_t) \tag{144}$$

Utility function is CRRA:

$$U(c-t) = \frac{c^{1-\sigma} - 1}{1-\sigma} \tag{145}$$

The investment equation is:

$$y = c + k' - (1-\delta)k \tag{146}$$

Agents can be employed or unemployed. If employed, it works one unit of labor or else it does not work. This is denoted with an ϵ. Employment status transitions via a Markov matrix.

On the market side, wages and rental rate equal their marginal benefit. In particular if aggregating across all capital ($\bar{k}$) and labor ($\bar{l}$):

$$w = (1-\alpha)z(\bar{k}/\bar{l})^{\alpha} r = \alpha z(\bar{k}/\bar{l})^{1-\alpha} \tag{147}$$

The states are the distribution of capital and labor, denoted Γ, as well as productivity. For each individual, they optimize via:

$$v(k, \epsilon; \Gamma, z) = max_{c,k'}\{U(c) + \beta E[v(k, \epsilon; \Gamma, z)|z, \epsilon]\} \tag{148}$$

subject to

$$c + k' = rk + wl\epsilon + (1-\delta)k\Gamma' = H(\Gamma, z, z')k' \geq 0 \tag{149}$$

Each agent optimizes according to their own k and ϵ. However the prices are chosen so that when agents are optimizing, the aggregate next period levels of k and ϵ match the levels implied by the prices. As such you are solving for a fixed point so that given a distribution of k and ϵ, the next period distribution of k and ϵ also correspond to the prices of k and ϵ that would induce agents to acquire those amounts. Winberry takes a different approach. Since you know the individual's law of motion and you can integrate the distribution to get the aggregate prices that generate that law of motion, you can derive a law of motion for the heterogeneous distribution. If you approximate this distribution via moments, you can derive a law of motion for the moments of the distribution. Then you can apply non-heterogeneous agent approaches to solve this model. In particular, Winberry solves the model in Dynare with perturbation. For more details on the solution method, see Winberry (2018).

Liu and Plagborg-Møller (2023) perform a likelihood estimation on a Krusell-Smith model solved via Winberry. Estimating full information heterogeneous agent models via a likelihood is difficult for

models solved via Winberry's approach. The reason is that when evaluating the micro-data, one has to calculate the likelihood of the micro data conditional on the distribution of the state after filtering conditional on macro variables. One does this by sampling from the Kalman filter state-space, then for each draw from the Kalman filter, evaluate the integral over the range of micro variables using a grid. This is quadratic in the number of evaluations required and is computationally intensive. Due to the fact that SNPE does not require likelihood calculation, this whole process is avoided.

D Background on Related Simulation-Based Approaches and Density Estimators

I will first discuss conditional mixture of Gaussians in Section DA which is an alternate density estimator for SNPE. Then I will discuss the GAN, which is used as a density estimator. Due to the differences imposed by the GAN on other aspects of the algorithm, the simulation-based GAN approach is different and is called Sequential Neural Ratio Estimation (SNRE). Additionally, the method Sequential Neural Variational Inference (SNVI) is in improvement on SNRE. I will talk about all these approaches in this section.

DA Conditional Mixture of Gaussians

The conditional mixture of Gaussians approach is a powerful tool for conditional density estimation and, of the three methods, is the easiest to explain. Starting from the unconditional mixture of Gaussians: as outlined in Bishop (1994), a(n) (unconditional) Mixture of Gaussians is a linear combination of N Gaussian distributions, each with its own mean, covariance, and weight. I will start by discussing unconditional mixture of Gaussian density estimations. Figure XXXI shows how a Mixture of Gaussians fits a distribution:

