Estimating Nonlinear Heterogeneous Agents Models with Neural Networks[*]

Hanno Kase
European Central Bank

Leonardo Melosi
University of Warwick
FRB Chicago & CEPR

Matthias Rottner
Deutsche Bundesbank

December 16, 2023

Abstract

We leverage recent advancements in machine learning to develop a method to solve and to perform likelihood estimation of the parameters of nonlinear, heterogeneous agents models. Neural networks are set up to obtain an accurate approximation of the model's nonlinear transition equations and likelihood function. The proposed method can retrieve the true parameter values of a prototypical model whose solution is worked out analytically. Furthermore, the parameters estimated with our method are consistent with those estimated using a state-of-the-art method, which can, however, only be applied to stylized nonlinear representative agent models. Using simulated data we show that the proposed method provides an accurate estimation of the parameters of a nonlinear Heterogeneous Agent New Keynesian (HANK) model with a zero lower bound (ZLB) constraint and aggregate uncertainty. When we estimate the HANK model using U.S. data, we find that the interplay between aggregate nonlinearities and uninsurable idiosyncratic risk plays a large role in explaining business cycles volatility. The estimation with real data provides a first evaluation of how nonlinear HANK models should be expanded to improve their empirical performance.

Keywords: Machine learning, neural networks, Bayesian estimation, global solution, heterogeneous agents, nonlinearities, aggregate uncertainty, HANK model, zero lower bound.
JEL classification: C11, C45, D31, E32, E52

[*]Emails: hanno.kase@ecb.europa.eu, leonardo.melosi@warwick.ac.uk, matthias.rottner@bundesbank.de. We thank Borağan Aruoba, Jesús Fernández-Villaverde, David Levine, Galo Nuño, Christian Matthes, Ed Herbst, and Frank Schorfheide for their very helpful suggestions. We thank our discussants Kieran Larkin and Danila Smirnov and seminar participants at the NBER Summer Institute 2023, Banque de France, Deutsche Bundesbank, Bank of Finland, Schumpeter-BSE-Seminar in Berlin, CEBRA 2022 Annual Meeting, Conference on advanced analytics by the ECB, BoE and KCL, EUI, Conference on Non-traditional Data, Machine Learning and Natural Language Processing in Macroeconomics at Sveriges Riksbank, Ghent University Workshop on Empirical Macroeconomics, 29th Symposium of the SNDE, SEM Conference 2022, EEA Conference 2022, Midwest Macro 2022, T2M 2023 Conference, and CEF 2023 Conference. We also thank Jesús Laso Pazos for his excellent work as research assistant. A previous version was titled "Solving and Estimating Macroeconomic Models of the Future". The views in this paper are solely those of the authors and should not be interpreted as reflecting the views of the Deutsche Bundesbank, the Eurosystem, the European Central Bank, the Federal Reserve Bank of Chicago, or any other person associated with the Federal Reserve System.

1 Introduction

Models with heterogeneous agents and nonlinear constraints have attracted growing attention among economists in recent years. For instance, these models have been used to study the impact of monetary and fiscal policies on the macroeconomy, the macroeconomic implications of social inequality, and the distortions introduced by nonlinear constraints, such as the zero-lower-bound (ZLB) constraint on nominal interest rates and by households' borrowing limits. Nevertheless, these models are very complex and strong assumptions – e.g., perfect foresight, linearization – are typically made to ensure tractability. Furthermore, the complexity of these models has largely prevented researchers from studying interesting interactions among nonlinear constraints, aggregate uncertainty, and agents heterogeneity.

We propose a machine-learning based approach to solve and to estimate models featuring nonlinear constraints, aggregate uncertainty, and agents heterogeneity without resorting to these simplifying assumptions. Subsequently, we apply our method to estimate a quantitative Heterogeneous Agent New Keynesian (HANK) model with a recurrent ZLB constraint and aggregate uncertainty using U.S. data.

Being able to estimate nonlinear, heterogeneous agents models with likelihood methods is a critical step to further advance our understanding of what these models can help us understand the existing models cannot and how these models can be improved to expand our knowledge about the working of economic systems. For instance, likelihood estimation of linearized, representative agents dynamic stochastic general equilibrium (DSGE) models (Smets and Wouters, 2007) contributed to improving our understanding of the role played by macroeconomic shocks in the business cycle and allowed conducting formal econometric inference (e.g. estimating key parameters and state variables, comparing model's fit, and forecasting) as well as studying the identification of structural parameters. Likelihood estimation of these models was a critical factor behind their rapid diffusion and growing popularity. We believe that similar developments are possible for nonlinear, heterogeneous agents models, such as HANK models, once methods to solve and perform likelihood estimation become broadly available. In this paper, we develop these methods by leveraging recent progress in machine learning.

Estimation of structural models requires searching for the parameters values that globally maximize the likelihood function or the posterior distribution if one takes a Bayesian approach.[1] This search necessarily calls for evaluating the likelihood function at a potentially very large number of parameter values. For *each* of these parameter values, two steps are typically taken in order to maximize the likelihood or the posterior distribution. First, the model is solved to characterize the laws of motion or transition equations governing the equilibrium dynamics of the model's state variables (*solution step*). Second, once the transition equations are known for

[1]In addition, Bayesian estimation requires simulating a large number of parameter draws in a neighborhood of the posterior mode to approximate posterior moments.

that parameters value, the likelihood is evaluated with a filter (*likelihood evaluation step.*)

The solution and likelihood evaluation steps are performed sequentially for a given value of the model parameters. Consequently, these two steps have to be performed multiple times to find the global maximum of the likelihood function or the posterior distribution. The repetition of the two steps is not at all problematic if the model at hand can be solved within a fraction of second, using a routine that can be easily automatized. For instance, this is the case of linearized, representative-agent DSGE models. However, the repetition of the solution and likelihood evaluation steps is a critical bottleneck on likelihood estimation of highly complex, nonlinear models, such as nonlinear HANK models with aggregate uncertainty. First, solving these models multiple times is often impractical because it takes too long to solve the model even for just one value of the parameters. Second, solving these highly complex models invariably requires careful customization of the routine to achieve a good degree of accuracy.

To get around this bottleneck, we proceed by approximating the transition equations without conditioning on one particular value of the model parameters. To do so, we treat the model parameters as inputs (or pseudo-state variables) of the model's transition equations.[2] While this approach rises the dimensionality of the domain of the model's solution mapping to approximate, we show that neural networks are quite successful in coping with this computational challenge even for models with a large number of parameters.

Neural networks are the fundamental building block of deep learning, which belongs to the family of machine learning methods. They offer a flexible functional form that can be efficiently used to approximate arbitrarily complex, non-linear functions potentially featuring a large set of inputs.[3] Indeed, one of the most appealing features of neural networks is *scalability*; i.e., increasing the number of inputs of the function to approximate with a neural network does not give rise to curse-of-dimensionality problems.

We leverage the scalability of neural networks to approximate the transition equations without conditioning on one particular value of the model parameters. On the one hand, the scalability of neural networks allows us to treat model parameters as inputs of the model's solution mapping turns out to raise the computational costs only modestly. On the other hand, approximating the transition equations as a function of the entire parameter space allows solving the model only once – before we run the maximization of the likelihood or the posterior distribution. At the maximization stage, the transition equations can, thereby, be obtained almost instantaneously, allowing us to bypass the time-consuming solution step. In addition, since one needs to approximate the transition equations of the model only once before the maximization of the likelihood, neural networks can be adequately trained to make sure they deliver a good approximation of the transition equations.

[2]Typically, the inputs of macro models' transition equations include past model's state variables and current shocks.

[3]Goodfellow et al. (2016) provides an overview of deep learning. Fernández-Villaverde et al. (2020) and Maliar et al. (2021), among others, discuss machine learning in the context of macroeconomic modeling.

Another computational challenge is the evaluation of the likelihood function of nonlinear models, which requires using filters based on Monte Carlo methods, such as the particle filter (Fernández-Villaverde and Rubio-Ramírez, 2007). This type of filters are more expansive computationally and less accurate than the Kalman filter, which can only be used to evaluate the likelihood of linearized DSGE models. Therefore, minimizing the number of times the likelihood is evaluated is particularly desirable.

To achieve this goal, we train, in a second step, another neural network using only a limited number of parameters values to evaluate the likelihood function.[4] This additional neural network is used to nonlinearly approximate the likelihood function across the parameters value of the training sample. It should be noted that to train this neural network we are required to solve the model at every point of the training sample. To perform this task fast enough to make the training of the neural network in the second step feasible, we rely on the solution mapping approximated by the neural network trained in the first step, which was described earlier. With the approximated solution of the model at hand, we can run the particle filter to compute the likelihood value at the cloud of the parameters of the training sample.

Endowed with the approximated likelihood function, one does not need to run a time-consuming Monte Carlo filter as many times as needed to maximize the likelihood or the posterior distribution. Rather, the likelihood or the posterior can be evaluated almost instantaneously at any parameter value using the neural network approximation of the likelihood function. With the approximated posterior distribution at hand, a Bayesian econometrician can straightforwardly approximate its moments. In addition, we show that the neural networks approximation irons out the bumps due to the noise introduced by the particle filter when evaluating the likelihood.

We provide three proofs of concept to validate our estimation method based on neural networks. First, we demonstrate that our neural network approach replicate the transition equations of a small-scale linearized DSGE model, which can be solved analytically. Second, we compare our estimation framework to a state-of-the-art estimation method for nonlinear models. While this alternative method is based on global solution methods and thereby retains the interactions between aggregate nonlinearity and households' heterogeneity as our method also does, it fails to break the sequentiality between the solution step and the likelihood evaluation step, limiting considerably its scope of application. In light of these limitations, we are forced to consider a stylized representative agents model featuring only one source of aggregate nonlinearity – the ZLB constraint. We show that the two methods deliver remarkably similar estimation outputs, suggesting the our method can do equally well as a state-of-the-art method but it has the advantage to be applicable to a much broader set of models that are too complex for existing methods to handle. Third, we simulate data from a HANK model that features hundreds of state variables and 13 parameters to estimate with likelihood methods. Using simulated data, we demonstrate that our method successfully recovers all the true parameter values of the model.

[4]Likelihood evaluation at this cloud of parameters values is conducted using the particle filter.

We then turn our attention to an application with actual data. Specifically, we solve and estimate a nonlinear HANK model using US time-series data from 1984:Q1 to 2019:Q4. The model features idiosyncratic and aggregate shocks, as well as idiosyncratic and aggregate nonlinearities in the form of individual borrowing limits and a ZLB constraint, respectively. The set of parameters we estimate includes the standard deviation of the idiosyncratic shocks. The HANK model fits the data satisfactorily. Nonetheless, the model cannot account for the volatility and serial correlation of the federal funds rate. This shortcoming characterizes medium-scale RANK models as well. Additional features can be added to the HANK model to mitigate this problem.

Standard aggregate data allow us to identify the degree of idiosyncratic risk. The reason is that the level of idiosyncratic risk impacts the interest rate level and affects the frequency of encountering the ZLB. If idiosyncratic risk were absent, the frequency of hitting the ZLB would be severely diminished, underscoring the importance of studying the interactions between model's aggregate nonlinearities and agents heterogeneity. Furthermore, we find that the zero lower bound amplifies the impact of shocks, underscoring the importance of nonlinearities in the model. As an external validation, the model evaluated at the posterior mode gives a wealth Gini coefficient of 0.80, aligning with the average value observed in the US during the estimation period.

The estimation also highlights another advantage of our method: speed. The described neural network-based estimation of a nonlinear HANK model can be completed in around three days using a modern desktop computer, which underlines the method's potential.[5]

Literature The paper is connected to the literature that develops global solution methods using neural networks to solve complex dynamic macroeconomic models such as Fernández-Villaverde et al. (2020), Ebrahimi Kahou et al. (2021), Maliar et al. (2021), Valaitis and Villa (2021), Azinovic et al. (2022), among others. Our contribution to this literature is to show how to use neural networks to not only solve but also to estimate nonlinear, heterogeneous agent models with likelihood methods. Likelihood estimation adds an additional important layer of complication in that it requires solving the model at hundreds of thousands of different points in the parameter space. As a result, only extremely fast and efficient solution methods are compatible with likelihood analysis.

Fernández-Villaverde et al. (2023) estimate the critical parameter of a model with a financial sector and heterogeneous households with neural networks using the time-series of GDP growth. We can expand the scale of the estimation exercise and estimate several parameters using multiple observable variables because (i) we combine the scalability of neural networks with the intuition of treating the model parameters as pseudo-state variables when solving the model and (ii) we use neural networks to facilitate the evaluation of the likelihood via the filter - which we call *neural network particle filter*. Furthermore, unlike that paper, we estimate a HANK model. To

[5]We use a Nvidia Tesla V100 32GB (GPU) for the neural network-based Bayesian estimation.

our knowledge, we are the first to estimate a HANK model in its nonlinear specification.

The first papers that pioneered the analysis of HANK models resort to MIT shocks and disregard aggregate uncertainty in order to facilitate the task of solving these models (e.g., Oh and Reis, 2012; McKay et al., 2016; Challe et al., 2017; Kaplan et al., 2018; Bayer et al., 2019; Bilbiie, 2020; Ottonello and Winberry, 2020; Acharya and Dogra, 2020). More recently, scholars have studied the role of aggregate uncertainty in HANK models by using linear perturbation methods (Reiter, 2009; Winberry, 2021; Ahn et al., 2018; Boppart et al., 2018 and Auclert et al., 2021). There are only very few papers that pioneered the solving of HANK models with macroeconomic uncertainty and non-negative constraints. Maliar and Maliar (2020), Gorodnichenko et al. (2021), Fernández-Villaverde et al. (2021) and Han et al. (2021) rely on neural networks, while Schaab (2020) and Lin and Peruffo (2022) study a nonlinear HANK model based on a solution method unrelated to machine learning. Unlike these papers, we show that computational gains from using neural networks are so significant to even make likelihood estimation of these models possible. To achieve this result, we leverage of the scalability of neural networks by treating model parameters as (pseudo) state variables and introduce the *neural network particle filter*.

We lay down an estimation method that can be applied to quantitative nonlinear HANK models with idiosyncratic and aggregate risk. Other papers have estimated HANK models by approximating aggregate risk to first order or on the assumption of perfect foresight regarding aggregate shocks or by simplifying the aggregate dynamics of the model by making assumptions (Bayer et al., 2020; Bilbiie et al., 2023; Lee, 2020; Auclert et al., 2021).

Outline The paper is organized as follows. In Section 2, we develop the neural network-based estimation method. Section 3 provides proofs of concept of our method. In Section 4, we solve and estimate a quantitative nonlinear HANK model with our developed approach using US data as well as simulated data. Section 5 concludes the paper.

2 An Estimation Framework Based on Neural Networks

In this section, we describe how our methodology to estimate heterogeneous, nonlinear models with neural network works. We first define what neural networks. Second, we explain how neural networks can be used to efficiently characterize the model transition equation treating the parameters as pseudo state variables. Third, we show how another neural network can be conveniently trained to characterize the deterministic steady state variables, which may enter the transition equations of the model.[6] Third, we show how to train neural networks to approximate the outcomes of the particle filter, which is used to evaluate the likelihood function of nonlinear models. Finally, we show how all these neural networks can work together to allow researchers to find out the posterior moments of the parameters of a heterogeneous agents, nonlinear model.

[6]In HANK models, the most notable example is the monetary policy reaction function.

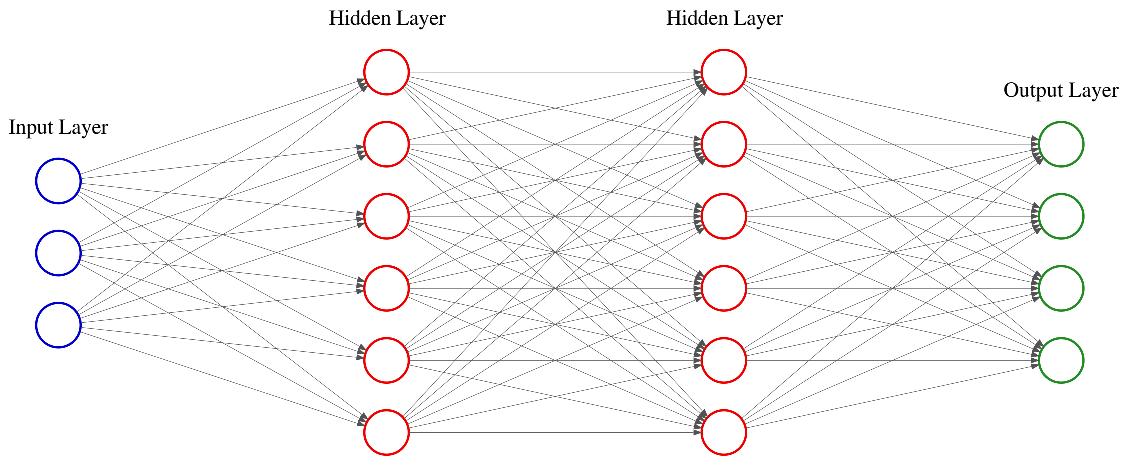

Figure 1: Example of a neural network. An illustrative example of a neural network with three inputs (blue circles), two hidden layers (the two rows of red circles) with six neurons (the red circles) each, and four outputs (the green circles).

2.1 Neural Networks and Deep Learning

Deep learning is a class of machine learning techniques with neural networks as the fundamental building block. Neural networks are function approximators that can represent a very wide class of interesting functions.[7] They are particularly well-suited to solve complex nonlinear macroeconomic models because they can handle many inputs and are an efficient method for quickly learning about complicated mathematical functions or mappings.[8]

A neural network is a mathematical function that maps some inputs S into outputs Y:

$$Y = \psi_{NN}(S|W), \tag{1}$$

where $\psi_{NN}(\cdot)$ is the neural network, and W are the weights of the neural network. The neural network consists of several layers, which can be divided into the input layer, hidden layer(s) in between, and the output layer. Each hidden layer consists of several neurons, which can be seen as nodes in the neural network. Figure 1 provides an example of a neural network with three inputs, two hidden layers with six neurons each, and four outputs.

The neural network combines mathematical functions performed at the single neurons in the network. A single neuron assigns its inputs $s_1, s_2, \ldots s_S$ some weights $w_1, w_2, \ldots, w_S$ and computes their sum (adjusted by a bias w_0). This value is then taken to a nonlinear activation function $h(\cdot)$, such as ReLU (rectified linear unit), CELU (continuously differentiable exponential

[7]The universal approximation theorem states that a feedforward network with at least one hidden layer can approximate any continuous function in a finite-dimensional space with any desired non-zero error given a sufficient width (Hornik et al., 1989; Cybenko, 1989).

[8]Neural networks allow us to break the curse-of-dimensionality as they can handle high dimensional problems much better than classical function approximators (Barron, 1993; Bach, 2017).

linear units), or hyperbolic tangent function, resulting in a single output $\tilde{y}$:

$$\tilde{y} = h(w_0 + \sum_{i=1}^{S} w_i s_i). \tag{2}$$

The activation function $h(\cdot)$ helps the neural network to capture nonlinear dynamics. The neural network can then be characterized as all the weights at each individual neuron, which can be stacked to a vector W. For example, the neural network illustrated in Figure 1 contains 94 weights (including biases).[9]

The vector W with all weights is optimized (trained) to minimize a loss function $\bar{\Phi}^L(W)$, e.g. the mean squared error between observed and predicted points. This loss is evaluated at a total batch B of points. The optimization relies on an iterative stochastic gradient descent method. Specifically, we optimize the neural networks using versions of the Adam algorithm (Kingma and Ba, 2017), which is a very popular optimizer. The neural network training usually exploits graphics processing units (GPUs) as they can process multiple computations simultaneously. PyTorch, Google Jax and TensorFlow are popular open-source machine learning libraries.

2.2 Extended Nonlinear Model Representation

We are interested in solving and estimating dynamic stochastic general equilibrium models in their nonlinear specification. A finite vector of state variables $\mathbb{S}_t$ describes the economy in each period. The economy is subject to exogenous shocks ν_t that affect the response of the state variables. The model features S state variables and U structural shocks. The last part is the model's structural parameters Θ. These objects describe the dynamics of the model and can be expressed as a transition equation:

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t | \, \Theta\right), \tag{3}$$

where f is a nonlinear function. This function f is generally unknown and is often characterized with numerical methods.

The critical step in solving these models is to determine the mapping from the state variables to the control variables ψ_t, which is captured by the policy functions $\psi\left(\cdot\right)$:

$$\psi_t = \psi(\mathbb{S}_t|\Theta), \tag{4}$$

where the policy functions are nonlinear and depend on the state variables. The number of control variables is denoted with O. Once the control variables are determined, the dynamics of

[9]The first hidden layer has $(3+1)\times 6 = 24$ weights (including biases); the second hidden layer has $(6+1)\times 6 = 42$; the output layer has $(6+1)\times 4 = 28$. The first term in the brackets represents the weights plus the bias at each neuron, and the second term indicates the number of neurons in the layer.

the state variables can be analytically calculated.[10] For this reason, the focus in our description is on the policy functions $\psi(\mathbb{S}_t|\Theta)$, which are derived as the solution to a set of optimality conditions:

$$F(\psi(\mathbb{S}_t|\Theta)) = 0, \tag{5}$$

where F denotes the function space.

Extended policy functions: Treating the model parameters as pseudo-state variables. A critical computational bottleneck for estimation is that one has to solve the policy functions for a possibly very large set of parameter values. In the context of nonlinear models, this is unmanageable in most cases. We get around this problem by considering the parameters to be estimated as pseudo-state variables when we approximate the policy functions. Consequently, once these equations are approximated, we would not need to solve the model repeatedly for several parameters values.

Since generally we estimate only a subset of parameters, we divide the set of parameters into two subsets:

$$\Theta = \{\tilde{\Theta}, \bar{\Theta}\}, \tag{6}$$

where $\tilde{\Theta}$ is the set of parameters to estimate and $\bar{\Theta}$ is the set of parameters to calibrate. We treat the parameters to estimate as pseudo-state variables of the economy. The *extended policy functions* – i.e., the policy functions expressed as functions of the current state variables $\mathbb{S}_t$ and the parameters to estimate $\tilde{\Theta}$ – can be written as:

$$\psi_t = \psi\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right), \tag{7}$$

where now the values for the parameters $\tilde{\Theta}$ are treated as pseudo-state variables.[11] The transition equations are given as

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{8}$$

Before explaining how we approximate the extended policy function using neural networks, let us consider the problem of solving nonlinear models with heterogeneous agents.

Incorporating Heterogeneity Our neural network-based solution method can be applied not only to representative-agent nonlinear models but also to models featuring heterogeneous

[10]We can separate between endogenous and exogenous state variables. The dynamics of the exogenous state variables do not depend on the control variables.

[11]Examples of computational papers that use pseudo-state variables in combination with machine learning techniques are Norets (2012), Duarte (2018) and Scheidegger and Bilionis (2019).

agents, multiple countries, or industries. These models feature individual states, $\mathbb{S}_t^i$ and individual shocks, ν_t^i – that is, states that characterize only a single agent or country, or industry i and shocks that only affect a single agent, country, or industry i. When heterogeneity applies to agents, it is often assumed that there exists a continuum of agents, distributed within the unit interval; that is, $i \in (0, 1)$.

Solving models with infinitely many types of agents whose individual states are endogenous requires keeping track of complex and time-varying distributions of individual state variables. The unbounded set of all the moments of these distributions, in principle, affect the equilibrium dynamics of prices and quantities, making the problem of solving these models obviously unmanageable.

We approximate the continuum of agents with a large but finite number of agents L. The distribution of the individual states and the agent-specific shocks for these L agents are summarized as follows:

$$\left\{\mathbb{S}_t^i\right\}_{i=1}^L \quad \text{and} \quad \left\{\nu_t^i\right\}_{i=1}^L. \tag{9}$$

The state variables and shock can, therefore, be written as

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} \quad \text{and} \quad \nu_t = \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}, \tag{10}$$

where with the superscript A we denote the aggregate state variables and aggregate shocks.[12] Similarly, we adjust the policy functions to separate between individual policy functions, which affect agent specific decisions (e.g. their individual asset holdings), and aggregate policy functions (e.g. total capital holdings):

$$\psi_t^i = \psi^I(\mathbb{S}_t^i, \mathbb{S}_t, \tilde{\Theta}, |\bar{\Theta}) \quad \text{and} \quad \psi_t^A = \psi^A(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}), \tag{11}$$

where the individual policy function of agent i is conditioned on the individual states $\mathbb{S}_t^i$ in addition to the entire state vector $\mathbb{S}_t$. It should be noted that the policy function ψ^I is not agent specific in that agents with the same idiosyncratic states, $\mathbb{S}_t^i$, will make the same decisions, ψ_t^i. Note that we define the set of all policy functions as $\psi_t(\mathbb{S}_t, \tilde{\Theta}, |\bar{\Theta})$.

The transition equations in the case of heterogeneity read

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} = f\left(\left\{\left\{\mathbb{S}_{t-1}^i\right\}_{i=1}^L, \mathbb{S}_{t-1}^A\right\}, \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}, \tilde{\Theta}|\bar{\Theta}\right) \tag{12}$$

The number of individual and aggregate state variables are S^i and S_t^A, respectively, so the total number of state variables is $S = S^i \times L + S^A$. The number of exogenous shocks and policy functions is similarly defined as $U = U^i \times L + U^A$ and $O = O^i \times L + O^A$, respectively.

[12] In case of with a finite number of countries, counties, sectors, or banks, this directly defines the state variables without an approximation.

In this paper, we work directly with all the states of the agents as, e.g., in Maliar et al. (2021). An alternative approach is to reduce the dimension of the individual states using reduction techniques for the distribution, such as exploiting the symmetry of the agents following Ebrahimi Kahou et al. (2021). Our method can be used with either approach.

2.3 Neural Network-Based Solution Method

We solve the model numerically using neural networks. In the following, we highlight five essential components. First, we set up two separate neural networks to approximate the extended individual and aggregate policy functions that needs to be trained.[13] Second, we define the loss function that the neural networks aim to minimize. The third component is a procedure to simulate the model to characterize the stochastic solution domain. This step is important because we want to have a tailored stochastic solution domain that accounts for the changing of parameter values. Fourth, we evaluate the expectations of the agents. Fifth, we treat the parameters as pseudo-state variables and define the parameter space. Additionally, if the deterministic steady state value affects the equilibrium dynamics of the model, we also train another neural network to approximate outcomes of the deterministic steady state.

Neural Networks and Extended Policy Functions. We use neural networks to approximate the individual and aggregate policy functions, ψ^I and ψ^A.[14] In particular, we set up two neural networks, ψ^I_{NN} and ψ^A_{NN}, that approximate the individual and aggregate policy function, respectively.[15] We train the neural networks to map the inputs, which consist of S state variables and P parameters we want to estimate, into several output variables O (policy functions).

Once trained, the neural networks return the approximated individual and aggregate policy functions in their extended form – i.e., including the parameters we want to estimate among their inputs/state variables. These functions can be expressed as follows:

$$\psi^i_t = \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right), \quad \text{and,} \quad \psi^A_t = \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right). \tag{13}$$

The set of all (individual and aggregate) policy functions from the neural network is given as:

$$\psi_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right) = \left\{\left\{\psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}, \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}. \tag{14}$$

This formulation highlights that our approach solves simultaneously for the individual and the aggregate policy functions.

[13]We use a separate neural network for the particle filter afterward, which will be explained in the next subsection.

[14]The description nests the scenario of a representative agent economy, which then only includes the aggregate state variables S^A and aggregate policy functions ψ^A.

[15]Setting up two neural networks helps for computational reasons in heterogeneous agent models.

Loss Function and Training of Neural Networks. The next step is to train the neural networks to approximate the extended policy functions. The neural networks ψ^I_{NN} and ψ^A_{NN} are trained to minimize a defined loss function Φ^L. In particular, we minimize the weighted squared residual error for the K optimality conditions (e.g., the consumption Euler equation):

$$\Phi^L = \sum_{k=1}^{K} \alpha_k [F_k(\psi(\mathbb{S}_t|\Theta))]^2. \tag{15}$$

where $F_k(\cdot)$ denotes the residual error for the $k-th$ optimality condition of the model and α_k determines the weight of this condition.[16] We include at least as many optimality conditions as the number of policy functions, so that $K \geq O^i \times L + O^A$.

Furthermore, the neural networks ψ_{NN} are trained on a batch with size B. This can be thought of as having B economies operating in parallel that are used to train the neural network. Consequently, we use deep learning techniques to minimize $B \times K$ equations in each iteration. The loss function is averaged over batch size B, that is:

$$\bar{\Phi}^L = \frac{1}{B}\sum_{b=1}^{B}\sum_{k=1}^{K} \alpha_k \left[F_k(\psi(\mathbb{S}_{t,b}|\Theta_b))\right]^2. \tag{16}$$

The neural network is then trained for tens of thousands of iterations using a stochastic gradient descent method, which minimizes the loss function.

Stochastic Solution Domain and Expectations. The grid points for the state variables and parameter values in each iteration are drawn randomly from the state and the parameter space. In particular, the neural network is trained on a stochastic solution domain, from which the values of the state variables are drawn randomly.

We are interested in training the algorithm only on the relevant n-dimensional domain of the state space. We ensure this via a simulation step, in which we we directly sample from the model to approximate the ergodic distribution of the model's state variables. Specifically, after training the neural network with the given draw of state variables over batch B in the current iteration, we simulate each economy (batch) for T^{sim} periods forward using our trained extended policy functions. The endpoint of the simulation gives the solution domain that we use in the next iteration for the optimization of the neural network. In that regard, we draw random points from a $S = S^i \times L + S^A$-dimensional domain, which covers the ergodic distribution.

The expectations are evaluated with a Monte Carlo approach, which relies on the antithetic variate to increase the precision.[17] This approach handles hundreds of shocks and is well suited

[16]The weight on the different equations can vary, enabling, for example, aggregate conditions to have higher weights than the idiosyncratic equations. Additionally, the losses in different equations can have different magnitudes.

[17]We randomly draw M sets of next period shocks, that is $\{\nu^m_{t+1}\}^M_{m=1}$, to approximate expectations. We use the antithetic variate method to increase the efficiency of Monte Carlo integration and reduce the number of necessary

to evaluate expectations in a stochastic setup with randomly drawn shocks. Additionally, we perform several rounds of optimization during each training step. For a given draw of state variables, we repeat the drawing of shocks and rerun the optimization for this different set of shock realizations.

Parameter Space and Pseudo-State Variables. The salient feature of our solution method is to treat model parameters as pseudo-state variables. We start by restricting each parameter that is estimated (and thus treated as pseudo-state variable) to lie inside some bounds:

$$\tilde{\Theta} = \left\{ \left[\tilde{\Theta}^{\underline{1}}, \tilde{\Theta}^{\overline{1}}\right], \left[\tilde{\Theta}^{\underline{2}}, \tilde{\Theta}^{\overline{2}}\right], \ldots, \left[\tilde{\Theta}^{\underline{P}}, \tilde{\Theta}^{\overline{P}}\right] \right\}, \tag{17}$$

where $\tilde{\Theta}^{\underline{i}}$ and $\tilde{\Theta}^{\overline{i}}$ is the lower bound and upper bound for each parameter that is estimated. Due to the bounded space, we can draw from a tighter distribution and increase the precision. For each iteration, we draw randomly from the parameter space.[18] Therefore, each economy (batch) has a different parameter combination, which is used to train the neural network.

After the maximizing step, we redraw the parameter space and then simulate each economy (batch) forward. This simulation step also ensures that we train each economy in its relevant stochastic solution domain for the drawn parameter combination. As an example, consider an economy with strong differences in the discount factor. The capital accumulation would then also vastly differ, resulting in a different stochastic solution domain for different parameterizations. Using the simulation step, we account for this and tailor the stochastic solution domain for each parameter combination. To sum up, we train the neural network over the entire parameter space as we consider a different set of economies in each training step. At the same time, we ensure via simulation that we are solving for the policy function in its relevant stochastic solution domain.

Neural Networks and the Deterministic Steady State. Often approximating the individual and aggregate policy functions requires knowing the deterministic steady state (DSS) of the model. Note that in the DSS of a heterogeneous agents economy, agents face and experience idiosyncratic risk, while aggregate risk is absent, in line with, e.g., Fernández-Villaverde et al. (2021). This task amounts to solving the Bewley-Huggett-Aiyagari economy, in which agents face idiosyncratic risk, but aggregate risk is absent. We can use an additional neural network to obtain a mapping from the parameter values to deterministic steady state (DSS) values, denoted as ψ^{SS}. An example in the context of a HANK model is the DSS nominal interest rate as it enters the Taylor rule.

draws. The antithetic variates technique creates to a given path $\{\nu_1, \nu_2, \nu_3, \ldots, \nu_{M/2}\}$ also its antithetic path $\{-\nu_1, -\nu_2, -\nu_3, \ldots, -\nu_{M/2}\}$.

[18]We draw from a Sobol sequence as it has good distributions in the unit hypercube. Other distributions, such as a truncated multivariate normal or the prior distribution, could also be used for drawing, and potentially combined with Latin hypercube sampling.

The inpus for this neural network are only the parameters. The mapping from parameters to the steady state values using a neural network as a function approximator can be written as:

$$\psi^{SS} = \psi^{SS}_{NN}\left(\tilde{\Theta}|\bar{\Theta}\right). \tag{18}$$

We need to adapt our approach slightly to incorporate this neural network in our solution and estimation procedure. Specifically, we train the neural network for the steady state values in the first step, which precedes solving the individual and aggregate policy functions in the second step. For this first step, we specify a separate loss function, in line with the description of this section, and train the neural network for the steady state values.[19]

This step of solving for the Bewley-Huggett-Aiyagary economy is typically a very time consuming step. One advantage of adopting neural networks to solve heterogeneous agents models is to reduce dramatically the computation burden involved in this step. This problem is especially aggravated when these models are estimated since estimation requires solving the model (and hence the Bewley-Huggett-Aiyagari model too) for several combinations of parameter values.

We then use this neural network for the DSS in the second step, during which we solve for the individual and aggregate policy functions. It is important to note that our approach solves for the entire equilibrium without distinguishing between variables that affect idiosyncratic and aggregate dynamics.

2.4 Neural Network-Based Evaluation of the Likelihood Function

The likelihood function of models is typically obtained by recasting the model in its state-space representation and then run a filter to compute the model's one-step ahead predictive densities over the available sample period, $p\left(y_t|y_{t-1},...,y_1\right)$, where y_t denotes the vector of observable variables in period t. The products of the model's predictive densities in every sample period $t \in \{1,...,T\}$ is the sought likelihood function. In symbols,

$$\mathcal{L}_{\bar{\Theta}}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) = \prod_{t=1}^{T} p\left(y_t|y_{t-1},...,y_1\right). \tag{19}$$

Thus, for a given data set of T periods, $\mathbb{Y}_{1:T}$, a filter can be thought as a mapping from the set of model parameters, $\tilde{\Theta}$, to the value of the likelihood function.[20] We train a neural network to accurately approximate this mapping.

In this section, we define a state-space model that encompasses the broad set of DSGE models analyzed in this paper. Second, we describe how an additional neural network can be trained to

[19]Note that it may also require training the individual policy functions for the DSS specifically.

[20]The notation used to express the likelihood makes it clear that while the likelihood function depends on all the parameters $\Theta \in \left\{\tilde{\Theta},\bar{\Theta}\right\}$, we want to emphasize the dependency of this function on the the subset $\tilde{\Theta}$, which collects the parameters that are estimated with likelihood methods.

efficiently approximate the likelihood function using a filter.

State-Space Representation. A state-space model is made of a set of transition equations, which, in our case, capture the equilibrium dynamics of a nonlinear general-equilibrium model with heterogeneous agents, and a set of measurement equations.

To evaluate the likelihood function, one needs to solve the transition equations, which have been the focus of the previous subsection. For convenience, we repeat the transition equations:

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t|\Theta\right). \tag{20}$$

To complete the state-space representation, we now accompany the transition equations with a set of measurement (or observation) equations linking the model's state variables $\mathbb{S}_t$ to the observable y_t. Formally,

$$y_t = g(\mathbb{S}_t|\Theta) + u_t, \tag{21}$$

where g is a function and u_t denotes a vector measurement errors.

The formulation highlights that the assessment of the likelihood requires the solving of the transition equations. To maximize the likelihood function, the model must be solved a very large number of times. But, for the type of nonlinear models considered in this paper, this requirement is a critical bottleneck that causes the likelihood estimation of these models to be totally unfeasible.

As explained earlier, we overcome this bottleneck by training neural networks to find the extended solution mapping of our nonlinear model, which corresponds to the transition equations for the entire parameter space shown in equation (12), which we report below for convenience:

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t, \tilde{\Theta}|\bar{\Theta}\right) \tag{22}$$

It should be noted that, unlike the specification in (20), these transition equations are now conditioned only on the parameter vector $\bar{\Theta}$ for which we do not need to evaluate the likelihood function because they are not estimated. The parameters that need to be estimated, $\tilde{\Theta}$ now appear on the left of the conditioning sign, stressing our choice of treating these parameters as pseudo-state variables. The advantage is that once we have solved for these transition equations, as discussed in the previous subsections, we can directly use them as input to run our filter. This is an important feature because speed is a critical prerequisite to be able to run the filter as many times as needed, especially in this nonlinear setting.

Particle Filter. The nonlinear model solution requires to use a Monte Carlo filter such as the particle filter (Fernández-Villaverde and Rubio-Ramírez, 2007; Herbst and Schorfheide, 2015).[21] The particle filter, which extracts the hidden states and shocks, can then be used to approximate the likelihood function. One challenge is that an insufficient number of particles results in a noisy approximation. However, an increase in the particles prolongs increasing the computational time substantially. Even though the particle filter could be directly used within our estimation approach, it can be very time-consuming. This is a particular problem as the filtering step is usually repeated hundreds of thousands of times, resulting in a massively increased computational time.

Training the neural network to approximate the likelihood. To overcome the limitation of time and an imprecise approximation, we are training a separate neural network with the objective to approximate the likelihood. Once successfully trained, this neural network returns a fast and reliable approximation of the likelihood that can be directly used to estimate the parameters of the model. In case of a Bayesian approach, the approximated likelihood function can be combined with the prior distribution to directly compute the moments of the resulting posterior distribution.

The neural network approximates the mapping from the parameters $\tilde{\Theta}$ to the likelihood, which can be expressed as follows:

$$\mathcal{L}^{NN}_{\bar{\Theta}}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \approx \mathcal{L}_{\bar{\Theta}}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \tag{23}$$

To train this neural network, we create a set of points that map the parameters to the corresponding likelihood using a filter. Specifically, we use a Sobol sequence to obtain several thousand quasi-random draws from the parameter space. We then run the particle filter to obtain the associated likelihood for each draw. The neural network is then trained to predict the likelihoods using these draws as inputs. For the loss function, we employ the mean squared error between the predicted values to the calculated likelihoods from the particle filter.[22] The model is trained over thousands of training steps, with the batch size determining the number of points used within each training step.

While we evaluate the likelihood at only several thousand points, we use the neural network to learn the connection between these points. Conceptually, we train the neural network so as to perform a nonlinear interpolation of the likelihood values. This allows us to evaluate the

[21]The particle filter has been used to filter highly nonlinear models with, for instance, occasionally binding constraints (Gust et al., 2017; Atkinson et al., 2020) or (endogenous) multiple equilibria (Aruoba et al., 2018; Rottner, 2021)

[22]It is often convenient to work directly with the log-likelihood instead of the likelihood. This approach entails summing the log of predictive densities across all the periods in the sample, T. It should be noted, however, that using the log-likelihood introduces a slight bias due to Jensen's inequality (specified for the logarithm as $E(\ln(X)) \leq (X))$ because our approach implicitly relies on averaging.

likelihood at points we did not initially assess. Once the neural network is trained, we can quickly evaluate the likelihood of millions of parameter combinations. Therefore, computing the posterior moments does not require running any filters as one can rely on the likelihood function approximated by the neural network.

As we shall show in Section 3.1, an important advantage of our likelihood approximation based on neural networks is the reduction of the computational noise and inaccuracy that sometimes characterizes the particle filter. The reason is that the neural network is trained to predict the likelihood using a large number of values. Therefore, the neural networks learns the structural mapping and reduces the noise that is sometimes associated with the particle filter. An important feature for this result is that the neural network is not overfitted to the sample. To prevent overfitting, the drawn points are divided into a training and validation sample. While the neural network is trained with the training sample to minimize the loss criteria, the validation sample is used to evaluate the loss on a sample that is not used for optimization. The loss in the validation sample provides a helpful metric for when to stop the training and to analyze the fit.

Real-Time Estimation and Forecasting. One challenge in practice, such as forecasting or policy analysis in real-time, is that the estimation needs to be updated with newly incoming data. While a typical estimation approach would need to be entirely repeated, our approach offers alternative routes. For instance, a strategy is to store the weights of each particle and the states in the last period of the sample. We can then run the particle filter for the additional periods to obtain a set of values of the new likelihood function. These values can then be used to retrain the neural network approximating the likelihood.

2.5 Estimation

We can now proceed to the final estimation step. Equipped with the neural network approximation of the likelihood function, we can estimate the model parameters with a standard optimization routine or approximate the moments of the posterior distribution in a straightforward way.

Maximum Likelihood Estimation. Likelihood maximization can be represented as follows:

$$\tilde{\Theta}^{ML} = \arg\max_{\tilde{\Theta}} \mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \tag{24}$$

We restrict the parameter space to be inside the boundaries of our neural network. This is not a constraining assumption, as we could extend the lower and upper bounds of the neural network. In practice, it should be ensured that the neural network covers a large enough parameter space so that the model likelihood lies very likely inside its bounds.

Bayesian Estimation Bayesian inference rests on approximating the posterior distribution for the model parameters. This distribution, $p\left(\tilde{\Theta}|\mathbb{Y}_{1:T}\right)$, combines the likelihood with a prior distribution, $p(\tilde{\Theta})$:

$$p\left(\tilde{\Theta}|\mathbb{Y}_{1:T}\right) = \frac{\mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \times p(\tilde{\Theta})}{\int \mathcal{L}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}\right) \times p(\tilde{\Theta})\, d\tilde{\Theta}}$$

where the numerator is the so-called posterior kernel and the denominator is the marginal likelihood, which does not depend on the model parameters to estimate. We are using truncated densities for the priors to ensure that the draws of the parameters are inside the boundaries of our solved neural network. Similarly to likelihood estimation, this truncation does not constrain our analysis in any meaningful way.

Now that we have approximated the likelihood function, we can compute the moments of the posterior distribution with the Random Walk Metropolis Hastings algorithm. The algorithm is described in detail in Appendix A.

Other Estimation Approaches: GMM and impulse-response matching. Other methods such as the method of moments, generalized method of moments, or impulse-response matching could be also embedded in this framework. Instead of using the neural network to approximate the likelihood, neural networks are trained using loss function defined with respect to moments or the impulse response functions of interest.

Refinements: Sequential Estimation Approach The accuracy of estimation can be improved by making the estimation approach sequential. We initially draw the parameter values from the unit hypercube when training the neural networks to approximate the extended policy functions as well as the likelihood function. This is a very agnostic approach that is very suitable if we have no precise information regarding the shape of the likelihood function. However, the neural network-based approximation of likelihood is likely to provide quite valuable information that can be exploited to increase the precision of the model solution as well as likelihood approximation. Instead of drawing parameters from the unit hypercube, we could draw now from the posterior distribution or likelihood when training the extended policy functions. Similarly, we can also draw from this distribution when generating the set of points to train the neural network that approximates the likelihood. We use this information then by putting more weights on specific areas of the parameter space when training the different neural networks. One can then go on training neural networks by following a sequential approach to achieve a satisfactory level of accuracy.

Multiplicity of equilibria. Little is known about existence and uniqueness of rational expectations equilibria for the type of nonlinear models studied in this paper. Our neural network solution method is based on minimizing the Euler equation errors and finds a solution if it exists. If the solution does not exist for a given set of parameter values, the neural network solution method fails to converge to a sufficient low error and, in this case we assign an extremely low value to the likelihood, as standard in the literature. As a result, the parameter values connected with nonexistence of an equilibrium will be effectively discarded in estimation. An equally popular method is to truncate the prior distributions so as to exclude parameters values that are inconsistent with existence or uniqueness of a rational expectation equilibrium. For instance, it is known that a highly persistent or highly volatile preference shock may lead to nonexistence of rational expectations equilibria in dynamic general equilibrium models featuring an occasionally binding ZLB constraint – Bianchi et al. (2021) call this situation deflationary spirals. We appropriately truncate the prior for the persistence parameter of this shock when we estimate the HANK model. In case of multiplicity of equilibria, the neural network solution method selects the rational expectations equilibrium with no sunspot and the model's likelihood is evaluated at that equilibrium.[23] Adding sunspots-driven dynamics is clearly beyond the scope of this paper.

2.6 Summary of the Neural Network Estimation Approach

Our neural network-based estimation approach consists of three key steps: i) Train the neural network to obtain the extended policy functions, ii) Train the particle filter neural network to get the likelihood over the parameter space, iii) Run maximum likelihood estimation or a Metropolis-Hastings algorithm. Our accompanied codes rely on *PyTorch* as the machine learning framework.

3 Validation of the method

We provide three proofs of concept to establish the validity of our estimation approach. Specifically, we assess our neural network-based method i) by comparing its estimation output to the true solution of a stylized linearized New Keynesian DSGE model that admits analytical solution; ii) by comparing its estimation output to the true solution of a nonlinear RANK model with a recurrently binding zero-lower-bound constraint using a state-of-the-art (non-neural network) method to solve nonlinear models; iii) by showing its ability to recover the true parameter values of a nonlinear HANK model with occasionally binding zero-lower-bound constraint.

[23]We can show this result when we study the linearized New Keynesian model in our first proof of concept. These results are available upon request.

Parameters		LB	UB	Parameters		LB	UB
β	Discount factor	0.95	0.99	θ_Π	Mon.pol. inflation response	1.25	2.5
σ	Relative risk aversion	1	3	θ_Y	Mon.pol. output response	0.0	0.5
η	Inverse Frisch elasticity	1	4	ρ_A	Persistence TFP shock	0.8	0.95
φ	Price duration	0.5	0.9	σ_A	Std. dev. TFP shock	0.02	0.1

Table 1: Model parameters values – first proof of concept. The panel shows the parameters of the three equation NK model. We solve the neural network for the parameter space spanned by the lower bound (LB) and upper bound (UB) of all parameters.