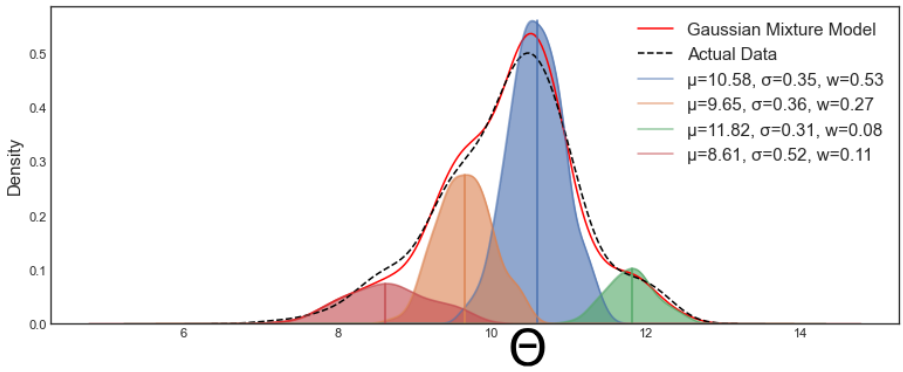

FIGURE XXXI: HOW A GAUSSIAN MIXTURE MODEL FITS A DISTRIBUTION

The y axis is probability mass and the x axis is parameter value. This is an example of a one-dimensional density estimation for intuition, but the Mixture of Gaussians can handle approximately 10-dimensional density estimation, given my sample size of 200 observations. As shown above, the Gaussian mixture is a weighted average of Gaussians with differing means and standard deviations. By modifying the weights, means and covariance matrices, one can combine a number of Gaussians to accurately represent a distribution. The probability density function of a point under this mixture can be calculated using the equation:

$$P(\theta) = \sum_i \pi_i * Q_i(\theta) \tag{150}$$

where $Q_i(\theta)$ is a normal distribution with mean and variance (covariance) specific to each Gaussian component. This represents the unconditional density estimator. However, SNPE requires a conditional density estimator. This can be achieved by using an optimizable function, such as a feed-forward Neural Network, that takes the X data as input and returns the parameters for the Conditonal Mixture of Gaussians. There is information on this in Section AD of the Appendix. However, for intuition, a high-dimensional polynomial regression or spline would also be able to learn this conditional distribution. The function would then return the weights, $\pi_i(X)$; means, $\mu(X)$; and covariances, $\sigma(X)$, for each Gaussian component as shown in Equation (151), resulting

in the following conditional density estimator:

$$P(\theta|X) = \sum_i \pi_i(X) * Q_i(\theta|\mu(X), \sigma(X)) \tag{151}$$

Finally, I will discuses the estimation procedure. The parameters of the neural network can be optimized via MLE, maximizing $P(\theta|X)$ by adjusting the parameters of the neural network or other function that maps X to the parameters of the conditional mixture of Gaussians. The approach maximizes $P(\theta|X)$ given the simulated conditioning data, X, and the feed-forward neural network will learn a mapping from X to the parameters of the conditional mixture of Gaussians (i.e. $\sigma(X)$, $\mu(X)$ etc.). This implies that the conditional mixture of Gaussians will converge to the true conditional distribution.

The problem of this approach begins to lead to over-fitting with more than 7-10 parameters. For example, Smets and Wouters (2007) has 36 parameters. With a conditional mixture of Gaussians, each covariance matrix of a Gaussian would have dimensionality $37 * 36/2 = 666$. Thus for higher dimensional problems, one will either have to use normalizing flows IIIC, as I generally do, or a GAN, which is discussed next.

DB Generative Adversarial Networks (GAN)

While the focus of this paper is on SNPE, there are related approaches that use a GAN (Goodfellow et al., 2014) as a density estimator. For example, SNRE, and another approach built on SNRE, SNVI, can be used. The SNVI approach is most useful in cases where the structural model can fail to converge within the support of the prior. While SNPE will extrapolate values for these regions that are difficult to predict, SNVI will learn to put zero probability mass in the regions. The center of the SNRE and SNVI approach is the GAN which is an algorithm that uses two competing models moving in tandem to generate data that matches data generated from the real world. Below is a diagram describing the structure of a basic GAN:

The generator is a neural network (or any differentiable model like a dynamic structural model)