3.1 Proof of concept 1: a linearized three-equation DSGE model.

We begin by solving and estimating a version of the linearized three equation New Keynesian model. This choice is because this model has an analytical solution. We can then compare the model solution obtained by applying our neural network approach to the analytical solution. We demonstrate that the approach based on neural networks provides a precise solution for the considered parameter space. Using simulated data, we also establish the precision of our likelihood approximation with a neural network.

The model is an off-the-shelf three-equation NK model with only one shock (TFP shock). The model is log-linearized around its unique steady-state equilibrium and the resulting equations are shown in Appendix B. This model is so tractable that a closed-form solution can be derived analytically. This solution is also shown in Appendix B.

3.1.1 Neural Network and Extended Policy Functions

We use our neural network approach to characterize the extended policy functions of inflation and output gap computationally as shown in the previous section. We train the neural network by minimizing the residual error for the Euler equation and the Phillips Curve. Importantly, the policy functions are simultaneously conditioned on the state variable and the parameters, which results in nine inputs.

Table 1 describes the upper and lower bounds of the parameters, for which we solve the neural network.

Training and Convergence We use 500,000 iterations to train the neural network.[24] The batch size is set to 1000, which can be thought of as having 1000 different parallel economies in each iteration to train the neural network. After each iteration, new parameters are drawn from the unit hypercube, and the economy is simulated for 10 periods to get a new draw for the state variable for each batch. The state is simulated using the law of motion of the exogenous state.

[24]The neural network contains five hidden layers with 256 neurons each. The activation function for the hidden layers is continuously differentiable exponential linear units (CELU). The optimizer is AdamW. The learning rate follows a cosine annealing scheduler starting from an initial rate of $1e-4$ and terminating at a rate of $1e-6$.

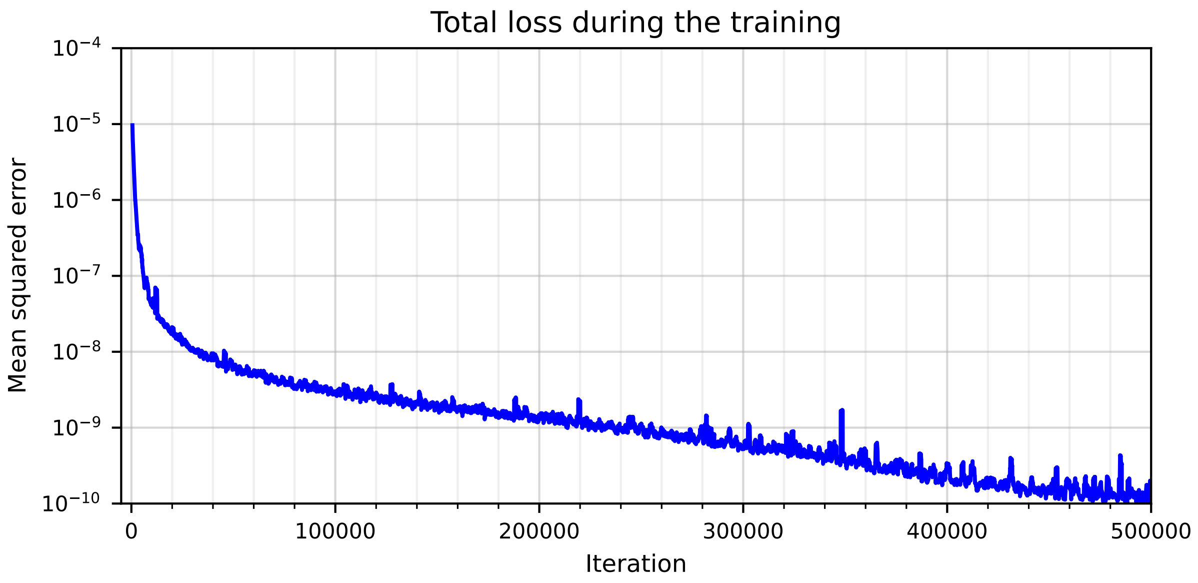

Figure 2: Convergence of the neural network solution. The figure shows the dynamics of the mean squared error averaged across the two equations that we minimize. The loss is calculated as moving average over 1000 iterations. The vertical axis has a logarithmic scale.

The expectations are evaluated using 50 draws and we conduct five internal optimization steps during each iteration.[25] The convergence of the neural network is shown in Figure 2, where the mean squared error averaged across the equations that we minimize is shown. At the end of the training, the neural network approximates the extended policy functions with a very high accuracy: the average mean squared error reaching about 10^{-10}.

Results Figure 3 shows the extended policy function of the output gap as the structural parameters of the model vary. The extended policy function is evaluated at a negative one standard deviation shock, and the unvaried parameters are fixed in the middle of their bounds. The comparison with the analytical solution demonstrates that our neural network is able to reconstruct the true solution very well as the lines almost perfectly coincide. The shown connection is also highly nonlinear, emphasizing the ability of neural networks to capture nonlinearities.[26] While we have focused on the output gap, Figure 10 in the Appendix shows that identical conclusions can be drawn from a similar graph regarding the policy function of inflation. The neural network-based solution coincides almost perfectly with the true solution.

[25]Our algorithm follows the general setup that we have derived in Section 2. Even though the simple setting could be exploited to fine-tune the method, we abstract from this for illustration purposes. For instance, we could directly sample from our state space as the only state is an exogenous shock. We also could pin down expectations without Monte Carlo integration by exploiting the linear setup.

[26]While the model is linearized, variations in the structural parameters generally engender nonlinear dynamics of the model variables.

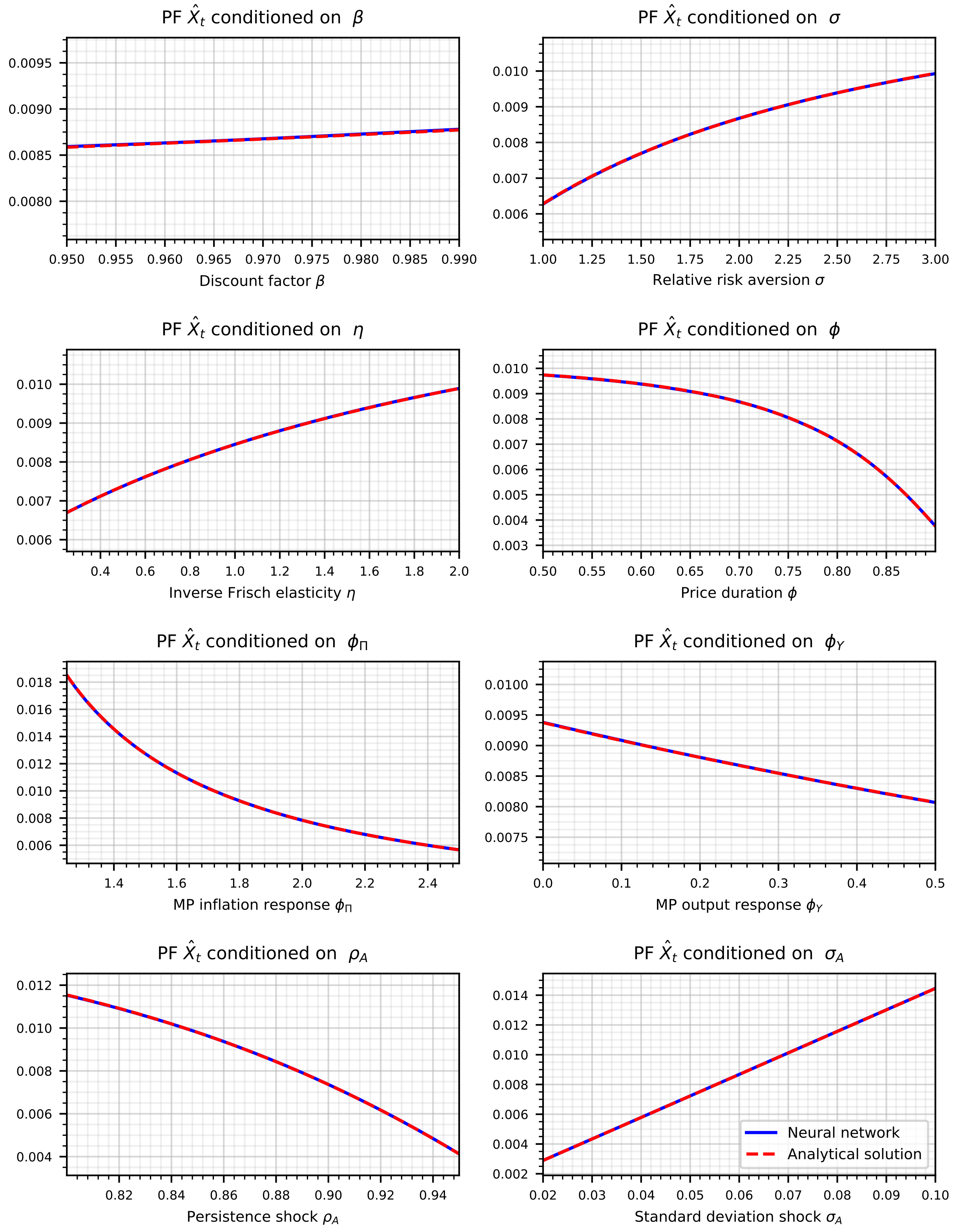

Figure 3: Accuracy of the neural network solution method. Comparison between the neural network and the analytical policy function of output. The plot shows how changes in each structural parameter affect the policy function for the output gap $\hat{X}_t$ approximated by the neural networks. The policy function is evaluated at a negative one-standard deviation shock. The unvaried parameters are fixed at their mean.

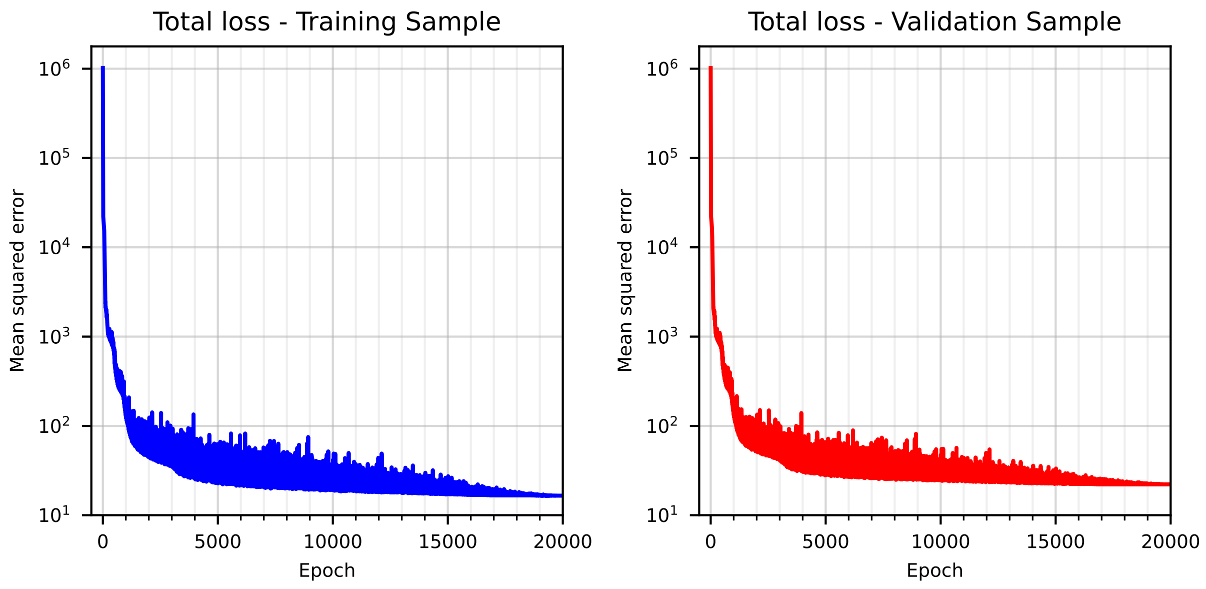

Figure 4: Training and validation samples convergence. The figure shows the dynamics of the mean squared error for predicting the likelihood throughout the training. The loss is shown for the training sample (left figure) and the validation sample (right figure) for each epoch. The vertical axis is expressed in a logarithmic scale.

3.1.2 Neural Network and Likelihood Estimation

As a next step, we want to estimate the likelihood of this model using our neural network approach. Using the linearized model as a data-generating process, we simulate inflation and output gap data for 100 periods for which we set the parameters to the middle of their bounds.[27]

Training We train the neural network for the likelihood approximation using 10,000 quasi-random draws from the parameter space as input.[28] Thus, we first use the particle filter to calculate the mapping between parameters and the likelihood for the drawn points. When running the particle filter, we choose on purpose a very low number of particles with 100 to highlight the ability of the neural network to reduce the noise. We divide the set of points into a training sample (80%) and a validation sample (20%). The neural network is trained using the training dataset across 20000 epochs, wherein each epoch the entire training dataset is processed.[29]

The loss during the training of the neural network is depicted in Figure 4. The left graph shows the error for the training sample, finishing at approximately 17. The neural network is

[27]We add a small measurement error (for each series 5% of its own variance) to avoid a degeneracy of the likelihood when using the particle filter.

[28]The neural network contains 4 hidden layers with 128 neurons each. The activation function for the hidden layers are CELU. The optimizer is AdamW. The learning rate follows a cosine annealing scheduler from an initial level of $2e-4$ and terminating at $1e-7$.

[29]The training sample is divided into batches of 100. Each batch is passed through the network in sequence until the entire sample has been used, completing one epoch. In our context, one epoch comprises 80 iterations, amounting to a total of 1.6 million iterations across all epochs.

trained to learn the structural mapping, with the level of the loss indicating the noise magnitude of the particle filter. If the training set is generated using more (less) particles, the error would be lower (higher). For instance, employing a scenario with 1000 particles results in a final loss near 11.

To address potential overfitting concerns, we inspect the error associated with the validation sample, as displayed in the right chart. Starting from an initial value near 10^6, both errors converge to levels around 17 and 22, respectively. The proximity of errors in both samples, coupled with the consistent decrease in the error of the validation sample throughout the training process, suggests that the neural network is not overfitted.

Results The next step is to demonstrate that the neural network offers a highly accurate approximation of the likelihood function. As shown in Figure 5, we examine approximation for the likelihood concerning the relative risk aversion parameter σ.[30] The orange dots show the mapping between the parameter and the likelihood directly coming from the particle filter, which is used as training data.[31] The neural network cuts through the cloud of points, effectively discarding the noise from the particle filter conducted with a low number of points. This also highlights that our approach accurately captures the shape of the likelihood. For further evidence, Appendix B shows the same exercise for the other parameters. This concludes the first proof of concept.

3.2 Proof of concept 2: Comparison to a state-of-the-art estimation method for nonlinear models

We assess whether our neural network estimation strategy can recover the true data-generating process of a stylized nonlinear RANK model augmented with a ZLB constraint, which we show in Appendix C. We then check that estimated parameters we obtained our neural network based approach are similar to the true ones and to the ones obtained adopting a state-of-the-art global solution method. This method relies on solving the model with global methods and evaluating the likelihood with a particle filter for every single draw of the Metropolis-Hastings algorithm (Herbst and Schorfheide, 2015). While this second method is too slow to allow us to estimate more complicated linear models, it provides a useful benchmark to assess the ability of our neural network approach to solve and estimate nonlinear models.[32] The neural network approach performs very well in recovering the parameters of the true data-generating process. The posterior

[30]The other parameters are set to their mean, while only σ is varied.

[31]As we train the neural network in an 8-dimensional space, we hold the other parameters at their mean and generate these specific depicted points solely for illustration purposes.

[32]The alternative approach is subject to the curse-of-dimensionality and is very time consuming since it requires running the particle filter repeatedly. Even though some scholars have estimated nonlinear RANK models (e.g., Gust et al., 2017; Atkinson et al., 2020), the scale and the scope of models that can be estimated with this approach is unavoidably quite limited.

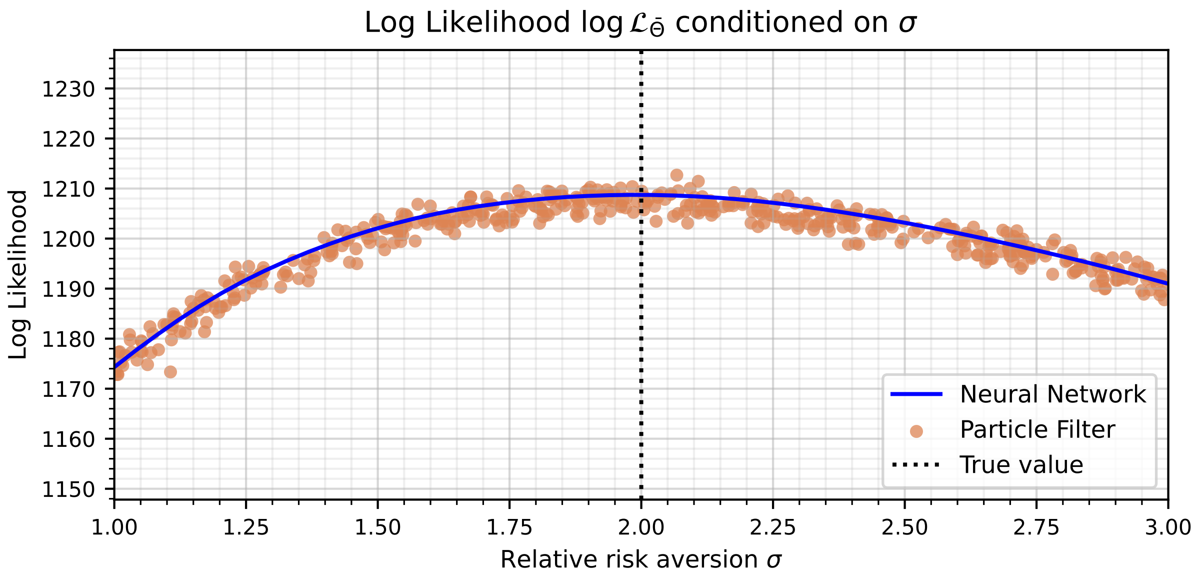

Figure 5: Accuracy of the neural networks approximating the model's policy functions. Comparison between the neural network and single draws from the particle filter. The plot shows how variations in the structural parameter σ affect the log likelihood $\mathbb{L}_{\bar{\Theta}}$. The unvaried parameters are fixed at their mean.

median is close to the true value and is always contained inside the 90% confidence interval. Furthermore, the shape of the posterior is very close to those obtained with the alternative approach relying on global solution methods. However, the alternative method is considerably slower than the proposed method. Results are shown in Appendix C.

3.3 Proof of concept 3: Estimation of a nonlinear HANK model using simulated data.

We now solve and estimate a nonlinear HANK model that features jointly idiosyncratic and aggregate risk. The households can trade one asset and face idiosyncratic income risk and a borrowing limit. The zero lower bound constrains monetary policy at times. The model features demand, supply, and monetary policy shocks. The HANK model is solved and estimated in its nonlinear specification. The HANK model is shown in details in Appendix E.

We estimate this model using simulated data generated from the same model. The objective is to assess whether our solution and estimation methods allow us to retrieve the "true" parameter values used to generate the simulated data. The model is simulated for 144 periods and the observable variables are output growth, inflation, and nominal interest rate. The estimation includes 12 structural parameters. Table 3 shows the priors for the estimated parameters. The prior densities for all parameters are truncated normal distributions. The prior mean corresponds to the true value, while the standard deviation is rather loose to avoid that the results are driven by the prior. As before, the truncation ensures that the drawn parameters lie inside the parameter space of the neural network.

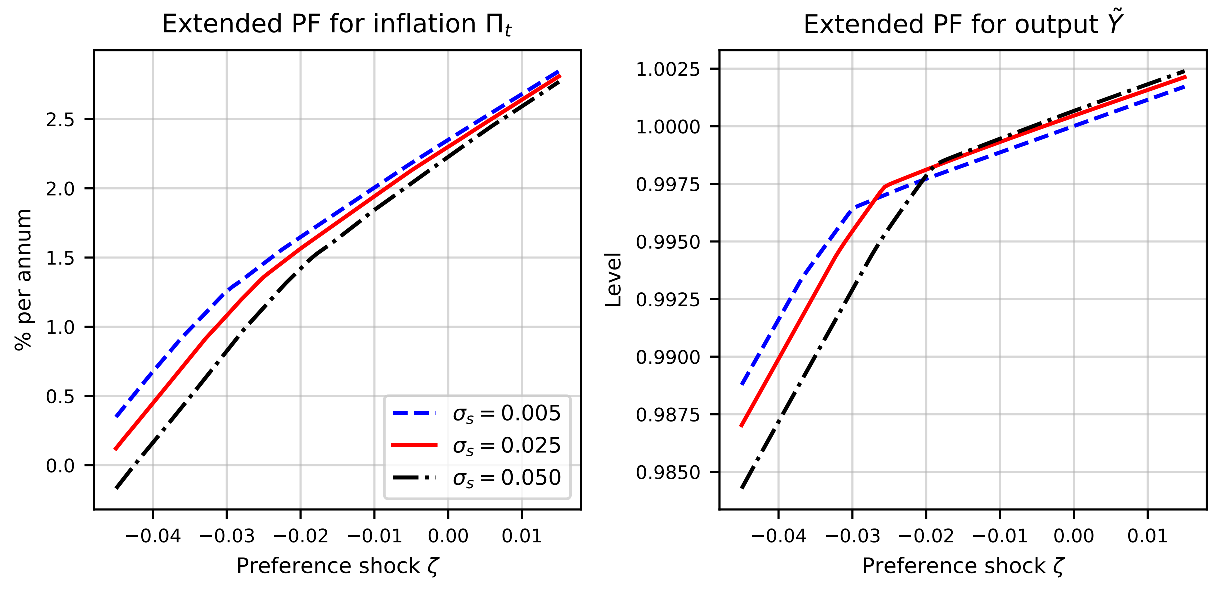

Figure 6: Aggregate policy functions of the nonlinear HANK model. The figure illustrates how the aggregate policy functions of inflation (left) and output (right) vary over several realizations of the preference shock, ζ. The three colored lines denote the two policy functions when the standard deviation of the idiosyncratic shock to households' income, σ_s, is set to be equal to 0.005, 0.025, and 0.050.

We train a separate neural network to perform the three following tasks: (i) characterization of the deterministic steady state (DSS) equilibrium; (ii) approximation of the individual and aggregate policy functions; (iii) approximation of the particle filter needed to evaluate the likelihood.

The deterministic steady state (DSS) equilibrium It is necessary to approximate the mapping from the model parameters to the (detrended) deterministic steady state first. The reason is that the monetary policy reaction function is specified using the value of inflation and output at the deterministic steady-state equilibrium. We set the number of agents L to 100, giving us with 205 state variables and 12 pseudo-state variables (parameters). The batch size is set to 100. The neural network trained to approximate the steady state has four layers. We use 128 neurons in each layer and rely on CELU and PReLU as acitivation functions.

We train a neural network ψ_{NN}^{SS} for the selected DSS values that is conditioned on the parameters.[33] The loss function is conditioned on the Euler equation adjusted for the borrowing limit and the clearing of markets.

[33]Solving the system of equations in the DSS requires that we also solve for the individual agents' policy functions. Thus, we use a temporary neural network that maps agents' individual states and parameters to agents' labor decisions.

Calibration for the data-generating process

Parameters		Value	Parameters		Value
β	Discount factor	0.99825	θ_Y	MP output response	0.98
η	Inverse Frisch elasticity	1	D	Government debt	0.25
ϵ	Price elasticity demand	11	$\underline{B}$	Individual borrowing limit	-0.15
σ	Relative risk aversion	1	ρ_s	Persistence labor productivity	0.8
g	Average growth rate	1.00	σ_s	Std. dev. labor productivity	0.045
γ^τ	Tax progressivity	0.18	ρ_ζ	Persistence preference shock	0.7
φ	Rotemberg pricing	100	σ_ζ	Std. dev. preference shock	0.015
χ	Disutility labor	0.91	σ_g	Std. dev. growth rate shock	0.004
Π	Inflation target	1.00625	σ_{mp}	Std. dev. MP Shock	0.00125
θ_Π	MP inflation response	2.6			

Estimation

Par.	True	Prior					Neural Network		
	Value	Type	Mean	Std Bound	Lower Bound	Upper Median	5%	Posterior 95%	
Parameters related to idiosyncratic risk									
$\underline{B}$	-0.15	Trc.N	-0.15	0.025	-0.5	0.0	-0.14	-0.018	-0.10
ρ_s	0.8	Trc.N	0.8	0.025	0.7	0.9	0.79	0.76	0.82
σ_s	0.045	Trc.N	0.045	0.01	0.01	0.06	0.53	0.047	0.058
Parameters related to aggregate risk									
β	0.99825	Trc.N	0.99825	0.001	0.995	0.9985	0.9981	0.9973	0.9984
g	1.0039	Trc.N	1.0039	0.001	1.0025	1.0055	1.0042	1.0037	1.0047
φ	100	Trc.N	100	5	50	150	101	941	108
θ_Π	2.6	Trc.N	2.6	0.025	1.5	3.0	2.59	2.55	2.64
θ_Y	0.98	Trc.N	0.98	0.025	0.25	1.25	1.00	0.96	1.04
ρ_ζ	0.7	Trc.N	0.7	0.01	0.5	0.8	0.70	0.68	0.72
σ_ζ	0.015	Trc.N	0.015	0.01	0.001	0.022	0.012	0.011	0.13
ρ_g	0.4	Trc.N	0.4	0.01	0.2	0.6	0.40	0.38	0.41
σ_g	0.004	Trc.N	0.004	0.001	0.0001	0.006	0.047	0.0422	0.053
σ_{mp}	0.00125	Trc.N	0.00125	0.001	0.0001	0.00375	0.00171	0.00137	0.00212

Table 2: The upper panel shows the calibration for the nonlinear HANK model with the ZLB and borrowing limit, which is used as data-generating process. The lower panel shows the prior and the posterior. The prior type indicate the prior density function, where Trc.N stands for a truncated normal distribution.

The policy function of the nonlinear HANK model The next step is to solve the HANK model in its nonlinear specification with idiosyncratic and aggregate risk. We use two neural networks to train the conditioning of the policy functions on the 205 states and 13 parameters.[34] These neural networks are jointly trained to minimize the residual error in selected equilibrium conditions from the individual agents' and aggregate equations. Appendix F.3 provides more

[34]The neural network for the individual policy functions is additionally conditioned on the agent's states. We capture the Karush-Kuhn-Tucker conditions due to the borrowing limit using the Fischer-Burmeister equation, as discussed in Appendix F. Alternatively, the constraints could be directly embedded in the neural network architecture, e.g., in Azinovic and Žemlička (2023).

information on the training of the neural networks.

In Figure 6, we show the aggregate policy functions for inflation (left panel) and output (right panel) as a function of realizations of the preference shocks (x-axis) evaluated at the "true" parameter values. The three lines denote three different levels of volatility of the idiosyncratic shocks to income. When the preference shock is negative, households want to save more and the aggregate demand contracts. Output and inflation decline as a result, explaining why the aggregate policy functions are upward sloping. For realizations of the preference shocks sufficiently negative, the slope of the aggregate policy functions become steeper. The kink occurs when the negative preference shock is sufficiently large to lead to a contraction in the aggregate demand that causes the ZLB constraint to become binding. By hindering monetary policy, the binding ZLB constraint makes the fall in aggregate demand even worse, leading to a more pronounced declined in output and (depending on the slope of the Phillips curve) in inflation.

Interestingly, the level of the uninsurable idiosyncratic risk (σ_s) has different effects depending on whether the economy is at the ZLB or not. Specifically, when the idiosyncratic risk is higher (move from the blue dashed line to the black dashed-dotted line), output increases and inflation falls. This is because households want to work more in face of the heightened income risk. The increase in the labor supply leads to higher output and lower marginal costs and inflation.

The binding ZLB reverses this mechanism. The reason is that a higher idiosyncratic income risk makes the ZLB more likely; that is, the ZLB constraint starts to bind for larger realization of the preference shocks. In the right Figure 6, we observe that the kink of the black dashed-dotted line, which is associated with the highest level of idiosyncratic risk, is rightmost relative to the other two lines. As observed by Fernández-Villaverde et al. (2021), the larger the idiosyncratic risk, the stronger the precautionary saving motive, the lower the real interest rate, the more frequent the ZLB constraint binds. A more frequently binding ZLB constraint hinders the central bank's ability to stabilize output in face of a negative preference shocks. As a result, the level of output is lower at the ZLB when the idiosyncratic income risk is higher. On the inflation side, inflation at the ZLB is lower if the idiosyncratic income risk is larger because the central bank's ability to stabilize prices is constrained.

The neural network trained to approximate the policy functions has five hidden layers. We use 128 neurons in each layer and rely on CELU and PReLU as activation functions.

The particle filter to evaluate the likelihood function We then turn to train the neural network that approximates the likelihood function. This neural network provides a direct mapping from the parameter values to the likelihood. We draw from the parameter space and generate 10,000 likelihood values with the particle filter. The neural network is then trained using these likelihood values. Specifically, we use 75% of these values to train the neural network (training sample) and the remaining values to validate the neural network (validation sample).[35] The neural network is trained over 10,000 iterations with a batch size of 100.

Once the training and the validation steps of the neural network is completed, we are able to evaluate the likelihood very fast for a large number of parameter values. Therefore, we are now in the position to run the Random Walk Metropolis-Hastings (RWMH) algorithm. We set the number of draws to 1 million (after a burn-in) to obtain the posterior distribution. It takes us around three days with a GPU (Nvidia Tesla V100 32GB) to estimate the HANK model with our approach.

Results We estimate 13 parameters of the nonlinear HANK model with our neural networks approach. The results are summarized in the lower panel of Table 2. The key takeaway is that the posterior median is very close to the true value. In particular, the true value is mostly contained in the 90% credible interval for all parameters despite having such few time-series. Furthermore, the posterior of the nonlinear HANK model is shown in Appendix F.3.

4 Estimating a Nonlinear HANK Model

We now estimate the nonlinear HANK model, which was introduced in the third proof of concept and is described in detail in Appendix E, with actual US data. We follow exactly the same steps as in the third proof of concept. However, the estimation is run on real data; that is, the GDP growth rate, the rate of inflation (the growth rate of the GDP deflator), and the Federal Funds rate. The sample period goes from 1984:Q1 until 2019:Q4.

We do not include any cross-sectional variable (e.g. moments of the wealth distribution in the US). Instead, we are going to compare the Gini coefficient of the wealth distribution implied by the model with with that coefficient measure in the US data after estimation. This exercise would provide an external validation that our estimated HANK model accounts for the dispersion of wealth across US household.

While this external validation approach is fairly standard in empirical works, one may wonder whether any identification of the idiosyncratic income risk can be achieved. Since these shocks are the key source of heterogeneity of the model, this is an important question which by no means has an obvious answer. We find that the standard deviation of the idiosyncratic income shocks can be identified by the volatility of the observed GDP growth. Indeed, we show that the existence of an occasionally binding ZLB constraint implies that the standard deviation of the idiosyncratic income shocks contributes considerably to output volatility. Consequently, the time series of GDP growth is very informative regarding the standard deviation of these shocks.

[35]The neural network features 64 nodes and four hidden layers. We use SiLU (Sigmoid Linear Unit) as an activation function.

Importantly, this sample period includes the first zero-lower-bound episode for the US. It should be noted that the ZLB constrained is a key aggregate non-linearity in the estimated HANK model. As said, the interaction between this aggregate nonlinear constraint with the uninsurable income risks is a critical novel feature of the HANK model studied in this paper.

4.1 Estimation: Data, measurement equation and priors

We estimate the nonlinear HANK model with our neural-network estimation method. We base the analysis on the quarterly GDP growth, annualized GDP deflator inflation, and the Federal Funds rate from 1984:Q1 until 2019:Q4, e.g., similar to Aruoba et al. (2021).[36] The measurement equation is given as follows:

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 100\left(\frac{Y_t}{Y_{t-1}/g_t} - 1\right) \\ 400\,(\Pi_t - 1) \\ 400\,(R_t - 1) \end{bmatrix} + u_t, \tag{25}$$

where the measurement error follows a Gaussian distribution $u_t \sim \mathcal{N}(0, \Sigma_u)$. The variance of the measurement error for each time series is the fraction $m_E = 0.05$ of its variance.

Estimated Parameters and Priors We estimate the standard deviations of the structural shocks. Importantly, our approach allows us to include the standard deviation of the idiosyncratic shock, which is directly related to household heterogeneity, as well as the aggregate shocks. The prior densities are truncated normal distributions for the parameters.

The remaining parameters are calibrated. The aim is to choose them so that the model can fit the current low interest rate environment and capture the heterogeneity of households in line with the nonlinear models of Gust et al. (2017), Bianchi et al. (2021) and Aruoba et al. (2021) as well as the heterogeneous agent frameworks of Kaplan et al. (2018) and Fernández-Villaverde et al. (2021). While most parameters are standard, a few choices should be highlighted. The central bank targets 2.5% inflation rate since the model is estimated with the GDP deflator. We set the level of government debt to 0.25 as in Fernández-Villaverde et al. (2021). The borrowing limit $\underline{B}$ is set to allow a Gini coefficient of the wealth distribution that aligns with the average over the considered period, which stands at 0.83.[37] The persistence of the labor productivity ρ_s is set to 0.8, as in Fernández-Villaverde et al. (2021). The persistence of the preference shock ρ_ζ is set to 0.7; a value that allows the model to match the frequency of hitting the ZLB observed in the data.

Neural Networks Based Estimation The estimation of the HANK model follows closely the steps we took to estimate the same model in the third proof of concept. We train one

[36]GDP growth is based on real gross domestic product per capita using the civilian noninstitutional population.
[37]The data is taken from the World Inequality Database.

Calibration

Parameters		Value	Parameters		Value
β	Discount factor	0.99825	θ_Π	MP inflation response	2.6
σ	Relative risk aversion	1	θ_Y	MP output response	0.98
η	Inverse Frisch elasticity	1	D	Government debt	0.25
ϵ	Price elasticity demand	11	$\underline{B}$	Individual borrowing limit	-0.15
χ	Disutility labor	0.91	φ	Rotemberg pricing	100
g	Average growth rate	1.0039	ρ_s	Persistence labor productivity	0.8
γ^τ	Tax progressivity	0.18	ρ_ζ	Persistence preference shock	0.7
Π	Inflation target	1.00625			

Estimation

Par.	Prior					Neural Network		
	Type	Mean	Std	Lower Bound	Upper Bound	Posterior Median	5%	95%
σ_s	Trc.N	4.5%	1.0%	1.0%	5.0%	4.24%	3.80%	4.66%
σ_ζ	Trc.N	1.5%	10.0%	1.0%	2.2%	1.36%	1.26%	1.47%
σ_g	Trc.N	0.4%	0.1%	0.2%	0.6%	0.47%	0.44%	0.53%
σ_{mp}	Trc.N	0.13%	0.01%	0.05%	0.38%	0.19%	0.17%	0.20%

Table 3: The upper panel shows the calibration for the nonlinear HANK model with the ZLB and borrowing limit, which is used as data-generating process. The lower panel shows the prior and the posteriror. The prior type indicates the prior density function, where Trc.N stands for a truncated normal distribution.

neural network to approximate the extended solution mapping (i.e., the transition equations) with parameters treated as pseudo stae variables. We train a second neural network to solve for Bewey-Huggett-Aiyagari economy or DSS equilibrium. Third, we train the neural network that approximates the likelihood values obtained by the particle filter. The specification of the neural networks and convergence statistics are given in Appendix F.2.

We estimate four parameters of the nonlinear HANK model with our neural networks approach. The results are summarized in the lower panel of Table 3. This table displays the posterior median, as well as the 5% and 95% quantiles.

The Bewley-Huggett-Aiyagari model The Bewley-Huggett-Aiyagari model is the exact same model as our estimated HANK model under the assumption of no aggregate risk and no ZLB constraint. This economy is commonly used to obtain the nominal rate and the output

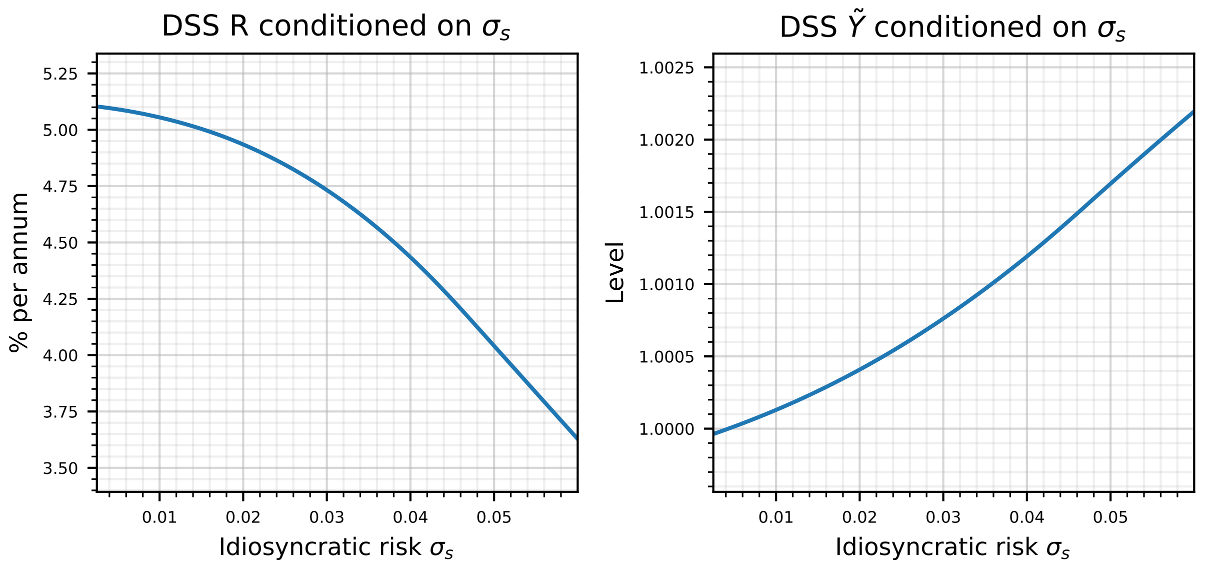

Figure 7: Deterministic steady-state (DSS) nominal interest rate and output. The figure shows the effects of the idiosyncratik risk σ_s on the DSS values of the nominal rate and output. The interest rate is expressed as net rate in %, while output is expressed in levels.

targets of the central bank.[38] It is worth noting that computing the solution to the Bewey-Huggett-Aiyagari economy is typically very time consuming using the existing methods. Once our neural network approximating the DSS is properly trained, it takes a fraction of second for us to obtain the relations shown in Figure 7.

It is instructive to analyze how the nominal interest rate and output vary in the Bewley-Huggett-Aiyagari model, which is basically the model studied at its deterministic steady state (DSS) equilibrium, with the size of the idiosyncratic income risk, σ_s. These relations are showed in Figure 7. The DSS interest rate goes down with the idiosyncratic risk as agents (especially lower income) desire to save more and work more. The supply of assets is fixed by the government so the DSS interest rate falls to ensure market clearing. In addition, agents work more and so output increases.

It is important to note that in the DSS or the Bewley-Huggett-Aiyagari model there is no aggregate uncertainty nor the ZLB constraint. So this economy is useful to construct counterfactuals, which help to shed light on the role of the ZLB constraint in the model. For instance, the relations shown in Figure 7 can be used to isolate the effects of idiosyncratic income risk on

[38]The RANK model would not provide an adequate benchmark for the central bank's targets because it would not take into account the effects of heterogeneity on the long-run nominal interest rate and output, which is basically what Figure 7 shows. Since the DSS does not feature the ZLB, the macroeconomic biases induced by the ZLB are still present in the nonlinear HANK model that we estimate.

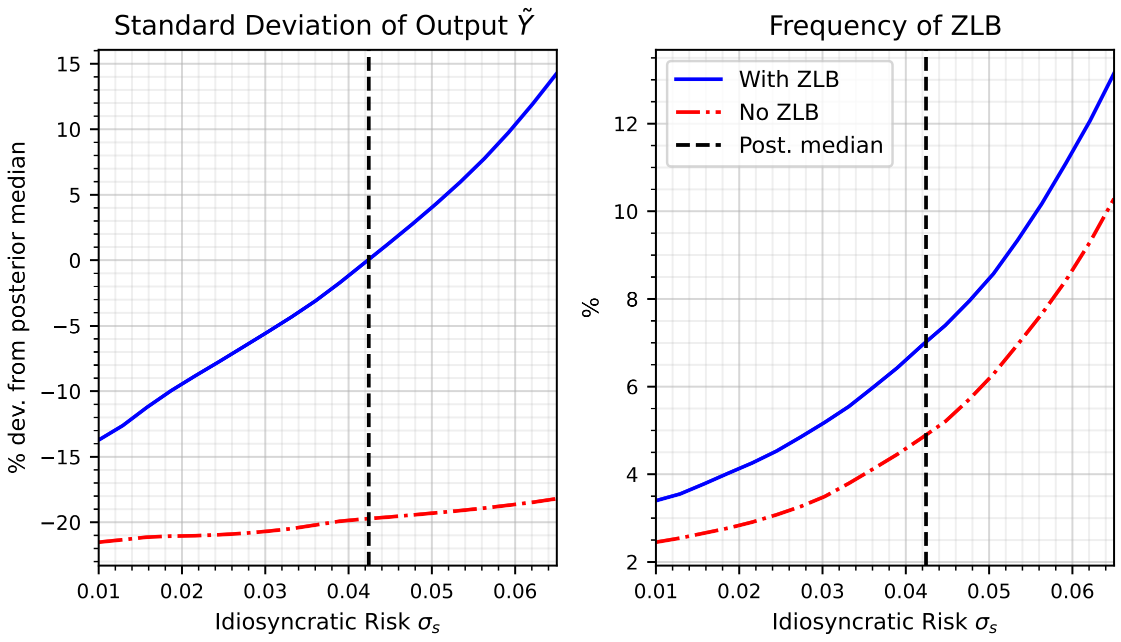

Figure 8: Volatility of output and ZLB frequency. *Left panel*: Standard deviation of output in percent deviations from its stochastic trend as a function of the idiosyncratic income risk, σ_s, in the estimated model (blue solid line) and in the case where the ZLB constraint is not enforced (red dashed-dotted line). The other parameters are fixed at their posterior median. *Right panel*: Frequency of ZLB episodes as a function of the idiosyncratic income risk, σ_s, in the estimated model (blue solid line) and in the case where the ZLB constraint is not enforced (red dashed-dotted line). The other parameters are fixed at their posterior median. In both panels, the vertical black dashed line marks the value of the posterior median of the standard deviation of the idiosyncratic income risk. We simulate the model for 100,000 periods and then calculate the standard deviation of output and the frequency of the ZLB with and without enforcing this constraint. When the ZLB constraint is not enforced (the red dashed-dotted line), the frequency of the ZLB is just the frequency at which the nominal interest rate becomes negative.

the the nominal interest rate and output in the absence of the ZLB constraint.

One advantage of preserving the nonlinear features of a HANK model is the ability to draw predictions on the changes in the long-run average inflation rate. We find that in the estimated model is around 2.2%, which is below the central bank's target of 2.5%. Thus, the model features a deflationary bias as in Bianchi et al. (2021). Similar to the phenomenon of aggregate volatility, this so-called deflationary bias is influenced by the degree of heterogeneity. The results highlight that variations in income risk influence the probability of hitting the zero lower bound, which, in turn, affects aggregate volatility. Our results should be interpreted as a conservative estimate of the interplay between idiosyncratic and aggregate risk, given the limited degree of heterogeneity in our model – e.g. agents only trade one asset. Models with more elaborated heterogeneity

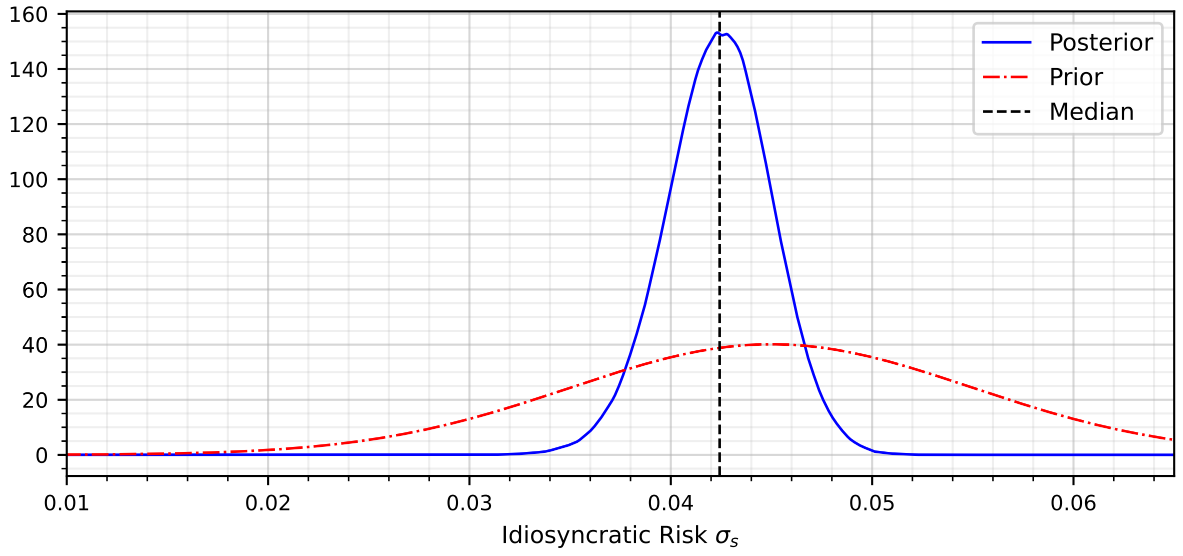

Figure 9: Posterior and prior distribution of the estimated idiosyncratic income risk. The blue line denotes the log-posterior density of the parameter controlling the level of idiosyncratic income risk, σ_s. The red dotted-dashed line denotes the prior distribution of the same parameter. The vertical black dashed line marks the posterior median.

settings could potentially reveal much more relevant interactions and introduce dynamics that cannot be captured by our simple HANK model.