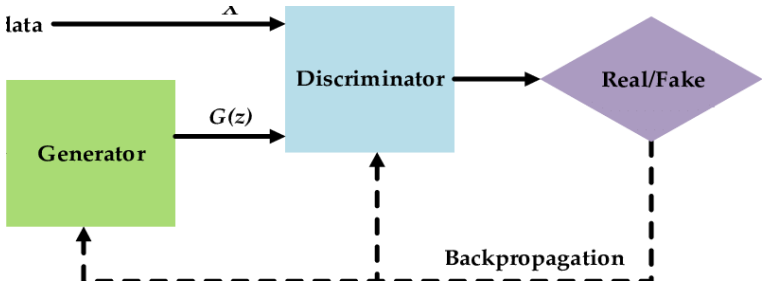

FIGURE XXXII: DIAGRAM OF A GAN

which attempts to produce data (unconditionally or conditionally via some inputs Z) that resembles data in the real world. The discriminator is another neural network (or any trainable model that can produce a logistic output) which outputs the probability a given input data comes from the underlying data generating process and not from the generator. The two networks play a game where the objective of the generator is to convince the discriminator to assign high likelihood to its output. The objective of the discriminator is to assign the correct probabilistic label, a number between 0 or 1 to each output. This number indicates the probability that discriminator believes the data is real and not generated by the generator. One way to think about a GAN is as an MSM algorithm where the moment conditions automatically evolve over many rounds to pick the most mismatched moments between the true data and the model. When thought in this way, it is intuitive to recognize that the GAN is nearly a full information and efficient estimation procedure. For more thorough proofs, see Kaji, Manresa and Pouliot (2020).

Mathematically, the discriminator loss is:

$$Loss_D = -E_{x\sim P_{data}(x)}[D(x)] - E_{z\sim P_{sim}(z)}[1 - D(G(z))] \tag{152}$$

This is just a logit loss applied to data either coming from the data generating process or the generator. Likewise, the generator's loss is to fool the discriminator:

$$Loss_G = E_{z\sim P_{sim}(z)}[1 - D(G(z))] \tag{153}$$

These two loss functions can be compactly represented as:

$$G, D = \min_G \max_D E_{x\sim P_{data}(x)}[D(x)] + E_{z\sim P_{sim}(z)}[1 - D(G(z))] \tag{154}$$

Equation (154) makes it clear the GAN objective is a minimax game with a solution that is a Nash equilibrium. However, what is important for simulation-based estimation is to recover likelihoods/probabilities. The next proposition shows how to do that:

Appendix Proposition 1.: For a fixed generator, the optimal discriminator will produce a logistic output:

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{generator}(x)} \tag{155}$$

See Goodfellow et al. (2014) for the proof. Using this, one can use the discriminator output to estimate a likelihood ratio.

DC Sequential Neural Ratio Estimation

In SNRE, Durkan, Murray and Papamakarios (2020) use a GAN, not to match underlying data like in Appendix Proposition 1, but to contrast between two different distributions to estimate a density. In macro, the data set is typically a single data point with correlated time-steps. Using a GAN to discriminate between real data and fake data would not work because the GAN would learn a delta function for the real data. In order to ameliorate this, the authors use contrastive learning where they differentiate between the generated distribution and the independent distribution, $X, \theta \sim P(X)P(\theta)$, where X is sampled uniformly from the X distribution of simulated data, ignoring

θ, breaking the joint relationship of X with θ. Then they contrast the independent distribution with the true joint: $X, \theta \sim p(X, \theta) = p(X|\theta)p(\theta)$. Furthermore, knowing the optimal discriminator gives the ratio discussed in (155). The authors then recover the ratio: $r(X, \theta) = \frac{p(X, \theta)}{P(X)P(\theta)}$, using the fact that $\frac{p(X, \theta)}{p(X, \theta) + p(X)P(\theta)}$ equals $\frac{r}{r+1}$. Thus, the density of the posterior is $P(\theta|X) = r(X, \theta)P(\theta)$, where $P(\theta)$ is the prior. This extends the Kaji, Manresa and Pouliot (2020) GAN approach in allowing estimation of models with a latent time dimension. One can extend this to multi-round inference where one uses the GAN as a better proposal distribution than the prior, by sampling θ from the GAN using MCMC. Then one uses importance sampling to deal with the mismatch in proposal distribution and prior. See Durkan, Murray and Papamakarios (2020), Hermans, Begy and Louppe (2020) and the SNRE theoretical sections of the Appendix for more information. Below, Algorithm 2 shows the steps for performing SNRE:

Algorithm 2: SNRE Algorithm

Input: Simulator $P(X|\theta)$, prior $P(\theta)$, data X', discriminator $d_\phi(X, \theta)$, Rounds R, Samples S;