Interplay between the idiosyncratic risk and aggregate nonlinearities. The zero lower bound constraint critically interacts with idiosyncratic income risk to raise the volatility of macroeconomic variables. We show the relation with real output, which is illustrated in Figure 8. In this chart, it is shown that an elevated idiosyncratic risk raises the volatility of output. This would not happen in the case in which the ZLB constraint is not enforced. An increase in the idiosyncratic risk lowers the level of the nominal interest rate, thereby influencing the frequency at which the ZLB is encountered (right panel). When the ZLB binds more frequently, it leads to more pronounced recessions and a drops in inflation, resulting in amplified macroeconomic volatility. If no idiosyncratic risk were present, the frequency of hitting the ZLB would fall to 2 percentage points (when parameters are at the posterior median) and the volatility of output and inflation would be more lower.

An important take-away is that the parameter, σ_s, controlling the level of idiosyncratic risk can be identified using the volatility of output and inflation. This insight is confirmed by looking at Figure 9, showing the shape of the posterior distribution (the solid blue line) around the posterior median for the idiosyncratic income risk, σ_s. The posterior is bell-shaped around the estimated value of the parameter. By comparing the posterior and the prior distribution (in red dahsed-dotted line), it is fairly clear that observing the three aggregate data leads to a relevant Bayesian updating.

How good is what we got? Very encouraging. In Table 4, we show the second moment of the observables in the estimated HANK model and in the data. We simulate the model for 100,000 periods and then calculate the volatility and serial correlation of GDP growth and inflation as well as the average Gini coefficient of the wealth distribution.

We observe that the standard deviation of GDP growth and inflation are quite close to the ones measured in the data. The serial correlation of output is a touch higher in the estimated model. Considering the the HANK model we have estimated is quite bare bone, we interpret these numbers as very encouraging. The table also shows that the Gini coefficient for the wealth distribution implied by the estimated model is also remarkably close to that measured in the data.

The match is unsatisfactory for what concerns the volatility and serial correlation of the nominal interest rates. More work has to be done to make sure that HANK models can explain these moments. One reason behind this shortcoming of the estimated model is certainly related to the fact that ZLB episodes tend to be much shorter lasting in the model than in the data. This is a well-known problem plaguing medium-scale linearized RANK models as well since it is hard to get a ZLB constraint binding for 8 years when the model is simulated. The same shortcoming applies to the HANK model. Adding interest rate inertia to the monetary policy reaction function would also help the model to explain the autocorrelation of the nominal interest rate in the data.

External validation: US wealth inequality. In Table 4, we show the average Gini index of the wealth distribution of the model, which we estimated without observing any cross-sectional data, and the same index measured by the World Inequality Database for the US.[39] The two Gini coefficients are remarkably close. The coefficient in the model is determined by the idiosyncratic income risk and the nonlinear endogenous response to changes in this risk. The coefficient is extremely close to the average Gini coefficient observed in the US data during the estimation period: 1984:Q1-2019:Q4.

Since we estimated our model using only aggregated time series data, this close match in the average Gini coefficient in the model and in the data can be regarded as an important external empirical validation of our estimation exercise. As shown earlier, the interactions between the model nonlinearities (primarily the ZLB constraint) and the presence of an uninsurable idiosyncratic income risk affect quite considerably aggregate output volatility, which is observed

[39]Bayer et al. (2020) use the same database to measure wealth inequality.

Standard deviations		
	Model	Data
GDP growth	0.6653	0.5735
Inflation	0.7330	0.9753
Federal funds rate	4.3803	2.9929

Autocorrelations		
	Model	Data
GDP growth	0.1287	0.3919
Inflation	0.7198	0.6362
Federal funds rate	0.6817	0.9751

Avg. Gini coefficient		
	Model	Data
Wealth distrib.	0.8932	0.8321

Table 4: Moments comparison: model vs. data. The upper part of this table shows the standard deviation, the autocorrelation coefficient in the model and in the data (Fred data set; Federal Reserve of St.Louis) for the three observable variables: GDP growth, GDP deflator growth (Inflation), and the federal funds rate. The bottom part of this table shows the average Gini coefficient in the model and in the data (World Inequality Database) for the wealth distribution – a proxy for social inequality. The model is simulated for 100,000 periods to assess the moments of these variables. The empirical moments are computed over the sample period used in estimation.

in estimation. See Figure 8 for a local quantification of the effects of these interactions on the volatility of GDP.

5 Conclusion

In this paper, we develop a novel estimation procedure using machine learning techniques. We exploit the advantages of neural networks to estimate complex models, which are probably out of reach otherwise. Our strategy rests on two key steps. First, we adapt the training of the neural network to treat the parameters, which are estimated, as pseudo-state variables. Second, we train a neural network as a surrogate model to approximate the likelihood in a computationally efficient manner.

Our method applies to a large class of economic models such as heterogeneous agents models, large representative agent models, sovereign default and endogenous bank run models, or multi-country (county) models. Our approach has three major advantages: i) it can account for many state variables, ii) it can capture nonlinear dynamics such as the zero lower bound or borrowing limits, iii) it does not impose any restrictions on the set of parameters to be estimated in a heterogeneous agent set up.

We apply our techniques to estimate a nonlinear HANK model using actual data. We show that standard aggregate data allow us to identify the degree of idiosyncratic risk. The reason is that the level of idiosyncratic risk impacts the interest rate level and affects the frequency of encountering the ZLB. If idiosyncratic risk were absent, the frequency of hitting the ZLB would be severely diminished. The estimated nonlinear HANK model surprisingly delivers an encouraging empirical fit. More work has to be done to improve the fit of the model but the first step is very promising. We highlighted some features of the model that are missing and could improve the empirical performance.

The proposed neural network-based estimation method opens new and exciting avenues for future research on the interaction between idiosyncratic and aggregate risk. For instance, the impact of aggregate nonlinearities on inequality can be evaluated with empirically estimated structural models.

References

Acharya, S., Dogra, K., 2020. Understanding hank: Insights from a prank. Econometrica 88, 1113–1158.

Ahn, S., Kaplan, G., Moll, B., Winberry, T., Wolf, C., 2018. When inequality matters for macro and macro matters for inequality. NBER Macroeconomics Annual 32, 1–75.

Aruoba, B., Cuba-Borda, P., Schorfheide, F., 2018. Macroeconomic dynamics near the zlb: A tale of two countries. The Review of Economic Studies 85, 87–118.

Aruoba, S.B., Cuba-Borda, P., Higa-Flores, K., Schorfheide, F., Villalvazo, S., 2021. Piecewise-linear approximations and filtering for dsge models with occasionally-binding constraints. Review of Economic Dynamics 41, 96–120.

Atkinson, T., Richter, A.W., Throckmorton, N.A., 2020. The zero lower bound and estimation accuracy. Journal of Monetary Economics 115, 249–264.

Auclert, A., Bardóczy, B., Rognlie, M., Straub, L., 2021. Using the sequence-space jacobian to solve and estimate heterogeneous-agent models. Econometrica 89, 2375–2408.

Azinovic, M., Gaegauf, L., Scheidegger, S., 2022. Deep equilibrium nets. International Economic Review .

Azinovic, M., Žemlička, J., 2023. Economics-Inspired Neural Networks with Stabilizing Homotopies. Technical Report.

Bach, F., 2017. Breaking the curse of dimensionality with convex neural networks. The Journal of Machine Learning Research 18, 629–681.

Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information theory 39, 930–945.

Bayer, C., Born, B., Luetticke, R., 2020. Shocks, Frictions, and Inequality in US Business Cycles. CEPR Discussion Papers 14364.

Bayer, C., Lütticke, R., Pham-Dao, L., Tjaden, V., 2019. Precautionary savings, illiquid assets, and the aggregate consequences of shocks to household income risk. Econometrica 87, 255–290.

Bianchi, F., Melosi, L., Rottner, M., 2021. Hitting the elusive inflation target. Journal of Monetary Economics 124, 107–122.

Bilbiie, F.O., 2020. The new keynesian cross. Journal of Monetary Economics 114, 90–108.

Bilbiie, F.O., Primiceri, G., Tambalotti, A., 2023. Inequality and Business Cycles. Working Paper 31729. National Bureau of Economic Research.

Boppart, T., Krusell, P., Mitman, K., 2018. Exploiting mit shocks in heterogeneous-agent economies: the impulse response as a numerical derivative. Journal of Economic Dynamics and Control 89, 68–92.

Challe, E., Matheron, J., Ragot, X., Rubio-Ramirez, J.F., 2017. Precautionary saving and aggregate demand. Quantitative Economics 8, 435–478.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems 2, 303–314.

Duarte, V., 2018. Gradient-Based Structural Estimation. mimeo.

Ebrahimi Kahou, M., Fernández-Villaverde, J., Perla, J., Sood, A., 2021. Exploiting symmetry in high-dimensional dynamic programming. Working Paper 28981. National Bureau of Economic Research.

Fernández-Villaverde, J., Hurtado, S., Nuno, G., 2023. Financial frictions and the wealth distribution. Econometrica 91, 869–901.

Fernández-Villaverde, J., Marbet, J., Nuño, G., Rachedi, O., 2021. Inequality and the Zero Lower Bound. mimeo.

Fernández-Villaverde, J., Nuño, G., Sorg-Langhans, G., Vogler, M., 2020. Solving High-Dimensional Dynamic Programming Problems using Deep Learning. mimeo.

Fernández-Villaverde, J., Rubio-Ramírez, J.F., 2007. Estimating Macroeconomic Models: A Likelihood Approach. The Review of Economic Studies 74, 1059–1087.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Gorodnichenko, Y., Maliar, L., Maliar, S., Naubert, C., 2021. Household Savings and Monetary Policy under Individual and Aggregate Stochastic Volatility. CEPR Discussion Papers 15614.

Gust, C., Herbst, E., López-Salido, D., Smith, M.E., 2017. The Empirical Implications of the Interest-Rate Lower Bound. American Economic Review 107, 1971–2006.

Han, J., Yang, Y., E, W., 2021. DeepHAM: A Global Solution Method for Heterogeneous Agent Models with Aggregate Shocks. Technical Report.

Herbst, E.P., Schorfheide, F., 2015. Bayesian estimation of DSGE models. Princeton University Press.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks 2, 359–366.

Kaplan, G., Moll, B., Violante, G.L., 2018. Monetary policy according to hank. American Economic Review 108, 697–743.

Lee, D., 2020. Quantitative Easing and Inequality. mimeo.

Lin, A., Peruffo, M., 2022. Aggregate Uncertainty, HANK, and the ZLB. Technical Report.

Maliar, L., Maliar, S., 2020. Deep Learning: Solving HANC and HANK Models in the Absence of Krusell-Smith Aggregation. mimeo.

Maliar, L., Maliar, S., Winant, P., 2021. Deep learning for solving dynamic economic models.

Journal of Monetary Economics .

McKay, A., Nakamura, E., Steinsson, J., 2016. The power of forward guidance revisited. American Economic Review 106, 3133–58.

Norets, A., 2012. Estimation of dynamic discrete choice models using artificial neural network approximations. Econometric Reviews 31, 84–106.

Oh, H., Reis, R., 2012. Targeted transfers and the fiscal response to the great recession. Journal of Monetary Economics 59, S50–S64.

Ottonello, P., Winberry, T., 2020. Financial heterogeneity and the investment channel of monetary policy. Econometrica 88, 2473–2502.

Reiter, M., 2009. Solving heterogeneous-agent models by projection and perturbation. Journal of Economic Dynamics and Control 33, 649–665.

Richter, A.W., Throckmorton, N.A., Walker, T.B., 2014. Accuracy, speed and robustness of policy function iteration. Computational Economics 44, 445–476.

Rottner, M., 2021. Financial Crises and Shadow Banks: A Quantitative Analysis. Economics Working Papers EUI ECO 2021/02. European University Institute.

Schaab, A., 2020. Micro and Macro Uncertainty. Technical Report.

Scheidegger, S., Bilionis, I., 2019. Machine learning for high-dimensional dynamic stochastic economies. Journal of Computational Science 33, 68–82.

Smets, F., Wouters, R., 2007. Shocks and frictions in us business cycles: A bayesian dsge approach. American Economic Review 97, 586–606.

Valaitis, V., Villa, A., 2021. A Machine Learning Projection Method for Macro-Finance Models. FRB of Chicago Working Paper.

Winberry, T., 2021. Lumpy investment, business cycles, and stimulus policy. American Economic Review 111, 364–96.

A Neural Network-Based Bayesian Estimation Algorithm

The following Random Walk Metropolis Hastings (RWMH) algorithm can be used to to estimate the parameters $\tilde{\Theta}$, where we use the neural network-based evaluation of the likelihood function. To integrate our neural network approach for Bayesian estimation, we adapt the RWMH algorithm of Atkinson et al. (2020).[40] The approach is as follows:

1. Initialize the algorithm: Obtain a first candidate density for the Metropolis Hastings algorithm, from which the parameters: $\tilde{\Theta}$ are drawn, as follows:
   - (a) Draw from the prior distribution a candidate vector $\tilde{\Theta}^{New}$.
   - (b) Evaluate the log likelihood of the candidate vector using the trained neural network for the particle filter $\log \mathcal{L}^{NN}_{\tilde{\Theta}}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}^{New}\right)$ and combine it with the prior to obtain the log posterior $\log g(\mathbb{Y}_{1:T}|\tilde{\Theta}^{New})$.
   - (c) Repeat these steps N^{init} times and collect all draws as $\tilde{\Theta}^{init}$.
   - (d) Approximate the covariance matrix with these draws:
     - i. Choose all draws where the likelihood is above the 90% quantile, which is denoted as $\ddot{\Theta} = \hat{\Theta} - \overline{\hat{\Theta}}$.
     - ii. Calculate the covariance matrix: $\Sigma = (\ddot{\Theta}'\ddot{\Theta})/(0.1N^{init})$.
     - iii. Define the mode, that is the draw associated with the highest likelihood, as $\breve{\Theta}$.
2. Run a Metropolis-Hastings algorithm as burn-in. Repeat the steps below N^{Burn} times:
   - (a) Draw a new parameter vector $\tilde{\Theta}^{New}$ from the candidate density to evaluate the log posterior. The candidate density is a multivariate normal distribution with mean vector $\breve{\Theta}$ and covariance matrix $c\Sigma$, where the parameter is set to achieve an acceptance ratio between 20 and 40%.
   - (b) Evaluate the log likelihood of the parameter vector using the trained neural network for the particle filter $\log \mathcal{L}^{NN}_{\tilde{\Theta}}\left(\mathbb{Y}_{1:T}|\tilde{\Theta}^{New}\right)$ and combine it with the prior to obtain the log posterior $\log g(\mathbb{Y}_{1:T}|\tilde{\Theta}^{New})$.
   - (c) Accept the draw if $\exp(\log g(\mathbb{Y}_{1:T}|\tilde{\Theta}^{New}) - \log g(\mathbb{Y}_{1:T}|\breve{\Theta}))$ is larger than the draw from a standard uniform distribution. If the draw is accepted, the mean of the candidate density is updated to $\breve{\Theta} = \tilde{\Theta}^{New}$.
3. Use these N^{Burn} draws to get an updated candidate density:
   - (a) Keep only the last 75% of draws, which are denoted as $\hat{\Theta}^b$.
   - (b) Calculate the deviations of each draw from the mean: $\ddot{\Theta}^b = \hat{\Theta}^b - \overline{\hat{\Theta}^b}$.

[40]See also Herbst and Schorfheide (2015) for more information on RWMH.

   (c) Calculate the covariance matrix: $\Sigma^b = (\ddot{\Theta}^{b\prime}\ddot{\Theta}^b)/(0.75N^{burn})$.

   (d) Define the mode, that is the draw associated with highest likelihood, as $\breve{\Theta}^b$.

4. Run the Metropolis-Hastings algorithm to obtain the posterior:

   (a) Use the proposal density defined in the previous step and repeat steps 2a - 2c N^{final} times.

   (b) N^{final} draws characterize the posterior distribution.

B Linearized DSGE Model

The model is a prototypical three-equation New Keynesian DSGE model with a TFP shock. The model is log-linearized around its unique steady-state equilibrium, which results in the following equations.

$$\hat{Y}_t = E_t\hat{Y}_{t+1} - \sigma^{-1}\left(\hat{R}_t - E_t\hat{\Pi}_{t+1}\right), \tag{26}$$

$$\hat{\Pi}_t = \kappa(\hat{Y}_t - \hat{Y}_t^*) + \beta E_t\hat{\Pi}_{t+1}, \tag{27}$$

$$\hat{R}_t = \phi_\Pi\hat{\Pi}_t + \phi_Y\hat{X}_t, \tag{28}$$

$$\hat{Y}_t^* = \omega\hat{A}_t, \tag{29}$$

$$\hat{A}_t = \rho_A\hat{A}_{t-1} + \sigma_A\epsilon_t^A, \tag{30}$$

where we define $\omega = (1+\eta)/(\eta+\sigma)$ as well as $\kappa = (1-\phi)(1-\phi\beta)(\sigma+\eta)/\phi$. Output is defined as $\hat{Y}_t = (Y_t - Y)/Y$, inflation as $\hat{\Pi}_t = \Pi_t - \Pi$, nominal rate as $\hat{R}_t = R_t - R$, output in the flex price economy as $\hat{Y}_t^* = (Y_t^* - Y^*)/Y^*$, TFP as $(A_t - Y)/A$, and the output gap as $\hat{X}_t = \hat{Y}_t - \hat{Y}_t^*$. The shock ϵ_t^A follows a standard normal distribution.

After standard manipulations, the model equations can be written as:

$$\hat{X}_t = E_t\hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi\hat{\Pi}_t + \phi_Y\hat{X}_t - E_t\hat{\Pi}_{t+1} - \hat{R}_t^*\right), \tag{31}$$

$$\hat{\Pi}_t = \kappa\hat{X}_t + \beta E_t\hat{\Pi}_{t+1}, \tag{32}$$

$$\hat{R}_t^* = \rho_A\hat{R}_{t-1}^* + \sigma(\rho_A - 1)\omega\sigma_A\epsilon_t^A, \tag{33}$$

where $\hat{R}_t^*$ is the natural rate of interest, which follows an exogenous process that is derived from the TFP process.

We train the neural network by minimizing the residual error in equations (31) and (32), while the law of motion of the exogenous state variable is described in equation (33).

The analytical solution for the output gap $\hat{X}_t$ and inflation $\hat{\Pi}_t$, which can be derived with the method of undetermined coefficients, is given as:

$$\hat{X}_t = \frac{1-\beta\rho_A}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\hat{R}_t^* \tag{34}$$

$$\hat{\Pi}_t = \frac{\kappa}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\hat{R}_t^* \tag{35}$$

The analytical solution underscores the idea of extended policy functions, which are simultaneously conditioned on the parameters and state variables. The analytical solution of the output gap and inflation depends on the (exogenous) state variable $\hat{R}_t^F$. At the same time, the output gap and inflation also depend on the parameters.

B.1 Mapping of the Model to the General Solution and Estimation Framework

We can map the linearized NK model in the general form of the outlined estimation procedure. The state variable and structural shock are:

$$\mathbb{S}_t = \left\{\hat{R}_t^f\right\}, \qquad \text{and} \qquad \nu_t = \left\{\epsilon_t^A\right\}. \tag{36}$$

The control variables of the model are:

$$\psi_t = \left\{\hat{X}_t, \hat{\Pi}\right\}. \tag{37}$$

The parameters of the model are divided in calibrated ($\bar{\theta}$) and estimated ones ($\tilde{\theta}$):

$$\bar{\Theta} = \{\}, \tag{38}$$

$$\tilde{\Theta} = \{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A\}. \tag{39}$$

where we choose to vary all parameters so that the set for the calibrated parameters is empty. The neural network ψ_{NN} is trained to determine the output gap and inflation:

$$\begin{pmatrix}\hat{X}_t \\ \hat{\Pi}_t\end{pmatrix} = \psi_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right). \tag{40}$$

B.2 Neural Network for Inflation

Figure 10 shows the extended policy function for inflation that is conditioned on the parameters.

B.3 Neural Network for Likelihood Approximation

Figure 11 shows the likelihood for varying the different parameters stemming from the neural network as blue line. The orange dots show the mapping between the parameter and the likelihood directly coming from the particle filter.

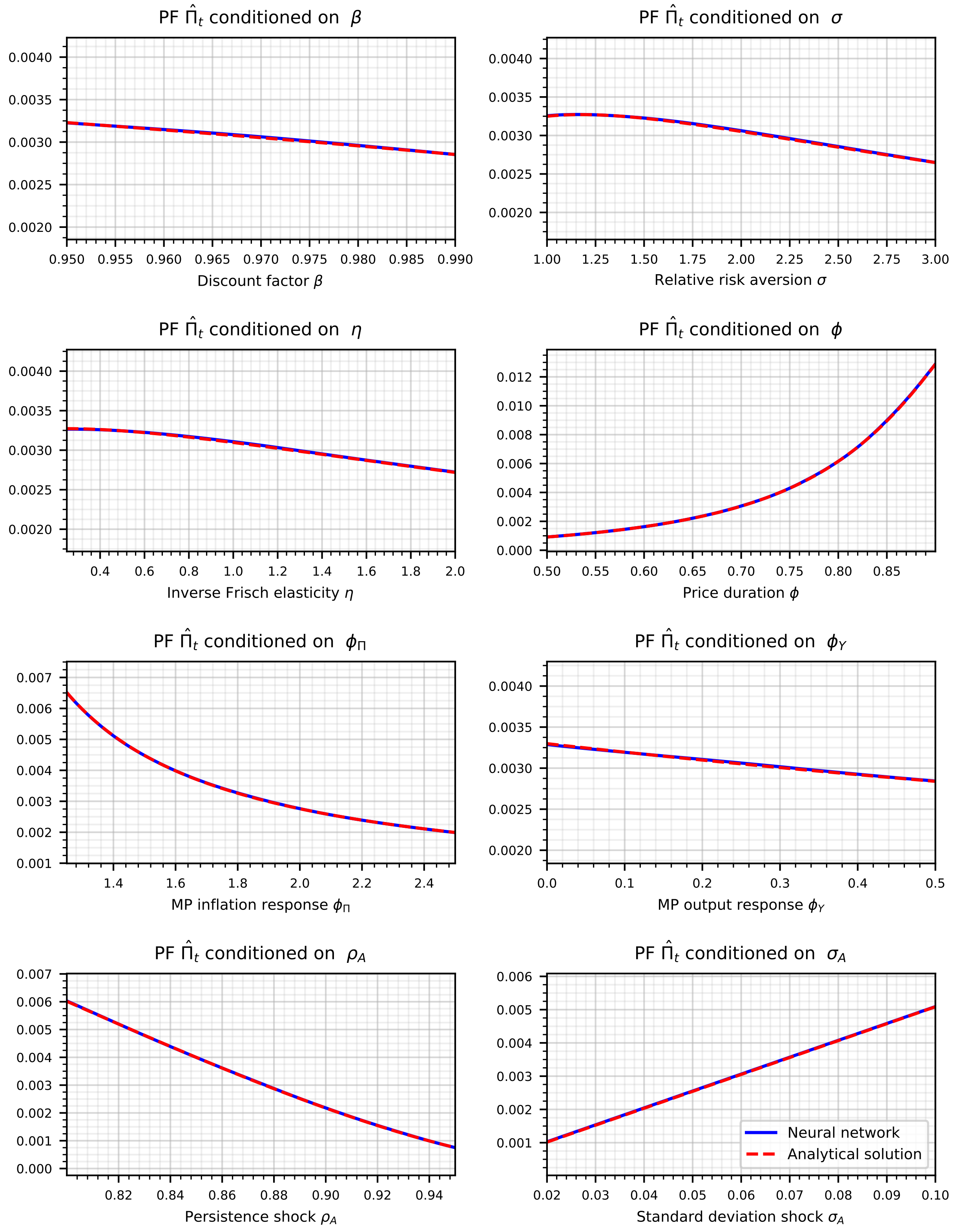

Figure 10: Comparison between the neural network and the analytical solution. The plot shows how variations in the structural parameters affect the extended policy function for the output gap $\hat{X}_t$. The policy function is evaluated at a negative one standard deviation shock. The unvaried parameters are fixed at their mean.

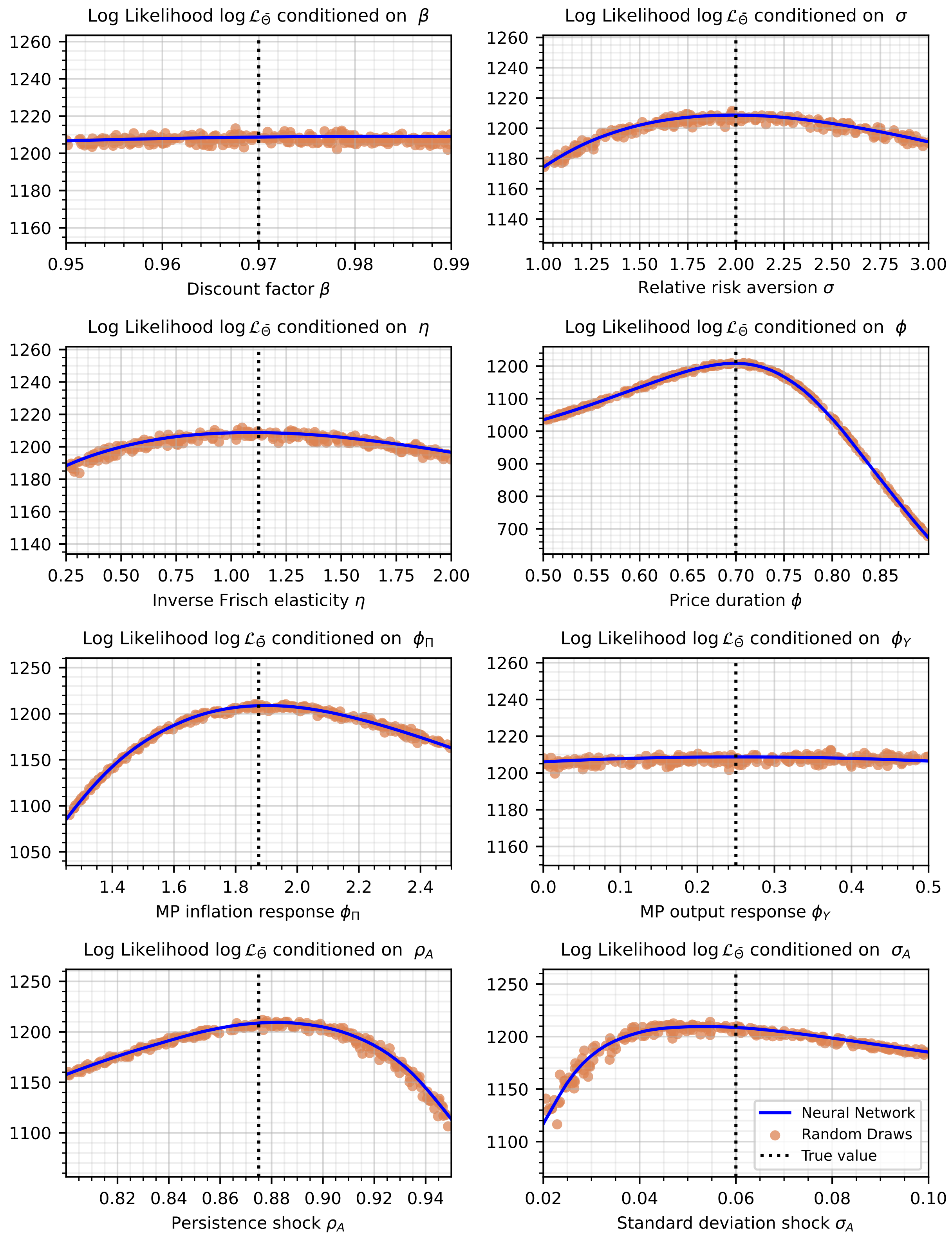

Figure 11: Comparison between the neural network and single draws from the particle filter. The plot shows how variations in the structural parameters affect the log likelihood $\mathbb{L}_{\bar{\Theta}}$. The unvaried parameters are fixed at their mean.

C Proof of concept 2: Comparison to a state-of-the-art estimation method for nonlinear models

We now assess whether our neural network estimation strategy can recover the true data-generating process of a nonlinear model. Specifically, we employ a Bayesian estimation for a nonlinear RANK model augmented with a ZLB. Additionally, we compare the results with a state-of-the-art estimation method of nonlinear macroeconomic models. This method relies on solving the model with global methods and evaluating the likelihood with a particle filter for every single draw of the Metropolis-Hastings algorithm (Herbst and Schorfheide, 2015).

We use a small-scale RANK model with the ZLB as a nonlinear element, for which we delegate the details to Appendix C.1. The calibrated model is used as a data-generating process, providing a controlled environment for the estimation. The upper panel of Table 5 summarizes the calibration of the model, with parameters chosen such that the model occasionally encounters the ZLB.

We estimate the nonlinear model in a Bayesian setup by employing our developed approach. We then compare the results to an estimation with a state-of-the-art (non-neural network) approach. The observables are the quarterly output growth rate, annualized quarterly inflation rate, and the nominal interest rate, which are generated with the calibrated model and cover a span of 500 periods. We also include a measurement error (for each time series 10% of its own variance) to avoid a degeneracy of the particle filter. The prior distributions are truncated normal densities.[41] The prior mean corresponds to the true value, while the standard deviation σ is loose to avoid the results being driven by the prior. The truncation ensures that the drawn parameters lie inside the bounds that have been imposed while solving the neural network. The lower panel of Table 5 summarizes the priors.

Neural Networks Based Estimation Approach The computationally most challenging step in the estimation is the first one, where we solve for the policy functions. The neural network approximates the labor and consumption policy and is conditioned on the state variables and parameters to be estimated. The network is trained by minimizing the Euler equation errors, where we solve the system of equations and then minimize the Euler equation as well as the New Keynesian Phillips Curve. We use 100,000 iterations to train the neural network.[42] After each iteration, the economy is simulated for 20 periods. The batch size is set to 100. We generate 15,000 likelihoods with the particle filter to train the neural network-based particle We then easily obtain the posterior with a Random Walk Metropolis-Hastings algorithm, which uses

[41]The probability density function of the truncated normal is $f(x;\mu,\sigma,a,b,) = \frac{1}{\sigma}\frac{\phi((x-\mu)/\sigma)}{\Phi((b-\mu)/\sigma)-\Phi((a-\mu)/\sigma)}$. If the bounds are not symmetric, the parameter μ does not correspond to the mean of the truncated normal. For simplicity, we refer to μ as the mean independent of the bounds.

[42]The neural network contains five hidden layers with 128 neurons each. The activation function for the hidden layers is parametric Rectified Linear Unit (PReLU).

Calibration for the data-generating process					
Parameters		Value	Parameters		Value
β	Discount factor	0.9975	θ_Π	Mon. pol. inflation response	2
σ	Relative risk aversion	1	θ_Y	Mon. pol. output response	0.25
η	Inverse Frisch elasticity	1	$4\log(\Pi)$	Inflation target (annualized)	2
ϵ	Price elasticity demand	11	Y	Output target	1
χ	Disutility labor	0.91	ρ_ζ	Persistence preference shock	0.7
φ	Rotemberg pricing	1000	$100\sigma^\zeta$	Std. dev. preference shock	2

Estimation											
Par.	Prior					Neural Network			Alternative Approach		
						Posterior			Posterior		
	Type	Mean	Std	Lower Bound	Upper Bound	Median	5%	95%	Median	5%	95%
θ_Π	Trc.N	2.0	0.1	1.5	2.5	2.11	1.92	2.24	2.06	1.93	2.20
θ_Y	Trc.N	0.25	0.05	0.05	0.5	0.248	0.236	0.259	0.248	0.237	0.260
φ	Trc.N	1000	50	700	1300	985	925	1048	970	909	1033
ρ_ζ	Trc.N	0.7	0.05	0.5	0.9	0.691	0.672	0.709	0.688	0.670	0.707
σ^ζ	Trc.N	0.02	$2.5e-3$	0.01	0.025	0.020	0.019	0.021	0.020	0.019	0.021

Table 5: The upper panel shows the calibration for the nonlinear RANK model with the ZLB, which is used as data-generating process. The lower panel shows the prior and compares the posterior for the neural network-based estimation with an alternative (non-neural network) approach. The prior type indicates the prior density function, where Trc.N stands for a truncated normal distribution.

50,000 draws after a burn-in.

We compare the results to a state-of-the-art nonlinear estimation approach that does not use machine learning techniques. The global solution method is based on time iteration with piecewise linear policy functions as in Richter et al. (2014) and the particle filter follows Herbst and Schorfheide (2015). The alternative approach also uses the nonlinear model as the data-generating-process.[43] However, this approach is much slower as we need to resolve the model and run the particle filter for each draw. These limitations rationalize our restriction to a relatively low amount of only 50,000 draws in the Metropolis-Hastings algorithm. Appendix C.2 shows how this model can be expressed in the general form that is outlined in Section 2.

Results The estimation results are summarized in Table 5. First, the neural network approach performs very well in recovering the true data-generating process. The posterior median is close to the true value and is always contained inside the 90% confidence interval. Furthermore, the

[43]We use the same sequence of structural shocks and measurement error shocks to generate the data. For the neural network, we feed the shocks in the model at the true parameters solved with neural networks. For the alternative non-neural network method, we feed the shocks in the model at the true parameters solved with standard global methods.

posterior median results for the alternative method are very similar. In addition to this, the ranges of the 90% confidence interval are also close.

C.1 Nonlinear RANK model with a zero lower bound

The model is a small nonlinear RANK model with a zero lower bound.

Households The economy consists of a representative household. The household chooses consumption C_t, labor H_t and assets B_t to maximize their utility:

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t) \left[\left(\frac{C_t}{1-\sigma} \right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta} \right) (H_t)^{1+\eta} \right],$$

where ζ_t is an aggregate preference shock, which follows an AR(1) process $\zeta_t = \rho_\zeta \zeta_{t-1} + \epsilon_t^\zeta$. C_t is aggregate consumption. The budget constraint in real terms can be written as:

$$C_t + B_t = W_t H_t + \frac{R_{t-1}}{\Pi_t} B_{t-1} - T_t + Div_t, \tag{41}$$

where Div_t is the real dividend, W_t is real wage, R_t is the gross nominal interest rate, Π_t is the gross inflation rate and T_t is real lump sum taxes. The first-order conditions are as follows:

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{\exp(\zeta_{t+1})}{\exp(\zeta_t)} \right) \left(\frac{C_t}{C_{t+1}} \right)^\sigma \frac{1}{\Pi_{t+1}} \right], \tag{42}$$

$$\chi (H_t)^\eta = (C_t)^{-\sigma} W_t. \tag{43}$$

Firms The firm sector consists of a continuum of final goods producers and intermediate goods firms. The final goods retailers buy the intermediate goods and transform them into a homogeneous final good using a CES production technology:

$$Y_t = \left(\int_0^1 (Y_t^j)^{\frac{\epsilon-1}{\epsilon}} dj \right)^{\frac{\epsilon}{\epsilon-1}}, \tag{44}$$

where Y_t^j is the output of intermediate goods firm j. The equilibrium price of the final good and the demand for the intermediate goods of firm j can be expressed as:

$$P_t = \left(\int_0^1 (P_t^j)^{1-\epsilon} dj) \right)^{\frac{1}{1-\epsilon}}, \quad \text{and} \quad Y_t^j = \left(\frac{P_t^j}{P_t} \right)^{-\epsilon} Y_t. \tag{45}$$

Intermediate goods producers are monopolistically competitive. The firm j uses labor N_t^j as input to produce output Y_t^j with the following production technology:

$$Y_t^j = Z N_t^j, \tag{46}$$

where Z is the total factor productivity. Labor is hired in competitive markets so that the wage is given as follows

$$W_t = ZMC_t. \tag{47}$$

The firm j sets the price of its goods to maximize its profit subject to the demand curve for intermediate goods and Rotemberg adjustment costs for changing prices:

$$\max_{P_t^j} P_t^j \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} \frac{Y_t}{P_t} - MC_t \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} Y_t - \frac{\varphi}{2}\left(\frac{P_t^j}{\Pi P_{t-1}^j} - 1\right)^2 Y_t, \tag{48}$$

where Π is the inflation target of the central bank. Imposing a symmetric equilibrium and discounting future profits with the real interest rate, the New Keynesian Phillips curve can be written as:

$$\left[\varphi\left(\frac{\Pi_t}{\Pi} - 1\right)\frac{\Pi_t}{\Pi}\right] = (1-\epsilon) + \epsilon MC_t + \varphi \mathbb{E}_t \frac{\Pi_{t+1}}{R_t}\left[\left(\frac{\Pi_{t+1}}{\Pi} - 1\right)\frac{\Pi_{t+1}}{\Pi}\frac{Y_{t+1}}{Y_t}\right], \tag{49}$$

where $\Pi_t = P_t/P_{t-1}$. The Rotemberg adjustment costs are given back as a lump sum. The real dividends of the firm sector are $Div_t = Y_t - W_t N_t$.

Policy makers The central bank sets the nominal interest R_t using a Taylor rule that responds to inflation and output deviations from their targets Π and Y. The rule is persistent as the interest rate response is smoothed with the previous period's interest rate. The zero lower bound restricts the nominal interest rate. The rule is given as follows:

$$R_t = \max\left[1, R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{Y_t}{Y}\right)^{\theta_Y}\right]. \tag{50}$$

The fiscal authority follows a passive policy rule, where it uses lump-sum taxes T_t to keep their debts D on a constant path:

$$D = \frac{R_{t-1}}{\Pi_t} D - T_t. \tag{51}$$

Measurement Equation We base the analysis on the quartlery output growth rate, annualized quarterly inflation rate and nominal interest rate. The sample is generated with the calibrated model and covers a span of 500 periods. The measurement equation is given as:

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 100\left(\frac{Y_t}{Y_{t-1}} - 1\right) \\ 400\left(\Pi_t - 1\right) \\ 400\left(R_t - 1\right) \end{bmatrix} + u_t, \tag{52}$$

where the measurement error follows a Gaussian distribution $u_t \sim \mathcal{N}(0, \Sigma_u)$. The variance of the measurement error for each time series is a fraction m_E of its own variance. We set $m_E = 0.1$.

C.2 Mapping of the Model to the General Framework

We can map the RANK model in the general form of the outlined estimation procedure. The state variable and structural shock are:

$$\mathbb{S}_t = \{\zeta_t\}, \qquad \text{and} \qquad \nu_t = \left\{\epsilon_t^{\zeta}\right\}. \tag{53}$$

The control variables of the model are:

$$\psi_t = \{C_t, H_t, T_t, Y_t, Div_t, MC_t\}. \tag{54}$$

The parameters of the model are divided in calibrated ($\bar{\theta}$) and estimated ones ($\tilde{\theta}$):

$$\bar{\Theta} = \{\beta, \sigma, \eta, \epsilon, \chi, \Pi, Y\}, \tag{55}$$

$$\tilde{\Theta} = \{\theta_\Pi, \theta_Y, \varphi, \rho_\zeta, \sigma_\zeta\}. \tag{56}$$

The neural network is trained to determine wage and inflation, which is sufficient to determine the other variables:

$$\begin{pmatrix} \Pi_t \\ W_t \end{pmatrix} = \psi_{NN}\left(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta}\right). \tag{57}$$

D Residual Error Neural Network

It is well known that economic models may only have a solution for some parameter combinations. This general problem for global solution methods also affects our method. To evaluate the solution over the parameter space, we suggest to analyse the residual error in the equations that the neural networks minimize. Importantly, the neural network may not be able to find a solution because there does actually not exist an equilibrium. In such a case, the neural network may find some (incorrect) solution, but the residual error is larger than in other correctly solved parts. It should be noted that global solution method often encounter numerical issues when they cannot find a solution. This makes it easy to spot a problem. Our experience is that the neural network is much less likely to encounter numerical problems, which result in a breakdown of the algorithm. Instead the neural network provides a solution with a large residual error.

Our strategy will be to evaluate the average residual error after we have solved for the neural network. The residual error is the weighted mean residual error when simulating the model for a sufficient large amount of time.[44] The residual error depends directly on the parameter:

[44]To obtain the mean residual error in practice, we simulate the model for a number of periods and calculate

$R(\tilde{\Theta})$. In particular, we are interested in finding a function that directly maps the parameter combination to the residual function:

$$R(\tilde{\Theta}) = \Omega^{RE}(\tilde{\Theta}) \tag{58}$$

where Ω^{RE} is an unknown function.

The usual approach to evaluate this function would be to evaluate the residual error at each single draw. However, this is time wise a very costly approach and, therefore, not suited for large models such as nonlinear HANK. To overcome this bottleneck, we propose to train a neural network model that provides the outcome of the residual error. Specifically, we train a neural network that provides the output of the function:

$$R(\tilde{\Theta}) = \Omega^{RE}_{NN}(\tilde{\Theta}) \tag{59}$$

where Ω^{RE}_{NN} is the neural network associated with the residual error. This type of neural network is also denoted as surrogate neural network as it allows to calculate the outcome in an efficient manner.

To train this separate neural network, we create a set of parameter values and corresponding mean residual errors.[45] The sample is divided in a training and validation sample. We train the neural network with the training sample and avoid overfitting with the validation sample. After we have trained $\Omega^{RE}_{NN}(\tilde{\Theta})$, the residual error of the model can be evaluated at a specific draw for negligible costs. While we calculate the residual errors at only several thousand parameter points, we use the neural network to learn the connection between these points. This allows us then to evaluate the likelihood at points that we did not assess initially. As a consequence, we can speed up the algorithm considerably. Even though, we have focused on residual errors here, other measures to validate the precision of the solution can be applied in a similar way.

Residual error The surrogate neural network model that contains the residual error is shown in Figure12. While the neural network is trained for all parameters, the figure shows the change in the residual error for the standard deviation of the shock.

E HANK Model

The model is a nonlinear HANK model that captures idiosyncratic and aggregate risk. The first essential ingredient is heterogeneity. The households face idiosyncratic income risk and a

in each period the residual error, where the expectations are approximated with a large amount of draws for the Monte Carlo integeration of expectations, and average then over the periods. We weight the different residual errors of the considered equations along with the weights in the loss function.

[45]The neural network of the model is used in the simulation to obtain the residual error.

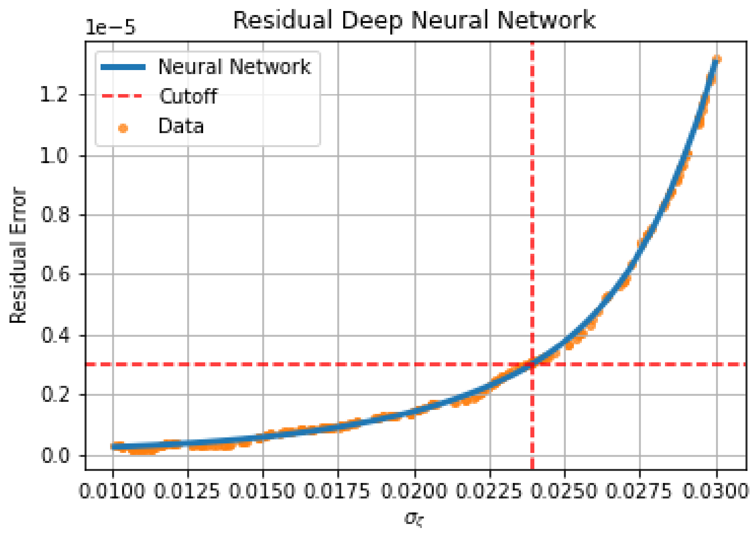

Figure 12: Surrogate neural network model that contains the residual error. It shows the trained neural network and contrasts it to the data that are used for training. The red line corresponds to a cut-off value at a residual error at .25e-5.

borrowing limit. The second key ingredient is that the zero lower bound constrains monetary policy. The model features demand, supply, and monetary policy shocks.

Households The economy consists of a continuum of households. The households choose consumption C_t^i, labor H_t^i, and assets B_t^i to maximize their utility:

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t) \left[\left(\frac{1}{1-\sigma}\right) \left(\frac{C_t}{Z_t}\right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta}\right) (H_t^i)^{1+\eta} \right],$$

where ζ_t is an aggregate preference shock, which follows an AR(1) process $\zeta_t = \rho_\zeta \zeta_{t-1} + \epsilon_t^\zeta$. C_t is aggregate consumption and Z_t is the total factor productivity. The budget constraint in real terms can be written as:

$$C_t^i + B_t^i = \tau_t \left(\frac{W_t}{Z_t} \exp(s_t^i) H_t^i\right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i + Div_t \exp(s_t^i), \tag{60}$$

where Div_t^i is the real dividend, W_t is real wage, R_t is the gross nominal interest rate, and Π_t is the gross inflation rate. The taxation is progressive, which is governed by γ_τ and τ_t determines the level of the post-tax income. The wage is divided by the productivity level to avoid a fiscal drag in taxation. The agents individual labor productivity s_t^i is stochastic and follows an AR(1) process in logs $s_t^i = \rho_s{}_{t-1}^i + \epsilon_t^{s,i}$. The dividend payout is scaled with the level of the idiosyncratic shock. The agents face a borrowing limit $\underline{B}$, which implies:

$$B_t \geq \underline{B}. \tag{61}$$

The first-order conditions can be written as

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{exp(\zeta_{t+1})}{exp(\zeta_t)} \right) \left(\frac{\lambda^i_{t+1}}{\lambda^i_t} \right) \frac{1}{\Pi_{t+1}} \right] + \mu^i_t, \tag{62}$$

$$\lambda^i_t = \left(\frac{C^i_t}{Z_t} \right)^{-\sigma} \frac{1}{Z_t}, \tag{63}$$

$$\chi (H^i_t)^\eta = (\lambda^i_t) \left[\tau_t (1-\gamma_\tau) \exp(s^i_t) \frac{W_t}{Z_t} \left(\exp(s^i_t) \frac{W_t}{Z_t} H^i_t \right)^{-\gamma_\tau} \right], \tag{64}$$

where $\mu^i_t \geq 0$ is the normalized multiplier on the individual borrowing limit in equation (61).

Firms The firm sector consists of a continuum of final goods producers and intermediate goods firms.