Initialize: Posterior $P^{(0)} = p(\theta)$, data set D = {};

for $i \leftarrow 1$ **to** R **do**

Sample $\theta^{(n)} \sim P^{(i-1)}$ for $n = 1...S$ with MCMC;

Simulate $X^{(n)} \sim P(X|\theta^{(n)})$ for $n = 1...S$;

Concatenate data $D = D \cup \{X^{(n)}, \theta^{(n)}\}_{n=1}^{S}$;

while $d_\phi(X, \theta)$ *not converged* **do**

Sample $\{X^{(i)}, \theta^{(i)}\}_i^B \sim D$ from D;

Sample K contrasting samples for each i sample $\{X^{c(k)}\}_k^{B*K}, \{\theta^{c(k)}\}_k^{B*K} \sim D\{X\}, D\{\theta\}$, breaking the joint distribution of D into independent marginals;

Train the GAN using the GAN loss function from the two distributions X, θ and X^c, θ^c and a softmax/multinomial logit objective $\sum_i^B log(\frac{exp(d_\phi(X^{(i)}, \theta^{(i)})}{\sum_k^K exp(d_\phi(X^{(k)}, \theta^{(k)})}$);

end

Update posterior $P^{(i)} \propto exp(d_\phi(X', \theta))p(\theta)$;

end

DD Sequential Neural Variational Inference (SNVI)

In SNVI, Glöckler, Deistler and Macke (2021), one performs SNRE, but instead of MCMC sampling from the GAN posterior, one projects the distribution derived by the GAN onto a normalizing flow with variational inference. As discussed in this article, MCMC takes a long time to converge to the posterior. Since one can calculate the pdfs of a sample point using the GAN ratio estimation, one can sample from a normalizing flow, calculate the pdf of the sample under both the flow and the GAN, and use this to minimize the Kullback-Leibler divergence

between the flow and the GAN. Thus the flow inherits the distribution of the GAN, and one can sample directly from the flow without resorting to MCMC. The downside of this approach is that using the reverse KL divergence, in finite iterations, leads to the learned posterior being too tight around the mode. The following Algorithm 3 describes the SNVI step by step:

Algorithm 3: SNVI Algorithm

Input: Simulator $P(X|\theta)$, prior $P(\theta)$, data X', discriminator $d_\phi(X, \theta)$, Rounds R, Samples S;

Initialize: Posterior $P^{(0)} = P(\theta)$, data set D = {};

for $i \leftarrow 1$ **to** R **do**

Sample $\theta^{(n)} \sim f_{\phi'}^{(i-1)}(\theta)$ for $n = 1...S$;

Simulate $X^{(n)} \sim P(X|\theta^{(n)})$ for $n = 1...S$;

Concatenate data $D = D \cup \{X^{(n)}, \theta^{(n)}\}_{n=1}^{S}$;

while $d_\phi(X, \theta)$ *not converged* **do**

Sample $\{X^{(i)}, \theta^{(i)}\}_i^B \sim D$ from D;

Sample K contrasting samples for each i sample $\{X^{c(k)}\}_k^{B*K}, \{\theta^{c(k)}\}_k^{B*K} \sim D\{X\}, D\{\theta\}$, breaking the joint distribution of D into independent marginals;

Train the GAN using the GAN loss function from the two distributions X, θ and X^c, θ^c and a softmax/multinomial logit objective $\sum_i^B log(\frac{exp(d_\phi(X^{(i)}, \theta^{(i)})}{\sum_k^K exp(d_\phi(X^{(k)}, \theta^{(k)})})$;

end

Train normalizing flow with variational inference $f_{\phi'}^i = argmin_{\phi'}(KL(f_{\phi'}^i(\theta)||exp(d_\phi(X', \theta))p(\theta))$;

end

An important benefit of SNVI and SNRE is that if the model fails to converge and produce a

joint sample, they will assign zero probability to those areas. On the other hand other, approaches will try to extrapolate a probability density function from known points. The issue with SNVI is that, when using the reverse KL, it is a well known problem that the derived distribution is mode seeking (Chan et al., 2022) – tighter around the modes with thinner tails than the true posterior. SNRE uses MCMC sampling and one does not get the speed and posterior accuracy benefits of SNPE.

Department of Economics, University of Michigan, Ann Arbor