The final goods retailers buy the intermediate goods and transform them into the homogeneous final good using a CES production technology:

$$Y_t = \left(\int_0^1 (Y^j_t)^{\frac{\epsilon-1}{\epsilon}} dj \right)^{\frac{\epsilon}{\epsilon-1}}, \tag{65}$$

where Y^j_t is the output of intermediate goods firm j. The equilibrium price of the final good and the demand for the intermediate goods of firm j can be expressed as:

$$P_t = \left(\int_0^1 (P^j_t)^{1-\epsilon} dj \right)^{\frac{1}{1-\epsilon}}, Y^j_t = \left(\frac{P^j_t}{P_t} \right)^{-\epsilon} Y_t. \tag{66}$$

Intermediate goods producers are monopolistically competitive. The firm j uses labor N^j_t as input to produce output Y^j_t with the following production technology:

$$Y^j_t = Z_t N^j_t. \tag{67}$$

Total factor productivity Z_t follows a stochastic trend

$$Z_t = g_t Z_{t-1}, \tag{68}$$

where the trend growth rate is subject to an AR(1) process

$$g_t = \bar{g} exp(z_t), \tag{69}$$

$$z_t = \rho_z z_{t-1} + \epsilon^g_t. \tag{70}$$

Labor is hired in competitive markets so that the wage is given as follows

$$W_t = Z_t MC_t. \tag{71}$$

The firm j sets the price of its goods to maximize its profit subject to the demand curve for intermediate goods and Rotemberg adjustment costs for changing prices:

$$\max_{P_t^j} P_t^j \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} \frac{Y_t}{P_t} - MC_t \left(\frac{P_t^j}{P_t}\right)^{-\epsilon} Y_t - \frac{\varphi}{2}\left(\frac{P_t^j}{\Pi P_{t-1}^j} - 1\right)^2 Y_t, \tag{72}$$

where Π is the inflation target of the central bank. Imposing a symmetric equilibrium and discounting future profits with the real interest rate, the New Keynesian Phillips curve can be written as:

$$\left[\varphi\left(\frac{\Pi_t}{\Pi} - 1\right)\frac{\Pi_t}{\Pi}\right] = (1-\epsilon)+\epsilon MC_t+\beta\varphi\mathbb{E}_t\left(\frac{\exp(\zeta_{t+1})}{\exp(\zeta_t)}\right)\left(\frac{\lambda_{t+1}}{\lambda_t}\right)\left[\left(\frac{\Pi_{t+1}}{\Pi} - 1\right)\frac{\Pi_{t+1}}{\Pi}\frac{Y_{t+1}}{Y_t}\right], \tag{73}$$

where $\Pi_t = P_t/P_{t-1}$ and $\lambda_t = (C_t/Z_t)^{-\sigma} Z_t{}^{-1}$. The Rotemberg adjustment costs are ex-post given back. The real dividends of the firm sector can then be written as

$$Div_t = Y_t - W_t N_t. \tag{74}$$

Policy makers The central bank sets the nominal interest R_t using a Taylor rule that responds to inflation and output deviations from their targets Π and Y. The rule is persistent because the interest rate response is smoothed with the previous period's interest rate. In addition, there are i.i.d. monetary policy shocks ϵ_t^{mp}. The zero lower bound restricts the nominal interest rate. The rule is given as follows:

$$R_t^N = \left(R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{Y_t}{Z_t Y}\right)^{\theta_Y}\right)\exp(\epsilon_t^{mp}), \tag{75}$$

$$R_t = \max\left[1, R_t^N\right], \tag{76}$$

where R and Y denote the deterministic steady state (DSS) values of the nominal interest rate and output (in the detrended economy). While households face idiosyncratic shocks in the deterministic steady state, the economy does not face (or expects) aggregate shocks.

The fiscal authority sets τ_t to keep their debts D_t on a constant path adjusted for productivity gains:

$$D_t = \frac{R_{t-1}}{\Pi_t} D_{t-1} - T_t, \tag{77}$$

$$T_t = \int \left(W_t \exp(s_t^i) H_t^i\right) di - \int \tau_t \left(\frac{W_t}{Z_t}\exp(s_t^i)H_t^i\right)^{1-\gamma_\tau} di. \tag{78}$$

Market Clearing Market clearing for the labor market, bond market, and goods market requires

$$N_t = \int N_t^j dj = \int \exp(s_t^i) H_t^i di, \tag{79}$$

$$D_t = \int B_t^i di, \tag{80}$$

$$Y_t = \int C_t^i di. \tag{81}$$

E.1 Equilibrium Conditions

To have stationarity, we need to define the variables as follows $\tilde{X}_t = \frac{X_t}{Z_t}$. The relevant conditions can then be written as:

$$\tilde{C}_t^i + \tilde{B}_t^i = \tilde{\tau}_t \left(\tilde{W}_t \exp(s_t^i) H_t^i\right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t g_t} \tilde{B}_{t-1}^i + \tilde{Div}_t^i, \tag{82}$$

$$1 = \beta R_t \mathbb{E}_t \left[\left(\frac{exp(\zeta_{t+1})}{exp(\zeta_t)} \right) \left(\frac{\tilde{C}_t^i}{\tilde{C}_{t+1}^i} \right)^{\sigma} \frac{1}{\Pi_{t+1} g_{t+1}} \right] + \mu_t^i, \tag{83}$$

$$\chi (H_t^i)^\eta = \left(\tilde{C}_t^i\right)^{-\sigma} \left[\tilde{\tau}_t (1-\gamma_\tau) \exp(s_t^i) \tilde{W}_t \left(\exp(s_t^i) \tilde{W}_t H_t^i \right)^{-\gamma_\tau} \right], \tag{84}$$

$$\tilde{Y}_t^j = N_t^j, \tag{85}$$

$$\tilde{W}_t = MC_t, \tag{86}$$

$$\tilde{Div}_t = \tilde{Y}_t - \tilde{W}_t \tilde{Y}_t, \tag{87}$$

$$\left[\varphi \left(\frac{\Pi_t}{\Pi} - 1 \right) \frac{\Pi_t}{\Pi} \right] = (1-\epsilon) + \epsilon MC_t + \beta \varphi \mathbb{E}_t \left[\left(\frac{\exp(\zeta_{t+1})}{\exp(\zeta_t)} \right) \left(\frac{\tilde{C}_{t+1}}{\tilde{C}_t} \right)^{-\sigma} \left(\frac{\Pi_{t+1}}{\Pi} - 1 \right) \frac{\Pi_{t+1}}{\Pi} \frac{\tilde{Y}_{t+1}}{\tilde{Y}_t} \right], \tag{88}$$

$$R_t^N = \left(R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_\Pi} \left(\frac{\tilde{Y}_t}{Y} \right)^{\theta_Y} \right) \exp(\epsilon_t^{mp}), \tag{89}$$

$$R_t = \max\left[1, R_t^N\right], \tag{90}$$

$$\tilde{D}_t = \frac{R_{t-1}}{\Pi_t g_t} \tilde{D}_{t-1} - \tilde{T}_t, \tag{91}$$

$$\tilde{T}_t = \int \left(\tilde{W}_t \exp(s_t^i) H_t^i \right) di - \int \tilde{\tau}_t \left(\tilde{W}_t \exp(s_t^i) H_t^i \right)^{1-\gamma_\tau} di. \tag{92}$$

F Neural Network-Based Solution Algorithm for HANK

The algorithm uses the outlined neural network approach to solve the HANK model over the parameter space.

The state variables and shocks of the HANK model are given as:

$$\mathbb{S}_t = \left\{ \left\{ \tilde{B}^i_{t-1} \right\}_{i=1}^{L}, \left\{ s^i_t \right\}_{i=1}^{L}, R^N_{t-1}, \tilde{C}_{t-1}, \zeta_t, g_t, \epsilon^{mp}_t \right\}, \tag{93}$$

$$\nu_t = \left\{ \left\{ \epsilon^{s,i}_t \right\}_{i=1}^{L}, \epsilon^{\zeta}_t, \epsilon^{g}_t, \epsilon^{mp}_t \right\}. \tag{94}$$

The parameters of the model are divided in calibrated ($\bar{\theta}$) and estimated ones ($\tilde{\theta}$). For instance for the case of our application with US data, this yields to:

$$\bar{\Theta} = \{\beta, \eta, \epsilon, \sigma, g, \gamma^{\tau}, \varphi, \chi, \Pi, \theta_{\Pi}, \theta_Y, D, \underline{B}, \rho_s, \rho_{\zeta}\}, \tag{95}$$

$$\tilde{\Theta} = \{\sigma_s, \sigma_{\zeta}, \sigma_g, \sigma_{mp}\}. \tag{96}$$

We use two neural networks to separate between individual and aggregate policy functions. The individual neural network $\psi^I_{NN}(\cdot)$ solves for labor supply:

$$\left\{ \left(H^i_t \right) = \psi^I_{NN} \left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta} | \bar{\Theta} \right) \right\}_{i=1}^{L}, \tag{97}$$

where $\mathbb{S}^i_t = \left\{ \tilde{B}^i_{t-1}, s^i_t \right\}$. The neural network for the aggregate policy functions $\psi^A_{NN}(\cdot)$ determines the wage and inflation:

$$\begin{pmatrix} \Pi_t \\ \tilde{W}_t \end{pmatrix} = \psi^A_{NN} \left(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta} \right). \tag{98}$$

One challenge is that we need to solve for the deterministic steady state values of the nominal rate and output, as the Taylor rule features them. To focus on the algorithm for the moment, we assume for the moment that we have trained already a neural network $\psi^{SS}_{NN}(\cdot)$ that maps the parameter values to the steady state values:

$$\begin{pmatrix} R \\ \tilde{Y} \end{pmatrix} = \psi^{SS}_{NN} \left(\tilde{\Theta} | \bar{\Theta} \right). \tag{99}$$

The training of the neural network precedes the training of the individiual and aggregate policy functions. It is described in Section F.1 and is very similar to the outlined algorithm below. This training process adapts the approach outlined below to the version without aggregate risk (e.g. Aiyagari version of the model).

The steps of the algorithm are as follows:

1. Set up the neural network to approximate the policy functions and guess the initial values for the neural network to initialize the algorithm:

(a) The neural network $\psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)$ for the individual policy functions

$$\left\{\left(H^i_t\right) = \psi^I_{NN}\left(\mathbb{S}^i_t, \mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}. \tag{100}$$

(b) The neural network $\psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right)$ for the aggregate policy functions:

$$\begin{pmatrix}\Pi_t \\ \tilde{W}_t\end{pmatrix} = \psi^A_{NN}\left(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}\right). \tag{101}$$

2. Solve for all time t variables for a given state vector of batch b. From the neural network, we have a current guess for the policy functions:

$$\{H^i_t\}^L_{i=1}, \Pi_t, \tilde{W}_t, \tag{102}$$

and the deterministic steady state values from the neural network that approximates the deterministic steady state values:

$$\begin{pmatrix}R \\ \tilde{Y}\end{pmatrix} = \psi^{SS}_{NN}\left(\tilde{\Theta}|\bar{\Theta}\right). \tag{103}$$

The next step is to calculate the following (aggregate) variables:

$$g_t = \bar{g}exp(z_t), \tag{104}$$

$$N_t = \left(\frac{1}{L}\sum_{i=1}^{L} H^i_t \exp(s^i_t)\right), \tag{105}$$

$$\tilde{Y}_t = N_t, \tag{106}$$

$$\tilde{T}_t = \left(\frac{R_{t-1}}{\Pi_t g_t} - 1\right) D, \tag{107}$$

$$\tilde{\tau}_t = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}_t \exp(s^i_t) H^i_t\right) - \tilde{T}_t}{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}_t \exp(s^i_t) H^i_t\right)^{1-\gamma_\tau}}, \tag{108}$$

$$MC_t = \tilde{W}_t. \tag{109}$$

As a next step, we can calculate the nominal interest rate, where we impose the zero lower bound:

$$R^N_t = R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{\tilde{Y}_t}{\tilde{Y}}\right)^{\theta_Y}\exp(\epsilon^{mp}_t), \tag{110}$$

$$R_t = \max\left[1, R^N_t\right]. \tag{111}$$

We pursue calculating each household's individual variables:

$$\left\{ \tilde{C}_t^i = \left(\frac{\tilde{\tau}_t(1-\gamma_\tau)\exp(s_t^i)\tilde{W}_t\left(\exp(s_t^i)\tilde{W}_t H_t^i\right)^{-\gamma_\tau}}{\chi(H_t^i)^\eta} \right)^{1/\sigma} \right\}_{i=1}^L, \tag{112}$$

$$\left\{ \tilde{Div}_t^i = \left(\tilde{Y}_t - \tilde{W}_t N_t\right)\exp(s_t^i) \right\}_{i=1}^L, \tag{113}$$

$$\left\{ \tilde{\omega}_t^i = \tilde{\tau}_t\left(\tilde{W}_t \exp(s_t^i) H_t^i\right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t g_t}\tilde{B}_{t-1}^i + \tilde{Div}_t^i \right\}_{i=1}^L, \tag{114}$$

$$\left\{ \tilde{B}_t^i = \tilde{\omega}_t^i - \tilde{C}_t^i \right\}_{i=1}^L. \tag{115}$$

Aggregate consumption is given as

$$\tilde{C}_t = \frac{1}{L}\sum_{i=1}^L \tilde{C}_t^i. \tag{116}$$

3. Use a Monte Carlo Approach to evaluate the expectations. We randomly draw M sets of next period shocks, which later allows us to evaluate the expectations. To increase the precision and reduce the number of necessary draws, we use the antithetic variate method. The antithetic variates technique creates to a given path $\{\nu_1, \nu_2, \nu_3, \dots, \nu_{M/2}\}$ also its antithetic path $\{-\nu_1, -\nu_2, -\nu_3, \dots, -\nu_{M/2}\}$. The drawn shocks are given as

$$\left\{ \left\{ \epsilon_{t+1}^{s,i,m} \right\}_{i=1}^L, \epsilon_{t+1}^{\zeta,m}, \epsilon_{t+1}^{g,m}, \epsilon_{t+1}^{mp,m} \right\}_{m=1}^M, \tag{117}$$

where the superscript m indicates to which shock draw the next period value is associated.

We then proceed to calculate the next period values of the stochastic state variables for all M draws, that is:

$$\left\{ \left\{ s_{t+1}^{i,m} = \rho_s s_t^i + \epsilon_{t+1}^{s,i,m} \right\}_{i=1}^L \right\}_{m=1}^M, \tag{118}$$

$$\left\{ \zeta_{t+1}^m = \rho_\zeta \zeta_t + \epsilon_{t+1}^{\zeta,m} \right\}_{m=1}^M, \tag{119}$$

$$\left\{ z_{t+1}^m = \rho_z z_t + \epsilon_{t+1}^{g,m} \right\}_{m=1}^M, \tag{120}$$

which then gives us the state variables for all M shock draws.

We can then use the neural networks for the individual and aggregate policy functions to calculate the individual and aggregate control variables:

$$\left\{ \left\{ \left(H_{t+1}^{i,m}\right) = \psi_{NN}^I\left(\mathbb{S}_{t+1}^{i,m}, \mathbb{S}_{t+1}^m, \tilde{\Theta}|\bar{\Theta}\right) \right\}_{i=1}^L \right\}_{m=1}^M, \tag{121}$$

$$\left\{ \begin{pmatrix} \Pi_{t+1}^m \\ \tilde{W}_{t+1}^m \end{pmatrix} = \psi_{NN}^A \left(\mathbb{S}_{t+1}^m, \tilde{\Theta} | \bar{\Theta} \right) \right\}_{m=1}^M . \tag{122}$$

The next step is to calculate the following (aggregate) variables for the period $t+1$ for all M draws:

$$\left\{ g_{t+1}^m = \bar{g} exp(z_{t+1}^m) \right\}_{m=1}^M , \tag{123}$$

$$\left\{ N_{t+1}^m = \left(\frac{1}{L} \sum_{i=1}^L H_{t+1}^{i,m} \exp(s_{t+1}^{i,m}) \right) \right\}_{m=1}^M , \tag{124}$$

$$\left\{ \tilde{Y}_{t+1}^m = N_{t+1}^m \right\}_{m=1}^M , \tag{125}$$

$$\left\{ \tilde{T}_{t+1}^m = \left(\frac{R_t}{\Pi_{t+1}^m g_{t+1}^m} - 1 \right) D \right\}_{m=1}^M , \tag{126}$$

$$\left\{ \tilde{\tau}_{t+1}^m = \frac{\frac{1}{L} \sum_{i=1}^L \left(\tilde{W}_{t+1}^m \exp(s_{t+1}^{i,m}) H_{t+1}^{i,m} \right) - \tilde{T}_{t+1}^m}{\frac{1}{L} \sum_{i=1}^L \left(\tilde{W}_{t+1}^m \exp(s_{t+1}^{i,m}) H_{t+1}^{i,m} \right)^{1-\gamma_\tau}} \right\}_{m=1}^M . \tag{127}$$

We pursue calculating for each household their individual variables in period $t+1$ for all M draws:

$$\left\{ \left\{ \tilde{C}_{t+1}^{i,m} = \left(\frac{\tilde{\tau}_{t+1}^m (1-\gamma_\tau) \exp(s_{t+1}^{i,m}) \tilde{W}_{t+1}^m \left(\exp(s_{t+1}^{i,m}) \tilde{W}_{t+1}^m H_{t+1}^{i,m} \right)^{-\gamma_\tau}}{\chi (H_{t+1}^{i,m})^\eta} \right)^{1/\sigma} \right\}_{i=1}^L \right\}_{m=1}^M , \tag{128}$$

$$\left\{ \left\{ \tilde{Div}_{t+1}^{i,m} = \left(\tilde{Y}_{t+1}^m - \tilde{W}_{t+1}^m N_{t+1}^m \right) \exp(s_{t+1}^{i,m}) \right\}_{i=1}^L \right\}_{m=1}^M , \tag{129}$$

$$\left\{ \left\{ \tilde{\omega}_{t+1}^{i,m} = \tilde{\tau}_{t+1}^m \left(\tilde{W}_{t+1}^m \exp(s_{t+1}^{i,m}) H_{t+1}^{i,m} \right)^{1-\gamma_\tau} + \frac{R_t}{\Pi_{t+1}^m g_{t+1}^m} \tilde{B}_t^i + \tilde{Div}_{t+1}^{i,m} \right\}_{i=1}^L \right\}_{m=1}^M , \tag{130}$$

$$\left\{ \left\{ \tilde{B}_{t+1}^{i,m} = \tilde{\omega}_{t+1}^{i,m} - \tilde{C}_{t+1}^{i,m} \right\}_{i=1}^L \right\}_{m=1}^M . \tag{131}$$

Aggregate consumption is given as

$$\left\{ \tilde{C}_{t+1}^m = \frac{1}{L} \sum_{i=1}^L \tilde{C}_{t+1}^{i,m} \right\}_{m=1}^M . \tag{132}$$

We need to ensure that the borrowing constraint of the households $B_t^i \geq \underline{B}$ are satisfied (see also equation (61)).[46] It is convenient to define $\bar{\lambda}_t^i$ as follows:

$$\bar{\lambda}_t^i = 1 - \mu_t^i. \tag{133}$$

We calculate the multiplier for all agents L using their Euler equation:

$$\left\{ \bar{\lambda}_t^i = \beta R_t \frac{1}{M} \sum_{m=1}^{M} \left[\left(\frac{\exp(\zeta_{t+1}^m)}{\exp(\zeta_t)} \right) \left(\frac{\tilde{C}_t^i}{\tilde{C}_{t+1}^{i,m}} \right)^{\sigma} \frac{1}{\Pi_{t+1}^m g_{t+1}^m} \right] \right\}_{i=1}^{L}, \tag{134}$$

where we can now evaluate the expectations by taking the mean of the M draws.

Equipped with this object, we use the Fischer-Burmeister function Ψ^{FB}, which can be used to capture computationally the complementary slackness conditions of the Karush-Kuhn-Tucker conditions to ensure that these conditions hold.[47] This can be written as

$$\Psi^{FB} \left(1 - \bar{\lambda}_t^i, \tilde{B}_t^i - \underline{B} \right). \tag{135}$$

We can now calculate the Euler error for the Euler equations via the Fischer-Burmeister function:

$$\left\{ L^{1,i} = \left(\Psi^{FB} \left(1 - \bar{\lambda}_t^i, \tilde{B}_t^i - \underline{B} \right) \right)^2 \right\}_{i=1}^{L}, \tag{136}$$

where $L^{1,i}$ captures the squared Euler error for agent's i Euler equation.

We also calculate the squared Euler error for the New Keynesian Phillips Curve, the resource constraint in period t and $t+1$, and market clearing for the bond in period t and $t+1$:

$$\begin{aligned} L^2 = \Bigg(& \left[\varphi \left(\frac{\Pi_t}{\Pi} - 1 \right) \frac{\Pi_t}{\Pi} \right] - (1-\epsilon) - \epsilon MC_t \\ & - \beta \varphi \frac{1}{M} \sum_{m=1}^{M} \left[\left(\frac{\exp(\zeta_{t+1}^m)}{\exp(\zeta_t)} \right) \left(\frac{\tilde{C}_{t+1}^m}{\tilde{C}_t} \right)^{-\sigma} \left(\frac{\Pi_{t+1}^m}{\Pi} - 1 \right) \frac{\Pi_{t+1}^m}{\Pi} \frac{\tilde{Y}_{t+1}^m}{\tilde{Y}_t} \right] \Bigg)^2, \end{aligned} \tag{137}$$

$$L^3 = \left(D - \frac{1}{L} \sum_{i=1}^{L} B_t^i \right)^2, \tag{138}$$

$$L^4 = \frac{1}{M} \sum_{m=1}^{M} \left(D - \frac{1}{L} \sum_{i=1}^{L} B_{t+1}^{i,m} \right)^2, \tag{139}$$

$$L^5 = \left(\tilde{Y}_t - \tilde{C}_t \right)^2, \tag{140}$$

[46]The complementary slackness conditions can be written as $\mu_t^i \geq 0,\ \left(\tilde{B}_t^i - \underline{B} \right) \geq 0,\ \mu_t^i \times \left(\tilde{B}_t^i - \underline{B} \right) = 0$.

[47]The complementary slackness conditions can be written for instance as: $e \geq 0,\ f \geq 0,\ e \times f = 0$. The Fischer-Burmeister function is defined as $\Psi^{FB}(e,f) = e + f - \sqrt{e^2 + f^2}$. If $\Psi^{FB}(e,f) = 0$, then the complementary slackness conditions are satisfied.

$$L^6 = \frac{1}{M}\sum_{m=1}^{M}\left(\tilde{Y}^m_{t+1} - \tilde{C}^m_{t+1}\right)^2. \tag{141}$$

4. Repeat steps 2 - 3 for each batch (B times), which gives the following loss components for the entire batch:

$$\left\{\left\{L^{1,i,b}\right\}_{i=1}^{L}, L^{2,b}, L^{3,b}, L^{4,b}, L^{5,b}, L^{6,b}\right\}_{b=1}^{B}. \tag{142}$$

5. Define the loss function:

$$\phi^L = \frac{1}{B}\sum_{b=1}^{B}\left[\sum_{i=1}^{L}\alpha_1^i L_1^{1,i,b} + \alpha_2 L^{2,b} + \alpha_3 L^{3,b} + \alpha_4 L^{4,b} + \alpha_5 L^{5,b} + \alpha_6 L^{6,b}\right], \tag{143}$$

where $\left\{\alpha_1^i\right\}_{i=1}^{L}, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6$ determine the weights for the different equations.

6. Optimize the parameters of the neural networks ψ^I_{NN} and ψ^A_{NN} to minimize the loss function ϕ^L with a stochastic gradient optimizer.

7. Repeat steps 2 - 6 N^{int} times to optimize several times. Redraw the shock realization in period $t+1$ at each internal optimization.

8. Simulate each batch economy b for T^{sim} periods using randomly drawn shocks. This creates then the state vector for the next iteration of the optimizer.

9. Repeat steps 2 - 8 N^{iter} times until the neural network converges on average to a sufficient low statistics.

F.1 Deterministic Steady State and Neural Network

We need the mapping from the parameters to the deterministic steady state values of the interest rate R and detrended output $\tilde{Y}$, as these objects enter the Taylor rule. To obtain this mapping, we solve the deterministic steady state of our HANK model, in which agents face and experience idiosyncratic risk, while aggregate risk is absent.

The DSS neural network solves for the deterministic steady state values:

$$\begin{pmatrix} R \\ \tilde{Y} \end{pmatrix} = \psi^{SS}_{NN}\left(\tilde{\Theta}|\bar{\Theta}\right). \tag{144}$$

To solve for this neural network, we also need to solve for the individual policy functions of the agents in the DSS.

$$\left\{\left(H_t^i\right) = \psi^{I,DSS}_{NN}\left(\mathbb{S}_t^i, \mathbb{S}, \tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}, \tag{145}$$

where DSS enters the superscript of the neural network for a clear distinction. Note that we let the deterministic steady state values of the aggregate state variables still enter this network. While this is not necessary, this is helpful as we use this trained neural network as starting point when we solve for the economy with aggregate risk.

1. Set up the neural network to approximate the policy functions and guess the initial values for the neural network to initialize the algorithm:

   (a) The neural network $\psi_{NN}^{SS}\left(\tilde{\Theta}|\bar{\Theta}\right)$ for the steady state parameters

   $$\begin{pmatrix} R \\ \tilde{Y} \end{pmatrix} = \psi_{NN}^{SS}\left(\tilde{\Theta}|\bar{\Theta}\right). \tag{146}$$

   (b) The neural network $\psi_{NN}^{I,DSS}\left(\mathbb{S}_t^i, \mathbb{S}, \tilde{\Theta}|\bar{\Theta}\right)$ for the individual policy functions without aggregate risk

   $$\left\{\left(H_t^i\right) = \psi_{NN}^{I,DSS}\left(\mathbb{S}_t^i, \mathbb{S}, \tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}. \tag{147}$$

2. Solve for all the current period individual variables. From the neural network, we have a current guess for the the deterministic steady state values

   $$\begin{pmatrix} R \\ \tilde{Y} \end{pmatrix} = \psi_{NN}^{SS}\left(\tilde{\Theta}|\bar{\Theta}\right), \tag{148}$$

   and individual policy functions

   $$\{H_t^i\}_{i=1}^{L}. \tag{149}$$

   Note that inflation is at target in the DSS, the wage equals its value in the DSS without idiosyncratic risk, and the growth rate is at its mean. This implies that we know

   $$\Pi, \tilde{W}, g. \tag{150}$$

   The next step is to calculate the following variables:

   $$N_t = \left(\frac{1}{L}\sum_{i=1}^{L} H_t^i \exp(s_t^i)\right), \tag{151}$$

   $$\tilde{Y}_t = N_t, \tag{152}$$

   $$\tilde{T} = \left(\frac{R}{\Pi g} - 1\right) D, \tag{153}$$

$$\tilde{\tau}_t = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}\exp(s_t^i)H_t^i\right) - \tilde{T}}{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}\exp(s_t^i)H_t^i\right)^{1-\gamma_\tau}}, \tag{154}$$

$$MC = \tilde{W}. \tag{155}$$

Note that we use here a subscript t to indicate that this variable is calculated based on the individual shock realizations. We assume that L is sufficiently large so that specific shock realizations and asset level of households do not affect the steady state value.[48] Once the network successfully approximates the DSS, aggregate variables such as (detrended) output are the same in each period, e.g. $\tilde{Y}_t \approx \tilde{Y}, \forall t$

We pursue calculating each household's individual variables:

$$\left\{\tilde{C}_t^i = \left(\frac{\tilde{\tau}_t(1-\gamma_\tau)\exp(s_t^i)\tilde{W}\left(\exp(s_t^i)\tilde{W}H_t^i\right)^{-\gamma_\tau}}{\chi(H_t^i)^\eta}\right)^{1/\sigma}\right\}_{i=1}^{L}, \tag{156}$$

$$\left\{\tilde{Div}_t^i = \left(\tilde{Y}_t - \tilde{W}N_t\right)\exp(s_t^i)\right\}_{i=1}^{L}, \tag{157}$$

$$\left\{\tilde{\omega}_t^i = \tilde{\tau}_t\left(\tilde{W}\exp(s_t^i)H_t^i\right)^{1-\gamma_\tau} + \frac{R}{\Pi g}\tilde{B}_{t-1}^i + \tilde{Div}_t^i\right\}_{i=1}^{L}, \tag{158}$$

$$\left\{\tilde{B}_t^i = \tilde{\omega}_t^i - \tilde{C}_t^i\right\}_{i=1}^{L}. \tag{159}$$

Aggregate consumption is given as

$$\tilde{C}_t = \frac{1}{L}\sum_{i=1}^{L}\tilde{C}_t^i. \tag{160}$$

3. Use a Monte Carlo Approach to evaluate the expectations, which requires randomly drawing M sets of next period shocks for the households (we use the antithetic variate method here as well). The drawn shocks are given as

$$\left\{\left\{\epsilon_{t+1}^{s,i,m}\right\}_{i=1}^{L}\right\}_{m=1}^{M}, \tag{161}$$

where the superscript m indicates to which shock draw the next period value is associated.

We then proceed to calculate the next period values of the stochastic state variables for all M draws, that is:

$$\left\{\left\{s_{t+1}^{i,m} = \rho_s s_t^i + \epsilon_{t+1}^{s,i,m}\right\}_{i=1}^{L}\right\}_{m=1}^{M}, \tag{162}$$

[48]Furthermore, the stochastic setup of the neural network allows us to average over several hundred thousands of draws as we consider in each iteration different draws for the batch B.

which then gives us the state variables for the households for all M shock draws.

We can then use the neural network for the extended individual policy functions in the DSS:

$$\left\{\left\{\left(H_{t+1}^{i,m}\right)=\psi_{NN}^{I,DSS}\left(\mathbb{S}_{t+1}^{i,m},\mathbb{S}^{m},\tilde{\Theta}|\bar{\Theta}\right)\right\}_{i=1}^{L}\right\}_{m=1}^{M}. \tag{163}$$

The next step is to calculate the following variables for the period $t+1$ for all M draws:

$$\left\{N_{t+1}^{m}=\left(\frac{1}{L}\sum_{i=1}^{L}H_{t+1}^{i,m}\exp(s_{t+1}^{i,m})\right)\right\}_{m=1}^{M}, \tag{164}$$

$$\left\{\tilde{Y}_{t+1}^{m}=N_{t+1}^{m}\right\}_{m=1}^{M}, \tag{165}$$

$$\left\{\tilde{\tau}_{t+1}^{m}=\frac{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}\exp(s_{t+1}^{i,m})H_{t+1}^{i,m}\right)-\tilde{T}}{\frac{1}{L}\sum_{i=1}^{L}\left(\tilde{W}\exp(s_{t+1}^{i,m})H_{t+1}^{i,m}\right)^{1-\gamma_\tau}}\right\}_{m=1}^{M}. \tag{166}$$

We pursue calculating for each household their individual variables in period $t+1$ for all M draws:

$$\left\{\left\{\tilde{C}_{t+1}^{i,m}=\left(\frac{\tilde{\tau}_{t+1}^{m}(1-\gamma_\tau)\exp(s_{t+1}^{i,m})\tilde{W}\left(\exp(s_{t+1}^{i,m})\tilde{W}H_{t+1}^{i,m}\right)^{-\gamma_\tau}}{\chi(H_{t+1}^{i,m})^{\eta}}\right)^{1/\sigma}\right\}_{i=1}^{L}\right\}_{m=1}^{M}, \tag{167}$$

$$\left\{\left\{\tilde{Div}_{t+1}^{i,m}=\left(\tilde{Y}_{t+1}^{m}-\tilde{W}N_{t+1}^{m}\right)\exp(s_{t+1}^{i,m})\right\}_{i=1}^{L}\right\}_{m=1}^{M}, \tag{168}$$

$$\left\{\left\{\tilde{\omega}_{t+1}^{i,m}=\tilde{\tau}_{t+1}^{m}\left(\tilde{W}\exp(s_{t+1}^{i,m})H_{t+1}^{i,m}\right)^{1-\gamma_\tau}+\frac{R}{\Pi g}\tilde{B}_{t}^{i}+\tilde{Div}_{t+1}^{i,m}\right\}_{i=1}^{L}\right\}_{m=1}^{M}, \tag{169}$$

$$\left\{\left\{\tilde{B}_{t+1}^{i,m}=\tilde{\omega}_{t+1}^{i,m}-\tilde{C}_{t+1}^{i,m}\right\}_{i=1}^{L}\right\}_{m=1}^{M}. \tag{170}$$

Aggregate consumption is given as

$$\left\{\tilde{C}_{t+1}^{m}=\frac{1}{L}\sum_{i=1}^{L}\tilde{C}_{t+1}^{i,m}\right\}_{m=1}^{M}. \tag{171}$$

We need to ensure that the borrowing constraint of the households $B_t^i \geq \underline{B}$ is satisfied. It is convenient to define $\bar{\lambda}_t^i$ as follows:

$$\bar{\lambda}_t^i=1-\mu_t^i. \tag{172}$$

We calculate the multiplier for all agents L using their Euler equation:

$$\left\{\bar{\lambda}_t^i = \beta R \frac{1}{M} \sum_{m=1}^{M} \left[\left(\frac{\tilde{C}_t^i}{\tilde{C}_{t+1}^{i,m}} \right)^{\sigma} \frac{1}{\Pi g} \right] \right\}_{i=1}^{L}, \tag{173}$$

where we can now evaluate the expectations by taking the mean of the M draws.

Equipped with this object, we use again the Fischer-Burmeister function Ψ^{FB} to capture computationally the complementary slackness conditions of the Karush-Kuhn-Tucker conditions to ensure that these conditions hold. This can be written as

$$\Psi^{FB}\left(1 - \bar{\lambda}_t^i, \tilde{B}_t^i - \underline{B}\right). \tag{174}$$

We can now calculate the Euler error for the Euler equations via the Fischer-Burmeister function:

$$\left\{L^{1,DSS,i} = \left(\Psi^{FB}\left(1 - \bar{\lambda}_t^i, \tilde{B}_t^i - \underline{B}\right)\right)^2\right\}_{i=1}^{L}, \tag{175}$$

where $L^{1,DSS,i}$ captures the squared Euler error for agent's i Euler equation.

We also calculate the squared Euler error for aggregated output, the resource constraint in period t and $t+1$, and market clearing for the bond in period t and $t+1$:

$$L^{2,DSS} = \left(\tilde{Y}_t - \tilde{Y}\right)^2,$$

$$L^{3,DSS} = \left(D - \frac{1}{L}\sum_{i=1}^{L} B_t^i\right)^2, \tag{176}$$

$$L^{4,DSS} = \frac{1}{M}\sum_{m=1}^{M}\left(D - \frac{1}{L}\sum_{i=1}^{L} B_{t+1}^{i,m}\right)^2, \tag{177}$$

$$L^{5,DSS} = \left(\tilde{Y}_t - \tilde{C}_t\right)^2, \tag{178}$$

$$L^{6,DSS} = \frac{1}{M}\sum_{m=1}^{M}\left(\tilde{Y}_{t+1}^m - \tilde{C}_{t+1}^m\right)^2. \tag{179}$$

4. Repeat steps 2 - 3 for each batch (B times), which gives the following loss components for the entire batch:

$$\left\{\left\{L^{1,DSS,i,b}\right\}_{i=1}^{L}, L^{2,DSS,b}, L^{3,DSS,b}, L^{4,DSS,b}, L^{5,DSS,b}, L^{6,DSS,b}\right\}_{b=1}^{B}. \tag{180}$$

5. Define the loss function:
$$\phi^L = \frac{1}{B}\sum_{b=1}^{B}\left[\sum_{i=1}^{L}\alpha_1^{i,DSS}L_1^{1,DSS,i,b} + \sum_{j=2}^{6}\alpha_j^{DSS}L^{j,DSS,b}\right], \tag{181}$$
where the weights for the equations are determined by $\left\{\alpha_1^{i,DSS}\right\}_{i=1}^{L}$ and $\alpha_j^{DSS}, \forall j = 2,3,4,5,6$.

6. Optimize the parameters of the neural networks ψ_{NN}^{SS} and $\psi_{NN}^{I,DSS}$ to minimize the loss function ϕ^L with a stochastic gradient optimizer.

7. Repeat steps 2 - 6 N^{int} times to optimize several times. Redraw the shock realization in period $t+1$ at each internal optimization.

8. Simulate each batch economy b for T^{sim} periods using randomly drawn shocks. This creates the state vector for the next iteration of the optimizer.

9. Repeat steps 2 - 8 N^{iter} times until the neural network converges on average to a sufficient low statistics.

F.2 HANK with real data: Additional Results

Training of DSS Neural Network The neural network to approximate output and nominal rate in the DSS is trained for 10,000 iterations. During each iteration step, we conduct 15 optimization steps, including a redrawing of the shocks realizations, and use 20 Monte Carlo draws. After each iteration, the economy is simulated for 20 periods. The parameters are redrawn at every 10th iteration and are initialized with a longer simulation of 200 periods. Note that this step requires to also train a temporary individual policy function, which provides the decision of households in the DSS. The convergence of the neural network is shown in Figure 13, where the mean squared error averaged across the equations that we minimize is shown. At the end of the training, the neural network approximates the DSS with a very high accuracy.

Training of Individual and Aggregate Policy Functions Neural Networks We use 30,000 iterations to jointly train the neural networks to approximate the extended individual and aggregate policy functions. During each iteration step, we conduct 15 optimization steps, including a redrawing of the shocks realizations, and use 20 Monte Carlo draws. After each iteration, the economy is simulated for 20 periods. The parameters are redrawn at every 10th iteration and are initialized with a longer simulation of 200 period. Figure 14 shows that the loss function converges to an error in the magnitude of 10e-6. For a better convergence, we slowly introduce aggregate risk between period 1000 and 2000 by scaling the standard deviations of the aggregate shocks from a tiny value to the actual values. Similarly, we introduce the ZLB

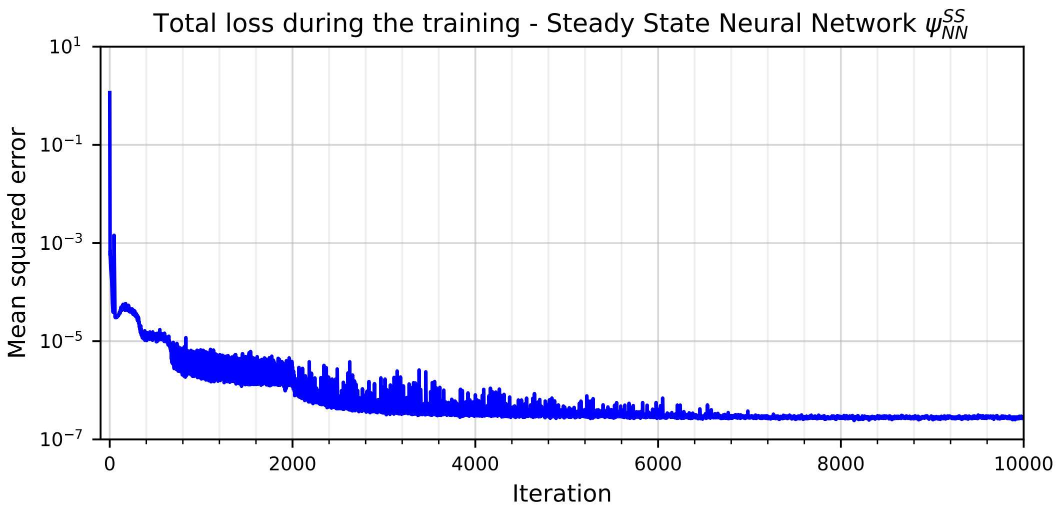

Figure 13: Figure shows the dynamics of the mean squared error, which is averaged across the equations that we minimize, associated with the deterministic steady state values of the nominal rate R and output Y. The vertical axis has a logarithmic scale.

between periods 5000 and 1000 by slowly scaling the impact of the ZLB. In particular, we use the following functional form:

$$R_t = \min[R_t^N, 1] + a^{ZLB} \max[R_t^N - 1, 0]. \tag{182}$$

The parameter a^{ZLB} is stepwise lowered from 1 to 0 to move from the economy without ZLB to the ZLB economy.

F.3 HANK with simulated data

To be updated and continued ...

Additional Results The posterior of the nonlinear HANK model is shown in Figure 15, in which we compare the median to the true value. This demonstrates that our method is well suited to estimating complex nonlinear models.

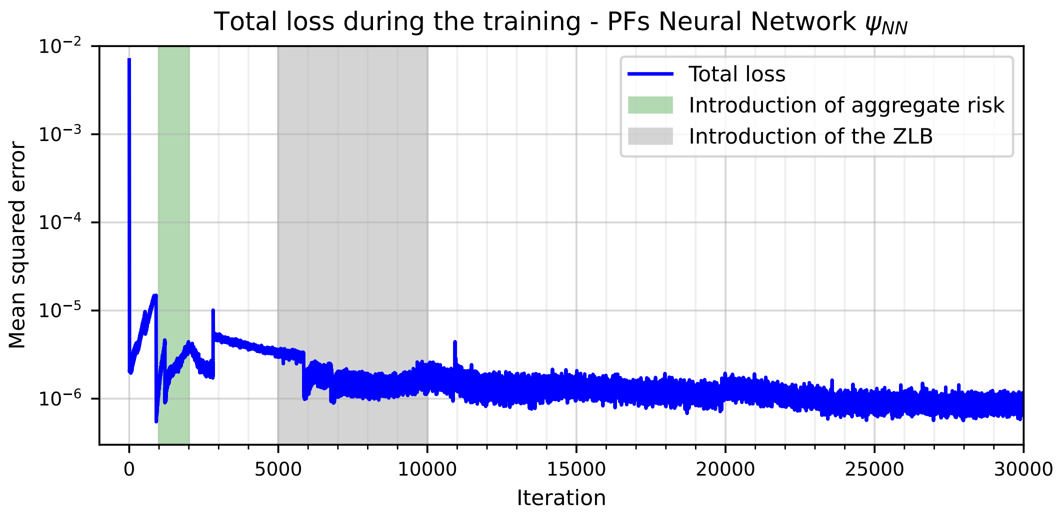

Figure 14: Figure shows the mean squared error over 30,000 iterations for training the neural networks associated with the individual and aggregate policy functions. The shaded areas indicate the periods in which we introduce aggregate risk and the zero lower bound.

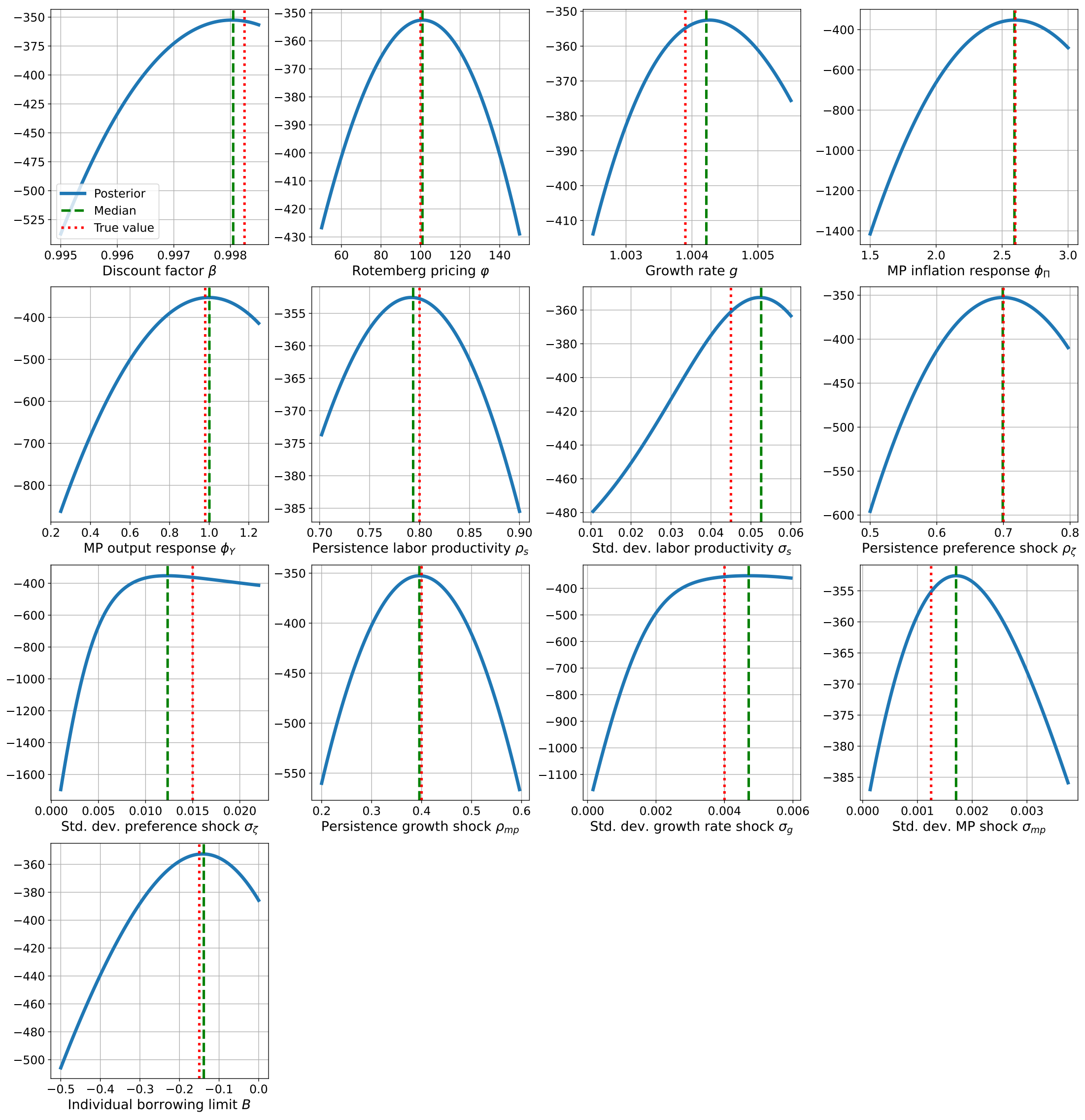

Figure 15: Posterior of the nonlinear HANK model estimated with the developed neural network estimation procedure. Each parameter is varied, while the other parameters are fixed at the posterior median. The posterior median is the green dashed line, while the true value from the data-generating process is the red dotted line.